This dissertation titled
Nonlinear Semi-supervised and Unsupervised Metric Learning with Applications in
Neuroimaging

by
PIN ZHANG

has been approved for
the School of Electrical Engineering and Computer Science
and the Russ College of Engineering and Technology by

Jundong Liu
Associate Professor of Electrical Engineering and Computer Science

Dennis Irwin
Dean, Russ College of Engineering and Technology
ABSTRACT

ZHANG, PIN, Ph.D., August 2018, Computer Science

Nonlinear Semi-supervised and Unsupervised Metric Learning with Applications in Neuroimaging (117 pp.)

Director of Dissertation: Jundong Liu

In many machine learning and data mining algorithms, pairwise distances (or dissimilarities) among data samples are computed based on the Euclidean metric, where all feature components are treated equally and assigned with the same weight. Learning a customized metric from the input data can often significantly improve the performance of the algorithms.

In this dissertation, we propose two nonlinear distance metric learning (DML) frameworks to boost the performance of semi-supervised learning (SSL) and unsupervised learning (USL) algorithms, respectively. Formulated under a constrained optimization framework, our proposed SSL-DML method learns a smooth nonlinear feature space transformation that makes the input data samples more linearly separable in Laplacian SVM (LapSVM). Our USL-ML solution, on the other hand, aims to increase data’s linear separability for $k$-means. A geometric model called Coherent Point Drifting (CPD) is utilized in both frameworks to move data points towards more desirable locations. The choice of CPD is with two considerations: 1) its remarkable capability in generating high-order yet smooth deformations; and 2) the available mechanism within CPD for assigning different levels of smoothness to data points.

Application-wise, we apply our SSL-DML to predict the conversion of Alzheimer’s Disease (AD) from its early stage: Mild Cognitive Impairment (MCI). The proposed USL-DML solution is utilized to improve the patient clustering. Using neuroimage data from the Alzheimer’s Disease Neuroimaging Initiative (ADNI) database, we evaluate
the effectiveness of the proposed frameworks. The experimental results demonstrate the improvements over the state-of-the-art solutions within the same category.
DEDICATION

To my parents Yuliang Zhang and Xiulan Wang, who always love and encourage me unconditionally. This dissertation is also dedicated to my wife Yue Zhao, who has been accompanying me along the way and overcoming the challenges in our life together.

Thank you all for your endless love!
ACKNOWLEDGMENTS

The path to this dissertation is full of challenges. Along the way, I am fortunate to have knowledgeable professors who guided me in my research, and valuable friends who helped me in both life and study.

First of all, I would like to express my sincere gratitude to my director, Dr. Jundong Liu. His earnest manner in research and inexhaustible patience in guiding me has benefited me greatly in my research and graduate study. I harvested both cutting-edge technologies and scientific researching methods from his professional advising.

Also, I am honored to have Drs. Razvan Bunescu, Jeffery Dill, Chang Liu, Li Xu and Aili Guo as my committee members. Their insightful thoughts and valuable advice have widely broadened my view of research. In addition, I also thank Dr. Charles Smith of the University of Kentucky, who has been cooperating with our laboratory in my research area.

Finally, I deeply appreciate the help and support from my friends at Ohio University: Bibo Shi, Yani Chen, Kevin H. Hobbs, Zhewei Wang, Tao Sun, Huihui Xu and Letian Lin. Thank you all for your ideas and suggestions for my dissertation.
# Table of Contents

Abstract ................................................................................. 3  
Dedication ............................................................................. 5  
Acknowledgments ................................................................... 6  
List of Tables .......................................................................... 9  
List of Figures ......................................................................... 10  
List of Notations ..................................................................... 12  

1 Introduction ......................................................................... 13  
  1.1 Semi-supervised Learning and Semi-supervised Metric Learning .... 13  
  1.2 Unsupervised Learning and Unsupervised Metric Learning ......... 18  
  1.3 Contributions ................................................................... 22  
  1.4 Overview of the Dissertation ............................................. 24  

2 Preliminaries ......................................................................... 26  
  2.1 Geometric Transformation .............................................. 26  
  2.2 Transformation Regularization and CPD Transformation Model .... 27  
  2.3 Metrics and Metric Learning ........................................... 30  

3 Literature Review .............................................................. 35  
  3.1 Semi-supervised Metric Learning .................................... 35  
  3.2 Unsupervised Metric Learning ........................................ 38  

4 Nonlinear Metric Learning for Semi-supervised Learning via CPD .... 42  
  4.1 Laplacian SVM ............................................................ 42  
  4.2 CPD based Metric Learning with LapSVM (CPD-LapSVM) ........ 43  
  4.3 Derivation of CPD-based Transductive SVM (CPD-TSVM) ....... 48  
  4.4 Experiments on Synthetic Data ....................................... 49  
  4.5 Experiments on UCI datasets ......................................... 53  

5 Nonlinear Metric Learning for Unsupervised Learning via CPD ....... 57  
  5.1 CPD based Unsupervised Metric Learning (CPD-UML) .......... 57  
  5.2 Experiments on Synthetic Data ....................................... 64  
  5.3 Experiments on UCI Datasets ......................................... 67
6 Application on Alzheimer’s Disease Early Detection .......................... 74
   6.1 Alzheimer’s Disease and Early Prediction ................................. 74
   6.2 Neuroimaging and Machine Learning ....................................... 77
   6.3 Alzheimer’s Disease Neuroimaging Initiative (ADNI) Database ........ 80
   6.4 Semi-Supervised Learning for Prediction of AD Conversion from MCI . 83
   6.5 Improve Patient Clustering via Nonlinear Feature Transformation and Embedding ................................................................. 87
   6.6 Quad-mesh based Feature Extraction ........................................... 91

7 Conclusions & Perspectives ............................................................. 101

References ....................................................................................... 104
LIST OF TABLES

Table                                                                 |
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Page</td>
</tr>
<tr>
<td>4.1 Seven UCI benchmark datasets used in experiments. Columns show</td>
</tr>
<tr>
<td>the name, numbers of samples, attributes and classes of each</td>
</tr>
<tr>
<td>dataset.</td>
</tr>
<tr>
<td>4.2 The mean and standard deviation of classification accuracy of</td>
</tr>
<tr>
<td>each tested method on seven benchmark UCI datasets. The prefix l</td>
</tr>
<tr>
<td>denotes the linear version and r denotes kernel version. Boldface</td>
</tr>
<tr>
<td>indicates the highest classification accuracy for each dataset. The</td>
</tr>
<tr>
<td>lower number in the parenthesis denotes the ranking score of each</td>
</tr>
<tr>
<td>method on the given dataset.</td>
</tr>
<tr>
<td>5.1 Six benchmark datasets used in experiments. Columns show the</td>
</tr>
<tr>
<td>name, numbers of samples, attributes and classes of each dataset.</td>
</tr>
<tr>
<td>5.2 Accuracy</td>
</tr>
<tr>
<td>5.3 Normalized Mutual Information</td>
</tr>
<tr>
<td>5.4 Purity</td>
</tr>
<tr>
<td>6.1 Demographics and clinical evaluations.</td>
</tr>
<tr>
<td>6.2 Performance comparison of CPD-LapSVM with other methods for AD</td>
</tr>
<tr>
<td>vs. NC classifications. Boldface denotes the best ACC &amp; AUC</td>
</tr>
<tr>
<td>performance.</td>
</tr>
<tr>
<td>6.3 Performance comparison of CPD-LapSVM with other methods for pMCI</td>
</tr>
<tr>
<td>vs. sMCI classifications. Boldface denotes the best ACC &amp; AUC</td>
</tr>
<tr>
<td>performance.</td>
</tr>
<tr>
<td>6.4 Comparisons of pMCI vs. sMCI classification solutions using</td>
</tr>
<tr>
<td>ADNI database.Boldface denotes the best performance for the</td>
</tr>
<tr>
<td>measure of ACC &amp; AUC.</td>
</tr>
<tr>
<td>6.5 Measurements of tested methods for AD/NC clustering.</td>
</tr>
<tr>
<td>6.6 Measurements of tested methods for pMCI/sMCI clustering.</td>
</tr>
<tr>
<td>6.7 Measurements of tested methods for AD/MCI/NC clustering.</td>
</tr>
<tr>
<td>6.8 Performance comparison of various RD/RDA features for AD/MCI/NC</td>
</tr>
<tr>
<td>classifications. Boldface denotes the best performance for each</td>
</tr>
<tr>
<td>measure.</td>
</tr>
<tr>
<td>6.9 comparison of the proposed method with other existing</td>
</tr>
<tr>
<td>hippocampus methods for AD/MCI classifications.</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Illustration for semi-supervised learning.</td>
</tr>
<tr>
<td>1.2</td>
<td>Illustration of clustering.</td>
</tr>
<tr>
<td>2.1</td>
<td>Illustrations for typical linear transformations.</td>
</tr>
<tr>
<td>2.2</td>
<td>Illustrations of the CPD transformation (Myronenko &amp; Song, 2010).</td>
</tr>
<tr>
<td>2.3</td>
<td>Illustration for a linear metric learning according to the color information.</td>
</tr>
<tr>
<td>2.4</td>
<td>Illustration for a linear metric learning according to shape information.</td>
</tr>
<tr>
<td>2.5</td>
<td>Illustration for a nonlinear metric learning.</td>
</tr>
<tr>
<td>3.1</td>
<td>An example of nonlinear dimensionality reduction. The 2-dimensional representation in (C) shows the mapping in LLE for the three-dimensional data (B) sampled from the manifold in (A). The mapping preserves the neighborhood relationship.</td>
</tr>
<tr>
<td>4.1</td>
<td>Illustration for a full graph.</td>
</tr>
<tr>
<td>4.2</td>
<td>Illustration of the concept of assigning different $\sigma$s to labeled and unlabeled points. Refer to text for details.</td>
</tr>
<tr>
<td>4.3</td>
<td>Left column: classification results of linear LapSVM (a), CPD-LapSVM with equal smoothness range (b), and CPD-LapSVM with larger smoothness range for labeled samples (c), respectively. Right column (d) - (f): initial deformation field, deformation field of (b) and deformation field of (c).</td>
</tr>
<tr>
<td>4.4</td>
<td>Left column: classification results of kernel LapSVM with RBF kernels width $= 1$, $8$, and $32$. Right column: results of kernel version CPD-LapSVM with RBF kernels width $= 1$, $8$, and $32$.</td>
</tr>
<tr>
<td>4.5</td>
<td>Classification accuracies of tested methods w.r.t. different prevalence levels of labeled data. The prefix $l$ denotes linear methods and $r$ denotes kernel methods using RBF kernels.</td>
</tr>
<tr>
<td>5.1</td>
<td>(a) clustering result of $k$-means; (b) clustering result of CPD-UML; (c) and (d) show the deformation field of (b). Color figures are best viewed on screen.</td>
</tr>
<tr>
<td>5.2</td>
<td>First column: clustering results of kernel $k$-means with RBF kernels width $= 4$, $8$, $16$ and $32$. Second column: results of kernel version CPD-UML with RBF kernels width $= 4$, $8$, $16$ and $32$.</td>
</tr>
<tr>
<td>6.1</td>
<td>Comparison between a healthy brain and the brain with AD.</td>
</tr>
<tr>
<td>6.2</td>
<td>Clinical disease stages. (Jack et al., 2010).</td>
</tr>
<tr>
<td>6.3</td>
<td>PET images showing the Amyloid accumulation with AD.</td>
</tr>
<tr>
<td>6.4</td>
<td>Working flow of AD identification using neuroimaging and machine learning.</td>
</tr>
</tbody>
</table>
6.5 (a): brain MRI of an ADNI subject. (b): anatomical segmentation of (a). Colors indicate different brain structures. .................................................. 81
6.6 ACC scores w.r.t different ratio of revealed labels in pMCI vs. sMCI classifications ................................................................. 86
6.7 An example of the Fiedler vector for a hippocampus. Warmer color stands for higher value. ................................................................. 95
6.8 An example of the parameterizations for a hippocampus. .................. 96
6.9 An example of the center line and radial distance for a hippocampus. .... 97
LIST OF NOTATIONS


L, S, W, Y, ψ

D, N, S, U  Sets of data instances.

V, S, X

u, v, x, y  Lowercase symbols and letters stand for scalars.

ψ, σ, λ, ξ

b, e, p, s  Lowercase bold Roman letters represent column vectors.

w, x, y, z

H, R, S  Blackboard bold letters represent manifolds or spaces.

G  Gaussian low-pass filter.

G̃  A column of Gaussian low-pass filters.

I  The identity matrix.

0  A matrix with all elements as 0.
1 INTRODUCTION

Machine learning techniques are drawing more attention these days as they have the abilities to identify hidden patterns or distributions from data without being programmed explicitly (Samuel, 2000). Based on the availability of the label information from given data, machine learning can be categorized into supervised learning, semi-supervised learning (SSL) and unsupervised learning (USL). Training samples are all labeled in supervised learning, partially labeled in SSL and not labeled in USL. While supervised learning has been well applied in various fields, SSL and USL have their unique advantage in the areas where the labeled instances are scarce due to the difficulty and high cost in labeling procedures. For example, the disease labels for patients are hard to get because the labeling requires professional skills and time-consuming follow-up examinations. By exploiting unlabeled data, SSL and USL learn classifiers or clusters based on certain assumptions regarding the distribution of input data.

1.1 Semi-supervised Learning and Semi-supervised Metric Learning

An example is illustrated in Fig. 1.1 to show how SSL can learn from unlabelled data. Assume in a binary classification problem, we are required to learn a classifier from the given training samples. While samples are abundant, there are only 2 samples with opposite labels, shown with blue and red colors in Fig. 1.1 (a). Using supervised learning, only these 2 labeled samples can be used to train a classifier, and the classification boundary can be any of the three lines shown in Fig. 1.1 (a). However, if we look closely into the unlabelled samples, it is clear the classification boundary in Fig. 1.1 (b) is intuitively more reasonable because each group separated by the new boundary are more compact. From the example, we can see semi-supervised learning algorithms can learn useful information from unlabelled training samples.
A variety of SSL paradigms have been proposed in the past 20 years or so. Label propagation, an earlier semi-supervised method was proposed in (X. Zhu & Ghahramani, 2002). In this method, unlabeled samples are labeled with soft labels by propagating the labels from labeled samples iteratively. Upon convergence, all samples are labeled with certain labels. Label propagation utilizes the information in the graph structure provided by a large amount of unlabeled data. Then, graph-based methods were developed using
a graph representation of the data to achieve classification. In (Belkin & Niyogi, 2004), the framework of manifold regularization was brought into semi-supervised learning. Two of the most commonly used algorithms based on the regularization of manifold structure: Laplacian support vector machine (LapSVM) and Laplacian regularized least squares (LapRLS) were proposed in (Belkin, Niyogi, & Sindhwani, 2006). These methods integrated the graph structure and the classification error together to improve the performance (Sindhwani, Niyogi, & Belkin, 2005). Other well-known graph-based methods are: graph min-cuts algorithm (Blum & Chawla, 2001) and harmonic function algorithms (X. Zhu, Ghahramani, Lafferty, et al., 2003). Besides, many improvements on LapSVM were also proposed in recent years, such as multi-kernel LapSVM (Yang & Fu, 2014) and the kernel classification framework applied on LapSVM (F. Wang, Zuo, Zhang, Meng, & Zhang, 2015). Another important category of semi-supervised learning methods is based on the low-density separation. The idea is to separate the samples into regions where there are few data points (both labeled and unlabeled data points). One of the commonly used algorithms within this category is the transductive support vector machine (TSVM), which is also named as Semi-supervised support vector machine (S3VM) (X. Zhu, 2005; Demiriz & Bennett, 2001; Fung & Mangasarian, 2001; Cristianini, 2004; Demiriz, Bennett, & Embrechts, 1999). Its approach is adding one more hinge loss item on SVM objective function for unlabeled points. Then, the optimization solution for the new objective function will separate the unlabeled samples from different classes with a large margin (Bennett, Demiriz, et al., 1999). The improvements on TSVM such as multi-kernel TSVM (X. Tian, Gasso, Canu, & de Rouen, 2011) were also proposed recently. There are also many other useful semi-supervised learning categories such as generative models (Nigam, McCallum, Thrun, & Mitchell, 2000; Baluja, 1999; Holub, n.d.; Baluja, 1998), multi-view algorithms (Yu, Wang, & Tao, 2012; Luo, Tao, Xu, & Li, 2013; Sindhwani et al., 2005; Brefeld, Gärtner, Scheffer, & Wrobel, 2006; de Sa, 1994),
self-training methods (Yarowsky, 1995; Rosenberg, Hebert, & Schneiderman, 2005; Riloff, Wiebe, & Wilson, 2003; Maeireizo, Litman, & Hwa, 2004) and neural network based solutions (Weston, Ratle, Mobahi, & Collobert, 2012; Ranzato & Szummer, 2008; Rifai, Dauphin, Vincent, Bengio, & Muller, 2011).

To make up for the lack of labeled samples, SSL solutions are commonly designed under certain assumption regarding the distribution of input data. For example, generative models assume the data follow an identifiable mixture distribution, and seek to determine the components through labeled samples (Baluja, 1998). The success of TSVM and its variants (Demiriz & Bennett, 2001; Cristianini, 2004) depends on the validity of the assumption that unlabeled data from different classes are linearly separable under the feature space. Graph-based SSL methods (Blum & Chawla, 2001; Belkin & Niyogi, 2004) construct graphs where nodes represent samples (both labeled and unlabeled), and edges (may be weighted) reflect the similarity between samples. Nodes connected with large-weight edges are assumed highly likely to be assigned with the same labels. The assumption behind the generative models is the given dataset has an identifiable mixture distribution, which could be determined by the labeled samples. In multi-view algorithms, it is assumed that every input sample is represented by different sets of features by splitting the given features.

The aforementioned assumptions in SSL, if violated in practice, could easily result in the limited validity of the models and subsequently poor classification performance. Geometrically transforming the input points to make them in accordance with the assumed data distribution would provide a remedy. For distance/similarity-based algorithms, learning such a transformation is equivalent to learning a new distance metric from the training samples. Distance metric learning (DML) has been extensively studied under supervised setting (Goldberger, Hinton, Roweis, & Salakhutdinov, 2004; Schultz & Joachims, 2004; Globerson & Roweis, 2005; Weinberger & Saul, 2009; Hoi, Liu, Lyu,
However, it is not trivial to generalize supervised DML solutions, especially those developed under nearest neighbor (NN) paradigm, to handle SSL problems.

Under semi-supervised setting, efforts for designing DML solutions have also been put forward, mainly along two directions: constrained clustering and semi-supervised learning/classification. Solutions for constrained clustering (Xing, Ng, Jordan, & Russell, 2003; Bilenko, Basu, & Mooney, 2004; Yeung & Chang, 2007; Kulis, Basu, Dhillon, & Mooney, 2009; Baghshah & Shouraki, 2010), where labels are not available, focus on integrating pairwise similarity (must-link) and/or dissimilarity (cannot-link) constraints to improve data grouping. Most of the methods learn global Mahalanobis metrics, which are equivalent to linear transformations applied to the data points. DML solutions for semi-supervised learning/classification commonly focus on formulating new regularizations to impose desired membership coherence throughout the data domain. LRML (Hoi, Liu, & Chang, 2010) and IDML (Dhillon, Talukdar, & Crammer, 2010) both assume the data lie approximately on a manifold of much lower dimension than the input space. The regularization term in LRML is a graph Laplacian, while IDML algorithm minimizes the harmonic energy over the data graph. Using a similar objective function as in LRML, SSM-DML (Yu et al., 2012) learns multiple Mahalanobis metrics for different feature sets. In SERAPH (Niu, Dai, Yamada, & Sugiyama, 2014), an information-theoretic regularization is used to specify neighborhood relationship. OLapSVM (Y. Gu & Feng, 2013) parameterizes graph weights through learning a Mahalanobis distance metric under Laplacian SVM. In JSL (Irina, Eric, Habrard, & Sebban, 2015), a linear metric and classifier are jointly learned, where the metric is sought to be a matrix-induced kernel function that satisfies the so-called \((\epsilon, \gamma, \tau)\)-goodness properties. Despite the reported
improvements, the existing semi-supervised DML solutions are mostly linear models performing under either input space or kernel space, which limit their capabilities in dealing with complex data.

1.2 Unsupervised Learning and Unsupervised Metric Learning

Clustering is a major unsupervised learning task. The goal of clustering is to separate dataset into different groups/clusters, where samples within the same group are similar to each other, and the dissimilar samples are separated into different groups. Fig. 1.2 illustrates the procedure of clustering.

Figure 1.2: Illustration of clustering.

$k$-means (Lloyd, 1982) is among the simplest and most widely used methods. With the specified cluster number $K$, $k$-means algorithm clusters the samples into $K$ clusters, which minimize the within-cluster sum-of-squares for the given datasets. $k$-means tries to get the clusters with the most internally coherent. While $k$-means is a simple and effective clustering method, it suffers from its assumption which assumes the clusters are convex and isotropic. Due to this drawback, it has poor performance on the elongated clusters and
irregular manifolds shapes. Gaussian mixture model (GMM) (Conley, 1999; Reynolds, 2015) is another mature clustering method which is based on a probabilistic model. GMM assumes that all the data samples are derived from a mixture of Gaussian distributions. Then the clustering task is converted into an optimization problem looking for the optimal parameters within the Gaussian, which maximize a posteriori estimation. An iterative expectation-maximization (EM) algorithm is used in solving the optimization problem. GMM can identify the elongated clusters and irregular manifolds shapes, which can not be handled by $k$-means. However, it often fails when there are insufficiently data instances provided for each mixture since the given data can not present the assumed Gaussian distribution well. Spectral clustering (Von Luxburg, 2007) methods treat clustering tasks as graph partitioning problems. The data instances are represented as graph vertices. The vertices are connected using edges, whose weights represent the similarity between vertices. Spectral clustering methods cluster the data points according to the spectrum of the similarity matrix. In detail, the solutions compute the eigenvalues and eigenvectors of the Laplacian matrix and map the sample points to a lower-dimensional space based on the achieved eigenvectors. The projected points have a better separability and can be clustered more easily in the mapped space. Hierarchical clustering (Johnson, 1967) solutions perform clustering analysis by trying to construct the hierarchy of clusters. The hierarchy has a tree shape, whose root is the set of all the samples. The hierarchy is constructed by splitting or merging the sub-clusters successively. There are two major types of hierarchical clustering strategies: agglomerative and divisive. Agglomerative is a "bottom-up" strategy. It merges a pair of clusters into one bigger cluster. The procedure moves up along the hierarchical structure from the bottom to top. The divisive strategy works in the opposite way. It is a "top-down" strategy, which starts from the root cluster and splits the clusters into pairs of smaller clusters. The procedure moves down along the hierarchical structure from the top to bottom. Hierarchical clustering methods depend heavily on the linkage
criteria used during merging or splitting. Typical linkage criteria are ward, complete and average linkage. The linkage types serve as different metrics determining the similarity between sub-clusters. The method density-based spatial clustering of applications with noise (DBSCAN) (Birant & Kut, 2007) treats the clustering tasks as seeking the low-density areas which can be used as the separation between high-density areas. DBSCAN is capable of finding clusters with arbitrary shape and sizes. However, the performance of this method depends heavily on the distance metric used in the procedure of determining the neighbors of each sample.

In measuring the similarities among data samples, the Euclidean distance is the most common choice in clustering algorithms. Under Euclidean distance, feature components are assigned with the same weight, which essentially assumes all features are equally important across the entire data space. In practice, such setup is often not optimal. Learning a customized metric function from the data samples can usually boost the performance of various machine learning algorithms (Bellet, Habrard, & Sebban, 2013).

While metric learning has been extensively researched under supervised (Xing et al., 2003; Weinberger & Saul, 2009; J. Wang, Kalousis, & Woznica, 2012; Noh, Zhang, & Lee, 2010) and semi-supervised settings (Peng, Heisterkamp, & Dai, 2002; Domeniconi, Gunopulos, et al., 2001; P. Zhang, Shi, Smith, & Liu, 2016; Niu et al., 2014), unsupervised metric learning (UML) remains a challenge, in part due to the absence of ground-truth label information to define a learning optimality. In this dissertation, we focus on the problem of UML for clustering.

As the goal of clustering is to capture the natural separations among data samples, one common practice in the existing UML solutions is to increase the data separability and make the separations more identifiable for the ensuing clustering algorithm. Such separability gain can be achieved by projecting data samples onto a carefully chosen low-dimensional manifold, under which the geometric relationships like the graph distribution
and pairwise distances can be preserved. The projections can be carried out linearly, as through the Principle Component Analysis, or nonlinearly, as via manifold learning solutions. Under the dimension-reduced spaces, clustering algorithms like GMM and k-means are utilized to perform the classification. Recent years have seen the developments of UML solutions exploring different setups for the low-dimensional manifolds. FME (Nie, Xu, Tsang, & Zhang, 2010) relies on the learning of an optimum linear regression function to specify the target low-dimensional space. (Abou-Moustafa, Schuurmans, & Ferrie, 2013) model local sample densities of the data to estimate a new metric space, and use the learned metric as the basis to construct graphs for manifold learning. Application-specific manifolds, such as Grassmann space (Z. Huang, Wang, Shan, & Chen, 2015) and Wasserstein geometry (Seguy & Cuturi, 2015), have also been studied. When utilized as a separate preprocessing step, dimensionality reduction UML solutions are commonly designed without considering the ensuing clustering algorithm and therefore cannot be fine-tuned accordingly. AML (J. Ye, Zhao, & Liu, 2007) takes a different approach, performing clustering and distance metric learning simultaneously. The joint learning under AML is formulated as a trace maximization problem, and numerically solved through an EM-like iterative procedure, where each iteration consists of a data projection step, followed by a clustering step via kernel k-means. The projection is parameterized by an orthogonal, dimension-reducing matrix. A kernelized extension of AML was proposed in (J. Chen, Zhao, Ye, & Liu, 2007). As the projection models are built on linear transformations, their capabilities to deal with complex nonlinear structures are limited.

With the remarkable representation learning capability brought by deep learning (DL), it has been a hot topic to develop DML solutions under the neural network framework. Within the supervised DML category, convolutional neural networks (CNNs) have been utilized to replace the “hand-craft” feature engineering step, and different architectures have been explored to learn CNN features and nonlinear transformations simultaneously.
(Hu, Lu, & Tan, 2014, 2015; Sun, Chen, Wang, & Tang, 2014; Yi, Lei, Liao, & Li, 2014; Hoffer & Ailon, 2015; Han, Leung, Jia, Sukthankar, & Berg, 2015). For unsupervised cases, the distinction between metric learning and feature/deep learning has become rather vague nowadays. Nevertheless, representations learned from DL models, such as auto-encoders, can be combined with clustering to improve the performance of the latter. Such integrations have been explored mainly with two strategies. The first approach learns latent features first, and then feed them into the clustering solution (J. Xu et al., 2015; F. Tian, Gao, Cui, Chen, & Liu, 2014). The second approach embeds an existing clustering method into DL structures, allowing features and clustering results to be updated jointly (Song, Liu, Huang, Wang, & Tan, 2013; Xie, Girshick, & Farhadi, 2016; K. Tian, Zhou, & Guan, 2017). Although many of the deep DML models produce state-of-the-art results, they require a large amount of training data, as a prerequisite, to perform effectively. As no constraints are generally imposed on the underlying nonlinear transformations, deep DML solutions tend to suffer from the overfitting problem, if without sufficient training samples.

1.3 Contributions

In light of the limitations and drawbacks described above, we exploit the power of geometric space transformations to address the gap between model assumptions and actual data distributions. More specifically, we apply a deformation model called Coherent Point Drifting (CPD) (Myronenko & Song, 2010) to make the transformed data points well conform to the underlying SSL and USL assumptions.

To improve the performance semi-supervised metric learning, we choose Laplacian SVM (LapSVM), a classic graph-based SSL model, as the host solution to develop and demonstrate the effectiveness of our approach. Tailored to semi-supervised learning problems, we assign labeled points with larger influence ranges than the unlabeled. The choice of CPD is with two considerations: 1) its remarkable capability in generating high-
order yet smooth deformations; and 2) the available mechanism within CPD for assigning different levels of smoothness to data points. To the best of our knowledge, this is the first attempt of utilizing globally smooth, nonlinear, dense spatial transformation models in semi-supervised learning. It should also be noted that, while the work present in this paper is based on LapSVM, our model has a broad applicability and can be readily extended to many other SSL solutions.

To boost the metric learning solutions under unsupervised setting, we propose a new nonlinear UML framework in this dissertation. Our solution integrates a nonlinear feature transformation and manifold embedding together to improve the data separability for $k$-means clustering. Our model can be regarded as a fully nonlinear generalization of AML, in which the transformation model is upgraded to a geometric model CPD. Data points are driven by CPD to reach a higher level of linear separability, which will be subsequently picked up by the manifold embedding component to generate well-separable sample projections. In the end, $k$-means is applied on the transformed, dimension-reduced embeddings to produce label predictions. The main contributions points of this work include the following: 1) our proposed fully nonlinear UML solution enhances data separability through the combination of CPD-driven deformation and spectral embeddings; 2) like our semi-supervised solution, this is the first work that utilizes dense, spatial varying deformations in unsupervised metric learning; 3) the CPD optimization has a closed-form solution, therefore can be efficiently computed; and 4) our model outperforms state-of-the-art UML methods on six benchmark databases, indicating promising performance in many real-world applications.

Application-wise, our proposed semi-supervised metric learning approach is applied to make the early prediction for Alzheimer’s disease (AD). The unsupervised metric learning solution is utilized to improve the patient clustering by classifying AD/MCI/NC subjects. In addition, a novel shape analysis framework has also been developed to extract
more discriminative features from hippocampal shapes to improve the performance of AD/MCI/NC classification. The developed framework utilizes harmonic mapping to align the hippocampal surfaces from longitudinal scans, which enables the accurate analysis of hippocampal atrophy on specific locations.

1.4 Overview of the Dissertation

This dissertation is organized as follows. Part I gives the introduction for the background knowledge, as well as a review of the works relevant to this dissertation:

- Chapter 1 introduces the background and existing problems for the metric learning solutions under semi-supervised and unsupervised settings.
- Chapter 2 describes the fundamental mathematics for geometric transformation, CPD model and distance metric learning.
- Chapter 3 gives a survey for the metric learning solutions under both SSL and USL settings.

Part II describes the details of our proposed solutions for semi-supervised and unsupervised metric learning:

- In Chapter 4, a nonlinear metric learning framework is proposed to boost the performance of SSL algorithms. Constructed on top of Laplacian SVM, the proposed method learns a smooth nonlinear feature space transformation that makes the input data points more linearly separable. Experiments performed on both synthetic and UCI datasets show the effectiveness of our CPD-LapSVM over the state-of-the-art metric learning solutions in SSL.
- In Chapter 5, we propose a nonlinear unsupervised metric learning framework to improve clustering algorithms. Under our framework, nonlinear distance metric
learning and manifold embedding are integrated and conducted simultaneously to increase the natural separations among data samples. Experiments on the synthetic datasets, as well as various real-world datasets, highlight the performance of our framework against existing unsupervised metric learning solutions.

Part III applies our proposed semi-supervised and unsupervised models on the field of neuroimaging:

- In Chapter 6, our proposed semi-supervised and unsupervised metric learning approaches are applied on ADNI datasets to demonstrate their applicability in AD early detection and clustering problems. In addition, the proposed novel feature extraction framework is also validated using ADNI datasets.

- Chapter 7 concludes the dissertation.
2 PRELIMINARIES

2.1 Geometric Transformation

As introduced in Section 1, we aim to design metric learning solutions through feature transformations that are regulated with certain geometric model(s). Generally speaking, a geometric transformation is a process which changes positions or shapes of geometric objects. In mathematics, it is considered as a mapping from the original object \( g(x, y) \) to the transformed object \( f(u, v) \), which can be presented as:

\[
f(u, v) = g(x(u, v), y(u, v))
\]

Geometric transformations can be categorized into linear and nonlinear transformations. Linear transformations are global transformations which transform domains homogeneously. Typical linear transformations include translation, scaling, rotation, and shearing. The illustrations of these transformations are given in Fig. 2.1. Since all the points are transformed in the same manner, linear transformations have limited power to deform feature spaces.

On the contrary, nonlinear transformations are performed locally. As they can deform feature spaces inhomogeneously, nonlinear transformations have the flexibility to transform targets to expected positions and shapes. In practice, in nonlinear transformation based applications (e.g. point matching), regularizations are commonly required to ensure the well-posedness of the problems, as well as to generate simple and smooth deformation fields. For example, the CPD model used in our work is regularized by the norm operator of the functional space. The detailed information will be introduced in the next section.
2.2 Transformation Regularization and CPD Transformation Model

In our framework, the CPD model is chosen to perform the transformation. Fig. 2.2 gives an example of CPD transformation which transforms the blue points towards the red points. Next, we will give the detailed derivation for the CPD formulation.
In transformation based applications (e.g. point matching), regularization is commonly required to ensure the well-posedness of the problem, as well as to generate a simple and smooth deformation field.

Let $v(x) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ denote a displacement function that moves dataset $x$ towards $y$. Estimation of an optimal $v(\cdot)$ has been commonly formulated under Tikhonov regularization framework,

$$
\mathcal{R}[v] = \mathcal{R}_{emp}[v] + \lambda \mathcal{R}_{reg}[v] \tag{2.1}
$$

where $N$ is the instance number of the dataset. $\| \cdot \|$ is the norm operator. $\lambda$ is a regularization parameter that controls the trade-off between the data term and the regularization term. $D$ is a linear differential operator.

The regularization functional on $v$ is to ensure certain extent of smoothness, which can be essentially determined by the norm operator of the functional space. Different norm operators would lead to different smoothness functionals. A general norm of $v$ in the Hilbert space $\mathbb{H}^m$ is given in (Z. Chen & Haykin, 2002):

$$
\|v\|_{\mathbb{H}^m}^2 = \int_{\mathbb{R}^d} \sum_{k=0}^{m} \|\partial_x^k v\|^2 dx \tag{2.2}
$$

Alternatively, the norm in the Reproducing Kernel Hilbert Space (RKHS) can also be defined as:

$$
\|v\|_{\mathbb{H}^m}^2 = \|v\|_\mathcal{K} = \int_{\mathbb{R}^d} \frac{|\hat{v}(s)|^2}{\mathcal{K}(s)} ds \tag{2.3}
$$

where $\mathcal{K}$ is the kernel function in the RKHS, whose Fourier transform is represented as $\hat{\mathcal{K}}$. $\hat{v}$ denotes $v$’s Fourier transform. $s$ is a vector variable in the frequency domain.
According to (Z. Chen & Haykin, 2002), the optimal solution that minimizes Eqn. (2.1) is given by a linear combination of particular kernel functions on each instance \( x \), plus the term \( \phi(z) \) in the null space of \( D \):

\[
v(z) = \sum_{i=1}^{N} \psi_i \mathcal{K}_{DD}(z, x_i) + \phi(z)
\]

where \( v(z) \) stands for the displacement of an arbitrary position \( z \) in the same vector space. The kernel function \( \mathcal{K}_{DD} \) is a Green’s function of the self-adjoint operator \( \hat{D}D \), where \( \hat{D} \) is the adjoint operator of \( D \). \( \psi_i \) (size \( d \times 1 \)) is the weight of the kernel functions.

In this work, the coherent point drifting (CPD) model (Myronenko & Song, 2010) is chosen to transform a feature space smoothly and nonlinearly. It was originally used as a registration solution to match point sets. CPD model is formulated using a particular regularization term where the kernel function \( K \) is chosen as a Gaussian low-pass filter \( G \) in Eqn. (2.3):

\[
R_{reg}[v] = \int_{\mathbb{R}^d} \frac{|\tilde{v}(s)|^2}{G(s)} ds
\]

This Gaussian choice is motivated by several considerations. First, the corresponding Green’s function (as in Eqn. 2.4) of this regularization term is also a Gaussian. Second, the size of the Gaussian function determines the frequency range filtered by the function, which can further influence the spatial smoothness. Therefore, the spatial smoothness at each data point can be controlled by applying appropriate Gaussian function sizes. As the null space term of the Gaussian kernel becomes 0, the optimal solution \( v(\cdot) \) for CPD is given by:

\[
v(z) = \sum_{i=1}^{N} \psi_i G(z, x_i) = \sum_{i=1}^{N} \psi_i G(\|z - x_i\|) = \sum_{i=1}^{N} \psi_i e^{-\frac{(z - x_i)^2}{2\sigma^2}}
\]
where the Green’s kernel function becomes a Gaussian $G(\cdot, \cdot)$. $\sigma$ is the width of the Gaussian filter, and it controls the overall level of smoothness in the deformation field.

The matrix format of $v(z)$ can be written as:

$$v(z) = \Psi \begin{pmatrix} G(z, x_1) \\ \vdots \\ G(z, x_n) \end{pmatrix} = \Psi \tilde{G}(z, x),$$

(2.7)

where $\Psi$ (size $d \times n$) is the weight matrix for the Gaussian kernel functions. $n$ is the number of instances in $x$.

In the original point-matching CPD algorithm, a uniform $\sigma$ was used in deformation field estimation. However, different $\sigma_i$ can be used to specify the stiffness of the deformation field around $x_i$. A large $\sigma$ corresponds to a smoother neighborhood and therefore a uniform deformation across a larger range. In other words, $\sigma$ defines the influence power of each data point. In this work, we take advantage of this flexibility equipped with the CPD model, and assign different influence ranges to labeled and unlabeled data points. More specifically, a larger $\sigma$ is assigned to labeled points allowing them to exert more significant influence in producing desired classification output. More details will be presented later in section 4.2.

2.3 Metrics and Metric Learning

2.3.1 Metrics

Metric is a measure of distance. A “good” metric is critical in many machine learning algorithms since the metric determines the distance and similarity between different samples. For example, the $k$ Nearest Neighbor (kNN) classifier (Dudani, 1976) assigns the label for a coming object purely based on the distance between it and the labeled samples. Some commonly used metrics are introduced below:
**Euclidean Metric** The Euclidean metric (Breu, Gil, Kirkpatrick, & Werman, 1995) is the most commonly used metric. It is the straight distance between samples under the Euclidean space. Given two points $\mathbf{p}_1$ and $\mathbf{p}_2$ under the Euclidean $n$-dimensional space, the distance between the two points is given as:

$$D_E(\mathbf{p}_1, \mathbf{p}_2) = \|\mathbf{p}_1 - \mathbf{p}_2\|_2 = \left( \sum_{i=1}^{n} |\mathbf{p}_{1i} - \mathbf{p}_{2i}|^2 \right)^{1/2}$$  \hfill (2.8)

The Euclidean metric is invariant to both translation and rotation.

**Discrete Metric** The Discrete metric (Thiel & Montanvert, 1992) is a simple metric defined as below:

$$D_D(\mathbf{p}_1, \mathbf{p}_2) = \begin{cases} 
1, & \text{if } \mathbf{p}_1 = \mathbf{p}_2 \\
0, & \text{if } \mathbf{p}_1 \neq \mathbf{p}_2 
\end{cases}$$  \hfill (2.9)

**Taxicab Metric** The Taxicab metric (Ozcan & Kaya, 2003) is the length sum of the projections from the straight line fragments between two points:

$$D_T(\mathbf{p}_1, \mathbf{p}_2) = \|\mathbf{p}_1 - \mathbf{p}_2\|_1 = \left( \sum_{i=1}^{n} |\mathbf{p}_{1i} - \mathbf{p}_{2i}|^2 \right)^{1/2}$$  \hfill (2.10)

The Taxicab Metric is invariant to translation.

**Hamming distance** The Hamming distance (Norouzi, Fleet, & Salakhutdinov, 2012) was originally used to count the number of different bits in fixed-length words. Given two words $\mathbf{w}_1$ and $\mathbf{w}_2$

$$D_H(\mathbf{w}_1, \mathbf{w}_2) = |i : \mathbf{w}_{1i} \neq \mathbf{w}_{2i}|$$  \hfill (2.11)

The Hamming distance is widely applied in the coding theory.

### 2.3.2 Distance Metric Learning (DML)

The Euclidean distance, which is introduced above, assigns an equal weight to each feature component. Metric learning is essentially aiming at learning an improved metric,
as the replacement of the Euclidean, from the training vectors \( \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathcal{X} \). As a generalization of the Euclidean distance, the Mahalanobis distance has become one of the most widely studied metrics in supervised DML research (Goldberger et al., 2004; Roweis & Saul, 2000; Globerson & Roweis, 2005; Weinberger & Saul, 2009; Hoi et al., 2006; Hong et al., 2011). The Mahalanobis distance is defined as: 

\[
D_M(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\mathbf{x}_i - \mathbf{x}_j}^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j),
\]

where \( \mathbf{M} \) is a positive semi-definite (PSD) matrix (denoted as \( \mathbf{M} \succeq 0 \)). Since \( \mathbf{M} \) can always be decomposed as \( \mathbf{M} = \mathbf{L}^T \mathbf{L} \), the Mahalanobis distance \( D_M \) can be rewritten as:

\[
D_M(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\mathbf{x}_i - \mathbf{x}_j}^T \mathbf{L}^T \mathbf{L} (\mathbf{x}_i - \mathbf{x}_j)
= \sqrt{\mathbf{L} \mathbf{x}_i - \mathbf{L} \mathbf{x}_j}^T (\mathbf{L} \mathbf{x}_i - \mathbf{L} \mathbf{x}_j).
\]

Distance metric learning for Mahalanobis distance is to learn an appropriate matrix \( \mathbf{M} \) to better present the given data samples.

### 2.3.3 Relationship between Geometric Transformation and Metric Learning

Eqn. (2.12) indicates that the Mahalanobis metric with matrix \( \mathbf{M} \) is essentially the Euclidean distance after a linear transformation \( f : \mathbf{x} \rightarrow \mathbf{L} \mathbf{x} \). To learn a Mahalanobis metric therefore is the same as to learn a linear transformation that ensures the resulted Euclidean distances would very well conform to the supervisory information. This observation also applies to nonlinear metric learning, and therefore the focus to learn a metric can be shifted to learn a geometric mapping.

Fig. 2.3 gives an example for a linear metric learning according to the color information. From the example, it is obvious that stretching the space along the x-axis will make the samples with the same color closer. Then, the distance-based algorithms, such as kNN or \( k \)-means, can make the correct decision for the sample labels. However, if
the samples are categorized according to their shapes, the space should be stretched along the y-axis as shown in Fig. 2.4 instead.

From the two figures, we can also observe that the linear metrics are essentially linear geometric transformations with different transformation matrices $L$ in the original space.
An example of nonlinear metric learning is illustrated in Fig. 2.5. The samples of different categories are pushed away from their opposite class in a nonlinear way.

Figure 2.5: Illustration for a nonlinear metric learning.
3 Literature Review

During the past years, efforts have been put forward for the metric learning solutions under both semi-supervised and unsupervised settings. In this section, we will give a brief description of the semi-supervised and unsupervised metric learning solutions.

3.1 Semi-supervised Metric Learning

**Constrained Clustering** The constrained clustering algorithms focus on integrating pairwise similarity (must-link) and/or dissimilarity (cannot-link) constraints to improve the data grouping. The first constrained clustering solution (Xing et al., 2003) learns a distance metric with the form below:

\[ d(x, y) = \sqrt{(x - y)^T A (x - y)} \]  

(3.1)

where \( A \), required to be a positive semi-definite matrix, is the parameter to be learned.

Given a set of point pairs with similar or dissimilar information, the solution formulate the metric learning problem into an optimization problem below:

\[
\min_A \sum_{(x_i, x_j) \in S} ||x_i - x_j||_A^2 \\
\text{s.t.} \sum_{(x_i, x_j) \in D} ||x_i - x_j||_A \geq 1, \quad A \geq 0
\]

(3.2)

Where \( S \) stands for the similar point pairs. \( D \) represents the dissimilar point pairs. The constrained clustering algorithms try to minimize the distance between the similar instances, with the constraint that \( A \) does not collapse the given samples into a point.

**OLapSVM** OLapSVM (Y. Gu & Feng, 2013) is a metric learning model designed based on LapSVM. It aims to optimize graph Laplacians and learns task-specific similarity metrics from the labeled samples. With a transformation matrix \( L \) of size \( m \times m \), where \( m \)
is the dimension of the data, the edge weight between two data samples \( x_i \) and \( x_j \) is defined as:

\[
w_{ij}(L) = \exp(-\|L(x_i - x_j)\|^2)
\]

The optimal matrix \( L^* \) in OLapSVM is determined by minimizing the following objective function:

\[
Q(L) = \sum_{(i,j) \in F} w_{ij}(L) - \sum_{(i,j) \in S} w_{ij}(L) - \lambda \sum_{i=1}^{l+u} \sum_{j \in N_i} w_{ij}(L)
\]

where \( S \) and \( F \) are equivalence and non-equivalence constraints defined by labels. \( N_i \) specifies the \( k \) nearest neighbors of \( x_i \) in the Euclidean space. \( l \) and \( u \) are the numbers of labeled and unlabeled data, respectively, and \( \lambda \) is a weighting factor.

**LRML** The Laplacian regularized metric learning (LRML) (Hoi et al., 2010) learns a metric with the same form as in Eqn. (3.1). It formulates the Laplacian regularized metric learning into a minimization problem as follows:

\[
\begin{align*}
\min_{A \geq 0} & \quad \text{trace}(XLX^T A) + \lambda_S \sum_{(x_i,x_j) \in S} \|x_i - x_j\|^2_A - \lambda_D \sum_{(x_i,x_j) \in D} \|x_i - x_j\|^2_A \\
\text{s.t.} & \quad \log(\det(A)) \geq 0
\end{align*}
\]

(3.3)

where \( X \) stands for the dataset. \( L \) is the Laplacian matrix, which has the form below:

\[
L = M - W
\]

\[
W_{ij} = \begin{cases} 
1, & \text{if } x_i \in N(x_j) \text{ or } x_j \in N(x_i) \\
0, & \text{otherwise}
\end{cases}
\]

(3.4)

where \( M \) is a diagonal matrix with the element \( M_{ii} = \sum_j W_{ij} \). \( N \) denotes the neighborhood of a sample \( x_i \).

The \( S \) and \( D \) in Eqn. (3.4) stand for the similar and dissimilar point pairs separately. LRML learns a matrix \( A \) by minimizing the Laplacian energy and the distance between the
similar instances together, with the constraint that \( A \) does not collapse the given samples into a point.

**SSM-DML**  The semi-supervised multiview distance metric learning (SSM-DML) (Yu et al., 2012) is a multiview distance metric learning strategy. It also utilizes the graph Laplacian as the regularizer in its optimization function. Multiview distance metrics are learned simultaneously from multiple feature sets using the available label information. The metric learning problem is formulated into the following optimization problem:

\[
\min_{A^{(k)}, F} \sum_{i,j=1}^{N} W_{ij}^{(k)} \frac{\| F_i - F_j \|}{\sqrt{D_{ii}^{(k)}}} + \lambda \sum_{i=1}^{N} \| F_i - Y_i \|^2
\]  (3.5)

where

\[
W_{ij}^{(k)} = \exp\left(-\left(\mathbf{x}_i^{(k)} - \mathbf{x}_j^{(k)}\right)^T A^{(k)} \left(\mathbf{x}_j^{(k)} - \mathbf{x}_i^{(k)}\right)\right) \tag{3.6}
\]

\( A^{(k)} \) is the linear transformation for the \( k \)th feature space. \( \mathbf{x}_i^{(k)} \) represents the \( k \)th feature value of the sample \( \mathbf{x}_i \). \( D^{(k)} \) stands for a diagonal matrix with the element \( D_{ii}^{(k)} = \sum_j W_{ij}^{(k)} \). \( Y_i \) is the available label information. \( F_{i,j} \) stands for the confidence of \( \mathbf{x}_i^{(k)} \) with the label \( Y_j \).

SSM-DML obtains \( A^{(k)} \) and \( F \) using an alternating optimization strategy.

**SERAPH**  The Semi-supervised metric learning paradigm with hyper-sparsity (SERAPH) (Niu et al., 2014) utilizes a posterior sparsity assumption (Ganchev, Taskar, Pereira, & Gama, 2009). The assumption claims that the sparsity the posterior distributions over the unobserved weak labels has a natural discriminative ability for given data samples. The proposed SERAPH integrates the supervised and unsupervised parts into a unified framework. The optimization problem in the model SERAPH is as below:

\[
\min_{A,k} L(A, k) = \sum_{S \cup U} \ln p_{i,j}^A(y_{i,j}) - \frac{\gamma}{2} k^2 + \mu \sum_{U} \sum_{y} p_{i,j}^A(y) \ln p_{i,j}^A(y) - \lambda \text{trace}(A) \tag{3.7}
\]
where $S$ and $D$ stand for the similar and dissimilar point pairs separately. $\gamma$, $\mu$ and $\lambda$ are regularization parameters with the values bigger than zero. $\mathcal{U}$ represents the sample set with label information. $y \in \{1, -1\}$ is the value for labels. $p_{i,j}^A(y_{i,j})$ is the probability of labeling the $i$th sample with label $y_j$ in the transformed feature space after the learned metric $A$. SERAPH tries to learn a linear metric to combine the supervised and unsupervised information from the information-theoretic perspective to promote the discriminative ability of the labels in the transformed spaces.

### 3.2 Unsupervised Metric Learning

**PCA** The principal component analysis (PCA) (Wold, Esbensen, & Geladi, 1987) is a linear dimensionality reduction technique. Unsupervised metric learning is commonly achieved via dimensionality reduction. PCA performs orthogonal transformations on the original data instances to project them into a lower dimensional space whose coordinates are linearly uncorrelated. The optimization problem of PCA is:

$$
\min_{\omega} \quad \omega^T S \omega \\
\text{subject to:} \quad \omega^T \omega = 1
$$

where

$$S = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T$$

and

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

where $N$ is the number of samples, and $\omega$ is the transformation vector to be optimized.

**LLE** The locally linear embedding (LLE) (Roweis & Saul, 2000) is a nonlinear manifold learning method. LLE learns the global structure of a nonlinear manifold by preserving the neighborhood between samples. An example of LLE is illustrated in Fig. 3.1. The LLE
method constructs a neighborhood preserving mapping by minimizing the cost function below:

$$\min_Y \sum_i |y_i - \sum_j W_{i,j} y_j|^2$$  \hspace{1cm} (3.11)

where $Y$ stands for the vectors that minimize the error function above with the given weight $W$. $W$ is computed by minimizing the cost function below to best reconstruct each data sample from its neighbors.

$$\min_W \sum_i |x_i - \sum_j W_{i,j} x_j|^2$$  \hspace{1cm} (3.12)

Figure 3.1: An example of nonlinear dimensionality reduction. The 2-dimensional representation in (C) shows the mapping in LLE for the three-dimensional data (B) sampled from the manifold in (A). The mapping preserves the neighborhood relationship

The steps in the LLE are as follows:

1. Find the neighborhood set for each data sample $x_i$. 

(Saul & Roweis, 2000)
2. Compute the weights $W$ minimizing the cost function in Eqn. (3.12).

3. Compute the transformation vectors minimizing the cost function in Eqn. (3.11).

**FME**  
The Flexible Manifold Embedding (FME) (Nie et al., 2010) relies on the learning of an optimum linear regression function to specify the target low-dimensional space. FME tries to compute the bias $b$, the projection matrix $W$, and the latent variable $f$ at the same time by minimizing the following objective function:

$$
\min_{f,W,b} \text{trace}(f^T L f) + \mu \|W\|^2 + \lambda \|X^T W + b^T - f\|^2)
$$

(3.13)

where $f$ can be seen as the latent variable, which is the representation in the lower dimensions. $\mu$ and $\lambda$ are the hyper-parameters to maintain the balance of different terms. $L$ has the same definition as in Eqn. (3.4). The procedure of FME as follows:

1. Compute the graph Laplacian matrix $L$.
2. Compute the latent variable $f$ using eigenvalue decomposition.
3. Compute the projection matrix $W$.

**AML**  
The adaptive distance metric learning (AML) (J. Ye et al., 2007) formulates the joint clustering and metric learning problem into a trace maximization problem, and solves them iteratively. AML projects the given data samples onto a lower dimensional manifold maximizing the data separability. The problem is formulated as follows:

$$
\max_{G:G^T=1} \frac{1}{N} \text{trace}(L^T \chi^T G(G^T \sum G)^{-1} G^T \chi L)
$$

(3.14)

where $L$ is the cluster indicator matrix, $G$ is the transformation matrix, $\chi$ is the datasets. $\sum$ is the class covariance matrix of the original data with the formulation below:

$$
\sum = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T + \lambda I_m
$$

$$
\mu = \frac{1}{N} \sum_{i=1}^{N} x_i
$$

(3.15)
AML solves the trace maximization problem in Eqn. (3.14) via an iterative algorithm under an EM framework.

**CPCM** The method of clustering predictions of cluster membership (CPCM) (Gupta, Foster, & Ungar, 2008) learns the metric by minimizing the blur ratio as blow:

\[
\min_{\Lambda, \mathcal{C}} \quad \text{BlurRatio} = \frac{SSC}{SST}
\]

\[
SSC = \sum_{k=1}^{K} \sum_{c(x_i) = C_k} (x_i - \mu_k)\Lambda(x_i - \mu_k)^T
\]

\[
SST = \sum_{i=1}^{N} (x_i - \mu)\Lambda(x_i - \mu)^T
\]

where \(SSC\) is the within-cluster variance and the \(SST\) is the total variance of all the clusters. \(\mu\) is the mean of all the samples and \(\mu_k\) is the center of \(k\)th cluster. The CPCM method optimizes the metric \(\Lambda\) and the cluster memberships in an iterative procedure.
4 NONLINEAR METRIC LEARNING FOR SEMI-SUPERVISED LEARNING VIA CPD

In this chapter, we propose a novel nonlinear metric learning framework for semi-supervised Learning via the transformation model CPD. The base classification algorithm applied in the framework is Laplacian SVM. Our work has been published in (P. Zhang et al., 2016), and received the best paper award.

4.1 Laplacian SVM

Laplacian SVM (LapSVM) is a popular graph-based SSL solution. Formulated based on the standard SVM models, LapSVM solves the classification problem by employing two regularization terms: one for SVM maximal margin classification, and the other for label smoothness across the graph - neighboring nodes should have identical or similar labels. LapSVM seeks to find the best class separation (maximal margin) while taking account of the graph structure that reflects the intrinsic similarities among data points.

Let \( \mathcal{X} = \{ x_i | x_i \in \mathbb{R}^d, i = 1, \cdots, l + u \} \) denote the whole training dataset. \( \{ x_i, y_i \}_{i=1}^l \) are labeled data with labels \( y_i \in \{-1, +1\} \), and unlabeled data instances are the remaining \( \{ x_i \}_{i=l+1}^{l+u} \). LapSVM learns a classifier \( f(x) \) from the training set \( \mathcal{X} \), by solving the following optimization problem (Belkin & Niyogi, 2004):

\[
\min_{f \in \mathcal{H}_K} J = \frac{1}{l} \sum_{i=1}^l \xi_i + \gamma_A \|f\|_K^2 + \gamma_I \sum_{j,k=1}^{l+u} D_{jk}(f(x_j) - f(x_k))^2
\]

s.t. \( y_i f(x_i) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i = 1 \ldots l; \)

where \( D_{ij} \) is the weight of the edge connecting \( x_i \) and \( x_j \) in the data adjacency graph. \( \xi_i \) is the slack variable from SVM. \( \gamma_A \) and \( \gamma_I \) are the hyper-parameters of the regularization terms. \( \|f\|_K^2 \) is the squared norm of \( f \) in \( \mathcal{H}_K \), the RKHS.
4.2 CPD based Metric Learning with LapSVM (CPD-LapSVM)

Similarly as in JSL (Irina et al., 2015) and SVML (Z. Xu et al., 2012), our framework jointly learns a spatial transformation and a classifier at the same time. The main distinction is that the metrics learned in our model are expressed with nonlinear global deformations, regulated via the CPD model. For each data point $x_i$, let $x_i^0$ be its initial coordinate. Through the displacement $v(x)$ in Eqn. (2.7), $x_i$ will be moved from $x_i^0$ to $x_i^1$:

$$x_i^1 = x_i^0 + v(x_i^0) = x_i^0 + \Psi \tilde{G}(x_i^0, x^0)$$

(4.2)

where $x^0$ is the initial dataset. For each test data point $z$, $\tilde{G}(z, x^0)$ will be calculated based on Eqn. 5.1. The weight matrix $\Psi$ captures the data abstraction from the training samples, and needs to be estimated in the training stage.

As we take LapSVM as the host algorithm, the classifier to be learned is a LapSVM classifier $f(x_i) = w^T x_i^1 + b$ under the CPD-transformed space. The kernelized version of both LapSVM and our CPD-LapSVM can lead to nonlinear decision boundaries in the input space, though they are still hyperplanes under the feature space. The CPD transformation can be applied in both the input space and the feature space. The latter is the kernel version of our CPD-LapSVM model, which will be presented later in section 5.1.3.

Under the input space, our linear version CPD-LapSVM (note: “linear” refer to decision boundary; CPD transformation is nonlinear) is built upon the LapSVM objective function in Eqn. (4.1). First, we use a quadratically smoothed hinge loss function as the slack variable item:

$$\xi_i = \max[0, 1 - y_i f(x_i^1)]^2$$

(4.3)

The choice of quadratic form is motivated by the mathematical convenience in computing the derivatives w.r.t. $f(\cdot)$ and $\Psi$. Second, the squared Frobenius norm of
\(\Psi\), denoted as \(\|\Psi\|_F^2\), is added to impose a smoothness constraint onto the estimated transformations. As a return, the chance of overfitting would be reduced.

With these two added terms, our linear CPD-LapSVM learns a nonlinear transformation and a linear classifier simultaneously through the minimization of the updated objective function:

\[
\min_{\Psi, w, b} J = \frac{1}{l} \sum_{i=1}^{l} \max\{0, 1 - y_i(w^T x_1^i + b)\}^2 + \gamma_A \|w\|_K^2 + \gamma_L \|\Psi\|_F^2
+ \gamma_I l + \sum_{j,k=1}^{l+u} D_{jk}(w^T x_1^j - w^T x_1^k)^2
\]

s.t. \(y_i(w^T x_1^i + b) \geq 1 - \max\{0, 1 - y_i(w^T x_1^i + b)\}\) \(\forall i = 1 \ldots l;\)

where \(\gamma_A, \gamma_I\) and \(\gamma_L\) are trade-off hyper-parameters.

**Optimization strategy** The objective function of CPD-LapSVM is parameterized with a transformation matrix \(\Psi\) and classifier parameters \(\{w, b\}\). To search for an optimal solution, we adopt an EM-like iterative minimization strategy that updates \(\Psi\) and \(\{w, b\}\) alternatingly. The matrix \(\Psi\) is initialized with all 0 entries, so are \(w\) and \(b\).

With \(\Psi\) fixed, Eqn. (4.4) reduces to the original LapSVM objective, performing on the transformed dataset \(x^1\). It can be easily optimized using the LapSVM solver in (Belkin et al., 2006) (a standard SVM solver with the quadratic forms). With \(\{w, b\}\) fixed, the classification decision boundary becomes explicit. We can then further update the deformation to make the transformed dataset better conform to the membership assigned via the decision boundary. Now the objective function is only on \(\Psi\) (note: \(x_1^i\) is also a function of \(\Psi\), as in Eqn. (4.2)), reformulated as follow,

\[
\min_{\Psi} J = \frac{1}{l} \sum_{i=1}^{l} \max\{0, 1 - y_i(w^T x_1^i + b)\}^2
+ \gamma_L \|\Psi\|_F^2 + \gamma_I l + \sum_{j,k=1}^{l+u} D_{jk}(w^T x_1^j - w^T x_1^k)^2
\]
While CPD is capable of producing rather sophisticated deformations, the smoothness term $\|\Psi\|^2_F$ in this objective greatly regularizes the deformations that can be generated. In this paper, we used full graphs as the adjacency graphs, where each pair of points are connected. An example of a full graph is illustrated in Fig. 4.1. The edge weight $D_{jk}$ between $x_j$ and $x_k$ is assigned as $D_{jk} = \exp(-\frac{1}{2\alpha^2}((\|x_j^1 - x_k^1\|^2)))$, where $\alpha$ is the parameter in heat kernel function. As the objective function Eqn. (4.5) is differentiable w.r.t. $\Psi$, a gradient based constrained optimization solver $^1$ is used to seek its local minima, as well as the optimal solutions of $\Psi$. The gradient $\frac{\partial J}{\partial \Psi}$ is given as follows:

$$\frac{\partial J}{\partial \Psi} = -\frac{2}{l} \sum_{i=1}^{l} \max[0, 1 - y_i(w^T x_i^1 + b)]y_i w^T \mathbf{G}^T(x_i^0, x^0) + 2\gamma L \Psi$$

$$+ 2\gamma_1 \sum_{j,k=1}^{l+l} D_{jk}(1 - \frac{1}{2\alpha^2}||x_j^1 - x_k^1||^2)(x_j^1 - x_k^1)$$

$$- \mathbf{G}^T(x_j^0, x^0)w^T w$$

$$\frac{\partial J}{\partial \Psi} = -\frac{2}{l} \sum_{i=1}^{l} \max[0, 1 - y_i(w^T x_i^1 + b)]y_i w^T \mathbf{G}^T(x_i^0, x^0) + 2\gamma L \Psi$$

$$+ 2\gamma_1 \sum_{j,k=1}^{l+l} D_{jk}(1 - \frac{1}{2\alpha^2}||x_j^1 - x_k^1||^2)(x_j^1 - x_k^1)$$

$$- \mathbf{G}^T(x_j^0, x^0)w^T w$$

Figure 4.1: Illustration for a full graph.

$^1$ Matlab “fmincon” optimizer.
The above derivations are based on a particular classifier, LapSVM. Integrating CPD with other SVM based models, e.g., TSVM, would be rather straightforward. In general, for SSL solutions formulated under an optimization framework, we can commonly utilize CPD to parameterize data points at new locations, and use the two-stage EM procedure to optimize the transformation and the classifier in an alternating fashion.

4.2.1 SSL Mechanism in CPD: Assign Larger Influence to Labeled Samples

In the CPD registration algorithm (Myronenko & Song, 2010), all points were treated equally and assigned the same Gaussian width. Such uniform assignment, if adopted in SSL, would fail to stress the importance of the membership certainty residing in labeled points. To exploit such certainty, we assign wider Gaussian widths to labeled data to amplify their control ranges. In this way, more samples will move coherently along with labeled samples to make the overall data set more separable. This concept is illustrated in Fig. 4.2. Black dots represent unlabeled data, and red and blue dots are labeled instances with opposite labels. Labeled samples have larger influence ranges (radii of the circles) than the unlabeled.

$$G(x_i, x_j) = \begin{cases} 
  e^{-\frac{(x_i-x_j)^2}{2\sigma_u^2}} & \forall x_i, x_j \in \text{unlabeled data}; \\
  e^{-\frac{(x_i-x_j)^2}{2\sigma_l^2}} & \text{otherwise}.
\end{cases}$$

Mapping this concept to implementation, we assign different $\sigma$s to the Gaussian function $G(x_i, x_j)$ in Eqn. (5.1). If $x_i$ and $x_j$ are both unlabeled samples, $G(x_i, x_j)$ is computed with a smaller width $\sigma_u$. Otherwise, a wider range $\sigma_l$ is used, as shown in Eqn. 4.7. In this way, the constructed Green’s kernel matrix in Eqn. (2.4) would still be maintained as symmetric.

The widths $\sigma_u$ and $\sigma_l$ are determined as follows. Let $d_{\text{min}}$ denote the distance from an instance $x_i$ to its nearest neighbor. $\sigma_u$ is computed as the mean of all $d_{\text{min}}$ values across
Figure 4.2: Illustration of the concept of assigning different $\sigma$s to labeled and unlabeled points. Refer to text for details.

the entire training set. $\sigma_l$ is calculated as the mean of $d_{\text{min}}$ values for only the labeled data. Since the labeled data are more sparsely positioned compared with the whole training set, the width $\sigma_l$ is always larger than $\sigma_u$. In addition, $\sigma_l$ increases along with the decrease of the number of labeled samples.

4.2.2 Kernelization of CPD-LapSVM

The CPD-LapSVM model we introduced as far works under the input space. Many machine learning algorithms, including various DML solutions, can be kernelized, and the idea is to embed the input features into a higher dimensional space, with the hope that the transformed data would have certain desired properties under the new domain.

SVMs and LapSVMs can be naturally kernelized as their dual formulations and solutions can both be expressed with inner products. Our CPD-LapSVM, with $\Psi$ as a parameter matrix, cannot be directly kernelized the same way – computation of $\Psi$ requires
the location information of the transformed samples. Therefore, we adopt a kernel principal component analysis (KPCA) based framework proposed in (C. Zhang, Nie, & Xiang, 2010; Chatpatanasiri, Korsrilabutr, Tangchanachaianan, & Kijsirikul, 2010). Given a chosen kernel function, we first project the input samples into a kernel feature space introduced by KPCA. We then train the CPD-LapSVM model under the kernel space to learn both the transformation and classifier. Proven to be equivalent to the traditional kernel trick, this KPCA based framework requires no derivation of any new mathematical formula. If a low-rank KPCA is used, this approach also provides a convenient way to accelerate a learner. For more technical details, we refer readers to (C. Zhang et al., 2010; Chatpatanasiri et al., 2010).

4.3 Derivation of CPD-based Transductive SVM (CPD-TSVM)

While the derivations presented in the above are based on LapSVM, integrating CPD with many other SSL algorithms is rather straightforward. In this section, we give the derivation of such integration for Transductive SVM (TSVM). Similar to LapSVM, TSVM also intends to estimate a membership function \( f(x) \) defined on the data graph, by solving the following optimization problem:

\[
\min_{f \in H_K} J = \frac{1}{l} \sum_{i=1}^{l} \xi_i + \gamma_A \|f\|_K^2 + \gamma_I \sum_{j=l+1}^{l+u} (1 - |f(x_j)|)_+ \\
\text{s.t. } y_i f(x_i) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i = 1 \ldots l;
\]

(4.7)

Like CPD-LapSVM, CPD-TSVM also learns a classifier \( f = w^T x + b \) while the data are deformed by a CPD transformation. Its objective is parameterized as follows:

\[
\min_{\Psi, w, b} J = \frac{1}{l} \sum_{i=1}^{l} \max[0, 1 - y_i (w^T x_i + b)]^2 + \gamma_A \|w\|_K^2 + \gamma_I \|\Psi\|_F^2 \\
+ \gamma_L \sum_{j=l+1}^{l+u} \max[0, 1 - (w^T x_j + b)]^2
\]

(4.8)
The same EM-like iterative minimization strategy can be applied to search the optimization solution of \( \{w, b\} \) and \( \Psi \) alternately.

### 4.4 Experiments on Synthetic Data

The first set of experiments are conducted on the two-moon synthetic dataset \(^2\). This dataset consists of 200 examples, equally divided into two classes (see Fig. 5.1). 10% of each class are chosen randomly as labeled samples in the following experiments. Both linear and kernel versions of our CPD-LapSVM were tested.

**Results from linear version CPD-LapSVM** This experiment is designed to show the ability of CPD-LapSVM in transforming data points for better separability under the input space. The effectiveness of assigning larger weights to labeled data for SSL is also demonstrated. The comparison is made with LapSVM, the host algorithm, to show the improvements.

Fig. 5.1 (a) and 5.1 (c) show the classification results of LapSVM and CPD-LapSVM, respectively. As a comparison, we also show in 5.1 (b) the result of a modified version of CPD-LapSVM, where equal Gaussian weights are assigned to both labeled and unlabeled samples. It is evident that, linear LapSVM cannot handle the non-separability in the data, while our CPD-LapSVM achieves a 100% accuracy by making the data points linearly separable through space transformation. Assigning equal width (Fig. 5.1.(b)) can also deform the space, but it does not work as effectively as CPD-LapSVM (Fig. 5.1.(c)). The corresponding deformation fields of Fig. 5.1 (b) and 5.1 (c) are shown in Fig. 5.1 (e) and 5.1 (f). From the comparison of the two fields, one can tell that the field of CPD-LapSVM appears smoother, labeled points are more linearly separated and the unlabeled points follow more closely with labeled samples. This can serve as a supporting evidence

\(^2\) http://manifold.cs.uchicago.edu/manifold_regularization/data.html
that assigning larger Gaussian widths to the labeled points indeed allows them to exert amplified influences of their label certainty.

**Results from kernel version CPD-LapSVM** In this experiment, the two-moon dataset is used to simulate linearly inseparable cases in the feature spaces induced by RBF kernels. Fig. 5.2 (a), 5.2 (b) and 5.2 (c) show the best classification results of LapSVM using RBF kernels with different widths. When an optimal or appropriate kernel width is in place, as in Fig. 5.2 (a), LapSVM can have the two classes well separated. However, it performs poorly when sub-optimal widths are used, as in 5.2 (b) and 5.2 (c). Finding an optimal width through cross-validation often entails a large number of width candidates and therefore many iterations. Our kernel CPD-LapSVM can greatly ease this procedure – deforming the kernel space through CPD provides a supplementary force to the RBF kernel in making the data points more linearly separable, just as it does under the input space. The effects are demonstrated in Fig. 5.2 (e) and 5.2 (f): CPD-LapSVM uses the same RBF kernels as in Fig. 5.2 (b) and 5.2 (c), but managed to obtain better classification accuracies. Due to the difficulty of visualization in high-dimensional space, the decision boundaries, which are hyperplanes under the kernel spaces, are shown under the original input space.
Figure 4.3: Left column: classification results of linear LapSVM (a), CPD-LapSVM with equal smoothness range (b), and CPD-LapSVM with larger smoothness range for labeled samples (c), respectively. Right column (d) - (f): initial deformation field, deformation field of (b) and deformation field of (c).
Figure 4.4: Left column: classification results of kernel LapSVM with RBF kernels width = 1, 8 and 32. Right column: results of kernel version CPD-LapSVM with RBF kernels width = 1, 8 and 32.
4.5 Experiments on UCI datasets

In this section, we employ the UCI machine learning repository datasets \(^3\) to evaluate our CPD-LapSVM for semi-supervised classification. Experiments are conducted to explore: 1) performance of CPD-LapSVM on datasets with a small number of labeled samples; and 2) the impact of the number of labeled samples on the classification accuracy. Seven UCI benchmark datasets were used in this study, and their basic info is summarized in Table 5.1. All datasets have been preprocessed through normalization.

Table 4.1: Seven UCI benchmark datasets used in experiments. Columns show the name, numbers of samples, attributes and classes of each dataset.

<table>
<thead>
<tr>
<th>Datasets</th>
<th># Samples</th>
<th># Attributes</th>
<th># Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balloons</td>
<td>76</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Haberman</td>
<td>306</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Liver</td>
<td>345</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Breast-cancer</td>
<td>286</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>Heart-statlog</td>
<td>270</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>Diabetes</td>
<td>768</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Sonar</td>
<td>208</td>
<td>60</td>
<td>2</td>
</tr>
</tbody>
</table>

Three semi-supervised methods, Laplacian regularized least squares (LapRLS), Laplacian SVM (LapSVM) and OLapSVM (Y. Gu & Feng, 2013), are utilized in all experiments as the competing solutions. These methods are tested with both linear and RBF kernels. Many SSL DML solutions have been proposed in recent years, under both SVM and \(k\)-NN paradigms. OLapSVM (Y. Gu & Feng, 2013) is chosen as a comparison

\(^3\) http://archive.ics.uci.edu/ml/
due to its close relevance to our solution, as it 1) also uses LapSVM as the host algorithm; 2) learns metrics to change the graph edge weights; and 3) has both linear and kernel versions.

**Semi-supervised classification** UCI datasets are all labeled. To simulate the SSL data scenario, 30% of the training data are randomly selected as labeled samples and the rest are treated as unlabeled. To better compare the classification performance, we run the experiment 50 times with different random 4-fold splits of each dataset, three for training and one for testing. The hyper-parameters $\gamma_A$, $\gamma_I$ and $\gamma_L$ are determined through cross-validation (CV) from $\{2^{-5} \sim 2^{15}\}$. $\gamma_A$ and $\gamma_I$ are the slackness tradeoff and graph regularization parameter used in all comparison models. $\gamma_L$ is the CPD regularization parameter used only in CPD-LapSVM. All the kernel methods have an additional parameter to tune: the RBF kernel width $\sigma$, which is also chosen through CV from $\{2^{-5} \sim 2^{10}\}$ in the experiments. The mean and standard deviation of each competing algorithm are calculated over the 50 runs and summarized in Table 4.2.

To better evaluate the relative performance of each algorithm, a pairwise Student’s $t$-test with a $p$-value 0.05 is conducted among the tested methods for each dataset. Then, a schema from (J. Wang et al., 2012) is applied to rank the tested algorithms. Each solution is compared with all competing methods: scores 1 if it is significantly better than one opponent in statistic; 0.5 points if there is a tie (no significant difference), and 0 if it performs worse. Table 4.2 summarizes the classification accuracies and comparison scores. Highest accuracy for each dataset has been identified in Boldface. It is evident that our CPD-LapSVM outperforms all other methods with significant margins in the statistic. It achieves the highest-ranking score in both linear and kernel groups. It is noteworthy that the linear CPD-LapSVM obtains comparable results (score 26.5) with the other competing methods using RBF kernels. Furthermore, our proposed CPD-LapSVM achieves significant improvements over the host method LapSVM.
Table 4.2: The mean and standard deviation of classification accuracy of each tested method on seven benchmark UCI datasets. The prefix $l$ denotes the linear version and $r$ denotes kernel version. Boldface indicates the highest classification accuracy for each dataset. The lower number in the parenthesis denotes the ranking score of each method on the given dataset.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Balloons</th>
<th>Haberman</th>
<th>Liver</th>
<th>Breast Cancer</th>
<th>Heart Statlog</th>
<th>Diabetes</th>
<th>Sonar</th>
<th>Total Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l$-LapRLS</td>
<td>82.67 ± 5.62 (0.5)</td>
<td>53.56 ± 7.05 (0.0)</td>
<td>62.10 ± 4.03 (0.5)</td>
<td>67.18 ± 3.79 (0.5)</td>
<td>78.87 ± 3.14 (1.0)</td>
<td>74.08 ± 2.79 (1.0)</td>
<td>69.01 ± 3.66 (0.5)</td>
<td>4</td>
</tr>
<tr>
<td>$l$-LapSVM</td>
<td>83.33 ± 5.21 (1.0)</td>
<td>56.57 ± 5.52 (1.0)</td>
<td>63.00 ± 3.64 (0.5)</td>
<td>68.53 ± 4.21 (1.0)</td>
<td>79.96 ± 2.85 (1.0)</td>
<td>74.19 ± 2.39 (1.0)</td>
<td>70.00 ± 3.48 (1.0)</td>
<td>6.5</td>
</tr>
<tr>
<td>$l$-OLapSVM</td>
<td>88.67 ± 4.98 (5.5)</td>
<td>65.15 ± 6.53 (1.0)</td>
<td>64.81 ± 3.07 (2.0)</td>
<td>69.15 ± 4.21 (2.0)</td>
<td>81.33 ± 2.59 (4.5)</td>
<td>74.97 ± 2.56 (2.0)</td>
<td>71.14 ± 3.34 (2.0)</td>
<td>20.5</td>
</tr>
<tr>
<td>$l$-CPD-LapSVM</td>
<td>86.67 ± 7.56 (3.5)</td>
<td>64.19 ± 6.31 (2.5)</td>
<td>67.81 ± 3.56 (4.0)</td>
<td>70.68 ± 3.70 (2.0)</td>
<td>82.78 ± 2.68 (4.5)</td>
<td>76.39 ± 3.45 (2.0)</td>
<td>72.07 ± 3.18 (2.0)</td>
<td>26.5</td>
</tr>
<tr>
<td>$r$-LapRLS</td>
<td>84.00 ± 6.05 (1.5)</td>
<td>71.21 ± 5.17 (5.5)</td>
<td>69.29 ± 3.37 (6.5)</td>
<td>72.76 ± 4.67 (4.5)</td>
<td>79.06 ± 3.92 (1.0)</td>
<td>76.78 ± 2.39 (6.0)</td>
<td>75.86 ± 3.62 (5.0)</td>
<td>30.5</td>
</tr>
<tr>
<td>$r$-LapSVM</td>
<td>87.33 ± 6.81 (4.0)</td>
<td>70.45 ± 5.43 (5.0)</td>
<td>67.71 ± 3.49 (4.0)</td>
<td>72.57 ± 3.29 (4.5)</td>
<td>80.33 ± 3.10 (3.5)</td>
<td>75.31 ± 2.03 (3.5)</td>
<td>75.59 ± 3.24 (3.5)</td>
<td>30</td>
</tr>
<tr>
<td>$r$-OLapSVM</td>
<td>89.33 ± 4.92 (6.0)</td>
<td>70.76 ± 5.33 (5.0)</td>
<td>68.10 ± 3.69 (4.0)</td>
<td>74.03 ± 3.13 (6.0)</td>
<td>81.41 ± 3.24 (5.0)</td>
<td>75.49 ± 2.37 (4.0)</td>
<td>75.80 ± 3.04 (5.0)</td>
<td>35.5</td>
</tr>
<tr>
<td>$r$-CPD-LapSVM</td>
<td>89.67 ± 6.44 (6.0)</td>
<td>72.32 ± 4.66 (6.5)</td>
<td>69.95 ± 2.27 (6.5)</td>
<td>75.63 ± 3.51 (7.0)</td>
<td>81.50 ± 3.09 (5.0)</td>
<td>76.95 ± 3.12 (6.0)</td>
<td>76.71 ± 3.05 (5.0)</td>
<td>42.5</td>
</tr>
</tbody>
</table>

Impact of the prevalence of labeled samples. We also explore the impact of varying prevalence levels of labeled samples on the performance of the tested algorithms. In this experiment, we used UCI liver and breast-cancer datasets as examples. The percentages of labeled data in training sets are set to an ascending sequence {20%, 40%, 60%, 80%, 100%}. The same experimental setting and hyper-parameters selection procedure are adopted from the previous experiments. The classification results are shown in Fig. 4.5.
Solid lines in both subfigures depict the results from kernel versions, while dash lines are for linear models. It is evident that CPD-LapSVM models consistently outperform the other competing methods. Kernel CPD-LapSVM, the solid red lines in both subfigures, achieves the highest classification accuracies among all solutions throughout all different label prevalence levels. In addition, linear CPD-LapSVM, dash read lines, often produces a comparable performance with other competing methods using RBF kernels.

Figure 4.5: Classification accuracies of tested methods w.r.t. different prevalence levels of labeled data. The prefix $l$ denotes linear methods and $r$ denotes kernel methods using RBF kernels.
5 **Nonlinear Metric Learning for Unsupervised Learning**

**via CPD**

5.1 **CPD based Unsupervised Metric Learning (CPD-UML)**

Many unsupervised machine learning algorithms have certain assumption regarding the distribution of the data to be processed. For example, $k$-means always produces clustering boundaries of hyperplanes, working best for the data set made of linearly separable groups. For data sets that are not linearly separable, even they are otherwise well-separable, $k$-means will fail to deliver. Nonlinearly displacing the data samples to make them linearly separable would provide a remedy, and learning such a transformation is the goal of our design. The application of such a smooth nonlinear transformation throughout feature space (either input space or kernel space) would change pairwise distances among samples, which is equivalent to assigning spatially varying metrics in different areas of the data space.

In our framework, the CPD model is also chosen to perform the transformation. The optimal solution $v(\cdot)$ for CPD is given here again as reference:

$$v(z) = \sum_{i=1}^{N} \psi_i G(z, x_i) = \sum_{i=1}^{N} \psi_i G(||z - x_i||)$$  \hspace{1cm} (5.1)

where $G(||z - x_i||) = e^{-\frac{(z-x_i)^2}{2\sigma^2}}$; $\sigma$ is the width of the Gaussian filter, which controls the overall level of smoothness in the deformation field. The matrix format of $v(z)$ can be written as:

$$v(z) = \Psi \begin{pmatrix} G(z, x_1) \\ \vdots \\ G(z, x_N) \end{pmatrix} = \Psi \tilde{G}(z, X),$$  \hspace{1cm} (5.2)

where $\Psi$ (size $d \times N$) is the weight matrix for the Gaussian kernel functions.
5.1.1 Formulation of CPD-UML

Let \( \mathcal{X} = \{ \mathbf{x}_i \mid \mathbf{x}_i \in \mathbb{R}^d, i = 1, \cdots, N \} \) denote a dataset. \( k \)-means clustering aims to partition the samples into \( K \) groups \( \mathcal{S} = \{ \mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_K \} \), through the minimization of the following objective function:

\[
\min_{\mathcal{S}} \quad J = \sum_{c=1}^{K} \sum_{\mathbf{x}_i \in \mathcal{S}_c} ||\mathbf{x}_i - \mu_c||^2 \quad \text{where} \quad \mu_c = \frac{\sum_{\mathbf{x}_i \in \mathcal{S}_c} \mathbf{x}_i}{N_c} \quad (5.3)
\]

where \( \mathcal{S}_c \) is the set of data samples in the \( c \)-th cluster. \( N_c \) is the number of data instances in cluster \( \mathcal{S}_c \), and \( \mu_c \) is the mean of \( \mathcal{S}_c \).

Allowing samples to be moved, we intend to learn a spatial transformation to improve the performance of \( k \)-means clustering by making groups more linearly separable, as well as by harnessing the updated distance measure under the transformed feature space. Let \( \mathbf{x}_i \) be the initial location of an instance. Through the motion in Eqn. (5.2), \( \mathbf{x}_i \) will be moved to a new position \( \mathbf{x}_i^1 \):

\[
\mathbf{x}_i^1 = \mathbf{x}_i + v(\mathbf{x}_i) = \mathbf{x}_i + \Psi \mathcal{G}(\mathbf{x}_i, \mathcal{X}) \quad (5.4)
\]

With Eqn. (5.4), Eqn. (5.3) can be reformulated as:

\[
\min_{\mathcal{S}^1, \Psi} \quad J = \sum_{c=1}^{K} \sum_{\mathbf{x}_i^1 \in \mathcal{S}_c^1} ||\mathbf{x}_i^1 - \mu_c^1||^2
\]

\[
\text{where} \quad \mu_c^1 = \frac{\sum_{\mathbf{x}_i^1 \in \mathcal{S}_c^1} \mathbf{x}_i^1}{N_c}; \quad \mathbf{x}_i^1 = \mathbf{x}_i + \Psi \mathcal{G}(\mathbf{x}, \mathcal{X}) \quad (5.5)
\]

Now \( \mathcal{S}^1 = \{ \mathcal{S}_1^1, \mathcal{S}_2^1, \ldots, \mathcal{S}_K^1 \} \) is a partition of the transformed dataset. \( \mu_c^1 \) is the mean vector of the instances in cluster \( \mathcal{S}_c^1 \). Our proposed CPD based unsupervised metric learning (CPD-UML) is designed to learn a spatial transformation \( \Psi \) and a clustering \( \mathcal{S}^1 \) at the same time. Eqn. (5.5) can be reformulated into a matrix format through the following steps. First, put the input dataset into a \( d \)-by-\( N \) data matrix. Second, define a Gaussian kernel function matrix for the CPD deformation as:

\[
\mathcal{G} = \mathcal{G}(\mathcal{X}, \mathcal{X}) = \{ \mathcal{G}(\mathbf{x}_1, \mathcal{X}), \mathcal{G}(\mathbf{x}_2, \mathcal{X}), \ldots, \mathcal{G}(\mathbf{x}_n, \mathcal{X}) \}
\]
The size of $\mathcal{G}$ is $N$-by-$N$. Third, let $\mathbf{e}$ be a vector of dimension $N_c$-by-1 with all elements equal to one, then the mean of the data instances within a cluster $S^1_c$ can be written as $
abla^2 = S^1_c \mathbf{e} / N_c$ (Zha, He, Ding, Gu, & Simon, 2001). With these three formulations, and let $\mathbf{E}$ be a permutation matrix, Eqn. (5.5) can be rewritten as:

$$
\begin{align*}
\min_{S^1, \Psi} J &= \sum_{c=1}^{K} \| S^1_c - \mu^1_c \mathbf{e}^T \|^2_F = \sum_{c=1}^{K} \| S^1_c - S^1_c \mathbf{e} \mathbf{e}^T / N_c \|^2_F \\
S^1 &= \mathbf{E} \mathbf{X}^1 = \mathbf{E}(\mathbf{X} + \Psi \mathbf{G}(\mathbf{X}, \mathbf{X}))
\end{align*}
$$

where $\mathbf{X}^1$ is the transformed data matrix, and $\| \cdot \|_F$ is the matrix norm. Since $\| A \|^2_F = \text{Tr}(A^T A)$, Eqn. (5.6) can be written in the form of the trace operation:

$$
\begin{align*}
\min_{S^1, \Psi} J &= \sum_{c=1}^{K} \text{Tr}((S^1_c (\mathbf{I} - \mathbf{e} \mathbf{e}^T / N_c))^T (S^1_c (\mathbf{I} - \mathbf{e} \mathbf{e}^T / N_c))) \\
&= \sum_{c=1}^{K} \text{Tr}(S^1_c (\mathbf{I} - \mathbf{e} \mathbf{e}^T / N_c) (\mathbf{I} - \mathbf{e} \mathbf{e}^T / N_c)^T (S^1_c)^T)
\end{align*}
$$

As $\text{Tr}(AB) = \text{Tr}(BA)$, and $\mathbf{e}^T \mathbf{e} = N_c$, the $J$ in Eqn. (5.7) can be further reformulated as:

$$
\begin{align*}
J &= \sum_{c=1}^{K} \text{Tr}((\mathbf{I} - \mathbf{e} \mathbf{e}^T / N_c) (\mathbf{I} - \mathbf{e} \mathbf{e}^T / N_c)^T (S^1_c)^T S^1_c) \\
&= \sum_{c=1}^{K} \text{Tr}((S^1_c)^T S^1_c - (\mathbf{e} / \sqrt{N_c}) (S^1_c)^T S^1_c (\mathbf{e} / \sqrt{N_c}))
\end{align*}
$$

Similar to (Zha et al., 2001), we define a $N$-by-$K$ orthonormal matrix $\mathbf{Y}$ as the cluster indicator matrix:

$$
\mathbf{Y} = [\mathbf{Y}_1, \mathbf{Y}_2, ..., \mathbf{Y}_K]
$$

where $\mathbf{Y}_c = (0, 0, \mathbf{e}^T, 0, 0, ..., 0) / \sqrt{N_c}$. With $\mathbf{X}^1 = \mathbf{X} + \Psi \mathbf{G}(\mathbf{X}, \mathbf{X})$ and the cluster indicator matrix in Eqn. (5.9), Eqn. (5.8) can be written into the following:

$$
\begin{align*}
\min_{\mathbf{Y}, \Psi} J &= \text{Tr}((\mathbf{X} + \Psi \mathbf{G})^T (\mathbf{X} + \Psi \mathbf{G})) - \text{Tr}(\mathbf{Y}^T (\mathbf{X} + \Psi \mathbf{G})^T (\mathbf{X} + \Psi \mathbf{G}) \mathbf{Y}) \\
&= \lambda \text{Tr}(\Psi^T \Psi)
\end{align*}
$$

To reduce overfitting, we add the squared Frobenius norm $\lambda \| \Psi \|^2_F = \lambda \text{Tr}(\Psi^T \Psi)$, to penalize any non-smoothness in the estimated transformations. $\lambda$ is a regularization
Finally, our nonlinear CPD-UML solution is formulated as a trace minimization problem, parameterized by $Y$ and $\Psi$:

$$
\min_{Y,\Psi} J = \text{Tr}((X + \Psi G)^T(X + \Psi G)) + \lambda \text{Tr}(\Psi^T \Psi) - \text{Tr}(Y^T(X + \Psi G)^T(X + \Psi G)Y)
$$

(5.11)

### 5.1.2 Optimization Strategy

To search for the optimal solutions of $Y$ and $\Psi$, an EM-like iterative minimization framework is adopted to update $Y$ and $\Psi$ alternately. The transformation matrix $\Psi$ is initialized with all 0 elements, and the cluster indicator is initialized with a $k$-means clustering result of the input data samples.

**Optimization for $Y$** With $\Psi$ fixed, Eqn. (5.11) reduces to a trace maximization problem:

$$
\max_Y J = \text{Tr}(Y^T X^T X Y)
$$

(5.12)

Since $Y$ is an orthonormal matrix: $Y^T Y = I_\mathcal{K}$, the spectral relaxation technique (Zha et al., 2001) can be adopted to compute the optimal $Y$. The solution is based on Ky Fan matrix inequalities below:

**Theorem.** (Ky Fan) If $A$ be a symmetric matrix with eigenvalues $\{\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n\}$. Let the corresponding eigenvectors be $\{v_1, v_2, ... v_n\}$, then

$$
\max_{Y^TY = I_\mathcal{K}} \text{Tr}(Y^T A Y) = \sum_{i=1}^{\mathcal{K}} \lambda_i
$$

where the optimal $Y^*$ is given by $Y^* = [v_1, v_2, ... v_\mathcal{K}]Q$ for any arbitrary orthogonal matrix $Q$.

This spectral relaxation solution can be regarded as a manifold learning method that projects data samples from the original space with the dimension of $d$ to a new space with the dimension of $K$. In our case, the $A$ matrix in Ky Fan Theorem takes the form of $X^T X$. In the implementation, we first compute the $K$ largest eigenvectors of $X^T X$, and then apply
the traditional \( k \)-means method, under the induced \( K \)-dimensional space, to compute the cluster assignments.

**Optimization for \( \Psi \)** With the \( Y \) generated from Eqn. (5.12), Eqn. (5.11) becomes a trace minimization problem w.r.t. \( \Psi \):

\[
\min_{\Psi} \quad J = \text{Tr}((X + \Psi G)^T(X + \Psi G)) + \lambda \text{Tr}(\Psi^T \Psi) \\
- \text{Tr}(Y^T(X + \Psi G)^T(X + \Psi G)Y)
\]  

(5.13)

Through a careful investigation of the gradient and Hessian matrix of Eqn. (5.13), we found the \( J \) could be proved to a smooth convex function, with its Hessian w.r.t. \( \Psi \) being positive definite (PD) everywhere. Therefore, the only stationary point of \( J \), where the gradient is evaluated to 0, locates the global minimum, and provides the optimal \( \Psi^* \). The convexity proof is given as follows.

**Convexity proof of \( J \) w.r.t. \( \Psi \):** Firstly, we update \( J \) in Eqn. (5.13), through several straightforward derivation steps to an equivalent form:

\[
J = \text{Tr}(X^T X) + 2\text{Tr}(G^T \Psi^T X) + \text{Tr}((\Psi G G^T \Psi)^T) - \text{Tr}(Y^T X^T X Y) \\
- 2\text{Tr}(Y^T G \Psi^T X Y) - \text{Tr}(\Psi G Y Y^T G^T \Psi^T) + \lambda \text{Tr}(\Psi^T \Psi)
\]  

(5.14)

The details of the derivation above are given below. Eqn. (5.13) can be expanded to:

\[
J = \text{Tr}(X^T X + X^T \Psi G + G^T \Psi^T X + G^T \Psi G) + \lambda \text{Tr}(\Psi^T \Psi) \\
- \text{Tr}(Y^T X^T X Y + Y^T G \Psi^T X Y + G^T \Psi Y Y^T G^T \Psi^T G Y)
\]  

(5.15)

As \( \text{Tr}(A) = \text{Tr}(A^T) \) for any matrix \( A \), we get \( \text{Tr}(X^T \Psi G) = \text{Tr}(G^T \Psi^T X) \) and \( \text{Tr}(Y^T X^T \Psi G Y) = \text{Tr}(Y^T G^T \Psi Y X) \). With these two equations, Eqn. (5.15) becomes:

\[
J = \text{Tr}(X^T X + 2\text{Tr}(G^T \Psi^T X) + G^T \Psi G) - \text{Tr}(Y^T X^T X Y) \\
- 2\text{Tr}(Y^T G \Psi^T X Y) - \text{Tr}(Y^T G^T \Psi Y G Y) + \lambda \text{Tr}(\Psi^T \Psi)
\]  

(5.16)

Through some simple matrix manipulations, as well as based on the fact that \( \text{Tr}(A B) = \text{Tr}(B A) \) for any matrix \( A \) and \( B \), Eqn. (5.16) can be finally updated to Eqn. (5.14).
Therefore, the gradient of $J$ w.r.t. $\Psi$ can then be computed as:

$$\frac{\partial J}{\partial \Psi} = 2XG^T - 2XYY^T G^T + 2\Psi GY^T G^T + 2\Psi GG^T I + 2\lambda \Psi$$  \hspace{1cm} (5.17)$$

To facilitate the convexity proof, we rewrite this gradient equation as:

$$\frac{\partial J}{\partial \Psi} = N + \Psi T$$

$$N = 2XG^T - 2XYY^T G^T$$

$$T = 2(G(I - YY^T)G^T + \lambda I)$$  \hspace{1cm} (5.18)$$

$N$ is a matrix of size $d \times N$. $T$ is a symmetric matrix of size $N \times N$, which can be proved positive definite, based on the theorem in (Horn & Johnson, 2012):

**Theorem.** "Suppose that $A_{m \times n}$ and $B_{n \times m}$ are two matrices with $m \leq n$. Then $BA$ has the same eigenvalues as $AB$, counting multiplicity, together with an additional $n - m$ eigenvalues equal to 0."

We know $Y^T \ast Y = I_K$, whose eigenvalues are all 1s. Then, according to this Theorem, the eigenvalues of $YY^T$ are 1s (multiplicity is $K$), and 0s (multiplicity is $N - K$). In the matrix $T$ of Eqn. (5.18), $I - YY^T$ is a positive semidefinite matrix as it is symmetric and its eigenvalues are either 0 or 1. $G$ is also positive definite because it is a Gram matrix with the Gaussian kernel. With $G$ being symmetric PD and $\lambda$ setting to be a positive number in our algorithm, the matrix $T$ is guaranteed to be a PD matrix.

Expanding the gradient formulated in Eqn. (5.18) to individual elements of $\Psi$, it can be further written as:

$$\frac{\partial J}{\partial \Psi_{ij}} = N_{ij} + \sum_{u=1}^{n} \Psi_{iu} T_{uj}$$  \hspace{1cm} (5.19)$$

where $1 \leq i \leq d; 1 \leq j \leq N$. With Eqn. (5.19), Eqn. (5.18) can be resized to a vector of size $d \times N$. Then, the Hessian matrix $H$ of $J$ w.r.t. $\Psi$ can be calculated as below:
It is clear that $H$ is a symmetric matrix with size $(d \times n) \times (d \times n)$. The diagonal of $H$ is composed of $d$ repeating $T$ matrices. Let $z$ be any non-zero column vector with size $(d \times n) \times 1$. To prove $H$ is a PD matrix, we want to show that $z^T H z$ is always positive. To this end, we rewrite $z$ as $[z_1, z_2, ..., z_d]$, where $z_i$ is the sub-column of $z$ with size $n \times 1$. Then $z^T H z$ can be computed as:

$$
z^T H z = [z_1^T T, z_2^T T, ..., z_d^T T] z = z_1^T T z_1 + z_2^T T z_2 + ... + z_d^T T z_d \quad (5.20)
$$
As $T$ has been proved to be a PD matrix, each item in Eqn. (5.20) is positive. Therefore, the summation $z^T H z$ is also positive. Since $z$ is an arbitrary non-zero column vector, this shows $H$ is PD.

With the Hessian matrix $H$ being PD everywhere, the objective function $J$ is convex w.r.t. $\Psi$. As a result, the stationary point of $J$ makes the unique global minimum solution $\Psi^*$. Let Eqn. (5.17) equal to 0, we get

$$\Psi^* = (X Y Y^T g^T - X X^T (I - Y Y^T) g^T + \lambda I)^{-1}$$

(5.21)

### 5.1.3 Kernelization of CPD-UML

So far, we developed and applied our proposed CPD-UML under input feature spaces. However, it can be further kernelized to improve the clustering performance for more complicated data. A kernel principal component analysis (KPCA) based framework (C. Zhang et al., 2010) is utilized in our work. After the input data instances are projected into kernel spaces introduced by KPCA, CPD-UML can be applied under the kernel spaces to learn both deformation field and clustering result, in the same manner as it is conducted under the original input spaces.

### 5.1.4 Main Algorithm

Based on the description above, our proposed CPD-UML algorithm can be summarized as the pseudo-code below:

### 5.2 Experiments on Synthetic Data

The two-moon synthetic dataset \(^4\) was tested in the first set of experiments. It consists of two classes with 100 examples in each class. (see Fig. 5.1). All the samples were treated

\(^4\) http://manifold.cs.uchicago.edu/manifold_regularization/data.html
Algorithm 1 Main Algorithm of CPD-UML

Input: Samples $X$, cluster number $K$, regularization parameter $\lambda$, and smoothness parameter $\sigma$ in CPD model

Output: Transformation matrix $\Psi$ and cluster indicator matrix $Y$

Initialize transformation matrix $\Psi$ using zero values;

Compute the initial cluster indicator matrix $Y$ using spectral relaxation;

Iterate the following two steps:

(1) Update $\Psi$ as in Eqn. (5.21);

(2) Update $Y$ as in Section 5.1.2 (Ky Fan Theorem);

Until convergence

Return $\Psi$ and $Y$;

as unlabeled samples in the experiments. Both linear and kernel versions of our CPD-UML were tested.

Linear version CPD-UML In this experiment, our CPD-UML was applied in deforming the data samples to achieve better separability under the input space. The effectiveness of our approach is demonstrated by comparing with the base algorithm $k$-means.

The clustering results of $k$-means and CPD-UML are shown in Fig. 5.1 (a) and 5.1 (b) respectively. The sample labels are distinguished using blue and red colors. The clustering results are shown using the decision boundary. It is obvious that $k$-means cannot cluster the two-moon data well due to the data's non-separability under the input space. Our CPD-UML, on the contrary, achieves a 99% clustering accuracy by making the data samples linearly separable via space transformations. The deformation field of Fig. 5.1 (b) in the input space is shown in Fig. 5.1 (c) and (d). It is evident that our nonlinear metric learning
model can deform feature spaces in a sophisticated yet smooth way to improve the data separability.

Kernel version CPD-UML In this set of experiments, various RBF kernels were applied on the two-moon dataset to simulate linearly non-separable cases under kernel spaces. The clustering results of kernel $k$-means with different RBF kernels ($width = 4$, \ldots)
8, 16, 32) are shown in Fig. 5.2 (a) – 5.2 (d). Colors and decision boundaries stand for the same meaning as those in Fig. 5.1. Obviously, the performance of kernel $k$-means was getting worse with sub-optimal kernels, as in 5.2 (b), 5.2 (c) and 5.2 (d). Searching for an optimal RBF kernel requires cross-validation among many candidates, which could result in a large number of iterations. This procedure can be greatly eased by our kernel CPD-UML. The CPD transformation under kernel spaces provides a supplementary force to the kernelization to further improve the data separability, the same as it performs under the input space. Fig. 5.2 (f) – 5.2 (h) demonstrate the effectiveness of our CPD-UML. Same RBF kernels as in Fig. 5.2 (b) – 5.2 (d) were used, but better clustering results were obtained. The ability to work with sub-optimal kernels should also be regarded as a computational advantage of our model.

5.3 Experiments on UCI Datasets

Experimental Setup In this section, we employ six benchmark datasets to evaluate the performance of our CPD-UML. They are five UCI datasets $^5$ : Breast, Diabetes, Cars, Dermatology, E. Coli and the USPS_20 handwritten data. Their basic information is summarized in Table 5.1.

Both linear and kernel versions of our proposed approach were tested. For linear version, $k$-means method was used as the baseline for comparison. In addition, three unsupervised metric learning solutions, AML (J. Ye et al., 2007), RPCA-OM (Nie, Yuan, & Huang, 2014) and FME (Nie et al., 2010) were utilized as the competing solutions. For kernel version, the baseline algorithm is kernel $k$-means. NAML (J. Chen et al., 2007), the kernel version of AML is adopted. Since RPCA-OM and FME do not have their kernel version, the same kernelization strategy in 5.1.3 was applied to kernelize these two solutions. RBF kernels were applied for all kernel solutions.

$^5$ http://archive.ics.uci.edu/ml/
Table 5.1: Six benchmark datasets used in experiments. Columns show the name, numbers of samples, attributes and classes of each dataset.

<table>
<thead>
<tr>
<th>Datasets</th>
<th># Samples</th>
<th># Attributes</th>
<th># Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast</td>
<td>683</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>Diabetes</td>
<td>768</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Cars</td>
<td>392</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>Dermatology</td>
<td>366</td>
<td>34</td>
<td>6</td>
</tr>
<tr>
<td>Ecoli</td>
<td>336</td>
<td>343</td>
<td>8</td>
</tr>
<tr>
<td>USPS20</td>
<td>1854</td>
<td>256</td>
<td>10</td>
</tr>
</tbody>
</table>

Each dataset was partitioned into seen and unseen data randomly. Optimal cluster centers and parameters are determined by the seen data. Clustering performance is evaluated via the unseen data, which are labeled directly based on their distances away from the cluster centers. Similar setups have been used in (Nie, Zeng, Tsang, Xu, & Zhang, 2011; Z. Huang et al., 2015). In the experiments, we performed 3-fold cross-validation, in which two folds were used as seen data and one fold as unseen data. In the competing solutions, the hyper-parameters were searched within the same range as in their publications. In our proposed approach, the regularization parameter $\lambda$ and smooth parameter $\sigma$ were searched from $\{10^0 \sim 10^{10}\}$ and $\{2^0 \sim 2^{10}\}$, respectively. The RBF kernel width for all kernel methods is chosen from $\{2^{-5} \sim 2^{10}\}$. Since the performance of tested methods depends on the initialization clusters, the clustering result of $k$-means was applied as the initialization clusters for all the competing solutions in each run. The performance of each algorithm was calculated over 20 runs.

**Results** We measured the performance using the ground truth provided in all six benchmark datasets. Three standard performance metrics were calculated: accuracy,
Table 5.2: Accuracy

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Breast</th>
<th>Diabetes</th>
<th>Cars</th>
<th>Dermatology</th>
<th>E. Coli</th>
<th>USPS20</th>
<th>Total Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$-means</td>
<td>96.39±0.34</td>
<td>96.42±0.12</td>
<td>42.67±0.35</td>
<td>69.64±7.73</td>
<td>57.11±3.23</td>
<td>62.29±3.12</td>
<td>19.5</td>
</tr>
<tr>
<td></td>
<td>(2.5)</td>
<td>(3.5)</td>
<td>(0.0)</td>
<td>(4.5)</td>
<td>(6.0)</td>
<td>(3.0)</td>
<td></td>
</tr>
<tr>
<td>AML</td>
<td>96.34±0.35</td>
<td>96.49±0.09</td>
<td>43.67±0.50</td>
<td>63.17±3.07</td>
<td>58.87±3.27</td>
<td>64.36±3.93</td>
<td>17.5</td>
</tr>
<tr>
<td></td>
<td>(1.5)</td>
<td>(4.5)</td>
<td>(1.5)</td>
<td>(0.0)</td>
<td>(6.0)</td>
<td>(4.0)</td>
<td></td>
</tr>
<tr>
<td>RPCA-OM</td>
<td>96.54±0.49</td>
<td>96.71±0.31</td>
<td>43.81±0.78</td>
<td>71.15±4.50</td>
<td>58.55±3.28</td>
<td>62.64±3.87</td>
<td>25.5</td>
</tr>
<tr>
<td></td>
<td>(3.5)</td>
<td>(6.0)</td>
<td>(1.5)</td>
<td>(5.0)</td>
<td>(6.0)</td>
<td>(3.5)</td>
<td></td>
</tr>
<tr>
<td>FME</td>
<td>96.36±0.30</td>
<td>96.57±0.35</td>
<td>48.86±2.11</td>
<td>77.72±3.68</td>
<td>58.59±3.54</td>
<td>66.27±4.09</td>
<td>30.5</td>
</tr>
<tr>
<td></td>
<td>(1.5)</td>
<td>(4.5)</td>
<td>(4.5)</td>
<td>(8.0)</td>
<td>(6.0)</td>
<td>(6.0)</td>
<td></td>
</tr>
<tr>
<td>CPD-UML</td>
<td>97.06±0.44</td>
<td>97.52±0.19</td>
<td>45.33±0.73</td>
<td>68.67±7.48</td>
<td>57.01±3.67</td>
<td>64.78±3.64</td>
<td>33.5</td>
</tr>
<tr>
<td></td>
<td>(8.5)</td>
<td>(8.0)</td>
<td>(3.0)</td>
<td>(4.0)</td>
<td>(5.5)</td>
<td>(4.5)</td>
<td></td>
</tr>
<tr>
<td>Kernel $k$-means</td>
<td>96.56±0.29</td>
<td>95.57±0.13</td>
<td>61.71±0.08</td>
<td>64.98±2.02</td>
<td>55.04±2.49</td>
<td>56.14±4.98</td>
<td>16.0</td>
</tr>
<tr>
<td></td>
<td>(4.0)</td>
<td>(1.0)</td>
<td>(6.0)</td>
<td>(1.5)</td>
<td>(3.0)</td>
<td>(0.5)</td>
<td></td>
</tr>
<tr>
<td>NAML</td>
<td>96.56±0.30</td>
<td>95.57±0.15</td>
<td>62.13±0.54</td>
<td>66.60±2.01</td>
<td>49.62±2.82</td>
<td>57.76±3.82</td>
<td>16.5</td>
</tr>
<tr>
<td></td>
<td>(4.0)</td>
<td>(1.0)</td>
<td>(7.0)</td>
<td>(3.5)</td>
<td>(0.5)</td>
<td>(0.5)</td>
<td></td>
</tr>
<tr>
<td>$r$-RPCA-OM</td>
<td>96.76±0.27</td>
<td>95.57±0.14</td>
<td>49.55±2.53</td>
<td>65.69±3.75</td>
<td>52.72±5.26</td>
<td>67.44±4.29</td>
<td>24.0</td>
</tr>
<tr>
<td></td>
<td>(6.5)</td>
<td>(1.0)</td>
<td>(4.5)</td>
<td>(2.5)</td>
<td>(2.0)</td>
<td>(7.5)</td>
<td></td>
</tr>
<tr>
<td>$r$-FME</td>
<td>96.73±0.25</td>
<td>96.79±0.14</td>
<td>63.38±1.51</td>
<td>79.45±3.86</td>
<td>50.78±2.53</td>
<td>68.13±5.47</td>
<td>36.5</td>
</tr>
<tr>
<td></td>
<td>(5.0)</td>
<td>(6.5)</td>
<td>(8.5)</td>
<td>(8.0)</td>
<td>(1.0)</td>
<td>(7.5)</td>
<td></td>
</tr>
<tr>
<td>$r$-CPD-UML</td>
<td>96.95±0.41</td>
<td>97.75±0.10</td>
<td>63.77±1.52</td>
<td>80.37±4.80</td>
<td>63.02±2.74</td>
<td>68.58±2.49</td>
<td>50.5</td>
</tr>
<tr>
<td></td>
<td>(8.0)</td>
<td>(9.0)</td>
<td>(8.5)</td>
<td>(8.0)</td>
<td>(9.0)</td>
<td>(8.0)</td>
<td></td>
</tr>
</tbody>
</table>

normalized mutual information and purity. To better compare the tested methods in statistics, we conducted a Student’s $t$-test with a $p$-value 0.05 between each pair of solutions for each dataset. The solutions were ranked using a scoring schema from (J. Wang et al., 2012). Compared with other methods, an algorithm scores 1 if it performs significantly better than one opponent in statistics; 0.5 if there is no significant difference, and 0 if it is worse.
Table 5.3: Normalized Mutual Information

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Breast</th>
<th>Diabetes</th>
<th>Cars</th>
<th>Dermatology</th>
<th>E. Coli</th>
<th>USPS20</th>
<th>Total Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$-means</td>
<td>76.10 ± 1.73</td>
<td>76.31 ± 0.63</td>
<td>18.17 ± 0.50</td>
<td>80.91 ± 4.96</td>
<td>56.77 ± 2.14</td>
<td>61.73 ± 1.49</td>
<td>23.5</td>
</tr>
<tr>
<td>AML</td>
<td>75.83 ± 1.80</td>
<td>76.58 ± 0.11</td>
<td><strong>20.53 ± 0.77</strong></td>
<td>79.15 ± 2.80</td>
<td>57.81 ± 2.07</td>
<td>62.51 ± 3.25</td>
<td>25.5</td>
</tr>
<tr>
<td>RPCA-OM</td>
<td>77.07 ± 2.52</td>
<td>77.88 ± 1.64</td>
<td>20.04 ± 0.64</td>
<td>82.27 ± 4.29</td>
<td>51.22 ± 2.42</td>
<td>60.78 ± 2.40</td>
<td>29.0</td>
</tr>
<tr>
<td>FME</td>
<td>75.93 ± 1.50</td>
<td>77.01 ± 1.77</td>
<td>19.45 ± 2.44</td>
<td>84.75 ± 3.58</td>
<td>56.73 ± 2.76</td>
<td>64.01 ± 2.73</td>
<td>34.0</td>
</tr>
<tr>
<td>CPD-UML</td>
<td>79.63 ± 2.35</td>
<td>83.31 ± 0.64</td>
<td>18.19 ± 0.51</td>
<td>79.46 ± 5.09</td>
<td>46.98 ± 8.29</td>
<td>61.12 ± 3.32</td>
<td>30.5</td>
</tr>
<tr>
<td>Kernel $k$-means</td>
<td>77.01 ± 1.58</td>
<td>77.06 ± 0.10</td>
<td>8.50 ± 0.05</td>
<td>74.65 ± 3.62</td>
<td>51.04 ± 1.76</td>
<td>59.48 ± 3.05</td>
<td>16.0</td>
</tr>
<tr>
<td>NAML</td>
<td>77.01 ± 1.58</td>
<td>77.06 ± 0.11</td>
<td>1.50 ± 0.95</td>
<td>79.86 ± 1.43</td>
<td>45.66 ± 4.03</td>
<td>60.87 ± 3.11</td>
<td>14.5</td>
</tr>
<tr>
<td>$r$-RPCA-OM</td>
<td>78.19 ± 1.50</td>
<td>77.08 ± 1.47</td>
<td>3.30 ± 1.90</td>
<td>79.84 ± 2.66</td>
<td>18.36 ± 1.85</td>
<td>66.12 ± 2.66</td>
<td>22.5</td>
</tr>
<tr>
<td>$r$-FME</td>
<td>77.95 ± 1.37</td>
<td>78.32 ± 0.79</td>
<td>6.38 ± 0.62</td>
<td><strong>85.96 ± 3.17</strong></td>
<td>49.74 ± 2.89</td>
<td><strong>67.02 ± 3.76</strong></td>
<td>33.5</td>
</tr>
<tr>
<td>$r$-CPD-UML</td>
<td>79.25 ± 2.43</td>
<td><strong>85.76 ± 0.81</strong></td>
<td>10.12 ± 2.26</td>
<td>81.46 ± 2.65</td>
<td><strong>57.85 ± 2.18</strong></td>
<td>65.34 ± 2.01</td>
<td><strong>41.0</strong></td>
</tr>
</tbody>
</table>

Tables 5.2, 5.3 and 5.4 summarize the clustering performance and ranking scores. The best performance is identified in Boldface for each dataset. It is evident that our CPD-UML outperforms other competing solutions in all three standard measurements with significant margins. Highest ranking scores in the performance tables are all achieved by our kernel version approach. In addition, significant improvements have been obtained by our proposed approach compared with the baseline algorithm $k$-means and kernel $k$-means.
Table 5.4: Purity

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Breast</th>
<th>Diabetes</th>
<th>Cars</th>
<th>Dermatology</th>
<th>E. Coli</th>
<th>USPS20</th>
<th>Total Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$-means</td>
<td>96.39 ± 0.34</td>
<td>96.42 ± 0.12</td>
<td>62.85 ± 0.36</td>
<td>81.36 ± 4.45</td>
<td>81.17 ± 1.58</td>
<td>69.78 ± 2.54</td>
<td>25.5</td>
</tr>
<tr>
<td></td>
<td>(2.5)</td>
<td>(3.5)</td>
<td>(4.5)</td>
<td>(4.0)</td>
<td>(7.5)</td>
<td>(3.5)</td>
<td></td>
</tr>
<tr>
<td>AML</td>
<td>96.34 ± 0.35</td>
<td>96.49 ± 0.09</td>
<td>62.91 ± 0.38</td>
<td>78.94 ± 1.81</td>
<td>81.39 ± 1.42</td>
<td>71.08 ± 4.32</td>
<td>25.0</td>
</tr>
<tr>
<td></td>
<td>(1.5)</td>
<td>(4.5)</td>
<td>(4.5)</td>
<td>(2.5)</td>
<td>(7.5)</td>
<td>(4.5)</td>
<td></td>
</tr>
<tr>
<td>RPCA-OM</td>
<td>96.54 ± 0.49</td>
<td>96.71 ± 0.31</td>
<td>63.85 ± 0.80</td>
<td>82.64 ± 3.85</td>
<td>74.50 ± 1.85</td>
<td>69.60 ± 3.58</td>
<td>27.5</td>
</tr>
<tr>
<td></td>
<td>(3.5)</td>
<td>(6.0)</td>
<td>(7.0)</td>
<td>(5.0)</td>
<td>(2.5)</td>
<td>(3.5)</td>
<td></td>
</tr>
<tr>
<td>FME</td>
<td>96.36 ± 0.30</td>
<td>96.57 ± 0.35</td>
<td>63.86 ± 1.07</td>
<td>85.27 ± 3.20</td>
<td>80.43 ± 2.11</td>
<td>72.68 ± 3.37</td>
<td>35.0</td>
</tr>
<tr>
<td></td>
<td>(1.5)</td>
<td>(4.5)</td>
<td>(7.0)</td>
<td>(8.0)</td>
<td>(7.0)</td>
<td>(7.0)</td>
<td></td>
</tr>
<tr>
<td>CPD-UML</td>
<td>97.06 ± 0.44</td>
<td>97.52 ± 0.19</td>
<td>63.10 ± 0.49</td>
<td>81.02 ± 4.45</td>
<td>74.88 ± 5.82</td>
<td>69.80 ± 3.61</td>
<td>31.5</td>
</tr>
<tr>
<td></td>
<td>(8.5)</td>
<td>(8.0)</td>
<td>(4.5)</td>
<td>(3.5)</td>
<td>(3.5)</td>
<td>(3.5)</td>
<td></td>
</tr>
<tr>
<td>Kernel $k$-means</td>
<td>96.56 ± 0.29</td>
<td>95.57 ± 0.13</td>
<td>62.16 ± 0.05</td>
<td>76.80 ± 1.94</td>
<td>76.03 ± 1.57</td>
<td>65.29 ± 4.29</td>
<td>10.5</td>
</tr>
<tr>
<td></td>
<td>(4.0)</td>
<td>(1.0)</td>
<td>(1.0)</td>
<td>(0.0)</td>
<td>(4.0)</td>
<td>(0.5)</td>
<td></td>
</tr>
<tr>
<td>NAML</td>
<td>96.56 ± 0.30</td>
<td>95.57 ± 0.15</td>
<td>62.21 ± 0.32</td>
<td>79.37 ± 1.02</td>
<td>70.90 ± 3.93</td>
<td>66.69 ± 3.38</td>
<td>10.5</td>
</tr>
<tr>
<td></td>
<td>(4.0)</td>
<td>(1.0)</td>
<td>(1.0)</td>
<td>(3.0)</td>
<td>(1.0)</td>
<td>(0.5)</td>
<td></td>
</tr>
<tr>
<td>$r$-RPCA-OM</td>
<td>96.76 ± 0.27</td>
<td>95.57 ± 0.14</td>
<td>62.29 ± 0.41</td>
<td>79.26 ± 2.16</td>
<td>52.84 ± 5.29</td>
<td>73.46 ± 2.95</td>
<td>19.0</td>
</tr>
<tr>
<td></td>
<td>(6.5)</td>
<td>(1.0)</td>
<td>(1.0)</td>
<td>(3.0)</td>
<td>(0.0)</td>
<td>(7.5)</td>
<td></td>
</tr>
<tr>
<td>$r$-FME</td>
<td>96.73 ± 0.25</td>
<td>96.79 ± 0.14</td>
<td>63.38 ± 1.76</td>
<td>85.44 ± 3.16</td>
<td>76.01 ± 2.62</td>
<td>72.77 ± 3.42</td>
<td>36.0</td>
</tr>
<tr>
<td></td>
<td>(5.0)</td>
<td>(6.5)</td>
<td>(5.5)</td>
<td>(8.0)</td>
<td>(4.0)</td>
<td>(7.0)</td>
<td></td>
</tr>
<tr>
<td>$r$-CPD-UML</td>
<td>96.95 ± 0.41</td>
<td>97.75 ± 0.10</td>
<td>64.86 ± 0.85</td>
<td>85.59 ± 2.56</td>
<td>82.09 ± 1.74</td>
<td>74.44 ± 2.14</td>
<td>49.5</td>
</tr>
<tr>
<td></td>
<td>(8.0)</td>
<td>(9.0)</td>
<td>(9.0)</td>
<td>(8.0)</td>
<td>(8.0)</td>
<td>(7.5)</td>
<td></td>
</tr>
</tbody>
</table>

It is also noteworthy that, the linear CPD-UML achieved comparable results with the other competing methods using RBF kernels, which further demonstrates the effectiveness of our nonlinear feature space transformation.
Figure 5.2: First column: clustering results of kernel $k$-means with RBF kernels width = 4, 8, 16 and 32. Second column: results of kernel version CPD-UML with RBF kernels width = 4, 8, 16 and 32.
6 Application on Alzheimer’s Disease Early Detection

In this chapter, we will apply our proposed CPD-LapSVM and CPD-UML models in the area of neuroimaging. We are targeting at improving the performance of computer-aided early prediction and clustering for Alzheimer’s Disease (AD) with the help of the features extracted from neuroimaging. This chapter is organized as follows. First, an introduction will be given for AD and its early prediction. Second, we will show how neuroimaging and machine learning techniques can be applied to make the classification for AD. The related works in the field are also introduced. Third, the Alzheimer’s Disease Neuroimaging Initiative (ADNI) dataset used in our experiments will be described in detail. Fourth, we will show the effectiveness of our proposed semi-supervised and unsupervised metric learning approaches in AD early prediction and clustering, compared to state-of-the-art metric learning solutions based on baseline MR images. Last, we proposed a novel surface analysis model to utilize the available longitudinal information. The features extracted from the model have been proved effective as discriminative structural features for AD vs. MCI diagnosis.

6.1 Alzheimer’s Disease and Early Prediction

Alzheimer’s disease can cause serious problems such as memory loss, difficulty in conversation, decreased judgment, etc. Two typical neuropathological signs of AD have been found during the autopsy for the AD patients. The first sign is that amyloid plaques are accumulating in the brain and obstacle the communications of nerve cells. The second is the neurofibrillar tangles, which consist of insoluble twisted fibers whose primary component is a protein called tau. The comparison between a healthy brain and the brain with AD is shown in Fig. 6.1. Up to now, there is no proven cure for AD.

According to (Jack et al., 2010), the development of AD can be categorized into three clinical disease stages: cognitively normal, Mild cognitive impairment (MCI) and
dementia, shown as Fig. 6.2. In each stage, different biological markers (biomarkers) are detected to have abnormal magnitude. The first stage is called preclinical AD. In this stage, the patients are cognitively normal and no outward symptoms are visible. However, A-beta, and tau value will increase in the blood. In the second stage - MCI, mild changes in memory and thinking can be noticed. But the change is not severe enough to affect everyday activities and functioning. In this stage, the brain structure and memory will be affected. The last stage is the dementia caused by AD. Memory, thinking and behavioral symptoms will impair a person’s ability to function in daily life.

Based on the AD clinical stages introduced above, MCI is often considered as the early stage of AD. While approximately 5% – 10% MCI patients will develop into AD each year, the others remain in this stage and never convert. Clinically, the former is called progressive MCI (pMCI) if the conversion happens within 3 years after baseline diagnosis, and the latter
is called stable MCI (sMCI). As MCI-to-AD conversion takes 3 years to detect, and many patients do not have follow-up diagnosis, a large number of MCI subjects are labeled as "Unknown MCI" (uMCI) (Moradi et al., 2015). Therefore, detecting progressive MCI can indicate a high risk of developing AD, which makes it possible to predict AD in its early stage.

Current treatments can not stop AD from progressing. However, they can temporarily slow down the development of AD and its symptoms, which can keep AD patients’ life quality. Nowadays, AD is commonly diagnosed with the dementia symptoms and neurological assessments. Since the symptoms already impair a person’s ability to function in daily life, this diagnosing strategy will inevitably miss the best time starting the treatment for AD patients. Therefore, efforts have been put forward to detect AD in its early stages. The prediction of potential AD can raise a red flag, and trigger the treatment to slow down the AD development before it causes severe damage.
6.2 Neuroimaging and Machine Learning

The typical biomarkers used to detect AD in the early stage are CSF amyloid and CSF tau. However, they can be only achieved through a spinal tap, which is an invasive observation method causing dramatic discomfort for the patients. Great efforts have been put forward on noninvasive observation methods. Under various clinical settings, neuroimaging or brain imaging are widely used to generate representations that can provide insights to view the structures or functions of human nervous systems. Neuroimaging solutions, such as positron emission tomography (PET) imaging and structure magnetic resonance (MR) imaging, have been proven effective to extract biomarkers for AD detection. Fig. 6.3 illustrates the comparison of Amyloid accumulation observed in PET images between a normal control and an AD patient.

![PET images](image)

(a) PET image of a normal brain  (b) PET image of a brain with AD

Figure 6.3: PET images showing the Amyloid accumulation with AD.

Machine learning, as a powerful tool, is often applied in the diagnosis procedure of nervous system disease. Using the features extracted from neuroimaging scans, machine
learning techniques can learn classifiers for target diseases. The learned classifiers are then used to make the diagnosis for new scans. For example, with machine learning techniques, AD can be distinguished from MCI using the features extracted from brain MRI scans. A typical working flow of AD identification using neuroimaging and machine learning is illustrated in Fig. 6.4.

Figure 6.4: Working flow of AD identification using neuroimaging and machine learning.

A number of solutions have been proposed in recent years to tackle AD/MCI early diagnosis problem. Common practices include the utilization of multi-modality (An, Adeli, Liu, Zhang, & Shen, 2016; Liu, Zhang, Yap, & Shen, 2017; Tong, Gray, Gao, Chen, & Rueckert, 2017; T. Ye, Zu, Jie, Shen, & Zhang, 2016) and longitudinal (M. Huang, Yang, Feng, & Chen, 2017; Thung, Wee, Yap, & Shen, 2016; Y. Zhu, Zhu, Kim, Shen, & Wu,
data to exploit complementary information, dimension reduction to diminish data redundancy and feature selection to extract the most discriminative feature set. Ensembles of classifiers from multiple domains or/and levels have also been employed to improve the overall classification performance (Liu et al., 2017; M. Huang et al., 2017; Y. Zhu et al., 2016; Cheng et al., 2016).

As an accurate diagnosis of MCI-AD conversion is often not available until a later time, semi-supervised learning (SSL), utilizing unlabeled data in conjunction with labeled samples (the valuable gold standard confirmed cases) to improve classification performance, is uniquely suitable to predict patients clinical trajectories. The solution in (D. Zhang & Shen, 2011) utilized MCI scans as unlabeled data to boost the accuracy in the AD vs. normal control (NC) classification. Compared with using AD/NC subjects only, a significant improvement was achieved. Similar approaches were proposed in (D. H. Ye, Pohl, & Davatzikos, 2011; Filipovych, Davatzikos, Initiative, et al., 2011) to predict disease labels for MCI subjects. A semi-supervised solution based on LDS was applied in (Moradi et al., 2015) to predict the conversion from MCI to AD. All these studies demonstrate that label augmentation through unlabeled data samples equips SSL with better predictive power over supervised learning.

Despite all the strides made in recent years, insufficient effort has been given to rationally selecting appropriate metrics to boost the power of various solutions. Learning a metric is the essentially the same as learning a feature transformation (Xing, Ng, Jordan, & Russell, 2002; B. Shi, Chen, Zhang, Smith, & Liu, 2017), and such transformations can often significantly boost the performance of the algorithms based on metrics.

In my work, we propose to enhance the prediction of MCI-AD conversion via our proposed metric learning models. We aim to predict MCI-AD conversion using only data at the baseline, obtained from the Alzheimer’s Disease Neuroimaging Initiative database. In addition, we also plan to apply our CPD-UML model to AD/MCI diagnosis, and evaluate
its effectiveness through three clustering problems: binary clustering of AD vs. Normal Control (NC), progressive MCI (pMCI) vs. stable MCI (sMCI) and ternary clustering of AD/MCI/NC.

6.3 Alzheimer’s Disease Neuroimaging Initiative (ADNI) Database

The data used in our work comes from the Alzheimer’s Disease Neuroimaging Initiative (ADNI) database (www.loni.usc.edu/ADNI) ADNI provides a wealth of structural and functional MRI images of normal elders, MCI and AD subjects. Using ADNI data, research has been well conducted for the diagnosis and prediction of AD/MCI.

Table 6.1: Demographics and clinical evaluations.

<table>
<thead>
<tr>
<th>Diagnosis</th>
<th>Number</th>
<th>Gender(M/F)</th>
<th>Age(mean std) [min-max]</th>
<th>MMSE(mean std) [min-max]</th>
<th>GDS(mean std) [min-max]</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>185</td>
<td>96/89</td>
<td>75.41 ± 7.54 [55 - 91]</td>
<td>23.38 ± 1.99 [20 - 27]</td>
<td>1.70 ± 1.45 [0 - 6]</td>
</tr>
<tr>
<td>NC</td>
<td>227</td>
<td>117/110</td>
<td>76.07 ± 5.02 [60 - 89]</td>
<td>29.12 ± 0.98 [25 - 30]</td>
<td>0.84 ± 1.13 [0 - 5]</td>
</tr>
<tr>
<td>pMCI</td>
<td>110</td>
<td>68/42</td>
<td>74.64 ± 7.46 [55 - 88]</td>
<td>26.46 ± 1.81 [22 - 30]</td>
<td>1.67 ± 1.36 [0 - 5]</td>
</tr>
<tr>
<td>sMCI</td>
<td>38</td>
<td>21/17</td>
<td>76.45 ± 6.79 [62 - 87]</td>
<td>27.29 ± 1.90 [24 - 30]</td>
<td>1.71 ± 1.45 [0 - 5]</td>
</tr>
<tr>
<td>uMCI</td>
<td>94</td>
<td>63/31</td>
<td>75.61 ± 7.79 [55 - 89]</td>
<td>26.84 ± 1.75 [24 - 30]</td>
<td>1.45 ± 1.31 [0 - 5]</td>
</tr>
</tbody>
</table>

We focus on the features extracted from baseline T1 weighted MRIs. Overall, 185 patients with AD, 242 with MCI and 227 with NC (654 subjects in total) were used in our
experiments. We provide the details about the demographics and clinical evaluations of the scans used in the experiments in Table 6.1. It includes Mini Mental State Examination (MMSE) and Geriatric Depression Scale (GDS) scores.

The features utilized in this study are the volumes of 113 cortical and subcortical brain structures of each subject, extracted from the FreeSurfer Cross-Sectional Processing aperc+aseg segmentation files, available under ADNI. These structures were segmented from the subjects’ Magnetic Resonance Imaging (MRI) scans. All features have been normalized by the corresponding whole brain volumes. The anatomical structures include left/right Hippocampi, left/right Caudates, etc., as shown in Fig. 6.5.

![Figure 6.5: (a): brain MRI of an ADNI subject. (b): anatomical segmentation of (a). Colors indicate different brain structures.](image)

The list of the structure names is provided as follows. Aseg list contains subcortical structures, and Aparc is for cortical structures.

Aseg structures Left-Lateral-Ventricle, Left-Inf-Lat-Vent, Left-Cerebellum-White-Matter, Left-Cerebellum-Cortex, Left-Thalamus-Proper, Left-Caudate, Left-Putamen, Left-

6.4 Semi-Supervised Learning for Prediction of AD Conversion from MCI

In this section, we evaluate the proposed CPD-LapSVM using two binary classification problems: AD vs. NC with MCI subjects as unlabeled samples, and progressive MCI (pMCI) vs. stable MCI (sMCI) with unknown MCI (uMCI) as unlabeled samples (the definition of “unknown MCI” will be given later). This work has been published in (P. Zhang, Shi, Smith, & Liu, 2017).

We compared our proposed method with various semi-supervised classification methods, using four measures: classification accuracy (ACC), sensitivity (SEN), specificity (SPE), and area under the receiver operating characteristic curve (AUC). Three semi-supervised methods: Laplacian regularized least squares (LapRLS), Laplacian SVM (LapSVM) and Optimized LapSVM (OLapSVM) (Y. Gu & Feng, 2013) are utilized in all experiments as the competing solutions. For each solution, both linear and RBF Gaussian kernel versions are evaluated. In the end, we also compare our method with five state-of-the-art pMCI/sMCI classification solutions (Cheng et al., 2016; D. H. Ye et al., 2011; Filipovych et al., 2011; Moradi et al., 2015; Suk, Lee, & Shen, 2017), which also used baseline T1-weighted MRIs from the ADNI database.

6.4.1 AD vs. NC with MCI as Unknown

The first set of experiments classify AD and NC subjects, with MCI subjects as unlabeled samples. The AD and NC groups, which are used as labeled subjects, were randomly divided into four folds for cross-validation (2 for training, 1 for validation and 1 for testing). All MCI subjects were shared as unlabeled data across folds.

The involved hyper-parameters $\gamma_A$, $\gamma_I$ and $\gamma_L$ are all chosen from $\{2^{-5} \sim 2^{10}\}$ over the cross-validations. $\gamma_A$ and $\gamma_I$ are the slackness tradeoff and graph regularization parameters used in all models. $\gamma_L$ is the regularization parameter, only used in our model. All the RBF
kernel versions of the methods have an additional parameter to tune: the RBF Gaussian kernel width \( \sigma \), which is also chosen from \( \{2^{-5} \sim 2^{10}\} \) in our experiments.

Table 6.2 summarizes the AD vs. NC classification results for all methods, averaged from 50 random repeats. It is evident that our CPD-LapSVM achieves the highest ACC and AUC scores among the competing methods, for both the linear and kernel versions. It is also noteworthy that the linear version CPD-LapSVM obtains comparable results with the kernel versions of other competing methods.

Table 6.2: Performance comparison of CPD-LapSVM with other methods for AD vs. NC classifications. Boldface denotes the best ACC & AUC performance.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Linear Kernel</th>
<th></th>
<th>RBF Kernel</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACC(%)</td>
<td>SEN(%)</td>
<td>SPE(%)</td>
<td>AUC(%)</td>
</tr>
<tr>
<td>LapRLSC</td>
<td>83.65</td>
<td>83.11</td>
<td>84.02</td>
<td>89.56</td>
</tr>
<tr>
<td>LapSVM</td>
<td>84.11</td>
<td>83.21</td>
<td>84.78</td>
<td>89.98</td>
</tr>
<tr>
<td>OLapSVM</td>
<td>85.36</td>
<td>84.26</td>
<td>86.38</td>
<td>90.89</td>
</tr>
<tr>
<td>CPD-LapSVM</td>
<td><strong>86.33</strong></td>
<td><strong>85.77</strong></td>
<td><strong>86.82</strong></td>
<td><strong>91.44</strong></td>
</tr>
</tbody>
</table>

6.4.2 pMCI vs. sMCI with uMCI as Unknown

The second set of experiments are designed to predict AD conversion from MCI patients through classification of pMCI vs sMCI, with uMCI as unlabeled samples. If the initial diagnosis was MCI at baseline, but the follow-up diagnosis is missing or not stable, the patient is categorized as “unknown MCI”. Overall, 110 patients with pMCI, 38 with sMCI and 227 with uMCI (242 MCI subjects in total) were used in our experiments.

The same experimental setting and hyperparameter selection approach as in the previous AD/NC classifications is adopted here. uMCI subjects were shared as unlabeled
data during 4-fold cross-validation. The results are reported in Table 6.3. Similar results as that in AD vs. NC classification can be observed. The ACC score of our linear-version CPD-LapSVM is significantly higher than all other methods. Comparing with LapSVM, which is the host solution of CPD-LapSVM, the ACC improvement on LapSVM is from 67.09% to 69.12%. For RBF Gaussian kernel versions, the highest ACC and AUC scores were both achieved by our model.

In order to investigate the effect of the number of labeled data on our method, a test was performed by decreasing the number of revealed labels. The ratio of revealed labels are decreased as: {100%, 80%, 60%, 40% and 20%}. The ACC scores with different labeled sample ratios are shown in Fig. 6.6. The Solid lines and prefix “r” denote the results from kernelized classifiers, while dashed lines and “l” are for linear classifiers. It is clear that the ACC values of our CPD-LapSVM with both linear and RBF Gaussian kernel are always performing the best.

Table 6.3: Performance comparison of CPD-LapSVM with other methods for pMCI vs. sMCI classifications. Boldface denotes the best ACC & AUC performance.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Linear Kernel</th>
<th>RBF Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACC(%) SEN(%) SPE(%) AUC(%)</td>
<td>ACC(%) SEN(%) SPE(%) AUC(%)</td>
</tr>
<tr>
<td>LapRLSC</td>
<td>67.09 62.66 80.42 79.11</td>
<td>76.09 92.82 29.61 76.49</td>
</tr>
<tr>
<td>LapSVM</td>
<td>66.37 61.86 79.61 79.03</td>
<td>76.14 88.70 40.17 76.46</td>
</tr>
<tr>
<td>OLapSVM</td>
<td>67.40 63.69 78.26 79.60</td>
<td>76.54 83.28 55.34 76.79</td>
</tr>
<tr>
<td>CPD-LapSVM</td>
<td><strong>69.12</strong> 67.92 72.03 78.07</td>
<td><strong>78.27</strong> 86.38 52.32 <strong>78.58</strong></td>
</tr>
</tbody>
</table>
6.4.3 Comparisons with State-of-the-art AD Early Prediction Methods

Finally, we summarize several recent works in pMCI vs. sMCI classification as a comparison in Table 6.4. To best of our knowledge, the best result so far was achieved by (Suk et al., 2017). Among the methods using baseline MRIs only, our work achieved the best performance in ACC score. However, it should be noted that direct comparisons of the published neuroimaging algorithms are often not feasible. When different datasets and experimental setups are utilized, higher accuracy or better results over a competing solution ought to be interpreted as more of a side evidence of the model efficacy, rather than the proof of superiority for head-to-head competitions.

Figure 6.6: ACC scores w.r.t different ratio of revealed labels in pMCI vs. sMCI classifications
<table>
<thead>
<tr>
<th>Methods</th>
<th>ACC(%)</th>
<th>SEN(%)</th>
<th>SPE(%)</th>
<th>AUC(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ye et al. (D. H. Ye et al., 2011)</td>
<td>56.10</td>
<td>94.10</td>
<td>40.80</td>
<td>73.00</td>
</tr>
<tr>
<td>Filipovych et al. (Filipovych et al., 2011)</td>
<td>—</td>
<td>79.40</td>
<td>51.7</td>
<td>69.00</td>
</tr>
<tr>
<td>Moradi et al. (Moradi et al., 2015)</td>
<td>74.74</td>
<td>88.85</td>
<td>51.46</td>
<td>76.61</td>
</tr>
<tr>
<td>Cheng et al. (Cheng et al., 2016)</td>
<td>73.80</td>
<td>69.00</td>
<td>77.40</td>
<td><strong>79.60</strong></td>
</tr>
<tr>
<td>Suk et al. (Suk et al., 2017)</td>
<td>74.82</td>
<td>70.93</td>
<td>78.82</td>
<td>75.89</td>
</tr>
<tr>
<td>CPD-LapSVM</td>
<td><strong>78.27</strong></td>
<td>86.38</td>
<td>52.32</td>
<td>78.58</td>
</tr>
</tbody>
</table>

Table 6.4: Comparisons of pMCI vs. sMCI classification solutions using ADNI database. Boldface denotes the best performance for the measure of ACC & AUC.

### 6.5 Improve Patient Clustering via Nonlinear Feature Transformation and Embedding

Clustering has been utilized to solve various neuroimaging problems, including brain tissue segmentation, fiber tracking, lesion detection, and brain network analysis. Clustering can also be used to discover subgroups among patients and identify sub-forms of particular diseases (Erro et al., 2013; Lee et al., 2011; Hrdlicka et al., 2005). Take Alzheimer’s Disease (AD) and its prodromal stage, Mild Cognitive Impairment (MCI), as an example: grouping patients into subpopulations with homogeneous clinical and biological characteristics can potentially lead to the identification of new biomarkers, pathological subtypes, and possible misdiagnosis of AD/MCI (Pasquini et al., 2014; Marinescu et al., 2017).

In this section, we apply our CPD-UML model to AD/MCI diagnosis, and evaluate its effectiveness through three clustering problems: binary clustering of AD vs. Normal Control (NC), progressive MCI (pMCI) vs. stable MCI (sMCI) and ternary clustering
of AD/MCI/NC. The same ADNI data sets described in Section 6.3 are applied in the experiment.

We use the same four measures as in Section 6.4: ACC, SEN, SPE, AUC, to evaluate our CPD-UML model, as well as the competing solutions. The baseline $k$-means method and three state-of-the-art UML solutions: AML (J. Ye et al., 2007), RPCA-OM (Nie et al., 2014) and FME (Nie et al., 2010) are utilized in all experiments as the competing solutions. For each solution, both linear and Gaussian RBF kernel versions are evaluated. NAML (J. Chen et al., 2007) is the kernel version of AML. As RPCA-OM and FME do not have tailored kernel versions, the same kernelization strategy in 5.1.3 was applied to kernelize these two solutions.

### 6.5.1 Binary clustering of AD/NC and sMCI/pMCI

The first two experiments were performed to cluster AD/NC, and sMCI/pMCI subjects. In each experiment, the data samples were randomly partitioned into seen and unseen groups, with a ratio of 2 : 1. Optimal cluster centers and parameters are determined by the seen data samples. Clustering performance is evaluated via the unseen instances, which are classified directly based on their distances away from the cluster centers. Similar setups have been used in (Nie et al., 2011; Song et al., 2013). In the competing solutions, the hyper-parameters were searched within the same range as in their publications. In our proposed approach, the regularization parameter $\lambda$ and smooth parameter $\sigma$ were searched from $\{10^0 \sim 10^{10}\}$ and $\{2^0 \sim 2^{10}\}$, respectively. The RBF kernel width for all kernel methods is chosen from $\{2^{-5} \sim 2^{10}\}$. Since the performance of tested methods depends on the initialization clusters, the clustering result of $k$-means was applied as the initialization clusters for all the competing solutions in each run. The results are reported for 50 random repeats.
Table 6.5: Measurements of tested methods for AD/NC clustering.

Table 6.5 and Table 6.6 summarize the results. The second row of each corresponding method in the table provides the standard deviation. Boldface denotes the highest ACC & AUC values. In the experiment of AD/NC, our CPU-UML achieved the best ACC and AUC results in both linear and kernel versions. The most significant improvements based on the baseline method are 2.72% in ACC and 2.75% in AUC using RBF kernels. In the second experiment, while the best AUC values were from the competing solutions, our proposed framework obtained the highest accuracies, with an improvement of 5.66% in linear version, and 4.00% in kernel version compared to the baseline method.

### 6.5.2 Ternary clustering of AD/MCI/NC

In the third experiment, we demonstrate the improvement achieved by our framework in clustering the data with multiple categories. All 654 subjects were used in this experiment. The same experimental settings were adopted from the previous experiments. We provide two pairs of sensitivity and specificity measures for the clustering results using
the “one vs. the rest” strategy. The first pair (SEN1 and SPE1) is calculated using AD subject as positive samples, and the rest as negative samples. The second pair (SEN2 and SPE2) treats AD and MCI as positive, and NC as negative. The AUC measures are also calculated, but they are not included in the table due to the space limit.

The experimental results are summarized in Table 6.7. Because of multi-class, the accuracies produced by the clustering algorithms are commonly lower than those of binary clustering. Nevertheless, our model improves the performance of the baseline \textit{k}-means by 1.61\% with the linear kernel, and 1.81\% with RBF kernels. Furthermore, our framework also gets the best accuracies among all the competing solutions, for both linear and kernel versions.

To sum it up, our proposed CPD-UML performs steadily well across the three experiments, and the improvements over the baseline \textit{k}-means clustering are stable and
significant. While we focus on AD/MCI patient clustering in this work, the proposed $k$-means + UML solution should be very well applicable to many other neuroimaging tasks.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Linear Kernel</th>
<th>RBF Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SEN1</td>
<td>SPE1</td>
</tr>
<tr>
<td>$k$-means</td>
<td>57.66</td>
<td>52.94</td>
</tr>
<tr>
<td>±1.62 ±1.68 ±1.43 ±2.13 ±1.91</td>
<td>±1.72 ±1.36 ±1.18 ±1.84 ±2.07</td>
<td></td>
</tr>
<tr>
<td>AML</td>
<td>57.37</td>
<td>53.76</td>
</tr>
<tr>
<td>±2.42 ±1.68 ±1.24 ±1.42 ±2.61</td>
<td>±2.54 ±2.12 ±1.49 ±2.08 ±2.45</td>
<td></td>
</tr>
<tr>
<td>RPCA-OM</td>
<td>59.08</td>
<td>53.55</td>
</tr>
<tr>
<td>±2.17 ±1.77 ±1.56 ±2.28 ±1.97</td>
<td>±2.33 ±1.62 ±1.41 ±2.19 ±2.03</td>
<td></td>
</tr>
<tr>
<td>FME</td>
<td>59.21</td>
<td>53.66</td>
</tr>
<tr>
<td>±2.77 ±2.09 ±1.47 ±2.16 ±2.03</td>
<td>±2.40 ±2.54 ±1.46 ±2.06 ±2.08</td>
<td></td>
</tr>
<tr>
<td>CPD-UML</td>
<td>59.18</td>
<td>54.59</td>
</tr>
<tr>
<td>±1.27 ±1.83 ±1.40 ±2.29 ±2.60</td>
<td>±1.27 ±1.61 ±1.35 ±2.30 ±2.29</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.7: Measurements of tested methods for AD/MCI/NC clustering.

### 6.6 Quad-mesh based Feature Extraction

In the experiments above, we improved the performance of AD early prediction and clustering by applying our proposed metric learning algorithms on the features extracted from baseline MRIs. For anatomical features extracted from structural MRIs, cortical thickness (Klöppel et al., 2008), volumetry of brain structures (Chupin et al., 2009; Suk, Lee, Shen, Initiative, et al., 2013) and voxel tissue probability maps (Fan, Shen, Gur, Gur, & Davatziko, 2007; Liu, Zhang, & Shen, 2014) across the whole brain or around certain regions of interest, are among the popular choices.

Hippocampal atrophy is one of the major detectable features of AD and mixed pathologies (Cahn et al., 1998; Petersen et al., 2000; Nelson et al., 2013). Observing the
longitudinal progress is crucial in detecting and differentiating the conditions. Solutions to analyze 3D hippocampal shapes and shape changes can be classified into two categories: medial structure based and surface-based approaches. Solutions in the first category (Fletcher, Lu, Pizer, & Joshi, 2004; Pizer et al., 1999; Bouix, Pruessner, Collins, & Siddiqi, 2005; Styner, Gerig, Lieberman, Jones, & Weinberger, 2003; Thompson et al., 2004; Yushkevich, 2009) rely on extracted hippocampal central lines to compare 3D shapes and conduct statistical group analysis. Medial axis radial distance models (Thompson et al., 2004; Chou et al., 2009) and m-rep (Styner et al., 2003; Yushkevich, 2009) are among the widely used solutions. Topological disparities due to shape variations cause branches of the medial axis and often pose a challenge for these models, where the correspondence of two topologically different medial trees needs to be carefully defined and sought. The second category (Csernansky et al., 2005; Gerig, Styner, Jones, Weinberger, & Lieberman, 2001; Shen et al., 2003; Shenton, Gerig, McCarley, Szekely, & Kikinis, 2002; Y. Shi et al., 2007; L. Wang et al., 2006; Y. Wang, Chiang, & Thompson, 2005; X. Gu & Vemuri, 2004; Y. Wang et al., 2010; T. Chen et al., 2012) uses entire hippocampal surfaces as the basis to compute subject/group dis-similarity, after the surfaces are aligned. Spherical mapping (X. Gu & Vemuri, 2004; T. Chen et al., 2012) and surface tensor-based morphometry (Y. Wang et al., 2010) have been applied to model hippocampal surfaces and atrophy. Finding the correspondence for surface vertices usually entails a non-rigid registration procedure, which tends to be computationally expensive and may fail to find satisfying alignments.

Despite the reported improvements, the existing feature extraction and feature space deformation solutions have their limitations and drawbacks. In our research, by exploiting new geometric models to overcome these drawbacks, we propose a novel quadrilateral mesh based feature extraction solution to better utilize the longitudinal information. The proposed method obtains evenly spaced quad meshes providing a common coordinate
system to align and compare shapes. We then utilized radial distances based on the quad-mesh representation to characterize surfaces as well as estimate their changes over time. In addition, we also extracted region-of-interest patches for radial distance and its change. We then examined their effectiveness as discriminative structural features for AD/MCI diagnosis. This work has been published in (Hobbs, Zhang, Shi, Smith, & Liu, 2016).

6.6.1 Harmonic Mapping

The term "mapping" refers to a function which pairs each element of its domain with each member of its range. For example, for a mapping from a Riemannian manifold $S$ to $R$, $S$ is the domain of the mapping and $R$ is the range. There are various kinds of mapping techniques. A mapping is called harmonic is it is a critical point of the Dirichlet energy functional

$$E_S(f) = \int_S \|df_x\|^2 dm_x \quad (6.1)$$

where $E(f)$ is the Dirichlet functional on the Riemannian manifold $S$. $f$ is the function to be optimized. $x$ is the element on $S$. $dm_x := dx^1...dx^m$.

Harmonic mapping minimizes the elastic membrane energy during the mapping. An intuitive way to understand the procedure is to imagine the manifold $S$ is made of rubber and $R$ is made of stone. Suppose we want to map $S$ onto $R$. The Dirichlet functional $E(f)$ can be seen as the elastic membrane energy generated during the mapping procedure. Under this setting, $f$ is a harmonic mapping if the rubber $S$ contacts with the stone $R$ everywhere after it is released.

Discrete harmonic mapping is widely used in the mesh parametrizations. The object surfaces are commonly represented using triangle meshes. Here we use $M$ to denote the surface mesh, $m_i$ as the vertex on an edge $[m_i, m_j]$ of a triangle $[m_i, m_j, m_k]$. For any point $P$ on the triangle, it can be represented using barycentric coordinates:
\[ p = g_i \mathbf{m}_i + g_j \mathbf{m}_j + g_k \mathbf{m}_k \]  
(6.2)

where \( g_i, g_j, \) and \( g_k \) are the coordinates, which can be computed as:

\[ g_i = \frac{\text{Area}(\mathbf{p}, \mathbf{m}_j, \mathbf{m}_k)}{\text{Area}(\mathbf{m}_i, \mathbf{m}_j, \mathbf{m}_k)} \]  
(6.3)

Then, the objective function to be optimized in the discrete harmonic mapping becomes a piecewise linear function \( f(p) = g_i f(\mathbf{m}_i) + g_j f(\mathbf{m}_j) + g_k f(\mathbf{m}_k) \). It is straightforward to compute the harmonic energy \( E(f) \) as follow

\[ E(f) = \sum v_{ij} (f(\mathbf{m}_i) - f(\mathbf{m}_j))^2 \]  
(6.4)

where \( v_{ij} \) is the weight on edge \([\mathbf{m}_i, \mathbf{m}_j]\). In triangles \([\mathbf{m}_i, \mathbf{m}_j, \mathbf{m}_k]\) and \([\mathbf{m}_i, \mathbf{m}_j, \mathbf{m}_h]\), if we define \( \theta_k \) and \( \theta_h \) are the corner angles with \( \mathbf{m}_k \) and \( \mathbf{m}_h \) as the apex individually, the weight \( v_{ij} \) can be computed as:

\[ v_{ij} = \frac{\cot \theta_k + \cot \theta_h}{2} \]  
(6.5)

If the edge \([\mathbf{m}_i, \mathbf{m}_j]\) is a boundary edge adjacent only to \([\mathbf{m}_i, \mathbf{m}_j, \mathbf{m}_k]\), then the weight becomes \( v_{ij} = \cot \theta_k/2 \). Therefore, for all the internal vertex, which are not on the boundary edges, the Laplace equation becomes a linear system:

\[ \sum v_{ij} (f(\mathbf{m}_i) - f(\mathbf{m}_j)) = 0 \]  
(6.6)

### 6.6.2 Method

Our quad-mesh based hippocampal analysis framework was developed by Kevin H. Hobbs and me. It consists of three major components as follows.

**Step 1. Quad-mesh generation**
We generate volume meshes of hippocampal masks using software ITK and CGAL in the first step. Then, Fiedler vectors (Fiedler, 1975) are computed according to the Gaussian weighted Laplacian matrices of generated volume meshes. Fiedler vector is the eigenvector corresponding to the second smallest eigenvalue. It presents the natural ordering of the vertex for a given mesh. An example of the Fiedler vector for a hippocampus is shown in Fig. 6.7. We define *Prime Meridian* as the curve which connects Fiedler extremes on a surface, shown as the green curve in Fig. 6.7. With Fiedler extremes and Prime Meridian, our proposed quad-meshes are generated by mapping hippocampal surfaces to a common needle cylinder along latitude and longitude directions respectively. The mapping is produced by a parameterization minimizing the harmonic energy in Eqn. (6.4) on the hippocampal triangular mesh.

![Figure 6.7: An example of the Fiedler vector for a hippocampus. Warmer color stands for higher value.](image)

Two harmonic functions specifying the parameterizations along latitude and longitude are applied respectively. For all interior vertex, they satisfy the discrete Laplace equation
and are both obtained by solving discrete Laplace equations with the boundary conditions Eqn. (6.6). The level sets of the final parameterizations are shown in Fig. 6.8.

Figure 6.8: An example of the parameterizations for a hippocampus.

**Step 2. Radial distances (RD) construction and group alignment**

The central line of a 3D surface is computed by connecting the mean positions of latitude level sets of the surface. We define *Radial Distance* in our model as the distance of each surface vertex to the mean position of the same level set. The center line and radial distance are illustrated in Fig. 6.9 (a) and (b). The group hippocampal alignment is performed by a 2D image registration between the extended RD images of subjects and template. The extended RD images are generated by gluing three RD images of the same object side by side to simulate periodic boundary conditions.

**Step 3. RD feature generation**

Our RD features are generated through a patch extraction procedure after the group alignment. A grid-wise t-test is first performed based on the group labels, i.e., MCI vs. normal controls (NC). The patches extracted from a hippocampal surface are sorted in
ascending order according to the mean p values of them. The first 10 patches are selected in a greedy solution. Finally, the average RD values in the selected patches are used as our RD features.

6.6.3 Experiments and Results

The data used in the experiments were obtained from the ADNI database. As our study is longitudinal in nature, we selected all the subjects for whom the baseline (M0) and 12-month follow-up information (M12), including Hippocampus masks, are available. As a result, 317 subjects were selected: 86 patients with AD, 115 with MCI and 116 normal controls (NC). The image modalities include T1-weighted baseline and follow-up MP-RAGE, Hippocampal Mask, MIDAS Whole Brain Mask and Total Intracranial Volume Brain Mask.

The effectiveness of our RD/RDA biomarkers is evaluated in this section, through two binary classification problems: AD vs. NC, and MCI vs. NC. The performance of various features is evaluated based on three measures: classification accuracy (ACC), sensitivity
(SEN) and specificity (SPE). In the end, we also compare our method with three related hippocampal biomarkers (Colliot et al., 2008; Chupin et al., 2009; Gerardin et al., 2009), which were also evaluated using ADNI dataset.

6.6.3.1 RD and RDA group averages and ROI

We conducted statistical analysis to assess the RD/RDA regions of interest between groups. Multi-resolution patches extracted in section 6.6.2 capture the major areas where the groups significantly differ. According to the statistical results, AD group has significantly ($p < 0.01$) smaller RDs almost everywhere across the hippocampus. AD group also have significantly ($p < 0.01$) higher atrophy rates in the medial side of the head and body and along the lateral side of the hippocampi.

6.6.3.2 RD/RDA ROI patches as discriminative features

To investigate the efficacy of RD/RDA biomarkers in distinguishing AD and MCI from normal controls, we evaluated four types of features, “RD+RDA Average”, “RD ROI”, “RDA ROI” and “RD+RDA ROI” based on three performance measures, ACC, SEN, and SPE. Linear support vector machine (SVM) is utilized as the classifier. To better compare the classification performance, we run each experiment 100 times with different random 3-fold splits (two folds for training, one fold for testing).

The classification results, averaging over the 100 runs, are summarized in Table 1. It is clear that combination of features through wrapper selection leads to improved classification performance over the single feature types: “RD+RDA ROI” feature has higher classification accuracy than both “RD ROI” and “RDA ROI”. It can also be observed that RD features are generally more indicative than RDA, especially for AD/NC. The “RD ROI” feature outperforms both “RD+RDA Average” and “RDA ROI”. A possible explanation is that the RD features of the AD patients, though estimated at baseline, may
bear the accumulated hippocampal atrophy for several years. The RDA features, on the other hand, record the atrophy for only 12 months.

<table>
<thead>
<tr>
<th>Feature</th>
<th>AD versus NC</th>
<th>MCI versus NC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACC (%)</td>
<td>SEN (%)</td>
</tr>
<tr>
<td>RD+RDA-Avg</td>
<td>78.35</td>
<td>66.59</td>
</tr>
<tr>
<td>RD ROI</td>
<td>84.27</td>
<td>74.05</td>
</tr>
<tr>
<td>RDA ROI</td>
<td>74.83</td>
<td>55.76</td>
</tr>
<tr>
<td>RD+RDA ROI</td>
<td><strong>85.89</strong></td>
<td><strong>79.13</strong></td>
</tr>
</tbody>
</table>

Table 6.8: Performance comparison of various RD/RDA features for AD/MCI/NC classifications. Boldface denotes the best performance for each measure.

<table>
<thead>
<tr>
<th>Method</th>
<th>Feature</th>
<th>AD versus NC</th>
<th>MCI versus NC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACC</td>
<td>SEN</td>
<td>SPE</td>
</tr>
<tr>
<td>Colliot et al.</td>
<td>Hippo Volume</td>
<td>73</td>
<td>63</td>
</tr>
<tr>
<td>Chupin et al.</td>
<td>Hippo Volume</td>
<td>72</td>
<td>71</td>
</tr>
<tr>
<td>Gerardin et al.</td>
<td>Hippo Shape</td>
<td>78</td>
<td>69</td>
</tr>
<tr>
<td>Proposed method</td>
<td>RD/RDA patches</td>
<td><strong>85.89</strong></td>
<td><strong>79.13</strong></td>
</tr>
</tbody>
</table>

Table 6.9: Comparison of the proposed method with other existing hippocampus methods for AD/MCI classifications.

Cuingnet et al. (Cuingnet et al., 2011) evaluated ten AD/MCI biomarker solutions using common ADNI dataset, among which three methods are based on hippocampal volumes and/or shapes (Colliot et al., 2008; Chupin et al., 2009; Gerardin et al., 2009). The classification results for AD/MCI/NC are listed in Table 6.9. Compared with them, our RD+RDA ROI biomarker achieves much higher accuracies for both AD/NC and MCI/NC.
classifications. While direct comparisons of the methods are not feasible, as different subjects, data and classifiers were used, the high accuracies from our model nevertheless can be regarded as an indirect evidence for the power of the proposed radial distance and atrophy biomarkers.
7 Conclusions & Perspectives

In this dissertation, we proposed our CPD-LapSVM and CpD-UML metric learning methods under the semi-supervised and unsupervised settings in light of the limitations and drawbacks of existing metric learning solutions in the two areas. Toward achieving the desired properties to overcome the existing limitations and drawbacks, we developed our theory carefully in each contribution. The proposed algorithms were validated using both synthetic and real-world datasets. The synthetic datasets illustrated the desired properties in an intuitive way. The real world datasets showed the effectiveness of our algorithms in the statistic. In addition, we also applied our proposed approaches on the Alzheimer’s disease early prediction task in the field of neuroimaging. The improved performance in the task further demonstrated the validity of our methods.

Firstly, we proposed the Coherent point drifting (CPD) based nonlinear metric learning algorithm for semi-supervised learning. The choice of CPD in the transformation is with the following considerations: The first is its capability of generating high-order yet smooth transformations; The second is larger smoothness weights can be assigned to labeled data points under CPD, allowing them to exert more significant influences in SSL. The proposed model is designed to learn a smooth nonlinear deformation under the feature spaces that transforms the input instances to have a better linear separability in LapSVM. The nonlinear transformation achieves the better linear separability by pulling the ”similar” points closer, and at the same time, pushing the ”dissimilar” points far away from each other. An EM-like minimization strategy is adopted in our model to search for the optimal values for the parameters iteratively. Our solution is named as CPD-LapSVM since Laplacian SVM (LapSVM) is applied as the base algorithm for our method. However, it should be noted that our framework has broad applicability, and it can be integrated with many other SSL classifiers than LapSVM.
Our second contribution is to improve the unsupervised metric learning via the same nonlinear feature space transformation strategy. \( k \)-means clustering is adopted as our base clustering algorithm. Our strategy integrates nonlinear feature transformation and manifold embedding together to improve the data separability for \( k \)-means clustering. The same CPD deformation model is applied to the transformation. Our proposed fully nonlinear UML solution enhances data separability through the combination of CPD-driven deformation and spectral embeddings. In this work, we derived a closed-form solution for the CPD optimization which can be solved efficiently. To the best of our knowledge, this is the first work that utilizes dense, spatial varying deformations in unsupervised metric learning.

In the last work, after being validated using both synthetic and real-world datasets, our proposed semi-supervised and unsupervised metric learning solutions are applied to improve the performance in the early prediction of Alzheimer’s disease. Accurate identification of patients with Mild Cognitive Impairment (MCI) at high risk for conversion to Alzheimer’s disease (AD) offers an opportunity to target the disease process early. Using baseline MR images from the Alzheimer’s Disease Neuroimaging Initiative (ADNI) database, we evaluate the effectiveness of the proposed frameworks and demonstrate the improvements over the state-of-the-art solutions within the same category. To best of our knowledge, our work achieved the best performance in accuracy among the methods using baseline MRIs only. Also, we applied our CPD-UML model to AD/MCI diagnosis, and evaluate its effectiveness through three clustering problems: binary clustering of AD vs. Normal Control (NC), progressive MCI (pMCI) vs. stable MCI (sMCI) and ternary clustering of AD/MCI/NC. Experimental results demonstrate that our model outperforms other state-of-the-art methods in the same category, and the improvements over the baseline \( k \)-means clustering are sable and significant. In addition, inspired by the additional information provided in the longitudinal datasets, we developed a new surfaces analysis model to compute the radial distance and atrophies of hippocampus over time. The region-
of-interest (ROI) patches extracted from the radial distance and atrophy have been proved effective as discriminative structural features for AD vs. MCI diagnosis.

In our future work, we can improve our proposed models along the following two directions: first, add mechanisms to decide their parameters automatically; second, combine the models with more base SSL and UML algorithms. Automatic parameter calculation will make our model more robust and efficient. Furthermore, under unsupervised setting, our method without parameters would have wider application. We proved that our CPD-lapSVM can be integrated with many other SSL classifiers. However, the CPD-UML model is derived only based on $k$-means. Combining the nonlinear CPD deformation with other clustering methods is the other direction for our future work. Application-wise, we plan to utilize longitudinal information into the Alzheimer’s disease early prediction problem. The proposed radial distance and atrophies features can be added to the baseline features to achieve better application performance.


664–672).


Holub, A. (n.d.). Exploiting unlabelled data for hybrid object classification..


and intelligent laboratory systems, 2(1-3), 37–52.


