Improved Distance Metric Learning for Nearest Neighbor Classification

Wang, Jun

This publication URL: https://archive-ouverte.unige.ch/unige:76989

Publication DOI: 10.13097/archive-ouverte/unige:76989

UNIVERSITÉ DE GENÈVE FACULTÉ DES SCIENCES

Département d'informatique

Professeur Christian Pellegrini
Docteur Alexandros Kalousis

Improved Distance Metric Learning for Nearest Neighbor Classification

Thèse

présentée à la Faculté des sciences de l'Université de Genève
pour obtenir le grade de Docteur ès sciences, mention informatique

par

Jun Wang

de

Chine

Thèse N^o 4837

GENÈVE
2015

UNIVERSITÉ DE GENÈVE

FACULTÉ DES SCIENCES

Doctorat ès sciences
Mention informatique

Thèse de *Monsieur Jun WANG*

intitulée :

"Improved Distance Metric Learning for Nearest Neighbor Classification"

La Faculté des sciences, sur le préavis de Monsieur C. PELLEGRINI, professeur honoraire et directeur de thèse (Département d'informatique), Monsieur S. MARCHAND-MAILLET, professeur associé et codirecteur de thèse (Département d'informatique), Monsieur A. KALOUSIS, docteur et codirecteur de thèse (Département d'informatique), Monsieur F. SHA, professeur (Department of Computer Science, University of Southern California, Los Angeles, California, United States of America) et Monsieur M. BRONSTEIN, professeur (Faculty of Informatics, University of Lugano, Switzerland), autorise l'impression de la présente thèse, sans exprimer d'opinion sur les propositions qui y sont énoncées.

Genève, le 2 juillet 2015

Thèse - 4837 -

Le Décanat [illegible]

N.B. - La thèse doit porter la déclaration précédente et remplir les conditions énumérées dans les "Informations relatives aux thèses de doctorat à l'Université de Genève".

Remerciement

Je remercie specialement mes superviseurs Alexandros Kalousis et Christian Pellegrini. Sans eux, je ne me serais pas aventuré dans le monde magnifique de l'apprentisage automatique. Je me rappelle toujours des discussions inspirantes que nous avons eux. Ces discussions ont profondement aiguisé ma pensée de recherche.

Je remercie aussi Stephane Marchand-Maillet, Melanie Hilario, Fei Sha, Phong Nguyen, Adam Woznica, Huyen Do, Pablo Strasser, Magda Gregorova et Ke Sun pour des discussions utiles sur nos idées farvelue. Sans eux, la recherche aurait été plus difficile.

De plus, je remercie mon support financier de recherche du Fonds national suisse de la recherche scientifique et du CUI, Université de Genève.

Finalement, je dédie cette thèse ma famille, ma femme, Xiaoyu Shi, et mon fils Steven.

Résumé

L'apprentissage automatique concerne l'apprentissage de modèles prédictifs partir de donnée. Pour bien generaliser, beaucoup d'algorithme d'apprentissage automatique suppose que les prédictions des points proches sont similaires. En conséquence, comment mesurer la proximitée des points de données devient donc un élément crucial d'algorithm d'apprentissage. L'apprentissage de métrique résout se défi en apprenant une métrique dépendantes des données. Pendant les dix dernière années, ceci a attiré beaucoup d'attention et est devenu une des approches principales pour adresser des problèmes d'apprentissage de distance/similarité.

L'approche la plus étudiée est l'apprentissage de métrique individuel. Cela apprend une mesure de distance dans tout l'espace d'entrée des données. Cette approche est souvent éfficace computationelement. Cependant, il peut ne pas etre suffisament flexible pour bien apprendre la distance dans different voisinage de donnée. Tel que, la distance dans different voisinage peut tre bien approximée. L'apprentissage de métrique non linéaire suit une approche differente pour augmenter la complexitée du modèle. Il envoie d'abord les points de donnée dans un nouvel espace de donnée par une application non linéaire d'attribut puis une métrique individuel est apprise dans le nouvel espace de donnée.

Malgré le succès de differents algorithmes d'apprentissage de métrique, chaque algorithme possède ses propres limitations. Dans cette thèse, nous nous concentrons dans leur limitations et developpons de nouveau algorithme d'apprenttisage de métrique. Quatre different types d'algorithme d'apprentissage de métrique ont été contribué, incluant un algorithme d'apprentissage de métrique individuel, un algorithme d'apprentissage de métrique local et deux algorithmes d'apprentissages non linéaire. Les algorithmes d'apprentissage de métrique developpé sont meilleur que les méthodes de l'état de l'art en terme de performance predictive et de scalabilitées algorithmique.

Abstract

Machine learning is about learning predictive models from data. To generalize well, many machine learning algorithms assume that the predictions of the near by data points are similar. Therefore, how to measure the closeness of data points becomes crucial in these learning algorithms. Metric learning solves this challenge by learning a data-dependent metric. Over the last ten years, it attracted a lot of attentions and becomes one of the main approaches to addressing the distance/similarity learning problem.

The most studied approach is single metric learning. It learns one distance metric in the whole input data space. This approaches is often computational efficient. However, it might not be flexible enough to learn well the distance in different data neighborhoods. This motivated local metric learning and nonlinear metric learning. In local metric learning approach, the algorithm learns one distance metric in a neighborhood. Such that, distance in different neighborhoods can be well fitted. The nonlinear metric learning follows a different approach to increase the model complexity. It first maps data points into a new data space through a nonlinear feature mapping and then a single metric is learned in the new data space.

Despite the success of various metric learning algorithms, each algorithm comes with its own set of limitations. In this thesis, we focus on their limitations and develop novel metric learning algorithms. We focus on metric learning for Nearest Neighbor (NN) classification; NN is one of the most well-known algorithms depending strongly on the distance metric. Four different of metric learning algorithms are contributed, including one single metric learning algorithm, one local metric learning algorithm and two nonlinear metric learning algorithms. The developed metric learning algorithms improved over the state-of-the-arts in terms of predictive performance and algorithmic scalability.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning is about learning models from data. Its target is to generalize its learned good predictive performance on the training data, the training error of the model, to that of unseen data, the true error of the model. In doing so, it relies on a number of fundamental assumptions. Probably, the most important one is that the given data points are assumed to be independent and identically distributed (i.i.d.). As a result of this assumption, the difference between the true error of the learned model and the training error is bounded in terms of the model space complexity; the bound is monotonically increasing with the model space complexity. Another important assumption is the **smoothness assumption**– the near by data points are assumed to have similar predictions. As a result of the smoothness assumption, the complexity of the model space and thus the error bound of learned models is reduced. The true error of the learned model is close to the training error. However, how to measure the closeness of data points is one of open questions in learning algorithms.

Often the Euclidean metric is the default choice. However, it may easily harm the performance of learning algorithms, since the smoothness assumption on the data might not always hold for Euclidean distance. A better way is learning a **data-dependent distance metric**, such that the learned distance respects the smoothness assumption. Metric learning follows exactly this idea. It has become a very active research field over the last decade (23; 32; 37; 62; 64; 69; 83; 105; 107).

Most of the existing distance metric learning algorithms can be described as a two-stage process. In the first stage, data points are mapped into a new space by learning a mapping function–representation learning. In the second stage, the distance is computed or learned in the new data space–distance learning. With different instantiations of the two-stage process, different distance metric learning algorithms can be divided into three categories.

The most studied and simplest form of metric learning is single metric learning (23; 32; 83; 105; 107). This approach first learns a linear function in the representation learning stage followed by a default distance computation, e.g. Euclidean distance, in the new space. Since the discriminatory power of the linear projection is limited, this approach is often not flexible enough to learn well the distance in different regions.

The second category is local metric learning which learns in each neighborhood one local metric (56; 73). In this approach, there is no learning process in the first stage. (The mapping function is the identity function). Its distance metric is learned by learning the local metrics in the original data space. When the local metrics vary smoothly in the data space, learning local metrics is equivalent to learning the Riemannian metric on the data manifold. Since the number of parameters of local metric learning is much larger than that of single metric learning, this approach can be prone to overfitting.

The last category is nonlinear metric learning. It can be divided into two main approaches, kernel-based methods (16; 46; 64) and deep neural network based methods (19; 66; 79). In kernelized metric learning, data points are first mapped into the Reproducing-Kernel Hilbert Space (RKHS) by a kernel function and then a global Mahalanobis metric is learned in the RKHS space. There are two limitations in this approach. The first one is that the kernel function most often is selected by cross-validation from a predefined set of kernel functions. This limits the choice of mapping function in the representation learning stage. The second one is that the kernel function must be Positive Semi-Definite (PSD). This limits the application domain of this approach, since not all learning problems can naturally construct a kernel function.

In the approach of deep neural network based metric learning, data points are first mapped into new space through a learned deep neural network and then followed by a default distance computation in the new data space (19; 66; 79). The main limitation of this approach is that often large number of data points are required to fit well the deep neural network.

1.1 Main Contributions

This thesis focuses on learning distance metrics for nearest neighbor (NN) classification. It contributes a number of new metric learning algorithms for NN classification in different categories, including single metric learning (103), local metric learning (100) and nonlinear metric learning (99; 102), which go beyond the state of the art. More precisely, it contains:

1. A novel single metric learning method, described in Chapter 4, Learning Neighbor-

hood Metric Learning (LNML), where we learn the Mahalanobis metric as well as the target neighborhoods. Our method outperforms the state of the art single metric learning methods in terms of predictive accuracy (103).

2. A novel Multiple Kernel Learning (MKL) framework for metric learning, described in Chapter 5, Metric Learning with Multiple Kernels (MKML), where we bring the MKL framework from the context of SVMs into metric learning. Similar to MKL in SVMs, our MKL method in metric learning achieves comparable empirical results to kernelized metric learning where the best kernel is selected by cross validation, while our method is computationally more efficient (99).

3. A novel local metric learning method, described in Chapter 6, Parametric Local Metric Learning (PLML), in which we learn a smooth metric tensor function over the data manifold. Using an approximation error bound of the metric tensor function, we learn local metrics as linear combinations of basis metrics defined on anchor points over different regions of the data space. Our metric learning method has excellent performance both in terms of predictive power and scalability (100).

4. A novel two-stage metric learning method, described in Chapter 7, Similarity-Based Fisher Information Metric Learning (SBFIML), where we first map data points onto finite discrete distributions by computing their similarities to a set of anchor points. Then, we define the distance in the input data space as the Fisher information distance in the associated statistical manifold. In contrast to kernel function in KML which must be PSD, the similarity function used in our SBFIML does not need to be PSD. It can even be asymmetric. In terms of predictive power, our method outperforms significantly other state-of-the-art metric learning methods (102).

1.2 Outline of the Thesis

The thesis is organized as follows. In chapter 2, Background, we cover the basic concept of distance metric and its fundamental role in different machine learning algorithms. It also reviews the state-of-the-art metric learning methods in different learning paradigms, namely supervised, semi-supervised and unsupervised learning paradigms. Chapter 3 reviews the state-of-the-art metric learning methods for NN classification. This is the most studied distance metric learning setting over the last years. In chapter 4, we describe our single metric learning algorithm, Learning Neighborhood Metric Learning. Chapter 5 presents the MKL framework for metric learning methods. In chapter 6 and 7, we present our contributions in local metric learning and two-stages metric learning respectively. Finally, in chapter 8, we conclude this thesis and discuss some future directions of metric learning. Note that, some remarks may be repeated over chapters so that each chapter is self-contained.

1.3 Published Work

This thesis is mainly based on the following publications.

- **Jun Wang**, Adam Woznica, Alexandros Kalousis. "Learning Neighborhoods for Metric Learning." In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2012.
- **Jun Wang**, Huyen Do, Adam Woznica, Alexandros Kalousis. "Metric Learning with Multiple Kernels." In *Advances in Neural Information Processing Systems*, 2011.
- **Jun Wang**, Alexandros Kalousis, Adam Woznica. "Parametric Local Metric Learning for Nearest Neighbor Classification." In *Advances in Neural Information Processing Systems*, 2012.
- **Jun Wang**, Ke Sun, Fei Sha, Stephane.Marchand-Maillet, Alexandros Kalousis. "Two-Stage Metric Learning". In *International Conference on Machine Learning*, 2014.
- Phong Nguyen, **Jun Wang**, Melanie Hilario, Alexandros Kalousis. "Learning Heterogeneous Similarity Measures for Hybrid-Recommendations in Meta-Mining". In *12th IEEE International Conference on Data Mining*, 2012.
- Huyen Do, Alexandros Kalousis, **Jun Wang**, Adam Woznica. "A metric learning perspective of SVM: on the relation of LMNN and SVM". In *Journal of Machine Learning Research - Proceedings Track 22: Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, 2012.

Chapter 2

Background

In this chapter, we present the definition of distance metric and its role in the smoothness assumption of machine learning algorithms. The critical role of distance metric in this assumption serves as the main motivation of the development of distance metric learning. We then review a number of different machine learning algorithms and see how distance metric learning can improve their predictive performances.

2.1 Distance Metrics

We first give the definition of a distance metric.

2.1.1 Definition

Definition 1. *A distance metric on a set $\mathcal{X}$ is a pairwise function, $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers. For all points $\mathbf{x}, \mathbf{x}', \mathbf{x}'' \in \mathcal{X}$, this function satisfies the following properties:*

1. $d(\mathbf{x}, \mathbf{x}') \geq 0$ *(non-negativity)*
2. $d(\mathbf{x}, \mathbf{x}') = 0$*, if and only if* $\mathbf{x} = \mathbf{x}'$ *(identity of indiscernibles)*
3. $d(\mathbf{x}, \mathbf{x}') = d(\mathbf{x}', \mathbf{x})$ *(symmetry)*
4. $d(\mathbf{x}, \mathbf{x}'') \leq d(\mathbf{x}, \mathbf{x}') + d(\mathbf{x}', \mathbf{x}'')$ *(triangle inequality).*

In distance metric learning, we often learn a pseudo-metric which satisifes all the conditions except condition 2. Learning a pseudo-metric is useful for regularizing the model complexity by eliminating irrelevant and redundant data subspaces (14; 61).

In the following section we present some of the most studied distance metrics.

2.1.2 Examples

Euclidean distance. Most probably, Euclidean distance is the most popular distance metric used in different problems. Its definition is given by:

$$d_2(\mathbf{x}, \mathbf{x}') = (\sum_{i=1}^{d} (x_i - x'_i)^2)^{\frac{1}{2}} \tag{2.1}$$

We can see that the Euclidean distance is the L_2 norm of the difference of the two vectors x and x'. From a machine learning point of view is assumes that all features are independent and equally important. These two properties lead to two generalizations of Euclidean distance, namely Minkowski and Mahalanobis distances.

Minkowski distances. Minkowski distances generalize the Euclidean distance through the use of L_p norm, where $(p \geq 1)$. They are defined as:

$$d_p(\mathbf{x}, \mathbf{x}') = (\sum_{i=1}^{d} (x_i - x'_i)^p)^{\frac{1}{p}} \tag{2.2}$$

Mahalanobis distance. The Mahalanobis distance generalizes the Euclidean distance by taking into consideration the correlation and importance of features, parametrized by a Positive Semi-Definite (PSD) matrix $\mathbf{M} \succeq 0$. Its definition is

$$d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = ((x - x')^T \mathbf{M}(x - x'))^{\frac{1}{2}} \tag{2.3}$$

Since matrix $\mathbf{M}$ can be decomposed into $\mathbf{M} = \mathbf{L}^T\mathbf{L}$, it can also be rewritten as

$$d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = ((\mathbf{Lx} - \mathbf{Lx}')^T (\mathbf{Lx} - \mathbf{Lx}'))^{\frac{1}{2}}, \tag{2.4}$$

Over the last years, the Mahalanobis distance has been extensively studied in distance metric learning (23; 32; 62; 105; 107). The Mahalanobis distance also has a close relation with the multivariate normal distribution.

Riemannian Distance. The Riemannian Distance generalizes the distance definition in the Euclidean space into the Riemannian manifold. In the following, we will briefly introduce its definition. More details can be found in the monograph (58).

Let $\mathcal{M}^d$ be a connected differential manifold of dimension d. A Riemannian metric g on $\mathcal{M}^d$ is a smooth function that defines the inner products of tangent vectors at any point $p \in \mathcal{M}^d$.

$$g_p : \mathcal{T}_p\mathcal{M}^d \times \mathcal{T}_p\mathcal{M}^d \to R$$

where $\mathcal{T}_p\mathcal{M}^d$ is the tangent space at p. Given a Riemannian metric g, let $c : [a, b]$ be a parametrized curve on $\mathcal{M}^d$. The length of c is defined as

$$L_a^b(c) := \int_a^b \sqrt{g(\bigtriangledown c(t), \bigtriangledown c(t))}\, \mathrm{d}t$$

where the $\bigtriangledown c(t)$ is the derivative of $c(t)$ at t.

The Riemannian distance between any two points x and x' on $\mathcal{M}^d$ is defined as

$$d(\mathbf{x}, \mathbf{x}') = \inf_{\gamma \in \Gamma} L(\gamma)$$

where Γ is the set of all curves that begins at $\mathbf{x}$ and ends at $\mathbf{x}'$.

The main difficulty about Riemannian distance is that its exact computation is often very expensive. It is approximated in various ways in Riemannian metric learing algorithms (73).

In the following section, we give a brief introduction of machine learning and its two fundamental assumptions. And then we will discuss the role of distance metric in various machine learning algorithms in section 2.3.

2.2 Machine Learning

Machine learning is about automatically learning predictive models from data. Its goal is to predict well on future unseen data. For example, one can learn a model from emails to categorize spam and non-spam email messages. The goal of the learned model is to classify new email into spam and non-spam folders as accurately as possible.

2.2.1 Types of Machines Learning

We can roughly divide the machine learning paradigms into supervised learning, semi-supervised learning (15), unsupervised learning and reinforcement learning (28). In this thesis, we will mainly focus on the first three learning paradigms.

Supervised Learning. In supervised learning, we are given a number of independent and identically distributed (i.i.d.) random learning pairs $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, generated according to some unknown distribution $p(\mathbf{x}, y), \mathbf{x} \in \mathcal{X}, y \in \mathcal{Y}$, where $\mathbf{x}$ is a learning instance in the $\mathcal{X}$ input data space and y is the associated output in the $\mathcal{Y}$ output space.

The target of supervised learning is to learn from some hypothesis space $\mathcal{H}$ a model h which minimizes the expected loss

$$\mathbb{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y)}(L(h(\mathbf{x}), y))$$

where $h(\mathbf{x})$ is the output of the model h given an instance $\mathbf{x}$ and L is a loss function which incurs a penalty if $h(x) \neq y$.

Most often, the input data spaces $\mathcal{X}$ is a d-dimensional Euclidean space $\mathbb{R}^d$ in which the learning instances are represented by d-dimensional Euclidean vectors. The $\mathcal{Y}$ output space varies depending in the different learning task. It can be a set of class labels $\{1, \dots, c\}$ in the classification task or the real numbers $\mathbb{R}$ in the regression task, or even more complex structures such as trees, graphs in structured output prediction.

The choice of loss function also depends on the type of learning task. For example, one of the appropriate loss functions L for classification task is the "$0-1$" loss, defined by

$$L(h(\mathbf{x}), y) = \begin{cases} 1 & h(x) \neq y \\ 0 & h(x) = y \end{cases}$$

It measures the classification error of the given model h.

Semi-Supervised Learning. Getting fully labeled data often involves a lot of expensive human effort in many learning problems. However, it is very often easy to get large amounts of unlabeled data. Semi-supervised learning is motivated by the availability of large amounts of and uses them to improve the predictive performance. The improvement strongly depends on the semi-supervised smoothness assumption: **if two points $\mathbf{x}$ and $\mathbf{x}'$ in a high-density region are close, then so should be the corresponding outputs y and y'** (15). If this assumption does not hold, semi-supervised learning will not yield an improvement over supervised learning. It might even harm the prediction accuracy. For more details on this assumption, please refer to (15).

As a result, in the setting of semi-supervised learning, in addition to a number of i.i.d. labeled learning pairs $\mathcal{L} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$ generated according to a distribution $p(\mathbf{x}, y)$, we are given a number of i.i.d. learning instances $\mathcal{U} = \{\mathbf{x}_{l+1}, \ldots, \mathbf{x}_n\}$ from the marginal distribution $p(\mathbf{x})$. Similar to supervised learning, the target of semi-supervised learning is to learn a model h which minimizes the expected loss

$$\mathbb{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y)}(L(h(\mathbf{x}), y))$$

Unsupervised Learning. In the setting of unsupervised learning, we are only given a number of i.i.d. learning instances $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. The target of unsupervised learning is to find "interesting patterns" in the data . Similar to supervised learning process, we learn from some hypothesis space $\mathcal{H}$ a model h which minimizes the expected loss

$$\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}L(h(\mathbf{x}))$$

where $h(\mathbf{x})$ is the output of the model h given an instance $\mathbf{x}$.

The output $h(\mathbf{x})$ can be the cluster labels in a clustering task, the new representation of the learning instance $\mathbf{x}$ in dimensionality reduction or its density in density estimation. Since in this setting there is no supervised signal, L is often a heuristic loss function. As a result, this is a much less well-defined problem comparing to supervised and semi-supervised learning paradigms.

2.2.2 Assumptions in Machine Learning

There is no free lunch in machine learning (28). Given finite learning instances, machine learning must make assumptions on the data in order to predict well on future unseen data. Two of these assumptions are the i.i.d. assumption on the generation process of learning instances and the smoothness assumption on the predictions of close data points.

I.I.D. Assumption. The i.i.d. assumption assumes that the given data are independent and identically distributed according to an unknown distribution, p. This is the most basic assumption in machine learning. Without this, the model learned from data cannot generalize on future unseen data. Under the i.i.d. assumption we can derive generalization error bounds such as the Probably Approximately Correct (PAC) bound which bounds the generalization error with the training error and the VC dimension of the hypothesis space (94).

Theorem 1. *Let $\mathcal{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ be n pairs of i.i.d. learning instances drawn from some distribution $p(\mathbf{x}, y)$, $\mathcal{H}$ a continuous model space with VC dimension $VC(\mathcal{H})$, L is a loss function and $\delta > 0$. For any $h \in \mathcal{H}$, with probability $1 - \delta$, we have:*

$$\mathbb{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y)}(L(h(\mathbf{x}), y)) \leq \sum_{(\mathbf{x},y)\in\mathcal{T}} L(h(\mathbf{x}), y) + \sqrt{\frac{VC(\mathcal{H})(\ln\frac{2n}{VC(\mathcal{H})} + 1) + \ln\frac{4}{\delta}}{n}} \tag{2.5}$$

Theorem 1 indicates that with high probability any model h can indeed generalize in the order of $O(\sqrt{\frac{\ln(n)}{n}})$ based on the i.i.d. assumption on the given data. However, in order to achieve small expected loss, we should minimize the sum of the empirical loss and the model complexity in terms of the VC dimension of the model space.

Smoothness Assumption. The smoothness assumptions states that: **if two points $\mathbf{x}$ and $\mathbf{x}'$ are close, then so should be the corresponding outputs y and y'.** This essentially assumes that the true model h^* of the data gives similar predictions to learning instances which are close to each other. As a result, the prediction of h^* cannot vary too much locally. Without this assumption, the complexity of the model space $\mathcal{H}$ containing h^* would be very large; as a result the learned model cannot generalize well with "limited" data.

On the grounds that smoothness assumption strongly depends on the distance metric (similarity measure) employed in learning algorithms, we now discuss the role of distance metric in machine learning.

2.3 Distance Metrics in Machine Learning

The smoothness assumption states that close learning instances should have similar predictions. However, as described in section 2.1, there is a very large number of distance metrics that can be used to measure the distance. This leads to two questions.

1. Which distance metric should be used to measure the distance of learning instances?

2. If the smoothness assumption does not hold with one distance metric, how can we discover/learn distance metrics for which the smoothness assumption holds?

The answers to these two questions depend on one feature of the learning algorithms – whether their designs intrinsically assume a distance metric or not. If the learning algorithms do, in this case the intrinsic metric must be used. For instance, in kernel-based methods, the the Euclidean distance is intrinsically used due to the 'kernel trick' (20; 26; 80). In order to improve these learning algorithms, we can learn a better feature space $\mathcal{X}'$ mapped from $\mathcal{X}$; the new feature space fits better the smoothness assumption. This gives rise to the research field of kernel/representation learning (7).

In contrast, if the learning algorithms do not intrinsically assume a distance metric, in this case the distance metric can be whichever metric we choose. Two popular examples of these algorithms are k-NN classification and k-Means clustering. There are two approaches to improve the smoothness assumption. First, as we do in the previous case, we can learn a better data space $\mathcal{X}'$ through representation learning. Second, we can also learn a better distance metric on the original data space $\mathcal{X}$. Note that, these two approaches can be used together.

We now describe in detail the role of distance metric in a number of classic machine learning algorithms. Latter in this chapter, we present the corresponding distance metric learning algorithms to improve the fit to the smoothness assumption.

2.3.1 Supervised Learning

k-Nearest Neighbor Classification. The k-Nearest Neighbor (NN) classifier is one of the simplest and most classical non-linear classification algorithms (28). It is a non-parametric method.

In k-NN classification, the posterior distribution of a given instance is defined based on its k closest training instances in the data space $\mathcal{X}$, given by

$$p(y|\mathbf{x}) = \frac{\sum_{\mathbf{x}' \in kNei(\mathbf{x})} \mathbb{I}_y(y')}{\sum_{y \in \mathcal{Y}} \sum_{\mathbf{x}' \in kNei(\mathbf{x})} \mathbb{I}_y(y')} \tag{2.6}$$

where $kNei(\mathbf{x})$ is the set of k closest learning instances of $\mathbf{x}$. The function $\mathbb{I}_y(y')$ is the indicator function. Its output is $\mathbb{I}_y(y') = 1$, if $y = y'$, otherwise 0. Following the decision rule of maximum a posteriori (MAP), an instance is classified into the most common class amongst its k nearest neighbors.

Theoretically, 1-NN is guaranteed to yield an error no worse than twice the Bayes error as the number of instances approaches infinity. However, with finite learning instances, its performance strongly depends on the use of an appropriate distance metric. k-NN does not assume an intrinsic distance metric. Consequently, in order to improve its performance, we can learn a better distance metric and (or) learn a better representation of the learning instances.

Learning with Kernels. Kernel-based learning methods have become one of the most popular methods in machine learning over the last two decades (20; 26; 80). In kernel-based learning methods, learning instances are mapped into the Reproducing-Kernel Hilbert Space (RKHS) through an implicit feature map function induced by a kernel function and then a model is learned in the RKHS space.

Consider a mapping $\phi(\mathbf{x})$ of instances $\mathbf{x}$ to some feature space $\mathcal{H}$, i.e. $\phi : \mathbf{x} \to \phi(\mathbf{x}) \in \mathcal{H}$. The dimensionality of $\mathcal{H}$ can be infinite. The power of kernel-based methods rely on the so called "kernel trick". To compute the inner product of two learning instances in $\mathcal{H}$, $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$, kernel methods do not necessarily compute the new representations $\phi(\mathbf{x}_i)$. There exists an associated kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ which computes exactly the inner product of two instances in the $\mathcal{H}$ feature space, i.e. $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$. Based on this "kernel trick", kernel-based methods can implicitly map learning instances into a high dimensional space with low computational complexity. One of most often used kernel functions is the Gaussian kernel function. It is defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{d_2^2(\mathbf{x}_i, \mathbf{x}_j)}{\sigma^2}) \tag{2.7}$$

where $d_2^2(\mathbf{x}_i, \mathbf{x}_j)$ is the squared Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ and σ is the kernel width. The dimensionality of its associated $\mathcal{H}$ space is infinite.

However, as kernel functions can only compute the inner product of instances in some $\mathcal{H}$ space, kernel-based methods thus are designed based on the Euclidean Geometry, i.e. the Euclidean distance is implicitly used in these methods according to

$$d_2^2(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_i) \rangle + \langle \phi(\mathbf{x}_j), \phi(\mathbf{x}_j) \rangle - 2\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$$

In this case, in order to improve the performance of kernel methods, we can only learn a better representation of $\mathcal{X}$. This can be done by learning a new representation of $\mathbf{x} \in \mathcal{X}$, representation learning, or learning a kernel function, kernel learning (4; 35). The latter approach implicitly learns the representation induced by the kernel function.

SVM. One of the most famous kernel machines is Support Vector Machine (SVM). It learns a linear model in some $\mathcal{H}$ space by maximizing the Euclidean margin between two classes (20). It solves the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \|\mathbf{w}\|_2^2 + \sum_i \xi_i \\ s.t \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \forall i \\ & \xi_i \geq 0, \forall i \end{aligned} \tag{2.8}$$

where $\mathbf{w}$ is the linear model in the $\mathcal{H}$ space and $y_i \in \{-1, 1\}$ is the class label of $\mathbf{x}_i$. To see the 'kernel trick' clearly, the problem (2.8) should be reformulated into its dual form,

$$\begin{aligned} \min_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \\ s.t \quad & \sum_i \alpha_i y_i = 0 \\ & \alpha_i \geq 0, \forall i \end{aligned} \tag{2.9}$$

As we see in (2.9), only the inner products of the learning instances, $k(\mathbf{x}_i, \mathbf{x}_j)$, are computed to solve the optimization problem.

After learning α, the threshold b can be obtained by solving the following linear system,

$$\bar{\mathbf{K}}\alpha + \mathbf{y}b = \mathbf{1} \tag{2.10}$$

where $\bar{\mathbf{K}}$ is a n by n matrix, whose (i, j) entry is $\bar{K}_{ij} = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$.

According to Karush–Kuhn–Tucker (KKT) conditions, the optimal solution of linear model $\mathbf{w}$ in $\mathcal{H}$ space is

$$\mathbf{w} = \sum_i \alpha_i y_i \phi(\mathbf{x}_i) \tag{2.11}$$

and thus its binary prediction for a given instance $\mathbf{x}$ is

$$h(\mathbf{x}) = sign(\sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b) \tag{2.12}$$

where the function $sign(a)$ is the signum function, whose output is 1, 0, or -1, corresponding to $a > 0$, $a = 0$, and $a < 0$.

2.3.2 Semi-supervised Learning

The success of semi-supervised learning strongly depends on the semi-supervised smoothness assumption. A prominent category of semi-supervised learning algorithms are graph-based learning algorithms (15).

Graph-Based Semi-Supervised Learning. In these learning algorithms, a similarity graph is built before learning. Its nodes correspond to both the labeled and unlabeled learning instances. Its edges encode similarities between learning instances. Essentially, this graph can be seen as a tool of encoding the neighborhood structure of all instances, such that close learning instances in high-density region are densely connected on the similarity graph. The basic idea of graph-based learning algorithm is to propagate label information through this predefined graph in order to label all the unlabeled learning instances. Various graph-based learning algorithms have been proposed the last decade in the literature (33; 49; 115; 117).

Very often the labeling (learning) process of graph-based learning algorithms is casted as a quadratic minimization problem. For example, in the binary classification algorithm of Harmonic Energy Minimization (HEM) (117), its quadratic optimization function is defined as

$$\min_{\mathbf{f}} \quad \sum_{ij} S_{ij}(f_i - f_j)^2 \tag{2.13}$$

where f_i is the soft label, if instance $\mathbf{x}_i$ is unlabeled, $\mathbf{x}_i \in \mathcal{U}$, otherwise, f_i is constrained to be equal to the given label y_i, $f_i = y_i \in \{0, 1\}$. The $S_{ij} \geq 0$ is the similarity between instances $\mathbf{x}_i$ and $\mathbf{x}_j$. Typically it is set by the Gaussian kernel of equation 2.7.

Setting the derivative of equation (2.13) to 0, the closed form solution of $\mathbf{f}_{\mathcal{U}}$ is obtained as

$$\mathbf{f}_{\mathcal{U}} = (\mathbf{I} - \mathbf{P}_{\mathcal{U}\mathcal{U}})^{-1}\mathbf{P}_{\mathcal{U}\mathcal{L}}\mathbf{f}_{\mathcal{L}} \tag{2.14}$$

where

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{\mathcal{L}\mathcal{L}} & \mathbf{P}_{\mathcal{L}\mathcal{U}} \\ \mathbf{P}_{\mathcal{U}\mathcal{L}} & \mathbf{P}_{\mathcal{U}\mathcal{U}} \end{bmatrix} := \mathbf{D}^{-1}\mathbf{S}.$$

$\mathbf{D}$ is a diagonal matrix, whose entries $D_{ii} = \sum_j S_{ij}$.

Since $y_i \in \{0, 1\}$, it can be shown that the soft labels f_i are bounded by $[0, 1]$, i.e. $0 \leq f_i \leq 1$. To predict the label of unlabeled instances, we can simply threshold the soft labels $\mathbf{f}_{\mathcal{U}}$ at 0.5 into binary class labels. [1]

By solving the optimization problem (2.13), the HEM algorithm predicts similar labels to instances with large S_{ij} similarity. However, its performance depends on the similarity measure of the graph. To improve its performance, we can learn a better similarity metric for graph-based learning algorithms.

2.3.3 Unsupervised Learning

Clustering is one of the most popular unsupervised learning paradigms (28). Its target is to partition n learning instances into k clusters, such that the (dis-)similarity betwen instances within the same cluster is large(small). Obviously, the performance of clustering algorithms relies on the (dis-)similarity measures used in the algorithms.

k-Means Clustering. k-means clustering is one of the most classic clustering methods in machine learning (28). It partitions instances into k clusters by minimizing the following objective.

$$\begin{aligned} \min_{\mathbf{Y},\mathbf{C}} \quad & \sum_{j=1}^{k}\sum_{i=1}^{n} Y_{ij} d_2^2(\mathbf{x}_i, \mathbf{c}_j) = \sum_{j=1}^{k}\sum_{i=1}^{n} Y_{ij} \|\mathbf{x}_i - \mathbf{c}_j\|_2^2 = \|\mathbf{X} - \mathbf{Y}\mathbf{C}\|_F^2 \\ s.t. \quad & \sum_{j} Y_{ij} = 1, Y_{ij} \in \{0,1\} \end{aligned} \tag{2.15}$$

where the vector $\mathbf{c}_j$ is the center of jth cluster. $Y_{ij} \in \{0,1\}$ is the cluster membership indicator, $Y_{ij} = 1$ if instance $\mathbf{x}_i$ belongs to the jth cluster, otherwise $Y_{ij} = 0$. It can be solved by an iterative algorithm optimizing alternatively $\mathbf{Y}$ and $\mathbf{C}$.

Spectral Relaxation of k-Means (54). We now present an another reformulation of problem (2.15) which will be used in the section 2.4.3 of distance metric learning for clustering. It is based on the fact that the optimal solution for the $\mathbf{C}$ matrix for a fixed $\mathbf{Y}$ is $\mathbf{C} = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{X}$.

Let $\mathbf{B} = \mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T$, we have the following reformulation of optimization problem 2.15,

$$\begin{aligned} \min_{\mathbf{B}} \quad & \|\mathbf{X} - \mathbf{B}\mathbf{X}\|_F^2 = tr(\mathbf{X}\mathbf{X}^T(\mathbf{I} - \mathbf{B})) \\ s.t. \quad & \mathbf{B} \in \mathcal{B}_k \end{aligned} \tag{2.16}$$

where the feasible set $\mathcal{B}_k$ of matrix $\mathbf{B}$ is specified by two constraints. The first constraint

[1] Note that, this strategy could fail when the number of instances is unbalanced in different classes. In (117), the authors proposed Class Mass Normalization (CMN) to considers the prior information on class ratio.

specifies that there exists an assignment matrix $\mathbf{Y}$ such that $\mathbf{B} = \mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T$. The second constraint is $tr(\mathbf{B}) = k$, the trace of matrix B is equal to the number of cluster, k.

Since the matrix $\mathbf{B}$ satisfies $\mathbf{B}^T\mathbf{B} = \mathbf{B}$, the problem (2.16) can be further rewritten as

$$\begin{aligned} \max_{\mathbf{B}} \quad & tr(\mathbf{X}\mathbf{X}^T\mathbf{B}) \\ s.t. \quad & \mathbf{B} \in \mathcal{B}_k \end{aligned} \tag{2.17}$$

With this new formulation, we can solve the problem (2.17) based on the classic idea of spectral relaxation, i.e. maximizing its upper bound,

$$\max_{\mathbf{B}\in\mathcal{B}_k} \quad tr(\mathbf{X}\mathbf{X}^T\mathbf{B}) \leq \max_{\mathbf{B}^T\mathbf{B}=\mathbf{B},tr(\mathbf{B})=k} tr(\mathbf{X}\mathbf{X}^T\mathbf{B}) \tag{2.18}$$

The matrix $\mathbf{B}$ of problem (2.18) has a closed form solution, $\mathbf{B} = \mathbf{U}\mathbf{U}^T$, where the matrix $\mathbf{U}$ is a $n \times k$ matrix whose columns are the k largest eigenvectors of matrix $\mathbf{X}\mathbf{X}^T$. The final cluster membership of learning instances can be heuristically obtained by applying k-means algorithm on matrix $\mathbf{U}$, the k largest eigenvectors.

Note that, the inner product matrix $\mathbf{X}\mathbf{X}^T$ can be replaced with a general kernel matrix, $\mathbf{K}$, e.g. Gaussian kernel in equation (2.7).

Spectral Clustering (96). In addition to k-Means clustering, spectral clustering is another important and popular clustering algorithm extensively studied over the last decade(96). It separates learning instances into different clusters according to their pair-wise similarity matrix, $\mathbf{S}$, which is very often defined by Gaussian kernel, equation (2.7). There are two types of spectral clustering, unnormalized (96) and normalized (68; 88).

The unnormalized spectral clustering is a spectral relaxation of the RatioCut minimization problem (96), defined as

$$\min RatioCut(\mathcal{C}_1, \cdots, \mathcal{C}_k) := \frac{1}{2}\sum_{i=1}^{k} \frac{W(\mathcal{C}_i, \bar{\mathcal{C}}_i)}{|\mathcal{C}_i|} \tag{2.19}$$

where $|\mathcal{C}|$ is the number of instances in cluster $\mathcal{C}$ and the cut function is defined as $W(\mathcal{C}, \bar{\mathcal{C}}) = \sum_{\mathbf{x}_i \in C, \mathbf{x}_j \notin \mathcal{C}} S_{ij}$.

The minimization of RatioCut can be further reformulated as the following discrete optimization problem (96),

$$\begin{aligned} \min_{(\mathcal{C}_1,\cdots,\mathcal{C}_k)} \quad & tr(\mathbf{H}^T\mathbf{L}\mathbf{H}) \\ s.t. \quad & \mathbf{H}^T\mathbf{H} = \mathbf{I} \\ & H_{ij} = \begin{cases} \frac{1}{\sqrt{|\mathcal{C}_j|}} & \mathbf{x}_i \in \mathcal{C}_j \\ 0 & \mathbf{x}_i \notin \mathcal{C}_j \end{cases} \end{aligned} \tag{2.20}$$

where matrix $\mathbf{L}$ is the Laplacian matrix. It is defined by $\mathbf{L} = \mathbf{D} - \mathbf{S}$, where $\mathbf{D}$ is a diagonal matrix, whose entries $D_{ii} = \sum_j S_{ij}$.

This optimization problem is NP-hard. The unnormalized spectral clustering relaxes this problem by removing the second constraint in problem (2.20) and the optimization problem now becomes

$$\begin{aligned} \min_{\mathbf{H}} \quad & tr(\mathbf{H}^T\mathbf{L}\mathbf{H}) \\ s.t. \quad & \mathbf{H}^T\mathbf{H} = \mathbf{I} \end{aligned} \tag{2.21}$$

where $\mathbf{H}$ is now a real value matrix with closed form solution $\mathbf{H} = \mathbf{U}$, where $\mathbf{U}$ is of size $n \times k$. Its columns are the k smallest eigenvectors of matrix $\mathbf{L}$. After computing $\mathbf{H}$, the cluster membership of learning instances is heuristically obtained by considering each row of matrix $\mathbf{H}$ as new representation of the learning instances and applying k-means algorithm on them. The work of (50) provides a theoretical foundation of this heuristic.

Normalized Spectral Clustering (88). The normalized spectral clustering is the spectral relaxation of the minimization of the Normalized cuts (Ncut) problem, defined as

$$\min NCut(\mathcal{C}_1, \cdots, \mathcal{C}_k) := \frac{1}{2}\sum_{i=1}^{k} \frac{W(\mathcal{C}_i, \bar{\mathcal{C}}_i)}{vol(\mathcal{C}_i)} \tag{2.22}$$

where $vol(\mathcal{C}) = \sum_{\mathbf{x}_i \in \mathcal{C}, \mathbf{x}_i \neq \mathbf{x}_j} S_{ij}$ is the volume function. Since in Ncut the cut function is normalized by the volumn of each instance, Ncut is a more appropriate measure than RatioCut if the volume of the different instances differs significantly. Similar to the minimization of RatioCut, the minimization of Ncut can be formulated as a discrete optimization problem (96),

$$\begin{aligned} \min_{(\mathcal{C}_1,\cdots,\mathcal{C}_k)} \quad & tr(\mathbf{H}^T\mathbf{L}\mathbf{H}) \\ s.t. \quad & \mathbf{H}^T\mathbf{D}\mathbf{H} = \mathbf{I} \\ & H_{ij} = \begin{cases} \frac{1}{\sqrt{vol(\mathcal{C}_j)}} & \mathbf{x}_i \in \mathcal{C}_j \\ 0 & \mathbf{x}_i \notin \mathcal{C}_j \end{cases} \end{aligned} \tag{2.23}$$

where the diagonal matrix $\mathbf{D}$ is exactly the same as the $\mathbf{D}$ in RatioCut, problem (2.20).

By allowing H to be a real matrix and removing the second constraint of problem (2.23), the normalized spectral clustering solves the following optimization,

$$\begin{aligned} \min_{\mathbf{H}} \quad & tr(\mathbf{H}^T\mathbf{L}\mathbf{H}) \\ s.t. \quad & \mathbf{H}^T\mathbf{D}\mathbf{H} = \mathbf{I} \end{aligned} \tag{2.24}$$

where the solution is $\mathbf{H} = \mathbf{U}$, the columns of which are the k smallest generalized eigenvectors of problem $\mathbf{L}\mathbf{u} = \lambda\mathbf{D}\mathbf{u}$.

By seting $\mathbf{Q} = \mathbf{D}^{\frac{1}{2}}\mathbf{H}$, the problem (2.24) is also equivalent to

$$\begin{aligned} \min_{\mathbf{Q}} \quad & tr(\mathbf{Q}^T\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\mathbf{Q}) \\ = \max_{\mathbf{Q}} \quad & tr(\mathbf{Q}^T\mathbf{D}^{-\frac{1}{2}}\mathbf{S}\mathbf{D}^{-\frac{1}{2}}\mathbf{Q}) \\ s.t. \quad & \mathbf{Q}^T\mathbf{Q} = \mathbf{I} \end{aligned} \tag{2.25}$$

where the solution of $\mathbf{Q}$ consists of the k largest eigenvector of matrix $\mathbf{D}^{-\frac{1}{2}}\mathbf{S}\mathbf{D}^{-\frac{1}{2}}$ and $\mathbf{H} = \mathbf{D}^{-\frac{1}{2}}\mathbf{Q}$.

In order to improve the performance of different clustering algorithms, we could learn a better distance metric in problem (2.15) or similarity matrix in problems (2.18), (2.21) and (2.25).

2.4 Distance Metric Learning

In the section of 2.3, we describes the critical role of distance metric (similarity measure) in different learning paradigms. However, despite of its importance, very often only the

Euclidean distance metric is used in various learning algorithms. This harms the performance of learning algorithms, if the Euclidean distance metric is not appropriate for the learning problem. In this section, we present different distance metric learning algorithms that lift this limitation by learning a "data dependent" distance metric. We start with distance metric learning for supervised learning.

2.4.1 Distance Metric Learning for Supervised Learning

In this section we will only present a distance metric learning algorithm for SVMs(108). We will discuss the bulk of the distance metric learning work that took place in the nearest neighbor setting in chapter 3.

SVML (108). Support Vector Metric Learning (SVML) learns a Mahalanobis distance based Gaussian kernel for SVM. It is motivated by the fact that very often the σ parameter in equation (2.7) is selected by Cross Validation (CV). The authors propose to learn the Mahalanobis metric by minimizing the classification error of SVM on a hold-out dataset, an alternative way for parameter selection.

The Gaussian kernel equipped with Mahalanobis metric in (108) is defined as

$$k_{\mathbf{L}}(\mathbf{x}_i, \mathbf{x}_j) = exp(-(\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_j)^T(\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_j)) \tag{2.26}$$

and the learning optimization problem is the following:

$$\min_{\mathbf{L}} \sum_{\mathbf{x}, y \in \mathcal{V}} s_a(yh(\mathbf{x})) + \lambda \|\mathbf{L} - \mathbf{L}_0\|_F^2 \tag{2.27}$$

where the function s_a is a soft approximation of "0-1" loss function, defined as

$$s_a(z) = \frac{1}{1 + exp(az)}$$

The parameter a controls the steepness of the s_a function. $\mathcal{V}$ is the validation dataset. Matrix $\mathbf{L}_0$ serves as a regularization parameter which penalizes the learned projection matrix $\mathbf{L}$ deviating from $\mathbf{L}_0$; in (108), it is set to the initial estimate of $\mathbf{L}$. $h(x)$ is the prediction function learned by SVM with Gaussian kernel $k_{\mathbf{L}}$, equation (2.26), on the training dataset $\mathcal{T}$. It can be obtained by solving the problems (2.9) and (2.10) with $k_{\mathbf{L}}$. The whole optimization problem (2.27) is solved with conjugate gradient descent (72).

Compared to the best Gaussian kernel selected by CV, the Gaussian kernel with Mahalanobis distance is much more powerful because it considers the correlation of input features. In (108), the authors empirically demonstrate that SVML achieves better predictive performance than SVM with best Gaussian kernel selected by CV.

2.4.2 Distance Metric Learning for Semi-Supervised Learning

In this section, we present two distance metric learning algorithms which learn the weight of input features for HEM, described in section 2.3.2. This is equivalent to learning a diagonal Mahalanobis distance metric.

EntMin (117). The approach of Entropy Minimization (EntMin) learns the weights of the input features by minimizing the entropy of unlabeled data. It is motivated by the intuition that a good distance metric should produce a confident labeling, i.e. all the unlabeled instances should have high predictive probability of belongin in of the two classes.

Since the solution of the soft label f_i is bounded in $[0, 1]$, the predictive probability of an unlabeled instance $p(y_i|\mathbf{x}_i)$ can be reasonably defined as

$$p(y_i|\mathbf{x}_i) = \begin{cases} f_i & y_i = 1 \\ 1 - f_i & y_i = 0 \end{cases}$$

Based on the definition of predictive probability, the optimization problem of learning the weights, $\mathbf{w}$, the input features is given:

$$\min_{\mathbf{w}} -\sum_{i=1}^{|\mathcal{U}|}(f_{\mathcal{U}_i} \log f_{\mathcal{U}_i} + (1 - f_{\mathcal{U}_i}) \log(1 - f_{\mathcal{U}_i})) \tag{2.28}$$

where $\mathbf{f}_{\mathcal{U}}$ is a function of $\mathbf{w}$, defined in equation (2.14). This optimization problem is solved by gradient descend in (117). The main drawback of this approach is that the idea of entropy minimization is rather a heuristic. It does not directly optimize the classification error.

LOOHL (116). The approach of Leave-One-Out Hyper-parameter Learning (LOOHL) learns the feature weights $\mathbf{w}$ by minimizing the leave-one-out error on labeled data. According to the closed form solution of $\mathbf{f}_{\mathcal{U}}$, equation (2.14), the soft label f_v of a validation instance $\mathbf{x}_v$ is computed by

$$f_v = \mathbf{s}^T \mathbf{f}_{\mathcal{U}}^v$$

where

$$\mathbf{s} = [1, 0, \cdots, 0]^T$$

and

$$\mathbf{f}^v_{\mathcal{U}} = (\mathbf{I} - \mathbf{P}^v_{\mathcal{U}\mathcal{U}})^{-1}\mathbf{P}^v_{\mathcal{U}\bar{\mathcal{L}}}\mathbf{f}_{\bar{\mathcal{L}}}$$

The set $\bar{\mathcal{L}}$ includes all the training instances except $\mathbf{x}_v$. Accordingly, $\mathbf{P}^v_{\mathcal{U}\mathcal{U}}$ is defined as

$$\mathbf{P}^v_{\mathcal{U}\mathcal{U}} = \begin{bmatrix} \mathbf{P}_{vv} & \mathbf{P}_{v\mathcal{U}} \\ P_{\mathcal{U}v} & P_{\mathcal{U}\mathcal{U}} \end{bmatrix},$$

and, in a similar manner, $\mathbf{P}^v_{\mathcal{U}\bar{\mathcal{L}}}$ is defined as

$$\mathbf{P}^v_{\mathcal{U}\bar{\mathcal{L}}} = \begin{bmatrix} \mathbf{P}_{v\bar{\mathcal{L}}} \\ \mathcal{P}_{\mathcal{U}\bar{\mathcal{L}}} \end{bmatrix}.$$

Given that, the objective of LOOHL is:

$$\min_{\mathbf{w}} \sum_{v=1}^{|\mathcal{L}|} g_v(f_v) + \lambda \sum_i (\frac{1}{w_i} - \frac{1}{\mu})^2 \tag{2.29}$$

where function g_v is the loss function. In (116), it is defined as

$$g_v(f_v) = \begin{cases} (1 - f_v)^2 & y_v = 1 \\ f_v^2 & y_v = 0 \end{cases}$$

The second term of the objective function (2.29) is a regularization which protects from degenerative solutions; μ is a predefined parameter.

Similar to MinEnt, the optimization problem (2.29) is solved by gradient descent. In (116), the authors empirically show that LOOHL is effective in improving the performance of HEM and outperforms its counterpart MinEnt in a statistically significant manner.

2.4.3 Distance Metric Learning for Unsupervised Learning

In this section, we present three distance metric learning approaches for clustering algorithms, described in section 2.3.3.

Xing et al.(107). Xing's distance metric learning method is one of the earliest Mahalanobis distance learning algorithms. It learns a Mahalanobis distance metric to improve the performance of k-means clustering by utilizing side information, i.e. must-link and cannot-link constraints. Instances subject to a must-link constraint should belong to the same cluster. And instances of a cannot-link constraint are forced into different clusters.

Xing's method is formulated based on a simple heuristic to fit these constraints. Specifically, it minimizes the distance of instances in the must-link constraints while maximizes the distance of instances in the cannot-link constraints. Accordingly, given a set of must-link constraints, $\mathcal{M}$, and a number of cannot-link constraints, $\mathcal{C}$, it learns a Mahalanobis metric, $\mathbf{M}$, by solving the following optimization problem,

$$\begin{aligned} \min_{\mathbf{M}} \quad & \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}} d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \\ s.t. \quad & \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}} d_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \geq 1 \\ & \mathbf{M} \succeq 0 \end{aligned} \tag{2.30}$$

where $d_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j)$ is the Mahalanobis distance defined in equation (2.3). This optimization problem is convex and thus has a global solution. Note that, in order to avoid a trivial solution for $\mathbf{M}$, the Mahalanobis distance instead of its square is used in the first constraint.

The authors propose an iterative projection algorithm to solve this optimization problem (2.30). After distance metric learning, the learned Mahalanobis distance, $d_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j)$, is plugged into k-means clustering algorithm, equation (2.15). In (107), the authors demonstrate empirically that the learned metrics can be used to significantly improve clustering performance. Its main drawback is that this method is based on a simple heuristic without considering in its problem formulation the error of clustering algorithm.

Bach & Jordan (50). The work of Bach & Jordan learns a distance metric for normalized spectral clustering, problem (2.25). More precisely, it learns the weight of the input features of the Gaussian kernel, equation (2.7), by minimizing a error between the true clustering partition and the solution of normalized spectral clustering.

Compared to Xing's method (107), this approach requires complete supervised information for all training instances. At first glance, this kind of fully supervised setting may seem not be practical for clustering algorithms. However, for instance, in image segmentation tasks, where spectral clustering algorithms are often applied to segment images (88), a number of fully labeled images are indeed easily available.

The error measure in (50) is defined based on the following observations. To obtain the true clustering assignment matrix $\mathbf{Y}$ of size n by k, the target solution $\mathbf{Q}^*$ of problem (2.25) ideally should satisfy the following two constraints, 1) there exists a real matrix $\mathbf{\Lambda}$ such that $\mathbf{D}^{-\frac{1}{2}}\mathbf{Q}^* = \mathbf{Y}\mathbf{\Lambda}$, and 2) $\mathbf{Q}^{*T}\mathbf{Q}^* = \mathbf{I}$. Furthermore, note that the eigenspace induced by the eigenvectors $\mathbf{Q}$ is rotation invariant. Therefore, the error measure in (88) is defined on the orthogonal projection operators, $\mathbf{Q}\mathbf{Q}^T$, as

$$J(\mathbf{S},\mathbf{E}) = \frac{1}{2}\|\mathbf{Q}\mathbf{Q}^T - \mathbf{Q}^*\mathbf{Q}^{*T}\|_F^2 \tag{2.31}$$

where $\mathbf{Q}^*\mathbf{Q}^{*T} = \mathbf{D}^{\frac{1}{2}}\mathbf{Y}\mathbf{\Lambda}\mathbf{\Lambda}^T\mathbf{Y}^T\mathbf{D}^{\frac{1}{2}} = \mathbf{D}^{\frac{1}{2}}\mathbf{Y}(\mathbf{Y}^T\mathbf{D}\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{D}^{\frac{1}{2}}$. It is easy to verify that $\mathbf{\Lambda}\mathbf{\Lambda}^T = (\mathbf{Y}^T\mathbf{D}\mathbf{Y})^{-1}$, since $\hat{\mathbf{Q}}^T\hat{\mathbf{Q}} = \mathbf{I}$. $\mathbf{Q}$ is the solution of normalized spectral clustering, problem (2.25).

Minimizing this error with respect to the similarity matrix $\mathbf{S}$ yields an algorithm for learning the similarity. As a result, given N training datasets and their target partitions, $\{(\mathbf{X}_1,\mathbf{Y}_1),\cdots,(\mathbf{X}_N,\mathbf{Y}_N)\}$, the following optimization problem is proposed to learn the weight of the input features in the Gaussian kernel,

$$\min_{\mathbf{w}} \frac{1}{2}\sum_{i=1}^{N} \|\mathbf{Q}_i\mathbf{Q}_i^T - \mathbf{Q}_i^*\mathbf{Q}_i^{*T}\|_F^2 + \lambda\|\mathbf{w}\|_1 \tag{2.32}$$

where the L_1 regularization on the weight vector $\mathbf{w}$ encourages a sparse solution which is useful to remove irrelevant input features. $\mathbf{Q}_i$ and $\mathbf{Q}_i^*$ are functions of $\mathbf{S}$ and $\mathbf{D}$ and thus functions of $\mathbf{w}$. However, since $\mathbf{Q}$ are the eigenvectors of the matrix $\mathbf{D}^{-\frac{1}{2}}\mathbf{S}\mathbf{D}^{-\frac{1}{2}}$, it is difficult to optimize $\mathbf{Q}\mathbf{Q}^T$. In (50), the authors solve this problem by approximating it based on the power method.

The authors empirically demonstrate on synthetic datasets that learning the weights of the input features is robust to irrelevant input features and improves significantly the performance of normalized spectral clustering.

LMMLP (54). The method of Large Margin Metric Learning for Partitioning (LMMLP) learns a Mahalanobis distance for the spectral relaxation of the k-Means clustering, problem (2.18).

Similar to the work of Bach & Jordan, LMMLP assumes the availability of fully labeled datasets. It learns the distance metric based on the large margin idea following the framework of structural SVM (92).

Specifically, by coupling the spectral relaxation of k-Means clustering, problem (2.18), with Mahalanobis metric learning, we have the following optimization problem,

$$\max_{\mathbf{M},\mathbf{B}^T\mathbf{B}=\mathbf{B},tr(\mathbf{B})=k} tr(\mathbf{X}\mathbf{M}\mathbf{X}^T\mathbf{B}) = tr(\mathbf{M}\mathbf{X}^T\mathbf{B}\mathbf{X}) \tag{2.33}$$

where $\mathbf{M}$ is the Mahalanobis metric that will be learned.

Given N training datasets and their target partitions, $\{(\mathbf{X}_1,\mathbf{Y}_1),\cdots,(\mathbf{X}_N,\mathbf{Y}_N)\}$, following the framework of structural SVM, the distance metric $\mathbf{M}$ is learned by the following optimization problem,

$$\begin{aligned}
\min_{\mathbf{M},\xi} \quad & \Omega(\mathbf{M}) + \lambda \sum_{i=1}^{N} \xi_i && (2.34)\\
s.t. \quad & \xi_i \geq 0 \\
& \xi_i \geq \max_{\mathbf{B}^T\mathbf{B}=\mathbf{B}, tr(\mathbf{B})=k} \delta(\mathbf{B}_i, \mathbf{B}_i^*) + tr(\mathbf{M}\mathbf{X}_i^T\mathbf{B}_i\mathbf{X}_i) - tr(\mathbf{M}\mathbf{X}_i^T\mathbf{B}_i^*\mathbf{X}_i) \\
& \mathbf{M} \succeq 0
\end{aligned}$$

where $\mathbf{B}_i^*$ is the target matrix defined based on true partition matrix $\mathbf{Y}_i$ according to the k-Means problem (2.15), $\mathbf{B}_i^* = \mathbf{Y}_i(\mathbf{Y}_i^T\mathbf{Y}_i)^{-1}\mathbf{Y}_i^T$. The margin $\delta(\mathbf{B}_i, \mathbf{B}_i^*)$ is defined by $\delta(\mathbf{B}_i, \mathbf{B}_i^*) = \|\mathbf{B}_i - \mathbf{B}_i^*\|_F^2$, which corresponds to a distance measure of partitions (54). Its use is mainly motivated by the observation that a closed form solution of $\mathbf{B}_i$ in the second constraint can be efficiently computed with this margin function, which is a classical computational bottleneck in structural SVM. The regularization term of $\Omega(\mathbf{M})$ can be instantiated with different regularizer, e.g. $\|\mathbf{M}\|_F^2$ and $tr(\mathbf{M})$. This problem (2.34) is convex but not smooth. The authors solve it with projected subgradient method.

In addition to learning the Mahalanobis metric $\mathbf{M}$ in problem (2.34), in a similar manner, LMMLP can learn the Mahalanobis metric of Gausian similarity matrix in the following problem,

$$\begin{aligned}
\min_{\mathbf{M},\xi} \quad & \Omega(\mathbf{M}) + \lambda \sum_{i=1}^{N} \xi_i && (2.35)\\
s.t. \quad & \xi_i \geq 0 \\
& \xi_i \geq \max_{\mathbf{B}^T\mathbf{B}=\mathbf{B}, tr(\mathbf{B})=k} \delta(\mathbf{B}_i, \mathbf{B}_i^*) + tr(\mathbf{S}\mathbf{B}_i) - tr(\mathbf{S}\mathbf{B}_i^*) \\
& \mathbf{M} \succeq 0
\end{aligned}$$

where S is the pariwise similarity matrix, whose (i, j) entry is

$$S_{ij} = exp(-(\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{M}(\mathbf{x}_i - \mathbf{x}_j)).$$

Unfortunately, the optimization problem (2.35) is concave-convex. Only a local convergence can be obtained.

The authors experiment with these two approaches, problems (2.34) and (2.35), on a number of datasets, including UCI datasets and image segmentation datasets. The empirical results show that learning the distance metric improves the performance of clustering algorithms.

2.4.4 Discussions

Motivated by the fundamental role of the smoothness assumption in machine learning algorithms, distance metric learning learns a "data dependent" distance metric to fit better the smoothness assumption. Most of the algorithms we reviewed in this section formulate the learning problem by minimizing some approximation of the generalization error of a target learning algorithm. In the learning process, the distance metric and the predictive model are learned jointly. As a result, distance metric learning can be seen as adding one more layer of learned structure on the base learning algorithm. The final model is learned by optimizing the approximation of the generalization error.

2.5 Conclusion

In this chapter, we first presented the definition of distance metric and briefly reviewed different types of machine learning. Then the importance of distance metric for various machine learning algorithms is discussed. The importance of distance metric in the smoothness assumption is the main motivation of the development of distance metric learning over the last decade. We also reviewed a number of distance metric learning algorithms and discuss their advantages and drawbacks. However we omitted probably the most well studied distance metric learning setting that of nearest neighbor classification. In the next chapter we will review these algorithms.

Chapter 3

Metric Learning for Nearest Neighbor Classification

In this chapter, we review distance metric learning for Nearest Neighbor (NN) classification, the most studied distance metric learning algorithms. A considerable volume of theoretical and empirical results have been established in this particular category of distance metric learning algorithm over the last decade. The structure of this chapter is as follows. We first briefly describe the key concepts of distance metric learning for NN in section 3.1. Then, we introduce different types of distance metric learning algorithms, starting from simple linear metric learning in section 3.2, its kernelized counterpart, kernelized metric learning, in section 3.3, and Riemannian metric learning in section 3.4. In section 3.5, we discuss various structural regularizations on distance metric. Finally, we review the works on distance metric learning theory.

3.1 Elements of Distance Metric Learning for NN Classification

We start this chapter with a big picture of distance metric learning for NN classification.

Very often a typical metric learning algorithm, h, learns a pseudo distance metric, d, by solving an optimization problem of the following form:

$$\begin{aligned} \min_{d} \quad & L_h(\mathcal{C}_h(d)) + \Omega_h(d) \\ s.t. \quad & d \text{ is a pseudo distance metric} \end{aligned} \tag{3.1}$$

where L_h is the loss function of the h algorithm, defined based on a set of constraints $\mathcal{C}_h(d)$. L_h can be a simple heuristic to improve the performance of NN or an approximation of classification error of NN. $\Omega_h(d)$ is a regularizer to prevent overfitting or to impose prior knowledge on the distance metric, d.

For instance, the most studied Mahalanobis metric learning algorithm often results in an optimization problem as

$$\begin{aligned} \min_{\mathbf{M}} \quad & L_h(\mathbf{C}_h(\mathbf{M})) + \Omega_h(\mathbf{M}) \\ s.t. \quad & \mathbf{M} \succeq 0 \end{aligned} \tag{3.2}$$

where L_h is a loss function of a set of constraints $\mathcal{C}_h(\mathbf{M})$ parametrized by the Mahalanobis metric $\mathbf{M}$. The regularizer $\Omega_h(\mathbf{M})$ could be the squared Frobenius norm, $\|\mathbf{M}\|_F^2$, the trace norm, $tr(\mathbf{M})$, or the structured sparsity inducing norm (5; 60; 112). These various regularizers will be discussed in section 3.5. The constraint, $\mathbf{M} \succeq 0$, ensures that the learned Mahalanobis distance, equation (2.3), is a pseudo distance.

The loss function L_h varies with different forms of constraints $\mathcal{C}_h(d)$ considered in metric learning algorithms. The two most popular forms of constraints are pairwise similarity and dissimilarity constraints (23; 107) and large margin triplet constraints (105).

3.1.1 Similarity and Dissimilarity Constraints

A pair of instances subject to a similarity constraint should have small distance. On the contrary, instance pair subject to a dissimilarity constraint should have large distance. More precisely, given a set of similarity constraints, $\mathcal{S}$, and a set of dissimilarity constraints,

$\mathcal{D}$, the optimization problem often has the following form:

$$\begin{aligned} \min_{d} \quad & \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S} \text{ or } \mathcal{D}} \xi_{ij} + \Omega_h(d) \\ s.t. \quad & d(\mathbf{x}_i,\mathbf{x}_j) \geq u + \xi_{ij}, \forall(\mathbf{x}_i,\mathbf{x}_j) \in \mathcal{D} \\ & d(\mathbf{x}_i,\mathbf{x}_j) \leq l + \xi_{ij}, \forall(\mathbf{x}_i,\mathbf{x}_j) \in \mathcal{S} \\ & \xi_{ij} \geq 0 \\ & d \text{ is a pseudo distance metric} \end{aligned} \tag{3.3}$$

where u and l are some constants, specifying the thresholds these constraints should satisfy. ξ_{ij} is a slack variable, incurring a penalty if a constraint is violated. The loss function L_h is defined as the sum of these penalties. Note that, to be more flexible, different thresholds u_{ij} and l_{ij} can be used for different pairs.

However, learning with similarity and dissimilarity constraints often leads to a heuristic loss function, since these constraints are not directly related to the NN classification error. These constraints are often used if only side information about instances are available.

3.1.2 Large Margin Triplet Constraints

The large margin triplet constraint originates from the classical large margin idea in machine learning (20). A large margin triplet constraint defined over triplets of instances $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ requires that their distances respect the following relationship,

$$d(\mathbf{x}_i,\mathbf{x}_k) \geq d(\mathbf{x}_i,\mathbf{x}_j) + \gamma,$$

where γ is the predefined margin distance. Similar to the optimization problem (3.3) with similarity and dissimilarity constraints, we can use the triplet constraints to define the following metric learning optimization problem given a set of $\mathcal{L}$ such constraints:

$$\begin{aligned} \min_{d} \quad & \sum_{(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k)\in\mathcal{L}} \xi_{ijk} + \Omega_h(d) \\ s.t. \quad & d(\mathbf{x}_i,\mathbf{x}_k) - d(\mathbf{x}_i,\mathbf{x}_j) \geq \gamma + \xi_{ijk}, \forall(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k) \in \mathcal{L} \\ & \xi_{ijk} \geq 0 \\ & d \text{ is a pseudo distance metric} \end{aligned} \tag{3.4}$$

where ξ_{ijk} is also a slack variable, and γ is often set by $\gamma = 1$. The loss function L_h in this case is the sum of margin violations.

The large margin triplet constraints are often defined based on class label information. For instance, we can define for each instance $\mathbf{x}_i$ a total of $k_1 * k_2$ constraints. These constraints are defined using its k_1 same-class nearest neighbors and its k_2 different-class nearest neighbors and constraining the distance of each instance to its k_2 different class nearest neighbors to be larger than those to its k_1 same class nearest neighbors with margin γ. Note that, the k_1 same-class nearest neighbors of each instance are also called its target neighbors (105). When triplet constraints are defined in such way, the margin violations can be seen as an approximation of the NN classification error. Learning distance metrics with large margin constraints often achieves state-of-the-art NN predictive performance.

In the following sections, we will briefly describe different types of metric learning algorithms always in the context of nearest neighbor classification. We will start from the simplest form of metric learning, linear metric learning in section 3.2; linear in the sense that the mapping that we learn is linear. And then we move to methods that lift the linearity limitation either through the use of kernels, section 3.3, or by learning a number of local metrics, section 3.4. We will also review briefly different regularization methods for linear metric learning, section 3.5, and finally with a brief review of more theoretical work on generalization error bounds for metric learning in section 3.6.

3.2 Linear Metric Learning

Linear metric learning is probably the earliest and most studied metric learning paradigm for kNN classification. Among all different metric lerning paradigms for kNN, it has the simplest form of distance function parametrization. Very often the methods that follow this paradigm learn a distance function of the following form:

$$d_{\mathbf{L}}(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{L}\mathbf{x}_i, \mathbf{L}\mathbf{x}_j) \tag{3.5}$$

The distance computation of equation (3.5) is a two-step process. In the first step, a linear transformation $\mathbf{L}$ is applied on all the learning instance in the input space $\mathcal{X}$. Then a predefined distance metric d is used to compute the distance in the projected space. Note that, the transformation matrix $\mathbf{L}$ is necessarily constrained to ensure the projected instances $\mathbf{L}\mathbf{x}$ being eligible for the distance function d. For instance, the well-studied Mahalanobis distance falls into this category by learning a linear transformation in the Euclidean space.

In the following sections, we review linear metric learning methods motivated from different aspects. We start with the two early works.

3.2.1 Early Approaches

Schultz & Joachims (83). This work learns a restricted pseudo-metric with the following form:

$$d_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{A}\mathbf{W}\mathbf{A}^{\mathbf{T}}(\mathbf{x}_i - \mathbf{x}_j) \tag{3.6}$$

where $\mathbf{A}$ is a predefined linear transformation matrix. $\mathbf{W}$ is a non-negative diagonal matrix that will be learned. Note that, this distance function can be a distance metric in a kernel space if we set $\mathbf{A} = \mathbf{\Phi}(\mathbf{X})$, where $\mathbf{\Phi}(\mathbf{X})$ are the images of the training instances in the kernel space.

With this distance parametrization, the authors proposed to learn the diagonal matrix $\mathbf{W}$ with large margin triplet constraints. This is also one of the earliest works that learns a distance metric using large margin triplet constraints. However, its main drawback is that the distance parametrization, equation 3.6, requires the use of a predefined $\mathbf{A}$ matrix and learns only a diagonal matrix $\mathbf{W}$.

POLA (84). Pseudo-metric Online Learning Algorithm (POLA) is the first online Mahalanobis metric learning algorithm. It learns the distance metric by maintaining a margin between distances of similar and dissimilar instances pairs; the distances of similar instances should have smaller distances than those of dissimilar instances. More specifically, the hinge loss function of POLA is defined as:

$$l(\mathbf{M}, n, \mathbf{x}_i, \mathbf{x}_j, S_{ij}) = \max(0, S_{ij}(d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - b) + 1) \tag{3.7}$$

where $S_{ij} = 1$ if instances $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar, otherwise $S_{ij} = -1$. b is the distance threshold indicating the separation of similar and dissimilar pairs. $\mathbf{M}$ is the Mahalanobis metric. Note that, $\mathbf{M} \succeq 0$ is constrained to ensure the learned distance is a pseudo-metric. It also constrains $b \geq 1$ to ensure the loss being well-defined.

The optimization problem is solved by orthogonal projection. At the kth iteration, given a pair of learning instances, $(\mathbf{x}_i, \mathbf{x}_j)$, POLA first updates $\mathbf{M}$ and b to reduce the loss of $(\mathbf{x}_i, \mathbf{x}_j)$, equation 3.7. This is done by a orthogonal projection which projects parameters $\mathbf{M}^{k-1}$ and b^{k-1} onto the halfspace defined by $S_{ij}(d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - b) + 1 \leq 0$. In the second step, it maintains the feasibility of $\mathbf{M}$ and b by projecting them back to their feasible region, defined by $\mathbf{M} \succeq 0$ and $b \geq 1$.

Note that, even though in POLA there is a margin maintained between distances of similar and dissimilar pairs, this margin definition is not directly related with the NN classification error. It is rather closely related with the definition of the "(dis)similar" concept in the learning problem.

3.2.2 Large Margin Approaches

In this section, we review three large margin linear metric learning algorithms. All of them learn distance metrics with large margin triplet constraints. We start with the work of Large Margin Nearest Neighbor (LMNN).

LMNN (105). LMNN learns a Mahalanobis metric $\mathbf{M}$ by optimizing the following problem:

$$\begin{aligned} \min_{\mathbf{M}} \quad & \sum_{(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k)\in\mathcal{L}} \xi_{ijk} + \mu \sum_{\mathbf{x}_j \in Ne(\mathbf{x}_i), y_i=y_j} d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \\ s.t. \quad & d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_k) - d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \geq 1 + \xi_{ijk}, \forall(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k) \in \mathcal{L} \\ & \xi_{ijk} \geq 0 \\ & \mathbf{M} \succeq 0 \end{aligned} \tag{3.8}$$

where $Ne(\mathbf{x}_i)$ is the k nearest neighbors of learning instance $\mathbf{x}_i$ and $d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j)$ is the squared Mahalanobis distance, defined by

$$d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{M}(\mathbf{x}_i - \mathbf{x}_j).$$

ξ_{ijk} is the slack variable incurring a penalty if the corresponding triplet constraints is violated. The set of triplet constraints $\mathcal{L}$ is defined by $\{(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k) : \mathbf{x}_j \in Ne(\mathbf{x}_i), y_i = y_j, y_i \neq y_k\}$. Note that, $\mathbf{x}_j$ is called the target neighbor of $\mathbf{x}_i$ if $\mathbf{x}_j \in Ne(\mathbf{x}_i), y_i = y_j$.

Intuitively, the problem (3.8) learns a Mahalanobis metric by minimizing the sum of the distances of each instance to its target neighbors while pushing all different class instances out of its target neighborhood. By learning a full Mahalanobis metric, LMNN successfully lifted the limitation of (83), section 3.2.1, and become one of the state-of-the-art Mahalanobis metric learning methods.

MLSVM (69). MLSVM differs slightly from LMNN by solving the following problem:

$$\begin{aligned} \min_{\mathbf{M}} \quad & \sum_i \xi_i + \mu\|\mathbf{M}\|_F^2 \\ s.t. \quad & \min_{\mathbf{x}_k \in Ne(\mathbf{x}_i), y_i \neq y_k} d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_k) - \max_{\mathbf{x}_j \in Ne(\mathbf{x}_i), y_i=y_j} d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \geq 1 + \xi_i \\ & \xi_i \geq 0 \\ & \mathbf{M} \succeq 0 \end{aligned} \tag{3.9}$$

where $\|\cdot\|_F$ is the Frobenius norm. Compared to the margin definition in LMNN, the margin of each instance defined in MLSVM is the margin between its furthest same class nearest neighbors and its nearest different class nearest neighbors. This margin definition is more closer to the margin definition in SVM. Motivated by this fact, the authors solve this problem with Pegasos (85), a SVM QP solver in the primal form. At each iteration, the Positive Semi-Definite property of $\mathbf{M}$ is maintained by an orthogonal projection step which projects $\mathbf{M}$ onto the PSD cone. In terms of predictive performance, the authors show MLSVM achieves slightly better results than that of LMNN.

χ^2-LMNN (53). The two previous works, LMNN and MLSVM, learn the Mahalanobis metric in the Euclidean space. χ^2 LMNN learns the χ^2 distance on the probability simplex. The main motivation of χ^2 LMNN is that there are many histogram datasets in computer vision, where each instance, $\mathbf{x}_i \in \{\mathbf{x}_i : \sum_k x_{ik} = 1, x_{ik} \geq 0, \forall k\}$, lies on the more restricted probability simplex rather than the Euclidean space. To learn a distance for these datasets, it would be better if we use the geometry structure of the simplex. Specifically, χ^2 LMNN learns the a modified χ^2 distance, defined by

$$\chi^2_{\mathbf{L}}(\mathbf{x}_i, \mathbf{x}_j) = \chi^2(\mathbf{L}\mathbf{x}_i, \mathbf{L}\mathbf{x}_j) \tag{3.10}$$

where

$$\chi^2(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2}\sum_k \frac{(x_{ik} - x_{jk})^2}{x_{ik} + x_{jk}}.$$

Furthermore, the constraints $\sum_i L_{ik} = 1, L \geq 0$ are added to ensure the projected learning instance $\mathbf{L}\mathbf{x}_i$ being on the simplex.

With this new distance parametrization on the simplex, χ^2 LMNN learns the χ^2 distance by solving an optimization similar to problem (3.8), where the Mahalanobis distance $d^2_{\mathbf{M}}$ is replaced by χ^2 distance in equation (3.10). By utilizing the geometrical structure of simplex, the authors show that χ^2 LMNN improves significantly the predictive performance for histogram datasets over that of LMNN. Note that, this is one of the first metric learning works that goes beyond distance metric learning in Euclidean space or kernel space.

3.2.3 Stochastic Approaches

In this section, we present metric learning approaches that follow the so-called stochastic approach. The key concept here is that of the stochastic nearest neighbor defined as:

$$p^d(\mathbf{x}_j|\mathbf{x}_i) = \frac{e^{-d^2(\mathbf{x}_i,\mathbf{x}_j)}}{\sum_{k\neq i} e^{-d^2(\mathbf{x}_i,\mathbf{x}_k)}} \tag{3.11}$$

where d is a distance metric. Intuitively, the conditional probability $p^d(\mathbf{x}_j|\mathbf{x}_i)$ can be interpreted as the probability of $\mathbf{x}_j$ being the nearest neighbor of $\mathbf{x}_i$ under the distance metric d.

NCA (34). Neighborhood Component Analysis (NCA) approximates the NN classification error based on the concept of stochastic nearest neighbor. Specifically, NCA defines the probability $p^d(\hat{y}_i = y_i|x_i)$ of $\mathbf{x}_i$ being correctly classified under metric d as:

$$p^d(\hat{y}_i = y_i|x_i) = \sum_{y_i=y_j} p^d(\mathbf{x}_i|\mathbf{x}_j)$$

where $\hat{y}_i$ is the predicted label of $\mathbf{x}_i$. Thus, the approximated NN error can be defined as

$$\sum_i (1 - p^d(\hat{y}_i = y_i|x_i)) \tag{3.12}$$

The optimization problem of NCA is to learn a linear transformation $\mathbf{L}$ by minimizing the approximated NN error, equation 3.12, under the distance metric

$$d^2_{\mathbf{L}}(\mathbf{x}_i,\mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{L}^T\mathbf{L}(\mathbf{x}_i - \mathbf{x}_j)$$

Compared to the large margin approach, section 3.2.2, which approximates the NN error based on predefined target neighborhood (triplet constraints), NCA bypasses need for a predefined neighborhoods by directly approximated NN error using the stochastic nearest neighbor concept. However, the main drawback of NCA is that the computational complexity of stochastic nearest neighbor is $O(n^2)$, where n is the number of instances. Furthermore, the optimization problem is not convex. In contrast, the optimization problem of LMNN is convex with computational complexity $O(n)$.

MCML (32). The algorithm of Maximally Collapsing Metric Learning (MCML) uses the stochastic nearest neighbor in a different manner. MCML learns a distance metric d to match, as well as possible, its induced conditional distribution $p^d(\mathbf{x}_j|\mathbf{x}_i)$ to an ideal distribution, defined as:

$$p_0(\mathbf{x}_j|\mathbf{x}_i) \propto \begin{cases} 1 & y_i = y_j \\ 0 & y_i \neq y_j \end{cases} \tag{3.13}$$

This ideal distribution corresponds to the scenario that all points in the same class were mapped to a single point and infinitely far from points in different classes. Specifically, MCML learns a Mahalanobis metric $\mathbf{M}$ by solving the following problem:

$$\begin{aligned} \min_{\mathbf{M}} \quad & \sum_i KL(p_0(\cdot|\mathbf{x}_i)|p^{d_{\mathbf{M}}}(\cdot|\mathbf{x}_i)) \\ s.t. \quad & \mathbf{M} \succeq 0 \end{aligned} \tag{3.14}$$

where $KL(p_0|p^{d_{\mathbf{M}}})$ is the Kullback-Leibler divergence between ideal distribution p_0 and $p^{d_{\mathbf{M}}}$.

The main advantage of MCML over NCA is that its optimization problem (3.14) is convex. However, the ideal target of MCML is over constrained for NN error minimization. In terms of predictive performance, MCML often performs worse than NCA.

3.2.4 Information Theoretic Approach

The third class of linear metric learning we will review is motivated from information theory. This approach proposes a better way on measuring the distance between Mahalanobis metrics, i.e. PSD matrices, by considering their PSD structure.

ITML (23). The work of Information-Theoretic Metric Learning (ITML) pioneers this approach. It is motivated by the fact that there exists a bijection between the set of Mahalanobis distances and the set of equal mean multivariate Gaussian distributions,

$$f(\mathbf{M}) \to p(\mathbf{x}, \mathbf{M}) = \frac{1}{Z} e^{-\frac{1}{2} d_{\mathbf{M}}(\mathbf{x},\mu)}$$

where $\mathbf{x}$ is the random variable of multivariate Gaussian distribution. μ is the mean of random variable and Z is the normalization constant. With this bijection, the authors propose to measure the distance between two Mahalanobis distances $\mathbf{M}_0$ and $\mathbf{M}$ by the Kullback-Leibler divergence between their corresponding multivariate Guassians $p(\mathbf{x}, \mathbf{M}_0)$ and $p(\mathbf{x}, \mathbf{M})$, defined as:

$$KL(p(\mathbf{x}, \mathbf{M}_0), p(\mathbf{x}, \mathbf{M})) = \frac{1}{2} d_{\text{LogDet}}(\mathbf{M}, \mathbf{M}_0) = tr(\mathbf{M}\mathbf{M}_0^{-1}) - \text{LogDet}(\mathbf{M}\mathbf{M}_0^{-1}) - n$$

where d_{LogDet} is the logdet divergence.

Its optimization problem of learning Mahalanobis metric $\mathbf{M}$ is formulated by minimizing the logdet divergence between $\mathbf{M}$ and a predefined target matrix $\mathbf{M}_0$ subject to a number of similarity and dissimilarity constraints, $\mathcal{S}$ and $\mathcal{D}$,

$$\begin{aligned} \min_{\mathbf{M}} \quad & d_{\text{LogDet}}(\mathbf{M}, \mathbf{M}_0) + \gamma \sum_{ij} \xi_{ij} \\ s.t. \quad & d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \leq l + \xi_{ij}, (i,j) \in \mathcal{S} \\ & d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq u - \xi_{ij}, (i,j) \in \mathcal{D} \end{aligned} \tag{3.15}$$

where very often matrix $\mathbf{M}_0$ is set to identity matrix.

Comparing to previous algorithms, ITML is fast and scalable, since it does not need eigen-decomposition computations or semi-denite programming in LMNN. Furthermore, thanks to the bijection between the multivariate Gaussians and the PSD covariance matrices, the logdet divergence measure also considers the PSD structure of the Mahalanobis metrics. This is missed in other approaches. Note that, the logdet divergence has been recently used in different metric learning algorithms (65; 75; 106).

3.3 Kernelized Metric Learning

In section 3.2, we discussed linear metric learning algorithms which learns a linear transformation in the original data space. However, very often a linear transformation cannot adequately represent the inherent complexities of a problem at hand. To address this limitation various works proposed kernelized versions of metric learning algorithms in order to learn a non-linear distance metric. In this section, we briefly review work falling into this category, starting from the early work.

3.3.1 Early Work

Since the dimensionality of kernel space might be infinite, early works of Kernelized Metric Learning (KML) learn a low rank Mahalanobis metric parametrized in a specific form.

Schultz & Joachims (83). As discussed in section 3.2.1, this work learns a restricted pseudo-metric with the following form:

$$d_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{A}\mathbf{W}\mathbf{A}^{\mathbf{T}} (\mathbf{x}_i - \mathbf{x}_j) \tag{3.16}$$

where $\mathbf{A}$ is a predefined matrix. $\mathbf{W}$ is a diagonal matrix. Under this parametrization, the pseudo-distance of $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ in a kernel space $\mathcal{H}$ can be effectively computed by defining $\mathbf{A} = \Phi(\mathbf{X})$,

$$\begin{aligned} d_{\mathbf{W}}(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)) \quad &= (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^T \boldsymbol{\Phi}(\mathbf{X})\mathbf{W}\boldsymbol{\Phi}(\mathbf{X})^{\mathbf{T}}(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) \\ &= (\mathbf{K}_i - \mathbf{K}_j)^T \mathbf{W}(\mathbf{K}_i - \mathbf{K}_j) \end{aligned} \tag{3.17}$$

where $\mathbf{K}_i$ is the ith row of kernel matrix $\mathbf{K}$, whose (i, j) entry $\mathbf{K}_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$.

With the distance parametrization of equation (3.17), learning the distance in the kernel space can be done in the same manner as learning the distance in the original input data space.

LMCA Large Margin Component Analysis (LMCA) (91) is the kernelized version of LMNN discussed in 3.2.2. LMNN learns an unrestricted Mahalanobis metric in the original input data space, defined as

$$d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j).$$

Similar to the work of Schultz & Joachims (83), LMCA learns a specific form of $\mathbf{M} = \Phi(\mathbf{X})\mathbf{B}\Phi(\mathbf{X})^T$. However, the $\mathbf{B}$ matrix here is a PSD matrix rather than a restricted diagonal matrix. Under this parametrization, the Mahalanobis distance in the kernel space can be defined as:

$$\begin{aligned} d^2_{\mathbf{B}}(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)) \quad &= (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^T \Phi(\mathbf{X})\mathbf{B}\Phi(\mathbf{X})^T(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) \\ &= (\mathbf{K}_i - \mathbf{K}_j)^T \mathbf{B}(\mathbf{K}_i - \mathbf{K}_j) \end{aligned} \tag{3.18}$$

Pluging euqation (3.18) into problem 3.8, LMCA learns a non-linear distance in the kernel space. Note that, the distance parametrization of equation (3.18) has also been used in MCML(32) and MLSVM (69).

ITML (23) The distance parametrization of kernelized ITML is slightly different from equations (3.17) and (3.18). It is motivated by the learning algorithm of ITML itself, i.e. Algorithm 1 in (23). More precisely, in the ITML learning algorithm, the Mahalanobis metric $\mathbf{M}^{k+1}$ at $k + 1$ iteration is updated by

$$\mathbf{M}^{k+1} = \mathbf{M}^k + \beta\mathbf{M}^k(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M}^k$$

where β is a stepsize-like parameter. By unrolling the matrix $\mathbf{M}$, we have

$$\mathbf{M} = \mathbf{M}_0 + \mathbf{M}_0\mathbf{X}\mathbf{B}\mathbf{X}^T\mathbf{M}_0$$

where $\mathbf{B}$ is PSD matrix. $\mathbf{M}_0$ is the predefined target matrix in problem 3.15.

As a result, the Mahalanobis distance metric $\mathbf{M}$ in kernelized ITML is parametrized by

$$\mathbf{M} = \mathbf{M}_0 + \mathbf{M}_0\Phi(\mathbf{X})\mathbf{B}\Phi(\mathbf{X}^T)\mathbf{M}_0$$

where $\mathbf{M}_0$ is the target Mahalanobis distance metric in kernel space, often set by $\mathbf{M}_0 = \mathbf{I}$. The distance in kernel space, if $\mathbf{M}_0 = \mathbf{I}$, is defined as

$$\begin{aligned} d^2_{\mathbf{B}}(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)) &= (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^T(\mathbf{I} + \Phi(\mathbf{X})\mathbf{B}\Phi(\mathbf{X}^T))(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) \\ &= (\mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij}) + (\mathbf{K}_i - \mathbf{K}_j)^T\mathbf{B}(\mathbf{K}_i - \mathbf{K}_j) \end{aligned} \tag{3.19}$$

Note that, there is an equivalence between learning the kernel matrix $\mathbf{K}$ and learning the Mahalanobis distance d in the kernel space, since

$$d^2(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)) = \mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij}$$

In (23), kernelized ITML learns the distance in the kernel space by learning a new kernel matrix $\mathbf{K}$.

3.3.2 KPCA Approach

The early work discussed in section 3.3.1 kernelizes the metric learning algorithms by optimizing a specific form of metric matrix in the kernel space. Depending on their objective functions, different learning algorithms might learn metric matrices of different forms. The work of (16) proposes an alternative way to kernelize the metric learning algorithms. This simple two step approach first applies Kernel PCA on all the learning instances and then it learns a non-linear distance in the kernel space by applying one linear metric learning algorithm on the projected learning instances, i.e. the output of applying Kernel PCA on learning instances.

The authors prove that for any objective function f of a metric learning algorithm, the optimal value of f using as input the learning instances $\phi(\mathbf{x}_i)$ is equal to the optimal value of f using as input the projected learning instances $\mathbf{z}_i$, where $\mathbf{z}_i$ is the new representation of $\phi(\mathbf{x}_i)$ after applying Kernel PCA.

Comparing to the approaches we presented in section 3.3.1, this approach does not require the use of metric matrices of a given structure. Also the code of linear metric learning can be reused.

3.3.3 A Representer Theorem for Metric Learning

Recently, the work of (46) proposed a representer theorem for kernelized metric learning algorithms. It is based on the following problem setting.

Most kernelized metric learning algorithm can be summarized into the following optimization problem,

$$\begin{aligned} \min_{\mathbf{M}} \quad & f(\mathbf{M}) \\ s.t. \quad & \mathbf{M} \succeq 0 \\ & g_i(\Phi(\mathbf{X})^T \mathbf{M} \Phi(\mathbf{X})) \leq b_i, i \in \{1, \cdots, m\} \end{aligned} \tag{3.20}$$

where $\mathbf{M}$ is Mahalanobis matrix in the kernel space, g_i is a function to produce the pairwise or triplet constraints used in metric learning algorithms and f is the objective function.

To state the representer theorem, we need the definition of the spectral function.

Definition 2. *(46) $f : \mathbb{R}^{n \times n} \to \mathbb{R}$ is a spectral function if $f(\mathbf{M}) = \sum_i f_s(\lambda_i)$, where $\lambda_1, \cdots, \lambda_n$ are the eigenvalues of $\mathbf{M}$ and $f_s : \mathbb{R} \to \mathbb{R}$ is a real-value scalar function. Note that, if f_s is a convex scalar function, then f is also convex.*

Now, we can present the following theorem that gives the optimal parametrization form of $\mathbf{M}$, i.e. the representer theorem of $\mathbf{M}$.

Theorem 2. *(46) Assume that f in problem 3.20 is a spectral function and the global minimum of the corresponding strictly convex scalar function f_s is $\alpha > 0$. Let $\mathbf{M}^*$ be an optimal solution of problem 3.20, then we have*

$$\mathbf{M}^* = \alpha \mathbf{I} + \Phi(\mathbf{X}) \mathbf{B} \Phi(\mathbf{X})^T \tag{3.21}$$

where $\mathbf{B}$ is a PSD matrix.

As a result of Theorem 2, most of metric learning algorithms can be easily kernelized. For instance, the optimal solutions of LMNN(105), POLA(84), MCML(32) and MLSVM (69) have a constant $\alpha = 0$.

3.4 Local Metric Learning

To lift the limitation of linear metric learning, kernelized metric learning learns implicitly a non-linear distance metric by learning a linear metric in the kernel space. In this section, we review an alternative approach, Local Metric Learning (Local ML). It directly learns a non-linear distance by learning a linear metric on each neighborhood. When the local metrics vary smoothly in the input data space, then learning them is equivalent to learning the Riemannian metric on the data manifold. As before we start by reviewing the early work that follows this approach.

3.4.1 Discriminative Adaptive Nearest Neighbor

Discriminative Adaptive Nearest Neighbor (DANN) (38) is one of the earliest (local) metric learning algorithms. It extends the idea of Linear Discriminant Analysis (LDA) into local metric learning. More precisely, the local metric $\mathbf{M}_i$ of instance $\mathbf{x}_i$ is iteratively learned by the following equation,

$$\mathbf{M}_i = \mathbf{W}^{-\frac{1}{2}}(\mathbf{W}^{-\frac{1}{2}}\mathbf{B}\mathbf{W}^{-\frac{1}{2}} + \lambda\mathbf{I})\mathbf{W}^{-\frac{1}{2}} \tag{3.22}$$

where $\mathbf{W}$ and $\mathbf{B}$ are the between and within sum-of-squares matrices computed based on the neighborhood of $\mathbf{x}_i$. λ is a regularizer. Note that, at each iteation the neighborhood is updated according to the learned local metric $\mathbf{M}$. Intuitively, the learned local metric $\mathbf{M}_i$ will shrink neighborhoods in directions orthogonal to the local decision boundaries and enlarge the neighborhoods parallel to the boundaries. However, since DANN is in fact a local LDA, it has the main limitations as LDA. Also it learns the local metrics independently with no regularization between them which makes it prone to overfitting.

3.4.2 MM-LMNN

Multiple Metric LMNN (MM-LMNN) is a variant of LMNN discussed in section 3.2.2 (105). Unlike DANN that learns for each instance one local metric, MM-LMNN simplifies the learning problem by learning one local metric for each cluster of learning instances to also address the fact the LMNN is much more computationally expensive compared to LDA.

Concretelly, MM-LMNN learns a number of local metrics $\{\mathbf{M}_1, \cdots, \mathbf{M}_m\}$ by solving the following optimization problem,

$$\begin{aligned} \min_{\mathbf{M}_1,\ldots,\mathbf{M}_m,\xi} \quad & \sum_{ijk} \xi_{ijk} + \mu \sum_{ij} d^2_{\mathbf{M}_{clus(\mathbf{x}_j)}}(\mathbf{x}_i, \mathbf{x}_j) \\ s.t. \quad & d^2_{\mathbf{M}_{clus(\mathbf{x}_k)}}(\mathbf{x}_i, \mathbf{x}_k) - d^2_{\mathbf{M}_{clus(\mathbf{x}_j)}}(\mathbf{x}_i, \mathbf{x}_j)) \geq 1 - \xi_{ijk} \ \forall i, j, k \\ & \xi_{ijk} \geq 0; \ \forall i, j, k \ \mathbf{M}_i \succeq 0; \ \forall i \end{aligned} \tag{3.23}$$

where $clus(\mathbf{x}_j)$ is the index of the cluster in which instance $\mathbf{x}_j$ belongs to. Note that, in

MM-LMNN, the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined by

$$d^2_{\mathbf{M}_{clus(\mathbf{x}_j)}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M}_{clus(\mathbf{x}_j)} (\mathbf{x}_i - \mathbf{x}_j).$$

This 'distance' in fact is not not symmetric, since $\mathbf{M}_{clus(\mathbf{x}_j)}$ might be different from $\mathbf{M}_{clus(\mathbf{x}_i)}$. In (105), the authors show that MM-LMNN successfully improves the predictive performance of LMNN. However, similar to DANN, MM-LMNN learns the metrics independently for each region making it also prone to overfitting since the local metrics will be overly specific to their respective regions.

3.4.3 GLML

Similar to DANN, Generalized Local Metric Learning (GLML) learns for each instance one local metric (73). Its local metric learning is motivated by the minimization of the expected NN classification error under a Multivariate Gaussian assumption for the learning instances of each class. GLML learns the local metric $\mathbf{M}$ of instance $\mathbf{x}$ by solving the following problem,

$$\begin{aligned} \min_{\mathbf{M}} \quad & (tr(\mathbf{M}^{-1}\mathbf{B}))^2 \\ s.t. \quad & |\mathbf{M}_i| = 1 \\ & \mathbf{M}_i \succeq 0 \end{aligned} \tag{3.24}$$

where $|\cdot|$ is the determinant. $\mathbf{B}$ is a matrix defined by

$$\mathbf{B} = \sum_{i=1}^{C} \nabla^2 p_i(\mathbf{x}) (\sum_{j \neq i} p_j^2(\mathbf{x}) - p_i(\mathbf{x}_i) \sum_{j \neq i} p_j(\mathbf{x}))$$

where $p_i(\mathbf{x})$ is the probability density of ith class at $\mathbf{x}$. $\nabla^2 p_i(\mathbf{x})$ is its Hessian matrix at $\mathbf{x}$. In the case of Multivariate Gaussian distribution,

$$\nabla^2 p_i(\mathbf{x}) = p_i(\mathbf{x})(\Sigma_i^{-1}(\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^T \Sigma_i^{-1} - \Sigma_i^{-1})$$

where Σ_i and μ_i are respectively the covariance matrix and mean of density function $p_i(\cdot)$.

GLML theoretically minimizes the expected NN classification error; however, the strong Multivariate Gaussian model assumptions do not hold for many real world learning problems.

3.5 Structured Regularization

The sparsity assumption is one of the most studied assumptions in machine learning over the last years. In this section, we review metric learning algorithms that exploit various sparse patterns on the metric matrix.

3.5.1 Off-Diagonal L_1 Regularization

The work of (75) proposes to learn a Mahalanobis metric with sparse off-diagonal elements. This is motivated by the fact that the off-diagonal elements of the inverse of the covariance matrix in a high dimensional space are often very small and can be ignored. Specifically, it learns the distance metric $\mathbf{M}$ by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{M}} \quad & tr(\mathbf{M}\mathbf{M}_0^{-1}) - \text{LogDet}(\mathbf{M}\mathbf{M}_0^{-1}) + \lambda\|\mathbf{M}\|_{1,\text{off}} + \eta\mathcal{L}(\mathcal{S},\mathcal{D}) \\ s.t. \quad & \mathbf{M} \succeq 0 \end{aligned} \tag{3.25}$$

where $\|\mathbf{M}\|_{1,\text{off}} = \sum_{i\neq j} |M_{ij}|$ is the L_1 off-diagonal regularization, $tr(\mathbf{M}\mathbf{M}_0^{-1}) - \text{LogDet}(\mathbf{M}\mathbf{M}_0^{-1})$ is the logdet divergence following ITML(23), discussed in section 3.2.4, and $\mathcal{L}(\mathcal{S},\mathcal{D})$ is a loss function defined based on similarity and dissimilarity constraints. To simplify the optimization problem, in (75), the authors define

$$\mathcal{L}(\mathcal{S},\mathcal{D}) := \sum_{(i,j)\in\mathcal{S}} d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) - \sum_{(i,j)\in\mathcal{D}} d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j)$$

Finally, the authors propose an efficient block coordinate descent algorithm to solve problem 3.25.

3.5.2 $L_{2,1}$ Regularization

The work of (61) studies the use of $L_{2,1}$ regularization for Mahalanobis distance metric learning. It is motivated by the fact that metric learning algorithms perform poorly when the input data contains a lot of irrelevant features. By adding an $L_{2,1}$ regularization on the metric matrix, irrelevant features are removed by setting their corresponding coefficients to 0. The $L_{2,1}$ norm of $\mathbf{M}$ is defined as

$$\|\mathbf{M}\|_{2,1} = \sum_i \|\mathbf{M}_{i\cdot}\|_2$$

where $\|\mathbf{M}_{i\cdot}\|_2$ is the L_2 norm of ith row of matrix $\mathbf{M}$.

The authors applied this regularization to the metric-learning-to-rank problem (64). The method learns a Mahalanobis metric by solving the optimization problem, defined as:

$$\begin{aligned}
\min_{\mathbf{M}} \quad & tr(\mathbf{M}) + \lambda\|\mathbf{M}\|_{2,1} + \mu \sum_{q \in \mathcal{Q}} \xi_q \\
s.t. \quad & tr(\mathbf{M}^T \phi(q, y_q)) - tr(\mathbf{M}^T \phi(q, y)) \geq \delta(y_q, y) - \xi_q, \forall y \in \mathcal{Y}, \forall q \in \mathcal{Q} \\
& \xi_q \geq 0 \\
& \mathbf{M} \succeq 0
\end{aligned} \tag{3.26}$$

where $\mathcal{Q}$ is the query space and $\mathcal{Y}$ is the permutation space of all documents to be ranked. $tr(\mathbf{M}^T \phi(q, y))$ can be intuitively interpreted as a match between query q and the permutation y of all documents, y_q is the target permutation of query q, and $\delta(y_q, y)$ is a predefined scalar, indicating the desired margin between y and y_q. In (61), the problem 3.26 are efficiently solved with alternating direction method of multipliers (ADMM) method (13).

The $L_{2,1}$ regularization will force whole rows and columns of $\mathbf{M}$ to be 0. In contrast, the trace norm regularization will force the PSD matrix $\mathbf{M}$ to be low rank. The work of (112) studies the trace norm regularization and solve the optimization problem with efficient Nesterov method (67).

3.6 Distance Metric Learning Theory

Learning theory studies the theoretical behavior of learning algorithms, e.g. their generalization error bounds. In this section, we briefly discuss work on distance metric learning theory. All such work focuses on the generalization error of "similar" and "dissimilar" pairs.

3.6.1 Generalization Error Bounds for Metric Learning

Regularized Distance Metric Learning(48) The work of (48) pioneered the study of generalization error bounds for distance metric learning. The authors consider the foilowing regularized metric learnign problem:

$$\begin{aligned}
\min_{\mathbf{M}} \quad & \frac{1}{2}\|\mathbf{M}\|_F + \frac{2C}{n(n-1)} \sum_{i<j} g(y_{ij}(1 - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)) \\
s.t. \quad & \mathbf{M} \succeq 0 \\
& tr(\mathbf{M}) \leq \eta(d)
\end{aligned} \tag{3.27}$$

where d and n are, respectively, the number of input features and the number of instances. $y_{ij} \in \{-1, 1\}$ indicates the relationship between instance $\mathbf{x}_i$ and $\mathbf{x}_j$; $y_{ij} = 1$ if $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}$, otherwise $y_{ij} = -1$. g is the convex loss function and Lipschitz continuous with Lipschitz constant L. $\eta(d)$ is a constant controlling the size of matrix $\mathbf{M}$.

The target here is to derive an estimation error bound between empirical loss and expected loss, defined by

$$I(\mathbf{M}_D) - I_D(\mathbf{M}_D)$$

where the empirical loss of $\mathbf{M}_D$ on the given data D is

$$I_D(\mathbf{M}_D) = \frac{2}{n(n-1)} \sum_{i<j} g(y_{ij}(1 - d^2_{\mathbf{M}_D}(\mathbf{x}_i, \mathbf{x}_j))$$

and the expected loss is

$$I(\mathbf{M}_D) = E_{(\mathbf{x}_i, \mathbf{x}_j, y_{ij})} g(y_{ij}(1 - d^2_{\mathbf{M}_D}(\mathbf{x}_i, \mathbf{x}_j))$$

Note that $\mathbf{M}_D$ is the metric learned on data D by solving problem 8.1.

Since the instance pairs $(\mathbf{x}_i, \mathbf{x}_j)$ are not i.i.d., the authors establish the generalization error bound based on stability analysis (11). The main result is given in the following theorem.

Theorem 3. *(48) Let D be a collection of n randomly selected instances, and $\mathbf{M}_D$ be the distance metric learned by solving problem 8.1. With probability $1 - \delta$, we have the following bound for the expected loss function $I(\mathbf{M}_D)$,*

$$I(\mathbf{M}_D) - I_D(\mathbf{M}_D) \leq \frac{32CL^2R^4}{n} + (32CL^2R^4 + 4Ls(d) + 2g_0)\sqrt{\frac{\ln(\frac{2}{\delta})}{2n}}$$

where R is the upper bound of the norm of the instances, i.e. $\sup_x \|x\|_2 \leq R$, g_0 measures the largest loss when metric is 0, i.e. $g_0 = \sup_{\mathbf{x}_i, \mathbf{x}_j, y_{ij}} |g(y_{ij}(1 - d^2_{\mathbf{0}}(\mathbf{x}_i, \mathbf{x}_j))|$, and $s(d) = min(\sqrt{2dg_0C}, \eta(d))$.

According to Theorem 3, the estimation error converges in the order of $O(\frac{s(d)}{\sqrt{n}})$.

Empirical Loss Minimization (9) In the work of (9), the authors study the distance metric learned by minimizing the empirical loss,

$$\begin{aligned} \min_{\mathbf{M},\gamma} \quad & \frac{2}{n(n-1)} \sum_{i<j} g(y_{ij}(\gamma - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j))) \\ s.t. \quad & \mathbf{M} \succeq 0 \\ & \gamma \geq 0 \end{aligned} \tag{3.28}$$

where γ is a positive variable denoting the decision threshold.

Similar to (48), this work also shows that the empirical loss of converges to the expected loss in the order of $O(\frac{1}{\sqrt{n}})$, i.e.

$$I(\mathbf{M}_D, \gamma_D) - I_D(\mathbf{M}_D, \gamma_D) = O(\frac{1}{\sqrt{n}}), \forall (\mathbf{M}, \gamma) \in \{(\mathbf{M}, \gamma) : \mathbf{M} \succeq 0, \gamma \geq 0\}$$

Furthermore, they also show that the metric $\mathbf{M}_D$ learning by solving problem 3.28 is consistent, i.e.

$$E(I(\mathbf{M}_D, \gamma_D) - I(\mathbf{M}^*, \gamma^*))^2 = O(\frac{1}{\sqrt{n}})$$

where $(\mathbf{M}^*, \gamma^*)$ is the optimal value that minimizes the expected loss.

Regularized Distance Metric Learning with General Matrix Norm (14) In (14) the authors extend the work of (48) and consider distance metric learning with general forms of matrix metric regularization, more concretely they examine cost functions of the following form:

$$\begin{aligned} \min_{\mathbf{M},b} \quad & \lambda \|\mathbf{M}\| + \frac{1}{n(n-1)} \sum_{i \neq j} [y_{ij}(d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - b)]_+ \\ s.t. \quad & \mathbf{M} \succeq 0 \\ & b \geq 0 \end{aligned} \tag{3.29}$$

where $\|\cdot\|$ is a general matrix norm, which could be trace norm, $L_{2,1}$ norm, Frobenious norm and element-wise L_1 norm and $[\cdot]_+ = \max(0, \cdot)$ is the hinge loss function.

The main results of this work is given in the following theorem.

Theorem 4. *(14) Let* $(\mathbf{M}_D, b_D)$ *be the solution of problem 3.29. Then, for any* $0 < \delta < 1$*, with probability* $1 - \delta$*, we have that*

$$I(\mathbf{M}_D, b_D) - I_D(\mathbf{M}_D, b_D) \leq \frac{4R_n}{\sqrt{\lambda}} + \frac{4(3 + 2\mathbf{X}_*/\sqrt{\lambda})}{\sqrt{n}} + 2(1 + \mathbf{X}_*/\sqrt{\lambda})\sqrt{\frac{2\ln(\frac{1}{\delta})}{n}}$$

where R_n is the Radmemcher complexity for metric learning and $\mathbf{X}_* = max_{\mathbf{x}_i,\mathbf{x}_j} \|(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T\|_*$, $\|\cdot\|_*$ *is the dual norm of* $\|\cdot\|$.

For different matrix norm regularizers for input data space $x \in [0,1]^d$, we have

- Frobenius-norm: $\mathbf{X}_* = d$ and $R_n \leq \frac{2d}{\sqrt{n}}$
- L_1 norm: $\mathbf{X}_* = 1$ and $R_n \leq \frac{4\sqrt{e \log d}}{\sqrt{n}}$
- $L_{2,1}$ norm: $\mathbf{X}_* = \sqrt{d}$ and $R_n \leq \frac{4\sqrt{ed \log d}}{\sqrt{n}}$

These results indicate that when d is large, the generalization error bound with element-wise L_1 norm regularization is much better.

3.7 Conclusion

In this chapter we reviewed metric learning algorithms across different dimensions, we started with linear metric learning in the context of the standard kNN classification algorithm or in stochastic kNN, then we moved to non-linear variants, such as kernelized versions and local metric learning, and finally to regularization and error bounds for metric learning. In the following chapters we will present the main contributions of the thesis and how these are addressing limitations in the state of the art. We will show in chapter 4 how we address the sensitivity of LMNN to the original given target neighborhood structure by actually making it a part of the learning problem, developing a new method that is similar in spirit to NCA but less computationally expensive. Then in chapter 5 we will show how we go beyond standard kernelized metric learning and learn the kernel that is optimal for a given problem. In the chapter 6 we propose yet another non-linear metric learning method, one that relies on smooth local metric learning. Finally in chapter 7 we propose a two-stage non-linear metric learning method which improves over kernelized metric learning and local metric learning methods.

Chapter 4

Learning Neighborhood Metric Learning

In this chapter, we propose a novel formulation of the metric learning problem that includes in the learning process the learning of the local target neighborhood relations. The formulation is based on the fact that many metric learning algorithms can be seen as directly maximizing the sum of some quality measure of the target neighbor relationships under an explicit parametrization of the target neighborhoods. We cast the process of learning the neighborhood as a linear programming problem with a totally unimodular constraint matrix (89). An integer 0-1 solution of the target neighbor relationship is guaranteed by the totally unimodular constraint matrix. The number of the target neighbors does not need to be fixed, the formulation allows the assignment of a different number of target neighbors for each learning instance according to the instance's quality. We propose a two-step iterative optimization algorithm that learns the target neighborhood relationships and the distance metric. The proposed neighborhood learning method can be coupled with standard metric learning methods to learn the distance metric, as long as these can be cast as instances of our formulation.

The chapter is organized as follows. In Section 4.1, we discuss the moviation of the proposed learning neighborhood metric learning. In Section 4.2 we present the optimization problem of the Learning Neighborhoods for Metric Learning algorithm (LNML) and in Section 4.3 we discuss the properties of LNML. In Section 4.4 we instantiate our neighborhood learning method on LMNN and MCML. In section 4.5, we discuss in more detail the related work. In Section 4.6 we present the experimental results and we finally conclude with Section 4.7.

4.1 Motivation

Metric learning for classification relies on two interrelated concepts, similarity and dissimilarity constraints, and the target neighborhood. The latter defines for any given instance the instances that should be its neighbors and it is specified using similarity and dissimilarity constraints. In the absence of any other prior knowledge the similarity and dissimilarity constraints are derived from the class labels; instances of the same class should be similar and instances of different classes should be dissimilar.

The target neighborhood can be constructed in a *global* or *local* manner. With a global target neighborhood all constraints over all instance pairs are active; *all* instances of the same class should be similar and *all* instances from different classes should be dissimilar (32; 107). These admittedly hard to achieve constraints can be relaxed with the incorporation of slack variables (23; 62; 69; 83). With a local target neighborhood the satisfiability of the constraints is examined within a local neighborhood (34; 69; 104; 105). For any given instance we only need to ensure that we satisfy the constraints that involve that instance and instances from its local neighborhood. The resulting problem is considerably less constrained than what we get with the global approach and easier to solve. However, the appropriate definition of the local target neighborhood becomes now a critical component of the metric learning algorithm since it determines which constraints will be considered in the learning process. (105) defines the local target neighborhood of an instance as its k, same-class, nearest neighbors, under the Euclidean metric in the original space. Goldberger et al. (34) initialize the target neighborhood for each instance to all same-class instances. The local neighborhood is encoded as a soft-max function of a linear projection matrix and changes as a result of the metric learning. With the exception of (34), all other approaches whether global or local establish a target neighborhood prior to learning and keep it fixed throughout the learning process. Thus the metric that will be learned from these fixed neighborhood relations is constrained by them and will be a reflection of them. However, these relations are not necessarily optimal with respect to the learning problem that one is addressing.

In this chapter, we propose a novel formulation of the metric learning problem that includes in the learning process the learning of the local target neighborhood relations. The proposed neighborhood learning method can be coupled with standard metric learning methods to learn the distance metric, as long as these can be cast as instances of our formulation. We experiment with two instantiations of our approach where the Large Margin Nearest Neighbor (LMNN) (105) and Maximally Collapsing Metric Learning (MCML) (32) algorithms are used to learn the metric; we dub the respective instantiations LN-LMNN and LN-MCML. We performed a series of experiments on a number of classification problems in order to determine whether learning the neighborhood relations improves over only learning the distance metric. The experimental results show that this is indeed the case. In addition, we also compared our method with other state-of-the-art metric learning methods and show that it improves over the current state-of-the-art performance.

4.2 Learning Target Neighborhoods for Metric Learning

Given a set of training instances $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ where $\mathbf{x}_i \in \mathbb{R}^d$ and the class labels $y_i \in \{1, 2, \ldots, c\}$, the Mahalanobis distance between two instances $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as:

$$D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j) \tag{4.1}$$

where $\mathbf{M}$ is a Positive Semi-Definite (PSD) matrix ($\mathbf{M} \succeq 0$) that we will learn.

We can reformulate many of the existing metric learning methods, such as (32; 69; 83; 105; 107), by explicitly parametrizing the target neighborhood relations as follows:

$$\begin{aligned} \min_{\mathbf{M}, \Xi} \quad & \sum_{ij, i \neq j, y_i = y_j} \mathbf{P}_{ij} \cdot F_{ij}(\mathbf{M}, \Xi) \\ s.t. \quad & \text{constraints of the original problem} \end{aligned} \tag{4.2}$$

The matrix $\mathbf{P}, \mathbf{P}_{ij} \in \{0, 1\}$, describes the target neighbor relationships which are established prior to metric learning and are not altered in these methods. $\mathbf{P}_{ij} = 1$, if $\mathbf{x}_j$ is the target neighbor of $\mathbf{x}_i$, otherwise, $\mathbf{P}_{ij} = 0$. Note that the parameters $\mathbf{P}_{ii}$ and $\mathbf{P}_{ij} : y_i \neq y_j$ are set to zero, since an instance $\mathbf{x}_i$ cannot be a target neighbor of itself and the target neighbor relationship is constrained to same-class instances. This is why we have $i \neq j, y_i = y_j$ in the sum, however, for simplicity we will drop it from the following equations. $F_{ij}(\mathbf{M}, \Xi)$ is the term of the objective function of the metric learning methods that relates to the target neighbor relationship $\mathbf{P}_{ij}$, $\mathbf{M}$ is the Mahalanobis metric that we want to learn, and Ξ is a set of other parameters in the original metric learning problems, e.g. slack variables. Regularization terms on the $\mathbf{M}$ and Ξ parameters can also be added into Problem 4.2 (69; 83).

The $F_{ij}(\mathbf{M}, \Xi)$ term can be seen as the 'quality' of the target neighbor relationship $\mathbf{P}_{ij}$ under the distance metric $\mathbf{M}$; a low value indicates a high quality neighbor relationship $\mathbf{P}_{ij}$. The different metric learning methods learn the $\mathbf{M}$ matrix that optimizes the sum of the quality terms based on the a priori established target neighbor relationships; however, there is no reason to believe that these target relationships are the most appropriate for learning.

To overcome the constraints imposed by the fixed target neighbors we propose the Learning the Neighborhood for Metric Learning method (LNML) in which, in addition to the metric matrix $\mathbf{M}$, we also learn the target neighborhood matrix $\mathbf{P}$. LNML has as objective function the one given in Problem 4.2 which we now optimize also over the target neighborhood matrix $\mathbf{P}$. We add some new constraints in Problem 4.2 which control for the size of the target neighborhoods. The new optimization problem is the following:

$$\begin{aligned}
\min_{\mathbf{M},\Xi,\mathbf{P}} \quad & \sum_{ij} \mathbf{P}_{ij} \cdot F_{ij}(\mathbf{M}, \Xi) \\
s.t. \quad & \sum_{i,j} \mathbf{P}_{ij} = K_{av} * n \\
& K_{max} \geq \sum_{j} \mathbf{P}_{i,j} \geq K_{min} \\
& 1 \geq \mathbf{P}_{ij} \geq 0 \\
& \text{constraints of the original problem}
\end{aligned} \tag{4.3}$$

K_{min} and K_{max} are the minimum and maximum numbers of target neighbors that an instance can have. Thus the second constraint controls the number of target neighbor that $\mathbf{x}_i$ instance can have. K_{av} is the average number of target neighbor per instance. It holds by construction that $K_{max} \geq K_{av} \geq K_{min}$. We should note here that we relax the target neighborhood matrix so that its elements $\mathbf{P}_{ij}$ take values in $[0, 1]$ (third constraint). However, we will show later that a solution $\mathbf{P}_{ij} \in \{0, 1\}$ is obtained, given some natural constraints on the K_{min}, K_{max} and K_{av} parameters.

4.2.1 Target neighbor assignment rule

Unlike other metric learning methods, e.g. LMNN, in which the number of target neighbors is fixed, LNML can assign a different number of target neighbors for each instance. As we saw the first constraint in Problem 4.3 sets the average number of target neighbors per instance to K_{av}, while the second constraint limits the number of target neighbors for each instance between K_{min} and K_{max}. The above optimization problem implements a target neighbor assignment rule which assigns more target neighbors to instances that have high quality target neighbor relations. We do so in order to avoid overfitting since most often the ’good’ quality instances defined by metric learning algorithms (32; 105) are instances in dense areas with low classification error. As a result the geometry of the dense areas of the different classes will be emphasized. How much emphasis we give on good quality instances depends on the actual values of K_{min} and K_{max}. In the limit one can set the value of K_{min} to zero; nevertheless the risk with such a strategy is to focus heavily on dense and easy to learn regions of the data and ignore important boundary instances that are useful for learning.

4.3 Optimization

4.3.1 Properties of the Optimization Problem

We will now show that we get integer solutions for the $\mathbf{P}$ matrix by solving a linear programming problem and analyze the properties of Problem 4.3.

Lemma 1. *Given* $\mathbf{M}, \Xi$, *and* $K_{max} \geq K_{av} \geq K_{min}$ *then* $\mathbf{P}_{ij} \in \{0,1\}$, *if* K_{min}, K_{max} *and* K_{av} *are integers.*

Proof. Given $\mathbf{M}$ and Ξ, $F_{ij}(\mathbf{M}, \Xi)$ becomes a constant. We denote by $\mathbf{p}$ the vectorization of the target neighborhood matrix $\mathbf{P}$ which excludes the diagonal elements and $\mathbf{P}_{ij} : y_i \neq y_j$, and by $\mathbf{f}$ the respective vectorized version of the F_{ij} terms. Then we rewrite Problem 4.3 as:

$$\begin{aligned}
\min_{\mathbf{p}} \quad & \mathbf{p}^T\mathbf{f} \\
s.t. \quad & (\underbrace{K_{max}, \cdots, K_{max}}_{n}, K_{av} * n)^T \geq \mathbf{Ap} \geq \\
& (\underbrace{K_{min}, \cdots, K_{min}}_{n}, K_{av} * n)^T \\
& 1 \geq \mathbf{p}_i \geq 0
\end{aligned} \tag{4.4}$$

The first and second constraints of Problem 4.3 are reformulated as the first constraint in Problem 4.4. $\mathbf{A}$ is a $(n+1) \times (\sum_{c_l} n_{c_l}^2 - n)$ constraint matrix, where n_{c_l} is the number of instances in class c_l

$$\mathbf{A} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \cdots & \mathbf{1} \end{bmatrix}$$

where $\mathbf{1}$ ($\mathbf{0}$) is the vector of ones (zeros). Its elements depends on the its position in the matrix $\mathbf{A}$. In its ith column, all $\mathbf{1}$ ($\mathbf{0}$) vectors have $n_i - 1$ elements, where n_i is the number of instances of class c_j with $c_j = y_{p_i}$. According to the sufficient condition for total unimodularity (Theorem 7.3 in (89)) the constraint matrix $\mathbf{A}$ is a totally unimodular matrix. Thus, the constraint matrix $\mathbf{B} = [\mathbf{I}, -\mathbf{I}, \mathbf{A}, -\mathbf{A}]^T$ in the following equivalent problem also is a totally unimodular matrix (pp.268 in (81)).

$$\begin{aligned} \min_{\mathbf{p}} \quad & \mathbf{p}^T\mathbf{f} \\ s.t. \quad & \mathbf{Bp} \leq \mathbf{e} \\ & e = (\underbrace{1, \cdots, 1}_{\sum_{c_l} n_{c_l}^2 - n}, \underbrace{0, \cdots, 0}_{\sum_{c_l} n_{c_l}^2 - n}, \underbrace{K_{max}, \cdots, K_{max}}_{n}, \\ & K_{av} * n, \underbrace{-K_{min}, \cdots, -K_{min}}_{n}, -K_{av} * n)^T \end{aligned} \tag{4.5}$$

Since $\mathbf{e}$ is an integer vector, provided K_{min}, K_{max}, and K_{av}, are integers, and the constraint matrix $\mathbf{B}$ is totally unimodular, the above linear programming problem will only have integer solutions (Theorem 19.1a in (81)). Therefore, for the solution $\mathbf{p}$ it will hold that $\mathbf{p}_i \in \{0, 1\}$ and consequently $\mathbf{P}_{ij} \in \{0, 1\}$. $\square$

Although the constraints to control the size of the target neighborhood are convex, the objective function in Problem 4.3 is not jointly convex in $\mathbf{P}$ and $(\mathbf{M}, \mathbf{\Xi})$. However, as shown in Lemma 1, the binary solution of $\mathbf{P}$ can be obtained by a simple linear program if we fix $(\mathbf{M}, \mathbf{\Xi})$. Thus, Problem 4.3 is individually convex in $\mathbf{P}$ and $(\mathbf{M}, \mathbf{\Xi})$, if the original metric learning method is convex; this condition holds for all the methods that can be coupled with our neighborhood learning method (32; 69; 83; 105; 107).

4.3.2 Optimization Algorithm

Based on Lemma 1 and the individual convexity property we propose a general and easy to implement iterative algorithm to solve Problem 4.3. The details are given in Algorithm 1. At the first step of the kth iteration we learn the binary target neighborhood matrix $\mathbf{P}^{(k)}$ under a fixed metric matrix $\mathbf{M}^{(k-1)}$ and $\mathbf{\Xi}^{(k-1)}$, learned in the $k-1$th iteration, by solving the linear programming problem described in Lemma 1. At the second step of the iteration we learn the metric matrix $\mathbf{M}^{(k)}$ and $\mathbf{\Xi}^{(k)}$ with the target neighborhood matrix $\mathbf{P}^{(k)}$ using as the initial metric matrix the $\mathbf{M}^{(k-1)}$. The second step is simply the application of a standard metric learning algorithm in which we set the target neighborhood matrix to the learned $\mathbf{P}^{(k)}$ and the initial metric matrix to $\mathbf{M}^{(k-1)}$. The convergence of proposed algorithm is guaranteed if the original metric learning problem is convex (8). In our experiment, it most often converges in 5-10 iterations.

Algorithm 1 LNML

```
Input: $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{M}^0$,$\Xi^0$, $K_{min}$, $K_{max}$ and $K_{av}$
Output: $\mathbf{M}$
repeat
  $\mathbf{P}^{(k)}$=LearningNeighborhood($\mathbf{X}, \mathbf{Y}, \mathbf{M}^{(k-1)}, \Xi^{(k-1)}$) by solving Problem 4.4
  ($\mathbf{M}^{(k)}, \Xi^{(k)}$)=MetricLearning($\mathbf{M}^{(k-1)}$,$\mathbf{P}^{(k)}$)
  $k := k + 1$
until convergence
```

4.4 Instantiating LNML

In this section we will show how we instantiate our neighborhood learning method with two standard metric learning methods, LMNN and MCML, other possible instantiations include the metric learning methods presented in (69; 83; 107).

4.4.1 Learning the Neighborhood for LMNN

The optimization problem of LMNN is given by:

$$\begin{aligned} \min_{\mathbf{M},\xi} \quad & \sum_{ij} \mathbf{P}_{ij}\{(1-\mu)D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) + \mu \sum_{l}(1-\mathbf{Y}_{il})\xi_{ijl}\} \\ s.t. \quad & D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_l) - D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \geq 1 - \xi_{ijl} \\ & \xi_{ijl} > 0 \\ & \mathbf{M} \succeq 0 \end{aligned} \tag{4.6}$$

where the matrix $\mathbf{Y}, \mathbf{Y}_{ij} \in \{0,1\}$, indicates whether the class labels y_i and y_j are the same ($\mathbf{Y}_{ij} = 1$) or different ($\mathbf{Y}_{ij} = 0$). The objective is to minimize the sum of the distances of all instances to their target neighbors while allowing for some errors, this trade off is controlled by the μ parameter. This is a convex optimization problem that has been shown to have good generalization ability and can be applied to large datasets. The original problem formulation corresponds to a fixed parametrization of $\mathbf{P}$ where its non-zero values are given by the k nearest neighbors of the same class.

Coupling the neighborhood learning framework with the LMNN metric learning method results in the following optimization problem:

$$\begin{aligned}
&\min_{\mathbf{M},\mathbf{P},\xi} && \sum_{ij} \mathbf{P}_{ij} \cdot F_{ij}(\mathbf{M}, \xi) \qquad (4.7)\\
= &\min_{\mathbf{M},\mathbf{P},\xi} && \sum_{ij} \mathbf{P}_{ij}\{(1-\mu)D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) + \mu\sum_{l}(1-\mathbf{Y}_{il})\xi_{ijl}\}\\
&s.t. && K_{max} \geq \sum_{j} \mathbf{P}_{i,j} \geq K_{min}\\
& && \sum_{i,j} \mathbf{P}_{ij} = K_{av} * n\\
& && 1 \geq \mathbf{P}_{ij} \geq 0\\
& && D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_l) - D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \geq 1 - \xi_{ijl}\\
& && \xi_{ijl} > 0\\
& && \mathbf{M} \succeq 0
\end{aligned}$$

We will call this coupling of LNML and LMNN LN-LMNN. The target neighbor assignment rule of LN-LMNN assigns more target neighbors to instances that have small distances from their target neighbors and low hinge loss.

4.4.2 Learning the Neighborhood for MCML

MCML relies on a data dependent stochastic probability that an instance $\mathbf{x}_j$ is selected as the nearest neighbor of an instance $\mathbf{x}_i$; this probability is given by:

$$p_{\mathbf{M}}(j|i) = \frac{e^{-D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j)}}{Z_i} = \frac{e^{-D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j)}}{\sum_{k\neq i} e^{-D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_k)}}, \quad i \neq j \qquad (4.8)$$

MCML learns the Mahalanobis metric that minimizes the KL divergence distance between this probability distribution and the ideal probability distribution p_0 given by:

$$p_0(j|i) = \frac{\mathbf{P}_{ij}}{\sum_k \mathbf{P}_{ik}}, \quad p_0(i|i) = 0 \qquad (4.9)$$

where $\mathbf{P}_{ij} = 1$, if instance $\mathbf{x}_j$ is the target neighbor of instance $\mathbf{x}_i$, otherwise, $\mathbf{P}_{ij} = 0$. The optimization problem of MCML is given by:

$$\begin{aligned} & \min_{\mathbf{M}} && \sum_i KL[p_0(j|i)|p_{\mathbf{M}}(j|i)] \\ = & \min_{\mathbf{M}} && \sum_{i,j} \mathbf{P}_{ij} \frac{(D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + \log Z_i)}{\sum_k \mathbf{P}_{ik}} \\ & s.t. && \mathbf{M} \succeq 0 \end{aligned} \tag{4.10}$$

Like LMNN, this is also a convex optimization problem. In the original problem formulation the ideal distribution is defined based on class labels, i.e. $\mathbf{P}_{ij} = 1$, if instances $\mathbf{x}_i$ and $\mathbf{x}_j$ share the same class label, otherwise, $\mathbf{P}_{ij} = 0$.

The neighborhood learning method cannot learn directly the target neighborhood for MCML, since the objective function of the latter cannot be rewritten in the form of the objective function in Problem 4.3, due to the denominator $\sum_k \mathbf{P}_{ik}$. However, if we fix the size of the neighborhood to $\sum_k \mathbf{P}_{i,k} = K_{av} = K_{min} = K_{max}$ the two methods can be coupled and the resulting optimization is given by:

$$\begin{aligned} & \min_{\mathbf{M},\mathbf{P}} && \sum_{ij} \mathbf{P}_{ij} \cdot F_{ij}(\mathbf{M}) \\ = & \min_{\mathbf{M},\mathbf{P}} && \sum_{i,j} \mathbf{P}_{ij} \frac{(D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + \log Z_i)}{K_{av}} \\ & s.t. && \sum_j \mathbf{P}_{i,j} = K_{av} \\ & && \mathbf{M} \succeq 0 \end{aligned} \tag{4.11}$$

We will dub this coupling of LNML and MCML as LN-MCML. The original MCML method follows the global approach in establishing the neighborhood, with LN-MCML we get a local approach in which the neighborhoods are of fixed size K_{av} for every instance.

4.5 Related Work

The early work of Xing et al., (107), learns a Mahalanobis distance metric for clustering that tries to minimize the sum of pairwise distances between similar instances while keeping the sum of dissimilar instance distances greater than a threshold. The similar and dissimilar pairs are determined on the basis of prior knowledge. Globerson & Roweis, (32) introduced the Maximally Collapsing Metric Learning (MCML). MCML uses a stochastic nearest neighbor selection rule which selects the nearest neighbor $\mathbf{x}_j$ of an instance $\mathbf{x}_i$ under some probability distribution. It casts the metric learning problem as an optimization problem that tries to minimize the distance between two probability distributions, an ideal one and a data dependent one. In the ideal distribution the selection probability is always one for instances of the same class and zero for instances of different class, defining in that manner the similarity and dissimilarity constraints under the global target neighborhood approach. In the data dependent distribution the selection probability is given by a soft max function of a Mahalanobis distance metric, parametrized by the matrix $\mathbf{M}$ to be learned. In a similar spirit Davis et al., (23), introduced Information-Theoretic Metric Learning. ITML learns a Mahalanobis metric for classification with similarities and dissimilarities constraints that follow the global target neighborhood approach. In ITML all same-class instance pairs should have a distance smaller than some threshold and all different-class instance pairs should have a distance larger than some threshold. In addition the objective function of ITML seeks to minimize the distance between the learned metric matrix and a prior metric matrix, modelling like that prior knowledge about the metric if such is available. The optimization problem is cast as a distance of distributions subject to the pairwise constraints and finally expressed as a Bregman optimization problem (minimizing the LogDet divergence). In order to be able to find a feasible solution they introduce slack variables in the similarity and dissimilarity constraints.

The so far discussed metric learning methods follow the global target neighborhood approach in which all instances of the same class should be similar under the learned metric, and all pairs of instances from different classes dissimilar. This is a rather hard constraint and assumes that there is a linear projection of the original feature space that results in unimodal class conditional distributions. Goldberger et al., (34), proposed the NCA metric learning method which uses the same stochastic nearest neighbor selection rule under the same data-dependent probability distribution as MCML. NCA seeks to minimize the soft error under its stochastic nearest neighbor selection rule. It uses only similarity constraints and the original target neighborhood of an instance is the set of all same-class instances. After metric learning some, but not necessarily all, same class instances will end up having high probability of been selecting as nearest neighbors of a given instance, thus having a small distance, while the others will be pushed further away. NCA thus learns the local target neighborhood as a part of the optimization. Nevertheless it is prone to overfitting, (109), and does not scale to large datasets. The large margin nearest neighbor method (LMNN) described in (104; 105) learns a distance metric which directly minimizes the distances of each instance to its local target neighbors while keeping a large margin between them and different class instances. The target neighbors have to be specified prior to metric learning and in the absence of prior knowledge these are the k same class nearest neighbors for each instance.

4.6 Experiments

With the experiments we wish to investigate a number of issues. First, we want to examine whether learning the target neighborhood relations in the metric learning process can improve predictive performance over the baseline approach of metric learning with an apriori established target neighborhood. Second, we want to acquire an initial understanding of how the parameters K_{min} and K_{max} relate to the predictive performance. To this end, we will examine the predictive performance of LN-LMNN with two fold inner Cross Validation (CV) to select the appropriate values of K_{min} and K_{max}, method which we will denote by LN-LMNN(CV), and that of LN-LMNN, with a default setting of $K_{min} = K_{max} = K_{av}$. Finally, we want to see how the method that we propose compares to other state of the art metric learning methods, namely NCA and ITML. We include as an additional baseline in our experiments the performance of the Euclidean metric (EucMetric). We experimented with twelve different datasets: seven from the UCI machine learning repository, Sonar, Ionosphere, Iris, Balance, Wine, Letter, Isolet; four text mining datasets, Function, Alt, Disease and Structure, which were constructed from biological corpora (51); and MNIST (57), a handwritten digit recognition problem. A more detailed description of the datasets is given in Table 4.1.

Since LMNN is computationally expensive for datasets with large number of features we applied principal component analysis (PCA) to retain a limited number of principal components, following (105). The datasets to which this was done were the four text mining datasets, Isolet and MNIST. For the two latter 173 and 164 principal components were respectively retained that explain 95% of the total variance. For the text mining datasets more than 1300 principal components should be retained to explain 95% of the total variance. Considering the running time constraints, we kept the 300 most important principal components which explained 52.45%, 47.57%, 44.30% and 48.16% of the total variance for respectively Alt, Disease, Function and Structure. We could experiment with NCA and MCML on full tranining datasets only with datasets with a small number of instances due to their computational complexity. For completeness we experimented with NCA on large datasets by undersampling the training instances, i.e. the learning process only involved 10% of full training instances which was the maximum number we could experiment for each dataset. We also applied ITML on both versions of the larger datasets, i.e. with PCA-based dimensionality reduction and the original ones.

For ITML, we randomly generate for each dataset the default $20c^2$ constraints which are bounded repectively by the 5th and 95th percentiles of the distribution of all available pairwise distances for similar and dissimilar pairs. The slack variable γ is chosen form $\{10^i\}_{i=-4}^{4}$ using two-fold CV. The default identity matrix is employed as the regularization matrix. For the different instantiations of the LNML method we took care to have the same parameter settings for the encapsulated metric learning method and the respective baseline metric learning. For LN-LMNN, LN-LMNN(CV) and LMNN the regularization parameter μ that controls the trade-off between the distance minimization component and the hinge loss component was set to 0.5 (the default value of LMNN). For LMNN the default number of target neighbors was used (three). For LN-LMNN, we set $K_{min} = K_{max} = K_{av} = 3$, similar to LMNN. To explore the effect of a flexible neighborhood, the values of the K_{min} and K_{max} parameters in LN-LMNN(CV) were selected from the sets $\{1, 4, 3\}$ and $\{2, 5, 3\}$ respectively, while K_{av} was fixed to three. Similarly for LN-MCML we also set $K_{av} = 3$. The distance metrics for all methods are initialized to the Euclidean metric. As the classification algorithm we used 1-Nearest Neighbor.

We used 10-fold cross validation for all datasets to estimate classification accuracy, with the exception of Isolet and MNIST for which the default train and test split was used. The statistical significance of the differences were tested with McNemar's test and the p-value was set to 0.05. In order to get a better understanding of the relative performance of the different algorithms for a given dataset we used a ranking schema in which an algorithm A was assigned one point if it was found to have a significantly better accuracy than another algorithm B, 0.5 points if the two algorithms did not have a significantly different performance, and zero points if A was found to be significantly worse than B. The rank of an algorithm for a given dataset is simply the sum of the points over the different pairwise comparisons. When comparing N algorithms in a single dataset the highest possible score is $N - 1$ while if there is no significant difference each algorithm will get $(N - 1)/2$ points.

4.6.1 Results

The results are presented in Table 6.1. Examining whether learning also the neighborhood improves the predictive performance compared to plain metric learning, we see that in the case of LN-MCML, and for the five small datasets for which we have results, learning the neighborhood results in a statistically significant deterioration of the accuracy in one out of the five datasets (balance), while for the remaining four the differences were not statistically significant. If we now examine LN-LMNN(CV), LN-LMNN and LMNN we see that here learning the neighborhood does bring a statistically significant improvement. More precisely, LN-LMNN(CV) and LN-LMNN improve over LMNN respectively in six (two small and four large) and four (two small and two large) out of the 12 datasets. Moreove, by comparing LN-LMNN(CV) and LN-LMNN, we see that learning a flexible neighborhood with LN-LMNN(CV) improves significantly the performance over LN-LMNN on two datasets. The low performance of LN-MCML on the balance dataset was intriguing; in order to take a closer look we tried to determine automatically the appropriate target neighborhood size, K_{av}, by selecting it on the basis of five-fold inner cross validation from the set $K_{av} = \{3, 5, 7, 10, 20, 30\}$. The results showed that the default value of K_{av} was too small for the given dataset, with the average selected size of the target neighborhood at 29. As a result of the automatic tunning of the target neighborhood size the predictive performance of LN-MCML jumped at an accuracy of 93.92% which represented a significant improvement over the baseline MCML for the balance dataset. For the remaining datasets it turned out that the choice of $K_{av} = 3$ was a good default choice. In any case, determining the appropriate size of the target neighborhood and how that affects the predictive performance is an issue that we wish to investigate further. In terms of the total score that the different methods obtain the LN-LMNN(CV) achieves the best in both the small and large datasets. It is followed closely by NCA in the small datasets and by LN-LMNN in the large datasets.

4.7 Conclusion and Future Work

In this chapter, we presented LNML, a general Learning Neighborhood method for Metric Learning algorithms which couples the metric learning process with the process of establishing the appropriate target neighborhood for each instance, i.e. discovering for each instance which same class instances should be its neighbors. With the exception of NCA, which cannot be applied on datasets with many instances, all other metric learning methods whether they establish a global or a local target neighborhood do that prior to the metric learning and keep the target neighborhood fixed throughout the learning process. The metric that is learned as a result of the fixed neighborhoods simply reflects these original relations which are not necessarily optimal with respect to the classification problem that one is trying to solve. LNML lifts these constraints by learning the target neighborhood. We demonstrated it with two metric learning methods, LMNN and MCML. The experimental results show that learning the neighborhood can indeed improve the predictive performance.

The target neighborhood matrix $\mathbf{P}$ is strongly related to the similarity graphs which are often used in semi-supervised learning (47), spectral clustering (97) and manifold learning (78). Most often the similarity graphs in these methods are constructed in the original space, which nevertheless can be quite different from true manifold on which the data lies. These methods could also profit if one is able to learn the similarity graph instead of basing it on some prior structure.

Table 4.1: Datasets.

Datasets	Description	# Sample	# Feature	# Class	# Retained PCA Components	% Explained Variance
Sonar		208	60	2	NA	NA
Ionosphere		351	34	2	NA	NA
Wine		178	13	3	NA	NA
Iris		150	4	3	NA	NA
Balance		625	4	3	NA	NA
Letter	character recognition	20000	16	26	NA	NA
Function	sentence classification	3907	2708	2	300	44.30%
Alt	sentence classification	4157	2112	2	300	52.45%
Disease	sentence classification	3273	2376	2	300	47.57%
Structure	sentence classification	3584	2368	2	300	48.16%
Isolet	spoken character recognition	7797	619	26	173	95%
MNIST	handwritten digit recognition	70000	784	26	164	95%

Table 4.2: Accuracy results.The superscripts $^{+-=}$ next to the LN-XXXX accuracy indicate the result of the McNemar's statistical test result of its comparison to the accuracy of XXXX and denote respectively a significant win, loss or no difference for LN-XXXX. Similarly, the superscripts $^{+-=}$ next to the LN-LMNN(CV) accuracy indicate the result of its comparison to the accuracies of LMNN and LN-LMNN. The bold entries for each dataset have no significant difference from the best accuracy for that dataset. The number in the parenthesis indicates the score of the respective algorithm for the given dataset based on the pairwise comparisons.

(a) Small datasets

Datasets	MCML	LN-MCML	LMNN	LN-LMNN	LN-LMNN(CV)	EucMetric	NCA	ITML
sonar	**82.69**(3.5)	**84.62**(3.5)$^{=}$	**81.25**(3.5)	**81.25**(3.5)$^{=}$	**83.17**(3.5)$^{==}$	**80.77**(3.5)	**81.73**(3.5)	**82.69**(3.5)
ionosphere	88.03 (3.0)	**88.89**(3.5)$^{=}$	**89.17**(3.5)	87.75 (3.0)$^{=}$	**92.02**(5.5)$^{=+}$	86.32 (3.0)	**88.60**(3.5)	87.75 (3.0)
wine	91.57 (3.0)	**96.07**(4.0)$^{=}$	94.38 (3.0)	**97.75**(5.5)$^{+}$	**97.75**(5.5)$^{+=}$	76.97 (0.0)	91.57 (3.0)	**94.94**(4.0)
iris	**98.00**(4.5)	**96.00**(3.5)$^{=}$	**96.00**(3.5)	94.00 (3.0)$^{=}$	94.00 (3.0)$^{==}$	**96.00**(3.5)	**96.00**(3.5)	**96.00**(3.5)
balance	91.20 (5.0)	78.08 (1.0)$^{-}$	78.56 (1.0)	89.12 (4.5)$^{+}$	89.28 (4.5)$^{+=}$	78.72 (1.0)	**96.32**(7.0)	87.84 (4.0)
Total Score	19.0	15.5	14.5	19.5	22.0	11.0	20.5	18.0

(b) Large datasets

Datasets	PCA+LMNN	PCA+LN-LMNN	PCA+LN-LMNN(CV)	EucMetric	PCA+EucMetric	PCA+NCA	ITML	PCA+ITML
letter	96.86 (5.0)	**97.71**(6.5)$^{+}$	**97.64**(6.5)$^{+=}$	96.02 (0.5)	96.02 (0.5)	96.48 (3.0)	96.39 (3.0)	96.39 (3.0)
func	76.30 (2.5)	76.73 (2.5)$^{=}$	**78.91**(6.0)$^{++}$	**78.73**(6.0)	76.48 (2.5)	72.36 (0.0)	**78.73**(6.0)	76.45 (2.5)
alt	83.98 (5.0)	**84.92**(6.5)$^{+}$	**85.37**(6.5)$^{+=}$	68.51 (0.5)	71.33 (2.0)	78.54 (4.0)	68.49 (0.5)	72.53 (3.0)
disease	**80.23**(4.0)	**80.14**(4.0)$^{=}$	**80.66**(4.0)$^{==}$	**80.60**(4.0)	**80.23**(4.0)	73.59 (0.0)	**80.60**(4.0)	**80.14**(4.0)
structure	77.87 (4.5)	**78.83**(6.0)$^{=}$	**79.37**(6.5)$^{+=}$	75.82 (1.5)	77.00 (4.0)	71.93 (0.0)	75.79 (1.5)	77.06 (4.0)
Isolet	**95.96**(6.0)	**95.06**(6.0)$^{=}$	**95.06**(6.0)$^{==}$	88.58 (1.5)	88.33 (1.5)	85.63(0.0)	92.05 (3.5)	91.08 (3.5)
MNIST	**97.66**(6.0)	**97.66**(6.0)$^{=}$	**97.73**(6.0)$^{==}$	96.91 (2.0)	96.97 (2.0)	96.58 (1.5)	96.93 (1.5)	97.09 (3.0)
Total Score	33	37.5	41.5	16	16.5	8.5	20	23

Chapter 5

Metric Learning with Multiple Kernels

The choice of the appropriate kernel function is a fundamental problem in kernelized metric learning. In this chapter, we show how to perform the Mahalanobis Metric Learning (ML) with Mutiple Kernel Learning (MKL). We propose a general framework of ML with MKL which can be instantiated with virtually any Mahalanobis ML algorithm h provided that the latter satisfies some stated conditions. Our approach can be seen as the counterpart of MKL with SVMs (55; 77; 90) for ML.

This chapter is organized as follows. In Section 5.1, we discuss the motivation of the proposed work. To be self-contained, we introduce the useful notations and preliminaries on ML and MKL in Section 5.2. Next, in Section 5.3 we present the framework of ML with MKL and describe the ML_h-$MKL_{\{\mu|\mathbf{P}\}}$ optimization problems, as well as the non-regularized extension *NR-*ML_h*-MKL*. In Section 5.4 we analyze the ML_h-$MKL_{\{\mu|\mathbf{P}\}}$ and *NR-*ML_h*-MKL* optimization problems and in Section 5.5 we describe the *LMNN* based instantiation. Finally, we present the experimental results in Section 5.6 and conclude with Section 5.7.

5.1 Motivation

Very often a linear projection cannot adequately represent the inherent complexities of a problem at hand. To address this limitation various works proposed kernelized versions of ML methods in order to implicitly compute a linear transformation and Euclidean metric in some non-linear feature space; this computation results in a non-linear projection and distance computation in the original input space (23; 32; 69; 105; 107). However, we are now faced with a new problem, namely that of finding the appropriate kernel function and the associated feature space matching the requirements of the learning problem.

The simplest approach to address this problem is to select the best kernel from a predefined kernel set using internal cross-validation. The main drawback of this approach is that only one kernel is selected which limits the expressiveness of the resulting method. Additionally, this approach is limited to a small number of kernels–due to computational constraints–and requires the use of extra data. Multiple Kernel Learning (MKL) (55; 77) lifts the above limitations by learning a linear combination of a number of predefined kernels. The MKL approach can also naturally handle the multiple-source learning scenarios where instead of combining kernels defined on a single input data, which depending on the selected kernels could give rise to feature spaces with redundant features, we combine different and complementary data sources.

In this chapter, we show how to perform the Mahalanobis ML with MKL. We first propose a general framework of ML with MKL which can be instantiated with virtually any Mahalanobis ML algorithm h provided that the latter satisfies some stated conditions. We examine two parametrizations of the learning problem that give rise to two alternative formulations, denoted by ML_h-MKL_μ and ML_h-$MKL_\mathbf{P}$. Our approach can be seen as the counterpart of MKL with SVMs (55; 77; 90) for ML. Since the learned metric matrix has a regularized form (i.e. it has internal structure) we propose a straightforward non-regularized version of ML with MKL, denoted by *NR*-ML_h-*MKL*; however, due to the number of free parameters the non-regularized version can only scale with very small number of kernels and requires ML methods that are able to cope with large dimensionalities. We performed a number of experiments for ML with MKL in which, for the needs of this work, we have chosen the well known Large Margin Nearest Neighbor (105) (*LMNN*) algorithm as the ML method h. The experimental results suggest that *LMNN*-$MKL_\mathbf{P}$ outperforms *LMNN* with an unweighted kernel combination and the single best kernel selected by internal cross-validation.

5.2 Preliminaries

In the different flavors of metric learning we are given a matrix of learning instances $\mathbf{X} : n \times d$, the i-th row of which is the $\mathbf{x}_i^T \in \mathbb{R}^d$ instance, and a vector of class labels $\mathbf{y} = (y_1, \dots, y_n)^T, y_i \in \{1, \dots, c\}$. Consider a mapping $\mathbf{\Phi}_l(\mathbf{x})$ of instances $\mathbf{x}$ to some feature space $\mathcal{H}_l$, i.e. $\mathbf{x} \to \mathbf{\Phi}_l(\mathbf{x}) \in \mathcal{H}_l$. The corresponding kernel function $k_l(\mathbf{x}_i, \mathbf{x}_j)$ computes the inner product of two instances in the $\mathcal{H}_l$ feature space, i.e. $k_l(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{\Phi}_l(\mathbf{x}_i), \mathbf{\Phi}_l(\mathbf{x}_j) \rangle$. We denote dimensionality of $\mathcal{H}_l$ (possibly infinite) as d_l. The squared Mahalanobis distance of two instances in the $\mathcal{H}_l$ space is given by

$$d^2_{\mathbf{M}_l}(\mathbf{\Phi}_l(\mathbf{x}_i), \mathbf{\Phi}_l(\mathbf{x}_j)) = (\mathbf{\Phi}_l(\mathbf{x}_i) - \mathbf{\Phi}_l(\mathbf{x}_j))^T \mathbf{M}_l (\mathbf{\Phi}_l(\mathbf{x}_i) - \mathbf{\Phi}_l(\mathbf{x}_j))$$

, where $\mathbf{M}_l$ is a Positive Semi-Definite (PSD) metric matrix in the $\mathcal{H}_l$ space ($\mathbf{M}_l \succeq 0$). For some given ML method h we optimize (most often minimize) some cost function F_h with respect to the $\mathbf{M}_l$ metric matrix[1] under the PSD constraint for $\mathbf{M}_l$ and an additional set of pairwise distance constraints $\mathcal{C}_h(\{d^2_{\mathbf{M}_l}(\mathbf{\Phi}_l(\mathbf{x}_i), \mathbf{\Phi}_l(\mathbf{x}_j)) \mid i, j = 1, \dots, n\})$ that depend on the choice of h, e.g. similarity and dissimilarity pairwise constraint (23) and relative comparison constraint (105). In the reminder of this chapter, for simplicity, we denote this set of constraints as $\mathcal{C}_h(d^2_{\mathbf{M}_l}(\mathbf{\Phi}_l(\mathbf{x}_i), \mathbf{\Phi}_l(\mathbf{x}_j)))$. The kernelized ML optimization problem can be now written as:

$$\begin{aligned} \min_{\mathbf{M}_l} \quad & F_h(\mathbf{M}_l) \\ s.t. \quad & \mathcal{C}_h(d^2_{\mathbf{M}_l}(\mathbf{\Phi}_l(\mathbf{x}_i), \mathbf{\Phi}_l(\mathbf{x}_j))) \\ & \mathbf{M}_l \succeq 0 \end{aligned} \tag{5.1}$$

Kernelized ML methods do not require to learn the explicit form of the Mahalanobis metric $\mathbf{M}_l$. It was shown in (44) that the optimal solution of the Mahalanobis metric $\mathbf{M}_l$ is in the form of $\mathbf{M}_l = \eta_h \mathbf{I} + \mathbf{\Phi}_l(\mathbf{X})^T \mathbf{A}_l \mathbf{\Phi}_l(\mathbf{X})$, where $\mathbf{I}$ is the identity matrix of dimensionality $d_l \times d_l$, $\mathbf{A}_l$ is a $n \times n$ PSD matrix, $\mathbf{\Phi}_l(\mathbf{X})$ is the matrix of learning instances in the $\mathcal{H}_l$ space (with instances in rows), and η_h is a constant that depends on the ML method h. Since in the vast majority of the existing ML methods (32; 42; 69; 82; 84; 105; 107) the value of constant η_h is zero, in this work we only consider the optimal form of $\mathbf{M}_l$ with $\eta_h = 0$. Under the optimal parametrization of $\mathbf{M}_l = \mathbf{\Phi}_l(\mathbf{X})^T \mathbf{A}_l \mathbf{\Phi}_l(\mathbf{X})$ the squared Mahalanobis

[1] The optimization could also be done with respect to other variables of the cost function and not only $\mathbf{M}_l$. However, to keep the notation uncluttered we parametrize the optimization problem only over $\mathbf{M}_l$.

distance becomes:

$$d^2_{\mathbf{M}_l}(\mathbf{\Phi}_l(\mathbf{x}_i), \mathbf{\Phi}_l(\mathbf{x}_j)) \quad = \quad (\mathbf{K}^i_l - \mathbf{K}^j_l)^T \mathbf{A}_l (\mathbf{K}^i_l - \mathbf{K}^j_l) = d^2_{\mathbf{A}_l}(\mathbf{\Phi}_l(\mathbf{x}_i), \mathbf{\Phi}_l(\mathbf{x}_j)) \tag{5.2}$$

where $\mathbf{K}^i_l$ is the i-th column of kernel matrix $\mathbf{K}_l$, the (i, j) element of which is $K_{l_{ij}} = k_l(\mathbf{x}_i, \mathbf{x}_j)$. As a result, (5.1) can be rewritten as:

$$\begin{aligned} \min_{\mathbf{A}_l} \quad & F_h(\mathbf{\Phi}_l(\mathbf{X})^T \mathbf{A}_l \mathbf{\Phi}_l(\mathbf{X})) \\ s.t. \quad & \mathcal{C}_h(d^2_{\mathbf{A}_l}(\mathbf{\Phi}_l(\mathbf{x}_i), \mathbf{\Phi}_l(\mathbf{x}_j))) \\ & \mathbf{A}_l \succeq 0 \end{aligned} \tag{5.3}$$

In MKL we are given a set of kernel functions $\mathbf{Z} = \{k_l(\mathbf{x}_i, \mathbf{x}_j) \mid l = 1 \dots m\}$ and the goal is to learn an appropriate kernel function $k_\mu(\mathbf{x}_i, \mathbf{x}_j)$ parametrized by μ under a cost function Q. The cost function Q is determined by the cost function of the learning method that is coupled with multiple kernel learning, e.g. it can be the SVM cost function if one is using an SVM as the learning approach. As in (55; 77) we parametrize $k_\mu(\mathbf{x}_i, \mathbf{x}_j)$ by a linear combination of the form:

$$k_\mu(\mathbf{x}_i, \mathbf{x}_j) = \sum_{i=l}^{m} \mu_l k_l(\mathbf{x}_i, \mathbf{x}_j), \;\; \mu_l \geq 0, \;\; \sum_{l}^{m} \mu_l = 1 \tag{5.4}$$

We denote the feature space that is induced by the k_μ kernel by $\mathcal{H}_\mu$, feature space which is given by the mapping $\mathbf{x} \rightarrow \mathbf{\Phi}_\mu(\mathbf{x}) = (\sqrt{\mu_1}\mathbf{\Phi}_1(\mathbf{x})^T, \dots, \sqrt{\mu_m}\mathbf{\Phi}_m(\mathbf{x})^T)^T \in \mathcal{H}_\mu$. We denote the dimensionality of $\mathcal{H}_\mu$ by d; it can be infinite. Finally, we denote by $\mathcal{H}$ the feature space that we get by the unweighted concatenation of the m feature spaces, i.e. $\forall \mu_i, \; \mu_i = 1$, whose representation is given by $\mathbf{x} \rightarrow \mathbf{\Phi}(\mathbf{x}) = (\mathbf{\Phi}_1(\mathbf{x})^T, \dots, \mathbf{\Phi}_m(\mathbf{x})^T)^T$.

5.3 Metric Learning with Multiple Kernel Learning

The goal is to learn a metric matrix $\mathbf{M}$ in the feature space $\mathcal{H}_\mu$ induced by the mapping $\mathbf{\Phi}_\mu$ as well as the kernel weight μ; we denote this metric by $d^2_{\mathbf{M},\mu}$. Based on the optimal form of the Mahalanobis metric $\mathbf{M}$ for metric learning method learning with a single kernel function (44), we have the following lemma:

Lemma 2. *Assume that for a metric learning method h the optimal parameterization of its Mahalanobis metric $\mathbf{M}^*$ is $\mathbf{\Phi}_l(\mathbf{X})^T \mathbf{A}^* \mathbf{\Phi}_l(\mathbf{X})$, for some $\mathbf{A}^*$, when learning with a single kernel function $k_l(\mathbf{x}, \mathbf{x}')$. Then, for h with multiple kernel learning the optimal* *parametrization of its Mahalanobis metric* $\mathbf{M}^{**}$ *is given by* $\mathbf{\Phi}_\mu(\mathbf{X})^T\mathbf{A}^{**}\mathbf{\Phi}_\mu(\mathbf{X})$*, for some* $\mathbf{A}^{**}$*.*

The proof of the above Lemma is similar to the proof of Theorem 1 in (44) (it is not presented here due to the lack of space). Following Lemma 2, we have:

$$\begin{aligned} d^2_{\mathbf{M},\mu}(\mathbf{\Phi}_\mu(\mathbf{x}_i),\mathbf{\Phi}_\mu(\mathbf{x}_j)) = & \quad (\mathbf{\Phi}_\mu(\mathbf{x}_i)-\mathbf{\Phi}_\mu(\mathbf{x}_j))^T\mathbf{\Phi}_\mu(\mathbf{X})^T\mathbf{A}\mathbf{\Phi}_\mu(\mathbf{X})(\mathbf{\Phi}_\mu(\mathbf{x}_i)-\mathbf{\Phi}_\mu(\mathbf{x}_j)) \quad (5.5) \\ = & \quad \sum_l \mu_l(\mathbf{K}^i_l-\mathbf{K}^j_l)^T\mathbf{A}\sum_l \mu_l(\mathbf{K}^i_l-\mathbf{K}^j_l) = d^2_{\mathbf{A},\mu}(\mathbf{\Phi}_\mu(\mathbf{x}_i),\mathbf{\Phi}_\mu(\mathbf{x}_j)) \end{aligned}$$

Based on (5.5) and the constraints from (5.4), the ML optimization problem with MKL can be presented as:

$$\begin{aligned} \min_{\mathbf{A},\mu} \quad & F_h(\mathbf{\Phi}_\mu(\mathbf{X})^T\mathbf{A}\mathbf{\Phi}_\mu(\mathbf{X})) \qquad (5.6) \\ s.t. \quad & \mathcal{C}_h(d^2_{\mathbf{A},\mu}(\mathbf{\Phi}_\mu(\mathbf{x}_i),\mathbf{\Phi}_\mu(\mathbf{x}_j))) \\ & \mathbf{A}\succeq 0 \\ & \mu_l \geq 0 \\ & \sum_l^m \mu_l = 1 \end{aligned}$$

We denote the resulting optimization problem and the learning method by *ML*$_h$-*MKL*$_\mu$; clearly this is not fully specified until we choose a specific ML method h.

Let $\mathbf{B} = \begin{bmatrix} (\mathbf{K}^i_1-\mathbf{K}^j_1)^T \\ \dots \\ (\mathbf{K}^i_m-\mathbf{K}^j_m)^T \end{bmatrix}$. We note that $d^2_{\mathbf{A},\mu}(\mathbf{\Phi}_\mu(\mathbf{x}_i),\mathbf{\Phi}_\mu(\mathbf{x}_j))$ from (5.5) can also be written as:

$$d^2_{\mathbf{A},\mu}(\mathbf{\Phi}_\mu(\mathbf{x}_i),\mathbf{\Phi}_\mu(\mathbf{x}_j)) = \mu^T\mathbf{B}\mathbf{A}\mathbf{B^T}\mu = tr(\mathbf{P}\mathbf{B}\mathbf{A}\mathbf{B^T}) = d^2_{\mathbf{A},\mathbf{P}}(\mathbf{\Phi}_\mathbf{P}(\mathbf{x}_i),\mathbf{\Phi}_\mathbf{P}(\mathbf{x}_j)) \qquad (5.7)$$

where $\mathbf{P}=\mu\mu^T$ and $tr(\cdot)$ is the trace of a matrix. We use $\mathbf{\Phi}_\mathbf{P}(\mathbf{X})$ to emphasize the explicit the dependence of $\mathbf{\Phi}_\mu(\mathbf{X})$ to $\mathbf{P}=\mu\mu^T$. As a result, instead of optimizing over μ we can also use the parametrization over $\mathbf{P}$; the new optimization problem can now be written as:

$$\begin{aligned}
\min_{\mathbf{A},\mathbf{P}} \quad & F_h(\mathbf{\Phi}_{\mathbf{P}}(\mathbf{X})^T\mathbf{A}\mathbf{\Phi}_{\mathbf{P}}(\mathbf{X})) \\
s.t. \quad & \mathcal{C}_h(d^2_{\mathbf{A},\mathbf{P}}(\mathbf{\Phi}_{\mathbf{P}}(\mathbf{x}_i),\mathbf{\Phi}_{\mathbf{P}}(\mathbf{x}_j))) \\
& \mathbf{A} \succeq 0 \\
& \sum_{ij} P_{ij} = 1 \\
& P_{ij} \geq 0 \\
& Rank(\mathbf{P}) = 1 \\
& \mathbf{P} = \mathbf{P}^T
\end{aligned} \tag{5.8}$$

where the constraints $\sum_{ij} P_{ij} = 1$, $P_{ij} \geq 0$, $Rank(\mathbf{P}) = 1$, and $\mathbf{P} = \mathbf{P}^T$ are added so that $\mathbf{P} = \mu\mu^T$. We call the optimization problem and learning method (5.8) as ML_h-$MKL_{\mathbf{P}}$; as before in order to fully instantiate it we need to choose a specific metric learning method h.

Now, we derive an alternative parametrization of (5.5). We need two additional matrices: $\mathbf{C}_{\mu_i\mu_j} = \mu_i\mu_j\mathbf{I}$, where the dimensionality of $\mathbf{I}$ is $n \times n$, and $\mathbf{\Phi}'(\mathbf{X})$ which is an $mn \times d$ dimensional matrix:

$$\mathbf{\Phi}'(\mathbf{X}) = \begin{bmatrix} \mathbf{\Phi}_1(\mathbf{X}) & \dots & \mathbf{0} \\ \dots & \dots & \dots \\ \mathbf{0} & \dots & \mathbf{\Phi}_m(\mathbf{X}) \end{bmatrix}$$

We have:

$$d^2_{\mathbf{A},\mu}(\mathbf{\Phi}_\mu(\mathbf{x}_i),\mathbf{\Phi}_\mu(\mathbf{x}_j)) = (\mathbf{\Phi}(\mathbf{x}_i) - \mathbf{\Phi}(\mathbf{x}_j))^T\mathbf{M}'(\mathbf{\Phi}(\mathbf{x}_i) - \mathbf{\Phi}(\mathbf{x}_j)) \tag{5.9}$$

where:

$$\mathbf{M}' = \mathbf{\Phi}'(\mathbf{X})^T\mathbf{A}'\mathbf{\Phi}'(\mathbf{X}) \tag{5.10}$$

and $\mathbf{A}'$ is a $mn \times mn$ matrix:

$$\mathbf{A}' = \begin{bmatrix} \mathbf{C}_{\mu_1\mu_1}\mathbf{A} & \dots & \mathbf{C}_{\mu_1\mu_m}\mathbf{A} \\ \dots & \dots & \dots \\ \mathbf{C}_{\mu_m\mu_1}\mathbf{A} & \dots & \mathbf{C}_{\mu_m\mu_m}\mathbf{A} \end{bmatrix}. \tag{5.11}$$

From (5.9) we see that the Mahalanobis metric, parametrized by the $\mathbf{M}$ or $\mathbf{A}$ matrix, in the feature space $\mathcal{H}_\mu$ induced by the kernel k_μ, is equivalent to the Mahalanobis metric in the feature space $\mathcal{H}$ which is parametrized by $\mathbf{M}'$ or $\mathbf{A}'$. As we can see from (5.11), *ML_h-MKL_μ* and *ML_h-$MKL_{\mathbf{P}}$* learn a *regularized* matrix $\mathbf{A}'$ (i.e. matrix with internal structure) that corresponds to a parametrization of the Mahalanobis metric $\mathbf{M}^{'}$ in the feature space $\mathcal{H}$.

5.3.1 Non-Regularized Metric Learning with Multiple Kernel Learning

We present here a more general formulation of the optimization problem (5.6) in which we lift the regularization of matrix $\mathbf{A}'$ from (5.11), and learn instead a full PSD matrix $\mathbf{A}''$:

$$\mathbf{A}'' = \begin{bmatrix} \mathbf{A}_{11} & \dots & \mathbf{A}_{1m} \\ \dots & \dots & \dots \\ \mathbf{A}_{1m} & \dots & \mathbf{A}_{mm} \end{bmatrix} \tag{5.12}$$

where $\mathbf{A}_{kl}$ is an $n \times n$ matrix. The respective Mahalanobis matrix, which we denote by $\mathbf{M}''$, still have the same parametrization form as in (5.10), i.e. $\mathbf{M}'' = \mathbf{\Phi}^{'}(\mathbf{X})^T\mathbf{A}''\mathbf{\Phi}^{'}(\mathbf{X})$. As a result, by using $\mathbf{A}''$ instead of $\mathbf{A}'$ the squared Mahalanobis distance can be written now as:

$$\begin{aligned} & d^2_{\mathbf{A}''}(\mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{x}_j)) = (\mathbf{\Phi}(\mathbf{x}_i) - \mathbf{\Phi}(\mathbf{x}_j))^T\mathbf{M}^{''}(\mathbf{\Phi}(\mathbf{x}_i) - \mathbf{\Phi}(\mathbf{x}_j)) \\ & = [(\mathbf{K}^i_1 - \mathbf{K}^j_1)^T, \dots, (\mathbf{K}^i_m - \mathbf{K}^j_m)^T]\mathbf{A}''[(\mathbf{K}^i_1 - \mathbf{K}^j_1)^T, \dots, (\mathbf{K}^i_m - \mathbf{K}^j_m)^T]^T \\ & = [\mathbf{\Phi}_{\mathbf{Z}}(\mathbf{x}_i) - \mathbf{\Phi}_{\mathbf{Z}}(\mathbf{x}_j)]^T\mathbf{A}''(\mathbf{\Phi}_{\mathbf{Z}}(\mathbf{x}_i) - \mathbf{\Phi}_{\mathbf{Z}}(\mathbf{x}_j)] \end{aligned} \tag{5.13}$$

where $\mathbf{\Phi}_{\mathbf{Z}}(\mathbf{x}_i) = ((\mathbf{K}^i_1)^T, \dots, (\mathbf{K}^i_m)^T)^T \in \mathcal{H}_{\mathbf{Z}}$. What we see here is that under the $\mathbf{M}''$ parametrization computing the Mahalanobis metric in the $\mathcal{H}$ is equivalent to computing the Mahalanobis metric in the $\mathcal{H}_{\mathbf{Z}}$ space. Under the parametrization of the Mahalanobis distance given by (5.13), the optimization problem of metric learning with multiple kernel learning is the following:

$$\begin{aligned} \min_{\mathbf{A}''} \quad & F_h(\mathbf{\Phi}^{'}(\mathbf{X})^T\mathbf{A}''\mathbf{\Phi}^{'}(\mathbf{X})) \\ s.t. \quad & \mathcal{C}_h(d^2_{\mathbf{A}''}(\mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{x}_j))) \\ & \mathbf{A}'' \succeq 0 \end{aligned} \tag{5.14}$$

We call this optimization problem *NR-ML_h-MKL*. We should note that this formulation has scaling problems since it has $O(m^2n^2)$ parameters that need to be estimated, and it clearly requires a very efficient ML method h in order to be practical.

5.4 Optimization

5.4.1 Analysis

The *NR-ML$_h$-MKL* optimization problem obviously has the same convexity properties as the metric learning algorithm h that will be used, since the parametrization $\mathbf{M}'' = \mathbf{\Phi}'(\mathbf{X})^T\mathbf{A}''\mathbf{\Phi}'(\mathbf{X})$ used in *NR-ML$_h$-MKL* is linear with $\mathbf{A}''$, and the composition of a function with an affine mapping preserves the convexity property of the original function (12). This is also valid for the subproblems of learning matrix $\mathbf{A}$ in *ML$_h$-MKL$_\mu$* and *ML$_h$-MKL$_\mathbf{P}$* given the weight vector μ.

Given the PSD matrix $\mathbf{A}$, we have the following two lemmas for optimization problems *ML$_h$-MKL$_{\{\mu|\mathbf{P}\}}$*:

Lemma 3. *Given the PSD matrix $\mathbf{A}$ the* ML$_h$-MKL$_\mu$ *optimization problem is convex with μ if metric learning algorithm h is convex with μ.*

Proof. The last two constraints on μ of the optimization problem from (5.6) are linear, thus this problem is convex if metric learning algorithm h is convex with μ. □

Since $d^2_{\mathbf{A},\mu}(\mathbf{\Phi}_\mu(\mathbf{x}_i), \mathbf{\Phi}_\mu(\mathbf{x}_j))$ is convex quadratic of μ, which can be easily proved based on the PSD property of matrix $\mathbf{BAB^T}$ in (5.7), many of the well known metric learning algorithms, such as Pairwise SVM (95), POLA (84) and Xing's method (107) satisfy the conditions in Lemma 3.

The *ML$_h$-MKL$_\mathbf{P}$* optimization problem (5.8) is not convex given a PSD $\mathbf{A}$ matrix because the rank constraint is not convex. However, when the number of kernel m is small, e.g. 20, there is an equivalent convex formulation.

Lemma 4. Given *the PSD matrix $\mathbf{A}$, the* ML$_h$-MKL$_\mathbf{P}$ *optimization problem (5.8) can be formulated as an equivalent convex problem with respect to $\mathbf{P}$ if the ML algorithm h is linear with $\mathbf{P}$ and the number of kernel m is small.*

Proof. Given the PSD matrix $\mathbf{A}$, if h is linear with $\mathbf{P}$, we can formulate the rank constraint problem with the help of the two following convex problems (22):

$$\begin{aligned} \min_{\mathbf{P}} \quad & F_h(\mathbf{\Phi_P}(\mathbf{X})^T \mathbf{A} \mathbf{\Phi_P}(\mathbf{X})) + w \cdot tr(\mathbf{P}^T \mathbf{W}) \\ s.t. \quad & \mathcal{C}_h(d^2_{\mathbf{A},\mathbf{P}}(\mathbf{\Phi_P}(\mathbf{x}_i),\ \mathbf{\Phi_P}(\mathbf{x}_j))) \\ & \mathbf{A} \succeq 0 \\ & \mathbf{P} \succeq 0 \\ & \sum_{ij} P_{ij} = 1 \\ & P_{ij} \geq 0 \\ & \mathbf{P} = \mathbf{P}^T \end{aligned} \tag{5.15}$$

where w is a positive scalar just enough to make $tr(\mathbf{P}^T\mathbf{W})$ vanish, i.e. global convergence defined in (5.17), and the direction matrix $\mathbf{W}$ is an optimal solution of the following problem:

$$\begin{aligned} \min_{\mathbf{W}} \quad & tr(\mathbf{P}^{*T}\mathbf{W}) \\ s.t. \quad & \mathbf{0} \preceq \mathbf{W} \preceq \mathbf{I} \\ & tr(\mathbf{W}) = m - 1 \end{aligned} \tag{5.16}$$

where $\mathbf{P}^*$ is an optimal solution of (5.15) given $\mathbf{A}$ and $\mathbf{W}$ and m is the number of kernels. Problem (5.16) has a closed form solution $\mathbf{W} = \mathbf{U}\mathbf{U}^T$, where $\mathbf{U} \in \mathbb{R}^{m \times m-1}$ is the eigenvector matrix of $\mathbf{P}^*$ whose columns are the eigenvectors which correspond to the $m-1$ smallest eigenvalues of $\mathbf{P}^*$. The two convex problems are iteratively solved until global convergence, defined as:

$$\sum_{i=2}^{m} \lambda(\mathbf{P}^*)_i = tr(\mathbf{P}^{*T}\mathbf{W}^*) = \lambda(\mathbf{P}^*)^T \lambda(\mathbf{W}^*) \equiv 0 \tag{5.17}$$

where $\lambda(\mathbf{P}^*)_i$ is the i-th largest eigenvalue of $\mathbf{P}^*$. This formulation is not a projection method. At global convergence the convex problem (5.15) is not a relaxation of the original problem, instead it is an equivalent convex problem (22).

Now we proof the convergence of problem (5.15). Suppose the objective value of problem (5.15) is f_i at iteration i. Since both problem (5.15) and (5.16) minimize the objective value of (5.15), we have $f_j < f_i$ for any iteration $j > i$. Since the infimum f^* of objective value of problem (5.15) corresponds to the optimal objective value of (5.15)

Algorithm 2 *ML_h-MKL_μ, ML_h-$MKL_{\mathbf{P}}$*

```
**Input:** $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{A}^0$, $\mu^0$, and matrices $\mathbf{K}_1, \dots, \mathbf{K}_m$
**Output:** $\mathbf{A}$ and $\mu$
**repeat**
  $\mu^{(i)}$=WeightLearning($\mathbf{A}^{(i-1)}$)
  $\mathbf{K}_{\mu^{(i)}} = \sum_k \mu_k^i \mathbf{K}_k$
  $\mathbf{A}^{(i)}$=MetricLearning$_h$($\mathbf{A}^{(i-1)}$,$\mathbf{X}$,$\mathbf{K}_{\mu^{(i)}}$)
  $i := i + 1$
**until** convergence
```

when the second term is removed. Thus the nonincreasing sequence of objective value is bounded below and as a result convergent because any bounded monotonic sequence in $\mathbb{R}$ is convergent. Now the local convergence of problem (5.15) is established.

Only local convergence can be established for problem (5.15) because objective $tr(\mathbf{P}^T\mathbf{W})$ is generally multimodal (22). However, as indicated in section 7.2 (22), when the size of m is small, the global optimal of problem (5.15) can be often achieved. It can be simply verified by comparing the difference between infimum f^* and optimal objective value f of problem (5.15). $\square$

For a number of known metric learning algorithms, such as LMNN (105), POLA (84), MLSVM (69) and Xing's method (107) linearity with respect to $\mathbf{P}$ holds given $\mathbf{A} \succeq 0$.

5.4.2 Optimization Algorithms

The *NR-ML_h-MKL* optimization problem can be directly solved by any metric learning algorithm h on the space $\mathcal{H}_{\mathbf{Z}}$ when the optimization problem of the latter only involves the squared pairwise Mahalanobis distance, e.g. LMNN (105) and MCML (32). When the metric learning algorithm h has regularization term on $\mathbf{M}$, e.g. trace norm (42) and Frobenius norm (69; 84), most often the *NR-ML_h-MKL* optimization problem can be solved by a slightly modification of original algorithm.

We now describe how we can solve the optimization problems of *ML_h-MKL_μ* and *ML_h-$MKL_{\mathbf{P}}$*. Based on Lemmas 3 and 4 we propose for both methods a two-step iterative algorithm, Algorithm 2, at the first step of which we learn the kernel weighting and at the second the metric under the kernel weighting learned in the first step. At the first step of the i-th iteration we learn the $\mu^{(i)}$ kernel weight vector under fixed PSD matrices $\mathbf{A}^{(i-1)}$, learned at the preceding iteration $(i-1)$. For *ML_h-MKL_μ* we solve the weight learning problem using Lemma 3 and for *ML_h-$MKL_{\mathbf{P}}$* using Lemma 4. At the second step we apply the metric learning algorithm h and we learn the PSD matrices $\mathbf{A}^{(i)}$ with the

$\mathbf{K}_{\mu^{(i)}} = \sum_l \mu_l^{(i)} \mathbf{K}_i$ kernel matrix using as the initial metric matrices the $\mathbf{A}^{(i-1)}$. We should make clear that the optimization problem we are solving is only individually convex with respect to μ given the PSD matrix $\mathbf{A}$ and vice-versa. As a result, the convergence of the two-step algorithm (possible to a local optima) is guaranteed (36) and checked by the variation of μ and the objective value of metric learning method h. In our experiments (Section 5.6) we observed that it most often converges in less than ten iterations.

5.5 *LMNN*-Based Instantiation

We have presented two basic approaches to metric learning with multiple kernel learning: *ML$_h$-MKL$_\mu$* (*ML$_h$-MKL$_\mathbf{P}$*) and *NR-ML$_h$-MKL*. In order for the approaches to be fully instantiated we have to specify the ML algorithm h. In this work we focus on the LMNN state-of-the-art method (105).

Due to the relative comparison constraint, LMNN does not satisfy the condition of Lemma 3. However, as we already mentioned LMNN satisfies the condition of Lemma 4 so we get the *ML$_h$-MKL$_\mathbf{P}$* variant of the optimization problem for LMNN which we denote by *LMNN-MKL$_\mathbf{P}$*. The resulting optimization problem is:

$$\begin{aligned}
\min_{\mathbf{A},\mathbf{P},\xi} \quad & \sum_{ij} \mathbf{S}_{ij}\{(1-\gamma) d^2_{\mathbf{A},\mathbf{P}}(\mathbf{\Phi}_\mathbf{P}(\mathbf{x}_i), \mathbf{\Phi}_\mathbf{P}(\mathbf{x}_j)) + \gamma \sum_k (1-\mathbf{Y}_{ik})\xi_{ijk}\} \qquad (5.18)\\
s.t. \quad & d^2_{\mathbf{A},\mathbf{P}}(\mathbf{\Phi}_\mathbf{P}(\mathbf{x}_i), \mathbf{\Phi}_\mathbf{P}(\mathbf{x}_k)) - d^2_{\mathbf{A},\mathbf{P}}(\mathbf{\Phi}_\mathbf{P}(\mathbf{x}_i), \mathbf{\Phi}_\mathbf{P}(\mathbf{x}_j)) \geq 1 - \xi_{ijk}\\
& \xi_{ijk} > 0\\
& \mathbf{A} \succeq 0\\
& \sum_{kl} P_{kl} = 1\\
& P_{kl} \geq 0\\
& Rank(\mathbf{P}) = 1\\
& \mathbf{P} = \mathbf{P}^T
\end{aligned}$$

where the matrix $\mathbf{Y}, \mathbf{Y}_{ij} \in \{0, 1\}$, indicates if the class labels y_i and y_j are the same ($\mathbf{Y}_{ij} = 1$) or different ($\mathbf{Y}_{ij} = 0$). The matrix $\mathbf{S}$ is a binary matrix whose S_{ij} entry is non-zero if instance $\mathbf{x}_j$ is one of the k same class nearest neigbors of instance $\mathbf{x}_i$. The objective is to minimize the sum of the distances of all instances to their k same class nearest neighbors while allowing for some errors, trade of which is controlled by the γ parameter. As the objective function of LMNN only involves the squared pairwise Mahalanobis distances, the instantiation of *NR-ML$_h$-MKL* is straightforward and it consists simply of the application of LMNN on the space $\mathcal{H}_{\mathbf{Z}}$ in order to learn the metric. We denote this instantiation by *NR-LMNN-MKL*.

5.6 Experiments

In this section we perform a number of experiments on real world datasets in order to compare the two of the LMNN-based instantiations of our framework, i.e. *LMNN-MKL*$_{\mathbf{P}}$ and *NR-LMNN-MKL*. We compare these methods against two baselines: *LMNN-MKL*$_{CV}$ in which a kernel is selected from a set of kernels using 2-fold inner cross-validation (CV), and LMNN with the unweighted sum of kernels, which induces the $\mathcal{H}$ feature space, denoted by *LMNN*$_{\mathcal{H}}$. Additionally, we report performance of 1-Nearest-Neighbor, denoted as 1-NN, with no metric learning. The PSD matrix $\mathbf{A}$ and weight vector μ in *LMNN-MKL*$_{\mathbf{P}}$ were respectively initialized by $\mathbf{I}$ and equal weighting (1 divided by the number of kernels). The parameter w in the weight learning subproblem of *LMNN-MKL*$_{\mathbf{P}}$ was selected from $\{10^i \mid i = 0, 1, \ldots, 8\}$ and was the smallest value enough to achieve global convergence. Its direction matrix $\mathbf{W}$ was initialized by $\mathbf{0}$. The number of k same class nearest neighbors required by LMNN was set to 5 and its γ parameter to 0.5. After learning the metric and the multiple kernel combination we used 1-NN for classification.

5.6.1 Benchmark Datasets

We first experimented with 12 different datasets: five from the UCI machine learning repository, i.e. Sonar, Ionosphere, Wine, Iris, and Wdbc; three microarray datasets, i.e. CentralNervous, Colon, and Leukemia; and four proteomics datasets, i.e. MaleFemale, Stroke, Prostate and Ovarian. The attributes of all the datasets are standardized in the preprocessing step. The $\mathbf{Z}$ set of kernels that we use consists of the following 20 kernels: 10 polynomial with degree from one to ten, ten Gaussians with bandwidth $\sigma \in \{0.5, 1, 2, 5, 7, 10, 12, 15, 17, 20\}$ (the same set of kernels was used in (31)). Each basic kernel $\mathbf{K}_k$ was normalized by the average of its $diag(\mathbf{K}_k)$. *LMNN-MKL*$_{\mathbf{P}}$, *LMNN*$_{\mathcal{H}}$ and *LMNN-MKL*$_{CV}$ were tested using the complete $\mathbf{Z}$ set. For *NR-LMNN-MKL* due to its scaling limitations we could only use a small subset of $\mathbf{Z}$ consisting of the linear, the second order polynomial, and the Gaussian kernel with the kernel width of 0.5. We use 10-fold CV to estimate the predictive performance of the different methods. To test the statistical significance of the differences we used McNemar's test and we set the p-value to 0.05. To get a better understanding of the relative performance of the different methods for a given dataset we used a ranking schema in which a method A was assigned one point

Table 5.1: Accuracy results. The superscripts $^{+-=}$ next to the accuracies of *NR-LMNN-MKL*and *LMNN-MKL*$_{\mathbf{P}}$indicate the result of the McNemar's statistical test of their comparison to the accuracies of *LMNN*$_{\mathcal{H}}$and *LMNN-MKL*$_{CV}$and denote respectively a significant win, loss or no difference. The number in the parenthesis indicates the score of the respective algorithm for the given dataset based on the pairwise comparisons of the McNemar's statistical test.

Datasets	*NR-LMNN-MKL*	*LMNN-MKL*$_{\mathbf{P}}$	*LMNN*$_{\mathcal{H}}$	*LMNN-MKL*$_{CV}$	1-NN
Sonar	$88.46^{+=}$(3.0)	$85.58^{==}$(2.0)	82.21(1.0)	88.46(3.0)	82.21(1.0)
Wine	$98.88^{==}$(2.0)	$98.88^{==}$(2.0)	98.31(2.0)	96.07(2.0)	97.19(2.0)
Iris	$93.33^{==}$(2.0)	$95.33^{==}$(2.0)	94.67(2.0)	94.00(2.0)	95.33(2.0)
Ionosphere	$93.73^{==}$(2.5)	$94.87^{=+}$(3.0)	92.59(2.5)	90.88(2.0)	86.89(0.0)
Wdbc	$94.90^{-=}$(1.0)	$97.36^{=+}$(3.5)	97.36(3.0)	95.96(1.5)	95.43(1.0)
CentralNervous	$55.00^{==}$(2.0)	$63.33^{==}$(2.0)	65.00(2.0)	65.00(2.0)	58.33(2.0)
Colon	$80.65^{==}$(2.0)	$85.48^{+=}$(2.5)	66.13(1.5)	79.03(2.0)	74.19(2.0)
Leukemia	$95.83^{+=}$(2.5)	$94.44^{+=}$(2.5)	70.83(0.0)	95.83(2.5)	88.89(2.5)
MaleFemale	$86.57^{==}$(2.5)	$88.81^{+=}$(3.0)	80.60(1.5)	89.55(3.0)	58.96(0.0)
Ovarian	$95.26^{+=}$(3.0)	$94.47^{+=}$(3.0)	90.51(0.5)	94.47(3.0)	87.35(0.5)
Prostate	$79.50^{==}$(2.0)	$80.43^{==}$(2.5)	79.19(2.0)	78.88(2.0)	76.71(1.5)
Stroke	$69.71^{==}$(2.0)	$72.12^{==}$(2.0)	71.15(2.0)	70.19(2.0)	65.38(2.0)
Total Score	26.5	30.0	20.0	27.0	16.5

if its accuracy was significantly better than that of another method B, 0.5 points if the two methods did not have a significantly different performance, and zero points if A was found to be significantly worse than B.

The results are reported in Table 6.1. First, we observe that by learning the kernel inside *LMNN-MKL*$_{\mathbf{P}}$ we improve performance over *LMNN*$_{\mathcal{H}}$ that uses the unweighted kernel combination. More precisely, *LMNN-MKL*$_{\mathbf{P}}$ is significantly better than *LMNN*$_{\mathcal{H}}$ in four out of the thirteen datasets. If we now compare *LMNN-MKL*$_{\mathbf{P}}$ with *LMNN-MKL*$_{CV}$, the other baseline method where we select the best kernel with CV, we can see that *LMNN-MKL*$_{\mathbf{P}}$ also performs better being statistically significant better in two dataset. If we now examine *NR-LMNN-MKL* and *LMNN*$_{\mathcal{H}}$ we see that the former method, even though learning with only three kernels, is significantly better in two datasets, while it is significantly worse in one dataset. Comparing *NR-LMNN-MKL* and *LMNN-MKL*$_{CV}$ we observe that the two methods achieve comparable predictive performances. We should stress here that *NR-LMNN-MKL* has a disadvantage since it only uses three kernels, as opposed to other methods that use 20 kernels; the scalability of *NR-LMNN-MKL* is left as a future work. In terms of the total score that the different methods obtain the best one is *LMNN-MKL*$_{\mathbf{P}}$ followed by *LMNN-MKL*$_{CV}$ and *NR-LMNN-MKL*.

Table 5.2: Accuracy results on the multiple source datasets.

Datasets	*LMNN-MKL*$_{\mathbf{P}}$	*LMNN*$_{\mathcal{H}}$	*LMNN-MKL*$_{CV}$	1-NN
Multiple Feature	$98.79^{++}(3.0)$	98.44(1.5)	98.44(1.5)	97.86(0.0)
Oxford Flowers	$86.01^{++}(3.0)$	85.74(2.0)	65.46(0.0)	67.38(1.0)

5.6.2 Multiple Source Datasets

To evaluate the proposed method on problems with multiple sources of information we also perform experiments on the Multiple Features and the Oxford flowers datasets (71). Multiple Features from UCI has six different feature representations for 2,000 handwritten digits (0-9); each class has 200 instances. In the preprocessing step all the features are standardized in all the data sources. Oxford flowers dataset has 17 category flower images; each class has 80 instances. In the experiment seven distance matrices from the website[2] are used; these matrices are precomputed respectively from seven features, namely clustered HSV values, SIFT features on the foreground region, SIFT features on the foreground boundary, HOG features and three vocabularies derived from color, shape and texture. The details of these features are described in (70; 71). For both datasets Gaussian kernels are constructed respectively using the different feature representations of instances with kernel width σ_0, where σ_0 is the mean of all pairwise distances. We experiment with 10 random splits where half of the data is used for training and the other half for testing. We do not experiment here with *NR-LMNN-MKL* here due to its scaling limitations.

The accuracy results are reported in Table 5.2. We can see that by learning a linear combination of different feature representations *LMNN-MKL*$_{\mathbf{P}}$ achieves the best predictive performance on both datasets being significantly better than the two baselines, *LMNN*$_{\mathcal{H}}$ and *LMNN-MKL*$_{CV}$. The bad performance of *LMNN-MKL*$_{CV}$ on the Oxford flowers dataset could be explained by the fact that the different Gaussian kernels are complementary for the given problem, but in *LMNN-MKL*$_{CV}$ only one kernel is selected.

5.7 Conclusions

In this chapter we combine two recent developments in the field of machine learning, namely metric learning and multiple kernel learning, and propose a general framework for learning a metric in a feature space induced by a weighted combination of a number of individual kernels. This is in contrast with the existing kernelized metric learning

[2]http://www.robots.ox.ac.uk/∼vgg/data/flowers/index.html

techniques which consider only one kernel function (or possibly an unweighted combination of a number of kernels) and hence are sensitive to the selection of the associated feature space. The proposed framework is general as it can be coupled with many existing metric learning techniques. In this work, to practically demonstrate the effectiveness of the proposed approach, we instantiate it with the well know LMNN metric learning method. The experimental results confirm that the adaptively induced feature space does bring an advantage in the terms of predictive performance with respect to feature spaces induced by an unweighted combination of kernels and the single best kernel selected by internal CV.

Chapter 6

Parametric Local Metric Learning

In this chapter we propose the Parametric Local Metric Learning method (PLML) which learns a smooth metric matrix function over the data manifold. It is motivated by the fact that most of the existing local metric learning algorithms learn the metrics independently for each region making them also prone to overfitting.

This chapter is organized as follows. In Section 6.1, we discuss the motivation of this work. In Section 6.2 we introduce the necessary notations and metric learning concepts to be self-contained. In Section 6.3 we show how to learn the metric matrix function. In Section 7.4 we present the experimental results and we conclude with Section 7.5.

6.1 Motivation

Mahalanobis metric learning (24; 43; 48; 98; 105; 112) learns a global distance metric which determines the importance of the different input features and their correlations. However, since the discriminatory power of the input features might vary between different neighborhoods, learning a global metric cannot fit well the distance over the data manifold. Thus a more appropriate way is to learn a metric on each neighborhood and *local metric learning* (10; 30; 38; 105) does exactly that. It increases the expressive power of standard Mahalanobis metric learning by learning a number of local metrics (e.g. one per each instance).

Local metric learning has been shown to be effective for different learning scenarios. One of the first local metric learning works, Discriminant Adaptive Nearest Neighbor classification (38), DANN, learns local metrics by shrinking neighborhoods in directions orthogonal to the local decision boundaries and enlarging the neighborhoods parallel to the boundaries. It learns the local metrics independently with no regularization between them which makes it prone to overfitting. The authors of LMNN-Multiple Metric (LMNN-MM) (105) significantly limited the number of learned metrics and constrained all instances in a given region to share the same metric in an effort to combat overfitting. In the supervised setting they fixed the number of metrics to the number of classes; a similar idea has been also considered in (10). However, they too learn the metrics independently for each region making them also prone to overfitting since the local metrics will be overly specific to their respective regions. The authors of (110) learn local metrics using a least-squares approach by minimizing a weighted sum of the distances of each instance to apriori defined target positions and constraining the instances in the projected space to preserve the original geometric structure of the data in an effort to alleviate overfitting. However, the method learns the local metrics using a learning-order-sensitive propagation strategy, and depends heavily on the appropriate definition of the target positions for each instance, a task far from obvious. In another effort to overcome the overfitting problem of the discriminative methods (38; 105), Generative Local Metric Learning, GLML, (73), propose to learn local metrics by minimizing the NN expected classification error under strong model assumptions. They use the Gaussian distribution to model the learning instances of each class. However, the strong model assumptions might easily be very inflexible for many learning problems.

In this work we propose the Parametric Local Metric Learning method (PLML) which learns a *smooth metric matrix function* over the data manifold. More precisely, we parametrize the metric matrix of each instance as a linear combination of basis metric matrices of a small set of anchor points; this parametrization is naturally derived from an error bound on local metric approximation. Additionally we incorporate a manifold regularization on the linear combinations, forcing the linear combinations to vary smoothly over the data manifold. We develop an efficient two stage algorithm that first learns the linear combinations of each instance and then the metric matrices of the anchor points. To improve scalability and efficiency we employ a fast first-order optimization algorithm, FISTA (6), to learn the linear combinations as well as the basis metrics of the anchor points. We experiment with the PLML method on a number of large scale classification problems with tens of thousands of learning instances. The experimental results clearly demonstrate that PLML significantly improves the predictive performance over the current state-of-the-art metric learning methods, as well as over multi-class SVM with automatic kernel selection.

6.2 Preliminaries

We denote by $\mathbf{X}$ the $n \times d$ matrix of learning instances, the i-th row of which is the $\mathbf{x}_i^T \in \mathbb{R}^d$ instance, and by $\mathbf{y} = (y_1, \ldots, y_n)^T$, $y_i \in \{1, \ldots, c\}$ the vector of class labels. The squared Mahalanobis distance between two instances in the input space is given by:

$$d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j)$$

where $\mathbf{M}$ is a PSD metric matrix ($\mathbf{M} \succeq 0$). A linear metric learning method learns a Mahalanobis metric $\mathbf{M}$ by optimizing some cost function under the PSD constraints for $\mathbf{M}$ and a set of additional constraints on the pairwise instance distances. Depending on the actual metric learning method, different kinds of constraints on pairwise distances are used. The most successful ones are the large margin triplet constraints. A triplet constraint denoted by $c(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$, indicates that in the projected space induced by $\mathbf{M}$ the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ should be smaller than the distance between $\mathbf{x}_i$ and $\mathbf{x}_k$.

Very often a single metric $\mathbf{M}$ can not model adequately the complexity of a given learning problem in which discriminative features vary between different neighborhoods. To address this limitation in local metric learning we learn a set of local metrics. In most cases we learn a local metric for each learning instance (38; 73), however we can also learn a local metric for some part of the instance space in which case the number of learned metrics can be considerably smaller than n, e.g. (105). We follow the former approach and learn one local metric per instance. In principle, distances should then be defined as geodesic distances using the local metric on a Riemannian manifold. However, this is computationally difficult, thus we define the distance between instances $\mathbf{x}_i$ and $\mathbf{x}_j$ as:

$$d^2_{\mathbf{M}_i}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M}_i (\mathbf{x}_i - \mathbf{x}_j)$$

where $\mathbf{M}_i$ is the local metric of instance $\mathbf{x}_i$. Note that most often the local metric $\mathbf{M}_i$ of instance $\mathbf{x}_i$ is different from that of $\mathbf{x}_j$. As a result, the distance $d^2_{\mathbf{M_i}}(\mathbf{x}_i, \mathbf{x}_j)$ does not satisfy the symmetric property, i.e. it is not a proper metric. Nevertheless, in accordance to the standard practice we will continue to use the term local metric learning following (73; 105).

6.3 Parametric Local Metric Learning

We assume that there exists a Lipschitz smooth vector-valued function $f(\mathbf{x})$, the output of which is the vectorized local metric matrix of instance $\mathbf{x}$. Learning the local metric of each instance is essentially learning the value of this function at different points over the data manifold. In order to significantly reduce the computational complexity we will approximate the metric function instead of directly learning it.

Definition 3. *A vector-valued function $f(\mathbf{x})$ on $\mathbb{R}^d$ is a (α, β, p)-Lipschitz smooth function with respect to a vector norm $\|\cdot\|$ if $\|f(\mathbf{x}) - f(\mathbf{x}')\| \leq \alpha \|\mathbf{x} - \mathbf{x}'\|$ and $\left\|f(\mathbf{x}) - f(\mathbf{x}') - \nabla f(\mathbf{x}')^T(\mathbf{x} - \mathbf{x}')\right\| \leq \beta \|\mathbf{x} - \mathbf{x}'\|^{1+p}$, where $\nabla f(\mathbf{x}')^T$ is the derivative of the f function at $\mathbf{x}'$. We assume $\alpha, \beta > 0$ and $p \in (0, 1]$.*

(113) have shown that any Lipschitz smooth real function $f(\mathbf{x})$ defined on a lower dimensional manifold can be approximated by a linear combination of function values $f(\mathbf{u}), \mathbf{u} \in \mathbf{U}$, of a set $\mathbf{U}$ of anchor points. Based on this result we have the following lemma that gives the respective error bound for learning a Lipschitz smooth vector-valued function.

Lemma 5. *Let $(\gamma, \mathbf{U})$ be a nonnegative weighting on anchor points $\mathbf{U}$ in $\mathbb{R}^d$. Let f be an (α, β, p)-Lipschitz smooth vector function. We have for all $\mathbf{x} \in \mathbb{R}^d$:*

$$\left\| f(\mathbf{x}) - \sum_{\mathbf{u} \in \mathbf{U}} \gamma_{\mathbf{u}}(\mathbf{x}) f(\mathbf{u}) \right\| \leq \alpha \left\| \mathbf{x} - \sum_{\mathbf{u} \in \mathbf{U}} \gamma_{\mathbf{u}}(\mathbf{x}) \mathbf{u} \right\| + \beta \sum_{\mathbf{u} \in \mathbf{U}} \gamma_{\mathbf{u}}(\mathbf{x}) \|\mathbf{x} - \mathbf{u}\|^{1+p} \quad (6.1)$$

The proof of the above Lemma 5 is similar to the proof of Lemma 2.1 in (113); for lack of space we omit its presentation. By the nonnegative weighting strategy $(\gamma, \mathbf{U})$, the PSD constraints on the approximated local metric is automatically satisfied if the local metrics of anchor points are PSD matrices.

Lemma 5 suggests a natural way to approximate the local metric function by parameterizing the metric $\mathbf{M}_i$ of each instance $\mathbf{x}_i$ as a *weighted linear combination*, $\mathbf{W}_i \in \mathbb{R}^m$, of a small set of metric basis, $\{\mathbf{M}_{b_1}, \dots, \mathbf{M}_{b_m}\}$, each one associated with an anchor point defined in some region of the instance space. This parametrization will also provide us with a global way to regularize the flexibility of the metric function. We will first learn the vector of weights $\mathbf{W}_i$ for each instance $\mathbf{x}_i$, and then the basis metric matrices; these two together, will give us the $\mathbf{M}_i$ metric for the instance $\mathbf{x}_i$.

More formally, we define a $m \times d$ matrix $\mathbf{U}$ of anchor points, the i-th row of which is the anchor point $\mathbf{u}_i$, where $\mathbf{u}_i^T \in \mathbb{R}^d$. We denote by $\mathbf{M}_{b_i}$ the Mahalanobis metric matrix associated with $\mathbf{u}_i$. The anchor points can be defined using some clustering algorithm, we have chosen to define them as the means of clusters constructed by the k-means algorithm. The local metric $\mathbf{M}_i$ of an instance $\mathbf{x}_i$ is parametrized by:

$$\mathbf{M}_i = \sum_{b_k} W_{ib_k} \mathbf{M}_{b_k}, \; W_{ib_k} \geq 0, \; \sum_{b_k} W_{ib_k} = 1 \tag{6.2}$$

where $\mathbf{W}$ is a $n \times m$ weight matrix, and its W_{ib_k} entry is the weight of the basis metric $\mathbf{M}_{b_k}$ for the instance $\mathbf{x}_i$. The constraint $\sum_{b_k} W_{ib_k} = 1$ removes the scaling problem between different local metrics. Using the parametrization of equation (6.2), the squared distance of $\mathbf{x}_i$ to $\mathbf{x}_j$ under the metric $\mathbf{M}_i$ is:

$$d^2_{\mathbf{M}_i}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{b_k} W_{ib_k} d^2_{\mathbf{M}_{b_k}}(\mathbf{x}_i, \mathbf{x}_j) \tag{6.3}$$

where $d^2_{\mathbf{M}_{b_k}}(\mathbf{x}_i, \mathbf{x}_j)$ is the squared Mahalanobis distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ under the basis metric $\mathbf{M}_{b_k}$. We will show in the next section how to learn the weights of the basis metrics for each instance and in section 6.3.2 how to learn the basis metrics.

6.3.1 Smooth Local Linear Weighting

Lemma 5 bounds the approximation error by two terms. The first term states that $\mathbf{x}$ should be close to its linear approximation, and the second that the weighting should be local. In addition we want the local metrics to vary smoothly over the data manifold. To achieve this smoothness we rely on manifold regularization and constrain the weight vectors of neighboring instances to be similar. Following this reasoning we will learn Smooth Local Linear Weights for the basis metrics by minimizing the error bound of (6.1) together with a regularization term that controls the weight variation of similar instances. To simplify the objective function, we use the term $\left\|\mathbf{x} - \sum_{\mathbf{u}\in\mathbf{U}} \gamma_{\mathbf{u}}(\mathbf{x})\mathbf{u}\right\|^2$ instead of $\left\|\mathbf{x} - \sum_{\mathbf{u}\in\mathbf{U}} \gamma_{\mathbf{u}}(\mathbf{x})\mathbf{u}\right\|$. By including the constraints on the $\mathbf{W}$ weight matrix in (6.2), the optimization problem is given by:

$$\begin{aligned} \min_{\mathbf{W}} g(\mathbf{W}) &= \|\mathbf{X} - \mathbf{W}\mathbf{U}\|_F^2 + \lambda_1 tr(\mathbf{W}\mathbf{G}) + \lambda_2 tr(\mathbf{W}^\mathbf{T}\mathbf{L}\mathbf{W}) \\ s.t. \quad & \mathbf{W}_{ib_k} \geq 0, \sum_{b_k} \mathbf{W}_{ib_k} = 1, \forall i, b_k \end{aligned} \tag{6.4}$$

where $tr(\cdot)$ and $\|\cdot\|_F$ denote respectively the trace norm of a square matrix and the Frobenius norm of a matrix. The $m \times n$ matrix $\mathbf{G}$ is the squared distance matrix between each anchor point $\mathbf{u}_i$ and each instance $\mathbf{x}_j$, obtained for $p = 1$ in (6.1), i.e. its (i, j) entry is the squared Euclidean distance between $\mathbf{u}_i$ and $\mathbf{x}_j$. $\mathbf{L}$ is the $n \times n$ Laplacian matrix constructed by $\mathbf{D} - \mathbf{S}$, where $\mathbf{S}$ is the $n \times n$ symmetric pairwise similarity matrix of learning instances and $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}_{ii} = \sum_k \mathbf{S}_{ik}$. Thus the minimization of the $tr(\mathbf{W}^\mathbf{T}\mathbf{L}\mathbf{W})$ term constrains similar instances to have similar weight coefficients. The minimization of the $tr(\mathbf{W}\mathbf{G})$ term forces the weights of the instances to reflect their local properties. Most often the similarity matrix $\mathbf{S}$ is constructed using k-nearest neighbors graph (114). The λ_1 and λ_2 parameters control the importance of the different terms.

Since the cost function $g(\mathbf{W})$ is convex quadratic with $\mathbf{W}$ and the constraint is simply linear, (6.4) is a convex optimization problem with a unique optimal solution. The constraints on $\mathbf{W}$ in (6.4) can be seen as n simplex constraints on each row of $\mathbf{W}$; we will use the projected gradient method to solve the optimization problem. At each iteration t, the learned weight matrix $\mathbf{W}$ is updated by:

$$\mathbf{W}^{t+1} = Proj(\mathbf{W}^t - \eta\nabla g(\mathbf{W}^t)) \tag{6.5}$$

where $\eta > 0$ is the step size and $\nabla g(\mathbf{W}^t)$ is the gradient of the cost function $g(\mathbf{W})$ at $\mathbf{W}^t$. The $Proj(\cdot)$ denotes the simplex projection operator on each row of $\mathbf{W}$. Such a projection operator can be efficiently implemented with a complexity of $O(nm\log(m))$ (27). To speed up the optimization procedure we employ a fast first-order optimization method FISTA, (6). The detailed algorithm is described in Algorithm 3. The Lipschitz constant β required by this algorithm is estimated by using the condition of $g(\mathbf{W}^i) \leq \widetilde{g}_{\beta,\mathbf{Y}^i}(\mathbf{W}^i)$ (3). At each iteration, the main computations are in the gradient and the objective value with complexity $O(nmd + n^2m)$.

To set the weights of the basis metrics for a testing instance we can optimize (6.4) given the weight of the basis metrics for the training instances. Alternatively we can simply set

Algorithm 3 Smoothl Local Linear Weight Learning

```
**Input:** $\mathbf{W^0}$, $\mathbf{X}$, $\mathbf{U}$, $\mathbf{G}$, $\mathbf{L}$, $\lambda_1$, and $\lambda_2$
**Output:** matrix $\mathbf{W}$
define $\widetilde{g}_{\beta,\mathbf{Y}}(\mathbf{W}) = g(\mathbf{Y}) + tr(\nabla g(\mathbf{Y})^T(\mathbf{W}-\mathbf{Y})) + \frac{\beta}{2}\|\mathbf{W}-\mathbf{Y}\|_F^2$
initialize: $t_1 = 1$, $\beta = 1$,$\mathbf{Y}^1 = \mathbf{W}^0$, and $i = 0$
**repeat**
  $i = i+1$, $\mathbf{W}^i = Proj((\mathbf{Y}^i - \frac{1}{\beta}\nabla g(\mathbf{Y}^i)))$
  **while** $g(\mathbf{W}^i) > \widetilde{g}_{\beta,\mathbf{Y}^i}(\mathbf{W}^i)$ **do**
    $\beta = 2\beta$, $\mathbf{W}^i = Proj((\mathbf{Y}^i - \frac{1}{\beta}\nabla g(\mathbf{Y}^i)))$
  **end while**
  $t_{i+1} = \frac{1+\sqrt{1+4t_i^2}}{2}$, $\mathbf{Y}^{i+1} = \mathbf{W}^i + \frac{t_i - 1}{t_{i+1}}(\mathbf{W}^i - \mathbf{W}^{i-1})$
**until** converges;
```

them as the weights of its nearest neighbor in the training instances. In the experiments we used the latter approach.

6.3.2 Large Margin Basis Metric Learning

In this section we define a large margin based algorithm to learn the basis metrics $\mathbf{M}_{b_1}, \ldots, \mathbf{M}_{b_m}$. Given the $\mathbf{W}$ weight matrix of basis metrics obtained using Algorithm 3, the local metric $\mathbf{M}_i$ of an instance $\mathbf{x}_i$ defined in (6.2) is linear with respect to the basis metrics $\mathbf{M}_{b_1}, \ldots, \mathbf{M}_{b_m}$. We define the relative comparison distance of instances $\mathbf{x}_i$, $\mathbf{x}_j$ and $\mathbf{x}_k$ as: $d^2_{\mathbf{M}_i}(\mathbf{x}_i, \mathbf{x}_k) - d^2_{\mathbf{M}_i}(\mathbf{x}_i, \mathbf{x}_j)$. In a large margin constraint $c(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$, the squared distance $d^2_{\mathbf{M}_i}(\mathbf{x}_i, \mathbf{x}_k)$ is required to be larger than $d^2_{\mathbf{M}_i}(\mathbf{x}_i, \mathbf{x}_j) + 1$, otherwise an error $\xi_{ijk} \geq 0$ is generated. Note that, this relative comparison definition is different from that defined in LMNN-MM (105). In LMNN-MM to avoid over-fitting, different local metrics $\mathbf{M}_j$ and $\mathbf{M}_k$ are used to compute the squared distance $d^2_{\mathbf{M}_j}(\mathbf{x}_i, \mathbf{x}_j)$ and $d^2_{\mathbf{M}_k}(\mathbf{x}_i, \mathbf{x}_k)$ respectively, as no smoothness constraint is added between metrics of different local regions.

Given a set of triplet constraints, we learn the basis metrics $\mathbf{M}_{b_1}, \ldots, \mathbf{M}_{b_m}$ with the following optimization problem:

$$\begin{aligned} \min_{\mathbf{M}_{b_1},\ldots,\mathbf{M}_{b_m},\xi} \quad & \alpha_1 \sum_{b_l} ||\mathbf{M}_{b_l}||_F^2 + \sum_{ijk} \xi_{ijk} + \alpha_2 \sum_{ij} \sum_{b_l} W_{ib_l} d^2_{\mathbf{M}_{b_l}}(\mathbf{x}_i, \mathbf{x}_j) \\ s.t. \quad & \sum_{b_l} W_{ib_l}(d^2_{\mathbf{M}_{b_l}}(\mathbf{x}_i, \mathbf{x}_k) - d^2_{\mathbf{M}_{b_l}}(\mathbf{x}_i, \mathbf{x}_j)) \geq 1 - \xi_{ijk} \ \forall i,j,k \\ & \xi_{ijk} \geq 0; \ \forall i,j,k \ \mathbf{M}_{b_l} \succeq 0; \ \forall b_l \end{aligned} \tag{6.6}$$

where α_1 and α_2 are parameters that balance the importance of the different terms. The large margin triplet constraints for each instance are generated using its k_1 same class nearest neighbors and k_2 different class nearest neighbors by requiring its distances to the k_2 different class instances to be larger than those to its k_1 same class instances. In the objective function of (6.6) the basis metrics are learned by minimizing the sum of large margin errors and the sum of squared pairwise distances of each instance to its k_1 nearest neighbors computed using the local metric. Unlike LMNN we add the squared Frobenius norm on each basis metrics in the objective function. We do this for two reasons. First we exploit the connection between LMNN and SVM shown in (25) under which the squared Frobenius norm of the metric matrix is related to the SVM margin. Second because adding this term leads to an easy-to-optimize dual formulation of (6.6) (86).

Unlike many special solvers which optimize the primal form of the metric learning problem (87; 105), we follow (86) and optimize the Lagrangian dual problem of (6.6). The dual formulation leads to an efficient basis metric learning algorithm. Introducing the Lagrangian dual multipliers γ_{ijk}, $\mathbf{p}_{ijk}$ and the PSD matrices $\mathbf{Z}_{b_l}$ to respectively associate with every large margin triplet constraints, $\xi_{ijk} \geq 0$ and the PSD constraints $\mathbf{M}_{b_l} \succeq 0$ in (6.6), we can easily derive the following Lagrangian dual form

$$\begin{aligned} \max_{\mathbf{Z}_{b_1},\ldots,\mathbf{Z}_{b_m},\gamma} \quad & \sum_{ijk} \gamma_{ijk} - \sum_{b_l} \frac{1}{4\alpha_1} \cdot \|\mathbf{Z}_{b_l} + \sum_{ijk} \gamma_{ijk} W_{ib_l} \mathbf{C}_{ijk} - \alpha_2 \sum_{ij} W_{ib_l} \mathbf{A}_{ij}\|_F^2 \\ s.t. \quad & 1 \geq \gamma_{ijk} \geq 0; \ \forall_{i,j,k} \ \mathbf{Z}_{b_l} \succeq 0; \ \forall b_l \end{aligned} \tag{6.7}$$

and the corresponding optimality conditions: $\mathbf{M}^*_{b_l} = \frac{(\mathbf{Z}^*_{b_l} + \sum_{ijk} \gamma^*_{ijk} W_{ib_l} \mathbf{C}_{ijk} - \alpha_2 \sum_{ij} W_{ib_l} \mathbf{A}_{ij})}{2\alpha_1}$ and $1 \geq \gamma_{ijk} \geq 0$, where the matrices $\mathbf{A}_{ij}$ and $\mathbf{C}_{ijk}$ are given by $\mathbf{x}_{ij}^T \mathbf{x}_{ij}$ and $\mathbf{x}_{ik}^T \mathbf{x}_{ik} - \mathbf{x}_{ij}^T \mathbf{x}_{ij}$ respectively, where $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$.

Compared to the primal form, the main advantage of the dual formulation is that the second term in the objective function of (6.7) has a closed-form solution for $\mathbf{Z}_{b_l}$ given a fixed γ. To drive the optimal solution of $\mathbf{Z}_{b_l}$, let $\mathbf{K}_{b_l} = \alpha_2 \sum_{ij} W_{ib_l} \mathbf{A}_{ij} - \sum_{ijk} \gamma_{ijk} W_{ib_l} \mathbf{C}_{ijk}$. Then, given a fixed γ, the optimal solution of $\mathbf{Z}_{b_l}$ is $\mathbf{Z}^*_{b_l} = (\mathbf{K}_{b_l})_+$, where $(\mathbf{K}_{b_l})_+$ projects the matrix $\mathbf{K}_{b_l}$ onto the PSD cone, i.e. $(\mathbf{K}_{b_l})_+ = \mathbf{U}[\max(\mathbf{diag}(\mathbf{\Sigma})), \mathbf{0})]\mathbf{U}^\mathbf{T}$ with $\mathbf{K}_{b_l} = \mathbf{U\Sigma U^T}$.

Now, (6.7) is rewritten as:

$$\begin{aligned} \min_{\gamma} \quad & g(\gamma) = -\sum_{ijk} \gamma_{ijk} + \sum_{b_l} \frac{1}{4\alpha_1} \|(\mathbf{K}_{b_l})_+ - \mathbf{K}_{b_l}\|_F^2 \\ s.t. \quad & 1 \geq \gamma_{ijk} \geq 0; \forall i,j,k \end{aligned} \tag{6.8}$$

And the optimal condition for $\mathbf{M}_{b_l}$ is $\mathbf{M}^*_{b_l} = \frac{1}{2\alpha_1}((\mathbf{K}^*_{b_l})_+ - \mathbf{K}^*_{b_l})$. The gradient of the objective function in (6.8), $\nabla g(\gamma_{ijk})$, is given by: $\nabla g(\gamma_{ijk}) = -1+\sum_{b_l} \frac{1}{2\alpha_1} \langle (\mathbf{K}_{b_l})_+ - \mathbf{K}_{b_l}, W_{ib_l}\mathbf{C}_{ijk}\rangle$. At each iteration, γ is updated by: $\gamma^{i+1} = BoxProj(\gamma^i - \eta\nabla g(\gamma^i))$ where $\eta > 0$ is the step size. The $BoxProj(\cdot)$ denotes the simple box projection operator on γ as specified in the constraints of (6.8). At each iteration, the main computational complexity lies in the computation of the eigen-decomposition with a complexity of $O(md^3)$ and the computation of the gradient with a complexity of $O(m(nd^2 + cd))$, where m is the number of basis metrics and c is the number of large margin triplet constraints. As in the weight learning problem the FISTA algorithm is employed to accelerate the optimization process; for lack of space we omit the algorithm presentation.

6.4 Experiments

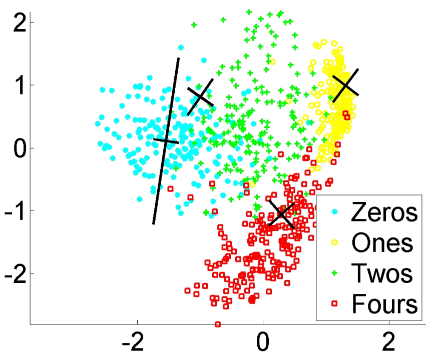
In this section we will evaluate the performance of PLML and compare it with a number of relevant baseline methods on six datasets with large number of instances, ranging from 5K to 70K instances; these datasets are Letter, USPS, Pendigits, Optdigits, Isolet and MNIST. We want to determine whether the addition of manifold regularization on the local metrics improves the predictive performance of local metric learning, and whether the local metric learning improves over learning with single global metric. We will compare PLML against six baseline methods. The first, SML, is a variant of PLML where a single global metric is learned, i.e. we set the number of basis in (6.6) to one. The second, Cluster-Based LML (CBLML), is also a variant of PLML without weight learning. Here we learn one local metric for each cluster and we assign a weight of one for a basis metric $\mathbf{M}_{b_i}$ if the corresponding cluster of $\mathbf{M}_{b_i}$ contains the instance, and zero otherwise. Finally, we also compare against four state of the art metric learning methods LMNN (105), BoostMetric (87)[1], GLML (73) and LMNN-MM (105)[2]. The former two learn a single global metric and the latter two a number of local metrics. In addition to the different metric learning methods, we also compare PLML against multi-class SVMs in which we use the one-against-all strategy to determine the class label for multi-class problems and select the best kernel with inner cross validation.

Since metric learning is computationally expensive for datasets with large number of features we followed (105) and reduced the dimensionality of the USPS, Isolet and MINIST datasets by applying PCA. In these datasets the retained PCA components explain 95% of their total variances. We preprocessed all datasets by first standardizing the input features, and then normalizing the instances to so that their L2-norm is one.

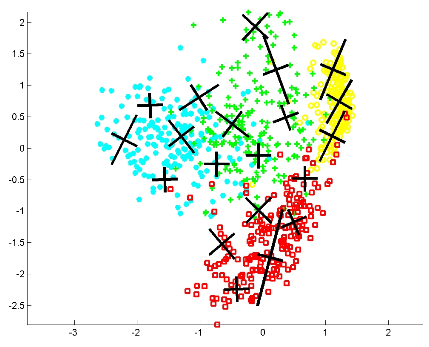
[1]http://code.google.com/p/boosting

[2]http://www.cse.wustl.edu/∼kilian/code/code.html.

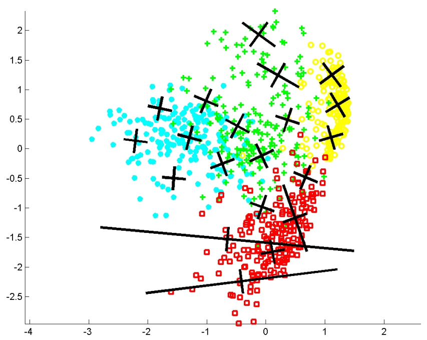
PLML has a number of hyper-parameters. To reduce the computational time we do not tune λ_1 and λ_2 of the weight learning optimization problem (6.4), and we set them to their default values of $\lambda_1 = 1$ and $\lambda_2 = 100$. The Laplacian matrix $\mathbf{L}$ is constructed using the six nearest neighbors graph following (114). The anchor points $\mathbf{U}$ are the means of clusters constructed with k-means clustering. The number m of anchor points, i.e. the number of basis metrics, depends on the complexity of the learning problem. More complex problems will often require a larger number of anchor points to better model the complexity of the data. As the number of classes in the examined datasets is 10 or 26, we simply set $m = 20$ for all datasets. In the basis metric learning problem (6.6), the number of the dual parameters γ is the same as the number of triplet constraints. To speedup the learning process, the triplet constraints are constructed only using the three same-class and the three different-class nearest neighbors for each learning instance. The parameter α_2 is set to 1, while the parameter α_1 is the only parameter that we select from the set $\{0.01, 0.1, 1, 10, 100\}$ using 2-fold inner cross-validation. The above setting of basis metric learning for PLML is also used with the SML and CBLML methods. For LMNN and LMNN-MM we use their default settings, (105), in which the triplet constraints are constructed by the three nearest same-class neighbors and all different-class samples. As a result, the number of triplet constraints optimized in LMNN and LMNN-MM is much larger than those of PLML, SML, BoostMetric and CBLML. The local metrics are initialized by identity matrices. As in (73), GLML uses the Gaussian distribution to model the learning instances from the same class. Finally, we use the 1-NN rule to evaluate the performance of the different metric learning methods. In addition as we already mentioned we also compare against multi-class SVM. Since the performance of the latter depends heavily on the kernel with which it is coupled we do automatic kernel selection with inner cross validation to select the best kernel and parameter setting. The kernels were chosen from the set of linear, polynomial (degree 2,3 and 4), and Gaussian kernels; the width of the Gaussian kernel was set to the average of all pairwise distances. Its C parameter of the hinge loss term was selected from $\{0.1, 1, 10, 100\}$.



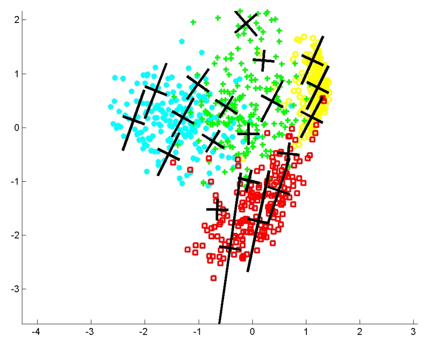
(a) LMNN-MM



(b) CBLML



(c) GLML

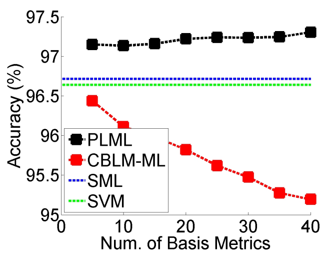


(d) PLML

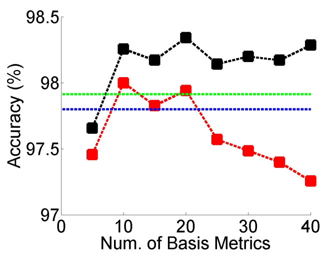
Figure 6.1: The visualization of learned local metrics of LMNN-MM, CBLML, GLML and PLML.

Table 6.1: Accuracy results. The superscripts $^{+-=}$ next to the accuracies of PLML indicate the result of the McNemar's statistical test with LMNN, BoostMetric, SML, CBLML, LMNN-MM, GMLM and SVM. They denote respectively a significant win, loss or no difference for PLML. The number in the parenthesis indicates the score of the respective algorithm for the given dataset based on the pairwise comparisons of the McNemar's statistical test.

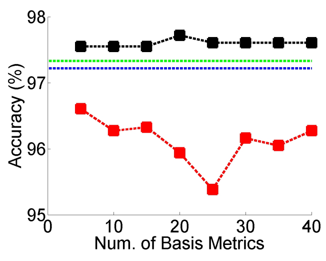
		Single Metric Learning Baselines			Local Metric Learning Baselines			
Datasets	PLML	LMNN	BoostMetric	SML	CBLML	LMNN-MM	GLML	SVM
Letter	**97.22**$^{+++\vert+++\vert+}$(7.0)	96.08(2.5)	96.49(4.5)	96.71(5.5)	95.82(2.5)	95.02(1.0)	93.86(0.0)	96.64(5.0)
Pendigits	**98.34**$^{+++\vert+++\vert+}$(7.0)	97.43(2.0)	97.43(2.5)	97.80(4.5)	97.94(5.0)	97.43(2.0)	96.88(0.0)	97.91(5.0)
Optdigits	**97.72**$^{===\vert+++\vert=}$(5.0)	**97.55**(5.0)	**97.61**(5.0)	**97.22**(5.0)	95.94(1.5)	95.94(1.5)	94.82(0.0)	**97.33**(5.0)
Isolet	**95.25**$^{=+=\vert+++\vert=}$(5.5)	**95.51**(5.5)	89.16(2.5)	**94.68**(5.5)	89.03(2.5)	84.61(0.5)	84.03(0.5)	**95.19**(5.5)
USPS	**98.26**$^{+++\vert+++\vert=}$(6.5)	97.92(4.5)	97.65(2.5)	97.94(4.0)	96.22(0.5)	97.90(4.0)	96.05(0.5)	**98.19**(5.5)
MNIST	**97.30**$^{=++\vert+++\vert=}$(6.0)	**97.30**(6.0)	96.03(2.5)	96.57(4.0)	95.77(2.5)	93.24(1.0)	84.02(0.0)	**97.62**(6.0)
Total Score	37	25.5	19.5	28.5	14.5	10	1	32.5



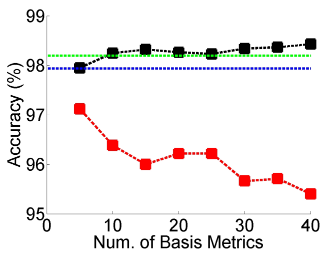
(a) Letter



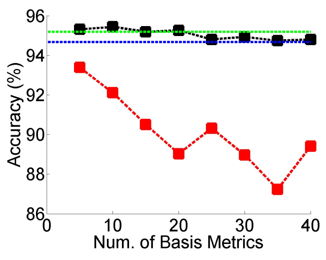
(b) Pendigits



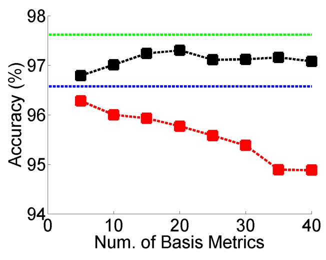
(c) Optdigits



(d) USPS



(e) Isolet



(f) MNIST

Figure 6.2: Accuracy results of PLML and CBLML with varying number of basis metrics.

To estimate the classification accuracy for Pendigits, Optdigits, Isolet and MNIST we used the default train and test split, for the other datasets we used 10-fold cross-validation. The statistical significance of the differences were tested with McNemar's test with a p-value of 0.05. In order to get a better understanding of the relative performance of the different algorithms for a given dataset we used a simple ranking schema in which an algorithm A was assigned one point if it was found to have a statistically significantly better accuracy than another algorithm B, 0.5 points if the two algorithms did not have a significant difference, and zero points if A was found to be significantly worse than B.

6.4.1 Results

In Table 6.1 we report the experimental results. PLML consistently outperforms the single global metric learning methods LMNN, BoostMetric and SML, for all datasets except Isolet on which its accuracy is slightly lower than that of LMNN. Depending on the single global metric learning method with which we compare it, it is significantly better in three, four, and five datasets (for LMNN, SML, and BoostMetric respectively), out of the six and never singificantly worse. When we compare PLML with CBLML and LMNN-MM, the two baseline methods which learn one local metric for each cluster and each class respectively with no smoothness constraints, we see that it is statistically significantly better in all the datasets. GLML fails to learn appropriate metrics on all datasets because its fundamental generative model assumption is often not valid. Finally, we see that PLML is significantly better than SVM in two out of the six datasets and it is never significantly worse; remember here that with SVM we also do inner fold kernel selection to automatically select the appropriate feature speace. Overall PLML is the best performing methods scoring 37 points over the different datasets, followed by SVM with automatic kernel selection and SML which score 32.5 and 28.5 points respectively. The other metric learning methods perform rather poorly.

Examining more closely the performance of the baseline local metric learning methods CBLML and LMNN-MM we observe that they tend to overfit the learning problems. This can be seen by their considerably worse performance with respect to that of SML and LMNN which rely on a single global model. On the other hand PLML even though it also learns local metrics it does not suffer from the overfitting problem due to the manifold regularization. The poor performance of LMNN-MM is not in agreement with the results reported in (105). The main reason for the difference is the experimental setting. In (105), 30% of the training instance of each dataset were used as a validation set to avoid overfitting.

To provide a better understanding of the behavior of the learned metrics, we applied PLML LMNN-MM, CBLML and GLML, on an image dataset containing instances of four different handwritten digits, zero, one, two, and four, from the MNIST dataset. As in (105), we use the two main principal components to learn. Figure 7.1 shows the learned local metrics by plotting the axis of their corresponding ellipses(black line). The direction of the longer axis is the more discriminative. Clearly PLML fits the data much better than LMNN-MM and as expected its local metrics vary smoothly. In terms of the predictive performance, PLML has the best with 82.76% accuracy. The CBLML, LMNN-MM and GLML have an almost identical performance with respective accuracies of 82.59%, 82.56% and 82.51%.

Finally we investigated the sensitivity of PLML and CBLML to the number of basis metrics, we experimented with $m \in \{5, 10, 15, 20, 25, 30, 35, 40\}$. The results are given in Figure 6.2. We see that the predictive performance of PLML often improves as we increase the number of the basis metrics. Its performance saturates when the number of basis metrics becomes sufficient to model the underlying training data. As expected different learning problems require different number of basis metrics. PLML does not overfit on any of the datasets. In contrast, the performance of CBLML gets worse when the number of basis metrics is large which provides further evidence that CBLML does indeed overfit the learning problems, demonstrating clearly the utility of the manifold regularization.

6.5 Conclusions

Local metric learning provides a more flexible way to learn the distance function. However they are prone to overfitting since the number of parameters they learn can be very large. In this chapter we presented PLML, a local metric learning method which regularizes local metrics to vary smoothly over the data manifold. Using an approximation error bound of the metric matrix function, we parametrize the local metrics by a weighted linear combinations of local metrics of anchor points. Our method scales to learning problems with tens of thousands of instances and avoids the overfitting problems that plague the other local metric learning methods. The experimental results show that PLML outperforms significantly the state of the art metric learning methods and it has a performance which is significantly better or equivalent to that of SVM with automatic kernel selection.

Chapter 7

Two-Stage Metric Learning

Over the last years various metric learning algorithms have been shown to perform well in different learning problems, however, each comes with its own set of limitations. In this chapter, we propose a novel two-stage metric learning algorithm, Similarity-Based Fisher Information Metric Learning (SBFIML). It has a few advantages comparing to existing metric learning methods. SBFIML is flexible and general; it can be applied to different types of data spaces with various non-negative similarity functions. Comparing to KML, SBFIML does not require the similarity measure to form a PSD matrix. Moreover, SBFIML can be interpreted as a local metric learning algorithm. Compared to the previous local metric learning algorithms which produce a non-metric distance (73; 101), the distance approximation in SBFIML is a well defined distance function with a closed form expression.

This chapter is organized as follows. In Section 7.1, we discuss the motivation of this work. In Section 7.2 we introduce statistical manifold and Fisher Information metric concepts to be self-contained. In Section 7.3 we present our two-stage metric learning algorithm. In Section 7.4 we present the experimental results and we conclude with Section 7.5.

7.1 Motivation

Over the last years various metric learning algorithms have been shown to perform well in different learning problems, however, each comes with its own set of limitations. Learning the distance metric with one global linear transformation is called single metric learning (24; 105). In this approach the distance computation is equivalent to applying on the learning instances a learned linear transformation followed by a standard distance metric computation in the projected space. Since the discriminatory power of the input features might vary locally, this approach is often not flexible enough to fit well the distance in different regions.

Local metric learning addresses this limitation by learning in each neighborhood one local metric (73; 101). When the local metrics vary smoothly in the feature space, learning local metrics is equivalent to learning the Riemannian metric on the data manifold (39). The main challenge here is that the geodesic distance endowed by the Riemannian metric is often computationally very expensive. In practice, it is approximated by assuming that the geodesic curves are formed by straight lines and the local metric does not change along these lines (73; 101). Unfortunately, the approximation does not satisfy the symmetric property and therefore the result is a non-metric distance.

Kernelized Metric Learning (KML) achieves flexibility in a different way (45; 99). In KML learning instances are first mapped into the Reproducing-Kernel Hilbert Space (RKHS) by a kernel function and then a global Mahalanobis metric is learned in the RKHS space. By defining the distance in the input feature space as the Mahalanobis distance in the RKHS space, KML is equivalent to learning a flexible non-linear distance in the input space. However, its main limitation is that the kernel matrix induced by the kernel function must be Positive Semi-Definite (PSD). Although Non-PSD kernel could be transformed into PSD kernel (17; 111), the new PSD kernel nevertheless cannot keep all original similarity information.

In this work, we propose a novel two-stage metric learning algorithm, Similarity-Based Fisher Information Metric Learning (SBFIML). It first maps instances from the data manifold into finite discrete distributions by computing their similarities to a number of predefined anchor points in the data space. Then, the Fisher information distance on the statistical manifold is used as the distance in the input feature space. This induces a new family of Riemannian distance metric in the input data space with two important properties. First, the new Riemannian metric is robust to density variation in the original data space. Without such robustness, an objective function can be easily biased towards data regions the density of which is low and thus dominates learning of the objective function. Second, the new Riemannian metric has largest distance discrimination on the manifold of anchor points and no distance in the directions being orthogonal to the manifold. So, the effect of locally irrelevant dimensions of anchor points is removed. To the best of our knowledge, this is the first metric learning algorithm that has these two important properties. We evaluate SBFIML on a number of datasets. The experimental results show that it outperforms in a statistically significant manner both metric learning methods and SVM.

7.2 Preliminaries

We are given a number of learning instances $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, where each instance $\mathbf{x}_i^T \in \mathcal{X}$ is a d-dimensional vector, and a vector of associated class labels $\mathbf{y} = (y_1, \dots, y_n)^T$, $y_i \in \{1, \dots, c\}$. We assume that the input feature space $\mathcal{X}$ is a smooth manifold. Different learning problems can have very different types of data manifolds with possibly different dimensionality. The most commonly used manifold in metric learning is the Euclidean space $\mathbb{R}^d$ (105). The probability simplex space $\mathcal{P}^{d-1}$ has also been explored (21; 53; 56).

We propose a general two-stage metric learning algorithm which can learn a flexible distance in different types of $\mathcal{X}$ data manifolds, e.g. Euclidean, probability simplex, hypersphere, etc. Concretely, we first map instances from $\mathcal{X}$ onto the statistical manifold $\mathcal{S}$ through a similarity-based differential map, which computes their non-negative similarities to a number of predefined anchor points. Then we define the Fisher information distance as the distance on $\mathcal{X}$. We have chosen to do so, since this induces a new family of Riemannian distance metric which enjoys interesting properties: 1) The new Riemannian metric is robust to density variations in the original data space, which can be produced for example by different intrinsic variabilities of the learning instances in the different categories. Distance learning over this new metric is hence robust to density variation. 2) The new Riemannian distance metric has largest distance discrimination on the manifold of the anchor points and has no distance in the directions being orthogonal to that manifold. So, the new distance metric can remove the effect of locally irrelevant dimensions of the anchor point manifold, see Figure 7.1 for more detials. In the remainder of this section, we will briefly introduce the necessary terminology and concepts. More details can be found in the monographs (2; 58).

Statistical Manifold. We denote by $\mathcal{M}^n$ a n-dimensional smooth manifold. For each point p on $\mathcal{M}^n$, there exists at least one smooth coordinate chart $(\mathcal{U}, \varphi)$ which defines a coordinate system to points on $\mathcal{U}$, where $\mathcal{U}$ is an open subset of $\mathcal{M}^n$ containing p and $\varphi : \mathcal{U} \longrightarrow \Theta$ is a smooth coordinate map $\varphi(p) = \theta \in \Theta \subset \mathbb{R}^n$. θ is the coordinate of p

defined by φ.

A statistical manifold is a smooth manifold whose points are probability distributions. Given a n-dimensional statistical manifold $\mathcal{S}^n$, we denote by $p(\xi|\theta)$ a probability distribution in $\mathcal{S}^n$, where $\theta = (\theta_1, \ldots, \theta_n) \in \Theta \subset \mathbb{R}^n$ is the coordinate of $p(\xi|\theta)$ under some coordinate map φ and ξ is the random variable of the $p(\xi|\theta)$ distribution taking values from some set Ξ. Note that, all the probability distributions in $\mathcal{S}^n$ share the same set Ξ.

In this work, we are particularly interested in the n-dimensional statistical manifold $\mathcal{P}^n$, whose points are finite discrete distributions, denoted by

$$\mathcal{P}^n = \{p(\xi|\theta = (\theta_1, \ldots, \theta_n)) : \sum_{i=1}^{n} \theta_i < 1, \forall i, \theta_i > 0\} \tag{7.1}$$

where ξ is the discrete random variable taking values in the set $\Xi = \{1, \ldots, n+1\}$ and $\theta \in \Theta \subset \mathbb{R}^n$ is called the m-affine coordinate (2). The probability mass of $p(\xi|\theta)$ is $p(\xi = i) = \theta_i$ if $i \neq n+1$, otherwise $p(\xi = n+1) = 1 - \sum_{k=1}^{n} \theta_k$.

Fisher Information Metric. The Fisher information metric is a Riemannian metric defined on statistical manifolds and endows a distance between probability distributions (76). The explicit form of the Fisher information metric at $p(\xi|\theta)$ is a $n \times n$ positive definite symmetric matrix $\mathbf{G}_{FIM}(\theta)$, the (i,j) element of which is defined by:

$$\mathbf{G}^{ij}_{FIM}(\theta) = \int_{\Xi} \frac{\partial \log p(\xi|\theta)}{\partial \theta_i} \frac{\partial \log p(\xi|\theta)}{\partial \theta_j} p(\xi|\theta) d\xi \tag{7.2}$$

where the above integral is replaced with a sum if Ξ is discrete. The following lemma gives the explicit form of the Fisher information metric on $\mathcal{P}^n$.

Lemma 6. *On the statistical manifold $\mathcal{P}^n$, the Fisher information metric $\mathbf{G}_{FIM}(\theta)$ at $p(\xi|\theta)$ with coordinate θ is*

$$\mathbf{G}^{ij}_{FIM}(\theta) = \frac{1}{\theta_i}\delta_{ij} + \frac{1}{1 - \sum_{k=1}^{n} \theta_k}, \forall i, j \in \{1, \ldots, n\} \tag{7.3}$$

where $\delta_{ij} = 1$ if $i = j$, otherwise $\delta_{ij} = 0$.

Properties of Fisher Information Metric. The Fisher information metric enjoys a number of interesting properties. First, the Fisher information metric is the unique Riemannian metric induced by all f-divergence measures, such as the Kullback-Leibler (KL) divergence and the χ^2 divergence (1). All these divergences converge to the Fisher information distance as the two probability distributions are approaching each other. Another important property of the Fisher information metric from a metric learning perspective is that the distance it endows can be approximated by the Hellinger distance, the cosine distance and all f-divergence measures (52). More importantly, when $\mathcal{S}^n$ is the statistical manifold of finite discrete distributions, e.g. $\mathcal{P}^n$, the cosine distance is exactly equivalent to the Fisher information distance (56; 59).

Pullback Metric. Let $\mathcal{M}^n$ and $\mathcal{N}^m$ be two smooth manifolds and $\mathcal{T}_p\mathcal{M}^n$ be the tangent space of $\mathcal{M}^n$ at $p \in \mathcal{M}^n$. Given a differential map $f : \mathcal{M}^n \longrightarrow \mathcal{N}^m$ and a Riemannian metric $\mathbf{G}$ on $\mathcal{N}^m$, the differential map f induces a pullback metric $\mathbf{G}^*$ at each point p on $\mathcal{M}^n$ defined by:

$$\langle \mathbf{v}_1, \mathbf{v}_2 \rangle_{\mathbf{G}^*(p)} = \langle D_p f(\mathbf{v}_1), D_p f(\mathbf{v}_2) \rangle_{\mathbf{G}(f(p))} \tag{7.4}$$

where $D_p f : \mathcal{T}_p\mathcal{M}^n \longrightarrow \mathcal{T}_{f(p)}\mathcal{N}^m$ is the differential of f at point $p \in \mathcal{M}^n$, which maps tangent vectors $\mathbf{v} \in \mathcal{T}_p\mathcal{M}^n$ to tangent vectors $D_p f(\mathbf{v}) \in \mathcal{T}_{f(p)}\mathcal{N}^m$.

Given the coordinate systems Θ and Γ of $\mathcal{U} \subset \mathcal{M}^n$ and $\mathcal{U}' \subset \mathcal{N}^m$ respectively, defined by some smooth coordinate maps $\varphi_{\mathcal{U}}$ and $\varphi_{\mathcal{U}'}$ respectively, then the explicit form of the pullback metric at point $p \in \mathcal{U} \subset \mathcal{M}^n$ with coordinate $\theta = \varphi_{\mathcal{U}}(p)$ is:

$$\mathbf{G}^*(\theta) = \mathbf{J}^T \mathbf{G}(\gamma) \mathbf{J} \tag{7.5}$$

where $\gamma = \varphi_{\mathcal{U}'}(f(p))$ is the coordinate of the $f(p) \in \mathcal{U}' \subset \mathcal{N}^m$ and $\mathbf{J}$ is the Jacobian matrix of the function $\varphi_{\mathcal{U}'} \circ f \circ \varphi_{\mathcal{U}}^{-1} : \Theta \longrightarrow \Gamma$ at point θ. Since $\mathbf{G}$ is a Riemannian metric, the pullback metric $\mathbf{G}^*$ is in general at least a PSD metric.

The following lemma gives the relation between the geodesic distances on $\mathcal{M}^n$ and $\mathcal{N}^m$.

Lemma 7. *Let $\mathbf{G}^*$ be the pullback metric of a Riemannian metric $\mathbf{G}$ induced by a differential map $f : \mathcal{M}^n \longrightarrow \mathcal{N}^m$, $d_{\mathbf{G}^*}(p', p)$ be the geodesic distance on $\mathcal{M}^n$ endowed by $\mathbf{G}^*$ and $d_{\mathbf{G}}(f(p'), f(p))$ the geodesic distance on $\mathcal{N}^m$ endowed by $\mathbf{G}$, then, it holds* $\lim_{p' \to p} \frac{d_{\mathbf{G}}(f(p'), f(p))}{d_{\mathbf{G}^*}(p', p)} = 1$

Proof. Let θ be the coordinate of $p \in \mathcal{U} \subset \mathcal{M}^n$ under some smooth coordinate map $\varphi_{\mathcal{U}}$ and γ be the coordinate of $f(p) \in \mathcal{U}' \subset \mathcal{N}^m$ under some smooth coordinate map $\varphi_{\mathcal{U}'}$. Since p' approaches p, we have $\theta' = \theta + d\theta$, where θ' is the coordinate of p' under the coordinate map $\varphi_{\mathcal{U}}$ and $d\theta$ is an infinitesimal small change approaching $\mathbf{0}$. Furthermore, since $f : \mathcal{M}^n \longrightarrow \mathcal{N}^m$ is a differential map, the function $\varphi_{\mathcal{U}'} \circ f \circ \varphi_{\mathcal{U}}^{-1} : \Theta \longrightarrow \Gamma$, which we will denote by g, is also differentiable. According to the Taylor expansion, we have $\gamma' = g(\theta') = g(\theta + d\theta) = g(\theta) + \bigtriangledown g(\theta) d\theta + R_g(d\theta, \theta) = \gamma + \mathbf{J} d\theta + R_g(d\theta, \theta)$, where γ' is the coordinate of $f(p')$ under the coordinate map $\varphi_{\mathcal{U}'}$, $\mathbf{J}$ is the Jacobian matrix of the function g at point θ and $R_g(d\theta, \theta)$

is the remainder term of linear approximation. Finally, according to the definition of pullback metric, we have $\lim_{d\theta \to \mathbf{0}} \frac{d_{\mathbf{G}(\gamma)}(\gamma',\gamma)}{d_{\mathbf{G}^*(\theta)}(\theta',\theta)} = \lim_{d\theta \to \mathbf{0}} \frac{(\mathbf{J}d\theta + R_g(d\theta,\theta))^T \mathbf{G}(\gamma)(\mathbf{J}d\theta + R_g(d\theta,\theta))}{d\theta^T \mathbf{J}^T \mathbf{G}(\gamma) \mathbf{J} d\theta} = 1$. This ends the proof. $\square$

In addition to approximating $d_{\mathbf{G}^*}(p', p)$ directly on $\mathcal{M}^n$ by assuming that the geodesic curve is formed by straight lines as previous local metric learning algorithms do (73; 101), Lemma 7 allows us to also approximate it with $d_{\mathbf{G}}(f(p'), f(p))$ on $\mathcal{N}^m$. Note that, both approximations have the same asymptotic convergence result.

7.3 Similarity-Based Fisher Information Metric Learning

We will now present our two-stage metric learning algorithm, SBFIML. In the following, we will first present how to define the similarity-based differential map $f : \mathcal{X} \longrightarrow \mathcal{P}$ and then how to learn the Fisher information distance.

7.3.1 Similarity-Based Differential Map

Given a number of anchor points $\{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$, $\mathbf{z}_i \in \mathcal{X}$, we denote by $s = (s_1, \ldots, s_n) : \mathcal{X} \longrightarrow \mathbb{R}^{+n}$ the differentiable similarity function. Each $s_k : \mathcal{X} \longrightarrow \mathbb{R}^+$ component is a differentiable function the output of which is a non-negative similarity between some input instance $\mathbf{x}_i$ and the anchor point $\mathbf{z}_k$. Based on the similarity function s we define the similarity-based differential map f as:

$$\begin{aligned} f(\mathbf{x}_i) \quad &= p(\xi|(\frac{s_1(\mathbf{x}_i)}{\sum_{k=1}^n s_k(\mathbf{x}_i)}, \ldots, \frac{s_{n-1}(\mathbf{x}_i)}{\sum_{k=1}^n s_k(\mathbf{x}_i)})) \\ &= (\bar{s}_1(\mathbf{x}_i), \ldots, \bar{s}_{n-1}(\mathbf{x}_i)) \end{aligned} \tag{7.6}$$

where $f(\mathbf{x}_i)$ is a finite discrete distribution on manifold $\mathcal{P}^{n-1}$. From now on, for simplicity, we will denote $f(\mathbf{x}_i)$ by $p^i(\xi)$. The probability mass of the kth outcome is given by: $p^i(\xi = k) = \bar{s}_k(\mathbf{x}_i) = \frac{s_k(\mathbf{x}_i)}{\sum_{k=1}^n s_k(\mathbf{x}_i)}$. In order for f to be a valid differential map, the similarity function s must satisfy $\sum_k s_k(\mathbf{x}_i) > 0$, $\forall \mathbf{x}_i \in \mathcal{X}$. This family of differential maps is very general and can be applied to any $\mathcal{X}$ space where a non-negative differentiable similarity function s can be defined. The finite discrete distribution representation, $p^i(\xi)$, of learning instance, $\mathbf{x}_i$, can be intuitively seen as an encoding of its neighborhood structure defined by the similarity function s. Note that, the idea of mapping instances onto the statistical manifold $\mathcal{P}$ has been previously studied in manifold learning, e.g. SNE (40) and t-SNE (93).

Akin to the appropriate choice of the kernel function in a kernel-based method, the choice of an appropriate similarity function s is also crucial for SBFIML. In principle, an appropriate similarity function s should be a good match for the geometrical structure of the $\mathcal{X}$ data manifold. For example, for data lying on the probability simplex space, i.e. $\mathcal{X} = \mathcal{P}^{d-1}$, the similarity functions defined either on $\mathbb{R}^d$ or on $\mathcal{P}^{d-1}$ can be used. However, the similarity function on $\mathcal{P}^{d-1}$ is more appropriate, because it exploits the geometrical structure of $\mathcal{P}^{d-1}$, which, in contrast, is ignored by the similarity function on $\mathbb{R}^d$ (53).

The set of anchor points $\{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ can be defined in various ways. Ideally, anchor points should be similar to the given learning instances $\mathbf{x}_i$, i.e. anchor points follow the same distribution as that of learning instances. Empirically, we can use directly training instances or cluster centers, the latter established by clustering algorithms. Similar to the current practice in kernel methods we will use in SBFIML as anchors points all the training instances.

Similarity Functions on $\mathbb{R}^d$. We can define the similarity on $\mathbb{R}^d$ in various ways. In this work we will investigate two types of differentiable similarity functions. The first one is based on the Gaussian function, defined as:

$$s_k(\mathbf{x}_i) = exp(-\frac{\|\mathbf{x}_i - \mathbf{z}_k\|_2^2}{\sigma_k}) \tag{7.7}$$

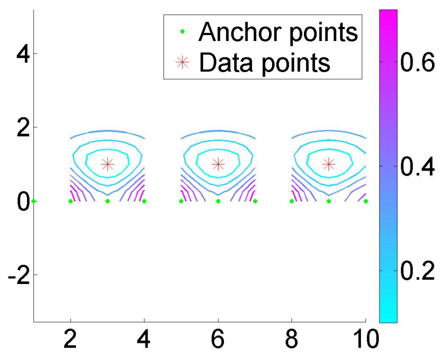
where $\|\cdot\|_2$ is the L_2 norm. σ_k controls the size of the neighborhood of the anchor point $\mathbf{z}_k$, with large values producing large neighborhoods. Note that the different σ_ks could be set to different values; if all of them are equal, this similarity function is exactly the Gaussian kernel. The second type of similarity function that we will look at is:

$$s_k(\mathbf{x}_i) = 1 - \frac{1}{\pi}\arccos(\frac{\mathbf{x}_i^T \mathbf{z}_k}{\|\mathbf{x}_i\|_2 \cdot \|\mathbf{z}_k\|_2}) \tag{7.8}$$

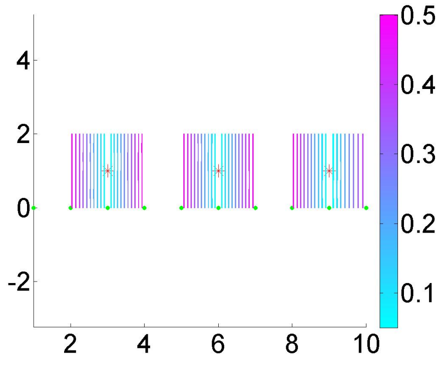
which measures the normalized angular similarity between $\mathbf{x}_i$ and $\mathbf{z}_k$. This similarity function can be explained as we first projecting all points from $\mathbb{R}^d$ to the hypersphere and then applying the angular similarity to points on a hypersphere. As a result, this similarity function is useful for data which approximately lie on a hypersphere. Note that this similarity function is also a valid kernel function (41).

One might say we can also achieve nonlinearity by mapping instances into the proximity space $\mathcal{Q}$ using the following similarity-based map $g : \mathcal{X} \longrightarrow \mathcal{Q}$:

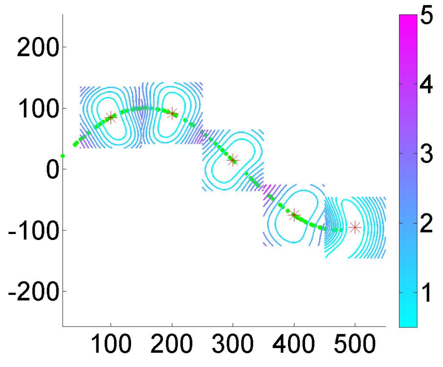
$$g(\mathbf{x}) = (s_1(\mathbf{x}), \ldots, s_n(\mathbf{x})) \tag{7.9}$$



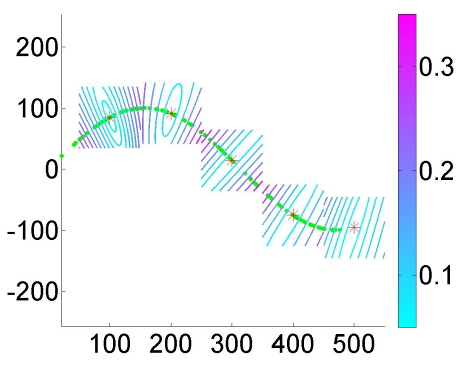
(a) $\mathbf{G}^*_{\mathcal{Q}}(\mathbf{x})$



(b) $\mathbf{G}^*_{\mathcal{P}}(\mathbf{x})$



(c) $\mathbf{G}^*_{\mathcal{Q}}(\mathbf{x})$



(d) $\mathbf{G}^*_{\mathcal{P}}(\mathbf{x})$

Figure 7.1: The visulization of equi-distance curves of pullback metrics $\mathbf{G}^*_{\mathcal{Q}}(\mathbf{x})$ and $\mathbf{G}^*_{\mathcal{P}}(\mathbf{x})$.

We now compare our similarity-based map f, equation 7.6 against the similarity-based map g, equation 7.9, in two aspects, namely representation robustness and pullback metric analysis.

Representation Robustness. Compared to the representation induced by the similarity-based map g, equation 7.9, our representation induced by the similarity-based map f, equation 7.6, is more robust to density variations in original data space, i.e. the density of the learning instances varies significantly between different regions. This can be explained by the fact that the finite discrete distribution is essentially a representation of the neighborhood structure of a learning instance normalized by a "scaling" factor, the sum of similarities of the learning instance to the anchor points. Hence the distance implied by the finite discrete distribution representation is less sensitive to the density variations of the different data regions. This is an important property. Without such robustness, an objective function based on raw distances can be easily biased towards data regions the density of which is low and thus dominates learning of the objective function. One example of this kind of objective is that of LMNN (105), which we will also use later in SBFIML to learn the Fisher information distance.

Pullback Metric Analysis. We also show how the two approaches differ by comparing the pullback metrics induced by the two similarity-based maps f and g. In doing so, we first need to specify the Riemannian metrics $\mathbf{G}_{\mathcal{Q}}$ in the proximity space $\mathcal{Q}$ and $\mathbf{G}_{\mathcal{P}}$ on the statistical manifold $\mathcal{P}^{n-1}$. Following the work of similarity-based learning (18), we use the Euclidean metric as the $\mathbf{G}_{\mathcal{Q}}$ in the proximity space $\mathcal{Q}$. On the statistical manifold $\mathcal{P}^{n-1}$ we use the Fisher information metric $\mathbf{G}_{FIM}$ defined in equation 7.3 as $\mathbf{G}_{\mathcal{P}}$. To simplify our analysis, we assume $\mathcal{X} = \mathbb{R}^d$. However, note that this analysis can be generalized to other manifolds, e.g. $\mathcal{P}^{d-1}$. We use the standard Cartesian coordinate system for points in $\mathbb{R}^d$ and $\mathcal{Q}$ and use m-affine coordinate system, equation 7.1, for points on $\mathcal{P}^{n-1}$.

The pullback metric induced by these two differential maps are given in the following lemma.

Lemma 8. *In $\mathbb{R}^d$, at $\mathbf{x}$ with Cartesian coordinate, the form of the pullback metric $\mathbf{G}^*_{\mathcal{Q}}(\mathbf{x})$ of the Euclidean metric induced by the differential map g of equation 7.9 is:*

$$\mathbf{G}^*_{\mathcal{Q}}(\mathbf{x}) = \nabla g(\mathbf{x}) \nabla g(\mathbf{x})^T = \sum_{i=1}^{n} \nabla s_i(\mathbf{x}) \nabla s_i(\mathbf{x})^T \tag{7.10}$$

where the vector $\nabla s_i(\mathbf{x})$ of size $d \times 1$ is the differential of ith similarity function $s_i(\mathbf{x})$. The form of the pullback metric $\mathbf{G}^*_{\mathcal{P}}(\mathbf{x})$ of the Fisher information metric induced by the differential map f of equation 7.6 is:

$$\mathbf{G}^*_{\mathcal{P}}(\mathbf{x}) \quad = \sum_{i=1}^{n} \frac{1}{\bar{s}_i(\mathbf{x})} (\nabla \bar{s}_i(\mathbf{x}) \nabla \bar{s}_i(\mathbf{x})^T) \tag{7.11}$$

where $\nabla \bar{s}_i(\mathbf{x}) = \bar{s}_i(\mathbf{x}) \left(\nabla \log(s_i(\mathbf{x})) - E\left(\nabla \log(s_i(\mathbf{x}))\right)\right)$ and the expectation of $\nabla \log(s_i(\mathbf{x}))$ is $E(\nabla \log(s_i(\mathbf{x}))) = \sum_{k=1}^{n} \bar{s}_k(\mathbf{x}) \nabla \log(s_i(\mathbf{x}))$.

Gaussian Similarity Function. The form of pullback metrics $\mathbf{G}^*_{\mathcal{Q}}(\mathbf{x})$ and $\mathbf{G}^*_{\mathcal{P}}(\mathbf{x})$ depends on the explicit form of the similarity function $s_i(\mathbf{x})$. We now study their differences using the Gaussian similarity function with kernel width σ, equation 7.7. We first show the difference between $\mathbf{G}^*_{\mathcal{Q}}(\mathbf{x})$ and $\mathbf{G}^*_{\mathcal{P}}(\mathbf{x})$ by comparing their m largest eigenvectors, the directions in which metrics have the largest distance discrimination.

The m largest eigenvectors $\mathbf{U}_{\mathcal{Q}}(\mathbf{x})$ of $\mathbf{G}^*_{\mathcal{Q}}(\mathbf{x})$ are:

$$\begin{aligned}\mathbf{U}_{\mathcal{Q}}(\mathbf{x}) &= \underset{\mathbf{U}^T\mathbf{U}=\mathbf{I}}{\arg\max}\, tr(\mathbf{U}^T\mathbf{G}^*_{\mathcal{Q}}(\mathbf{x})\mathbf{U}) \\ &= \underset{\mathbf{U}^T\mathbf{U}=\mathbf{I}}{\arg\max} \sum_{k=1}^{m}\sum_{i=1}^{n} \frac{4}{\sigma^2}(\mathbf{u}_k^T s_i(\mathbf{x})(\mathbf{x}-\mathbf{z}_i))^2\end{aligned} \tag{7.12}$$

where $tr(\cdot)$ is the trace norm and $\mathbf{u}_k$ is the kth column of matrix $\mathbf{U}$. The m largest eigenvectors $\mathbf{U}_{\mathcal{P}}(\mathbf{x})$ of the pullback metric $\mathbf{G}^*_{\mathcal{P}}(\mathbf{x})$ are:

$$\begin{aligned}\mathbf{U}_{\mathcal{P}}(\mathbf{x}) &= \underset{\mathbf{U}^T\mathbf{U}=\mathbf{I}}{\arg\max}\, tr(\mathbf{U}^T\mathbf{G}^*_{\mathcal{P}}(\mathbf{x})\mathbf{U}) \\ &= \underset{\mathbf{U}^T\mathbf{U}=\mathbf{I}}{\arg\max} \sum_{k=1}^{m}\sum_{i=1}^{n} \frac{4\bar{s}_i(\mathbf{x})}{\sigma^2}(\mathbf{u}_k^T(\mathbf{z}_i - E(\mathbf{z}_i))^2\end{aligned} \tag{7.13}$$

where $E(\mathbf{z}_i) = \sum_{k=1}^{n} \bar{s}_k(\mathbf{x})\mathbf{z}_k$

We see one key difference between $\mathbf{U}_{\mathcal{P}}(\mathbf{x})$ and $\mathbf{U}_{\mathcal{Q}}(\mathbf{x})$. In equation 7.13, $\mathbf{U}_{\mathcal{P}}(\mathbf{x})$ are the directions which maximize the sum of expected variance of $\mathbf{u}_k^T\mathbf{z}_i, k \in \{1,\dots,m\}$, with respected to its expected mean. In contrast, the directions of $\mathbf{U}_{\mathcal{Q}}(\mathbf{x})$ in equation 7.12 maximize the sum of the unweighted "variance" of $\mathbf{u}_k^T s_i(\mathbf{x})(\mathbf{x}-\mathbf{z}_i), k \in \{1,\dots,m\}$, without centralization. Their difference can be intuitively compared to the difference of doing local PCA with or without centralization. Therefore, $\mathbf{U}_{\mathcal{P}}(\mathbf{x})$ is closer to the principle directions of local anchor points. Second, since $\mathbf{G}^*_{\mathcal{P}}(\mathbf{x}) = \sum_{i=1}^{n} \frac{4\bar{s}_i(\mathbf{x})}{\sigma^2}(\mathbf{z}_i - E(\mathbf{z}_i))(\mathbf{z}_i - E(\mathbf{z}_i))^T$, it is also easy to show that $\mathbf{G}^*_{\mathcal{P}}(\mathbf{x})$ has no distance in the orthogonal directions of the affine subspace spanned by the weighted anchor points of $\bar{s}_i(\mathbf{x})\mathbf{z}_i$. So, $\mathbf{G}^*_{\mathcal{P}}(\mathbf{x})$ removes the effect of locally irrelevant dimensions to the anchor point manifold.

To show the differences of pullback metrics $\mathbf{G}^*_{\mathcal{Q}}(\mathbf{x})$ and $\mathbf{G}^*_{\mathcal{P}}(\mathbf{x})$ intuitively, we visualize their equi-distance curves in Figure 7.1, where the Guassian similarity function, euqation 7.7, is used to define the similarity maps in equations 7.9 and 7.6. As shown in Figure 7.1, we see that the pullback metric $\mathbf{G}^*_{\mathcal{P}}(\mathbf{x})$ emphasizes more the distance along the principle direction of the local anchor points than the pullback metric $\mathbf{G}^*_{\mathcal{Q}}(\mathbf{x})$. Furthermore, in Figure 7.1(b) we see that $\mathbf{G}^*_{\mathcal{P}}(\mathbf{x})$ has a zero distance in the direction being orthogonal to the manifold of anchor points, the straight line which the (green) anchor points lie on. Therefore, $\mathbf{G}^*_{\mathcal{P}}(\mathbf{x})$ is more discriminative on the manifold of the anchor points. To explore the effect of these differences, we also experimentally compare these two approaches in section 7.4 and the results show that learning the Fisher information distance on $\mathcal{P}$ outperforms in a significant manner learning Mahalanobis distance in proximity space $\mathcal{Q}$.

7.3.2 Large Margin Fisher Information Metric Learning

By applying on the learning instances the differential map f of equation (7.6) we map them on the statistical manifold $\mathcal{P}^{n-1}$. We are now ready to learn the Fisher information distance from the data.

Distance Parametrization. As discussed in section 7.2, the Fisher information distance on $\mathcal{P}^{n-1}$ can be exactly computed by the cosine distance (56; 59):

$$d_{FIM}(\mathbf{p}^i, \mathbf{p}^j) = 2 \arccos(\sqrt{\mathbf{p}^i}^T \sqrt{\mathbf{p}^j}) \tag{7.14}$$

where $\mathbf{p}^i$ is the probability mass vector of the finite discrete distribution $p^i(\xi)$. To parametrize the Fisher information distance, we apply on the probability mass vector $\mathbf{p}^i$ a linear transformation $\mathbf{L}$. The intuition is that, the effect of the optimal linear transformation $\mathbf{L}$ is equivalent to locating a set of hidden anchor points such that the data's similarity representation is the same as the transformed representation. Thus the parametric Fisher information distance is defined as:

$$\begin{aligned} d_{FIM}(\mathbf{L}\mathbf{p}^i, \mathbf{L}\mathbf{p}^j) = \quad & 2 \arccos(\sqrt{\mathbf{L}\mathbf{p}^i}^T \sqrt{\mathbf{L}\mathbf{p}^j}) \\ s.t. \quad & \mathbf{L} \geq 0, \sum_i L_{ij} = 1, \forall j \end{aligned} \tag{7.15}$$

$\mathbf{L}$ has size $k \times n$. k is the number of hidden anchor points. To speedup the learning process, in practice we often learn a low rank linear transformation matrix $\mathbf{L}$ with small k. The constraints $\mathbf{L} \geq 0$ and $\sum_i L_{ij} = 1, \forall j$ are added to ensure that each $\mathbf{L}\mathbf{p}^i$ is still a finite discrete distribution on the manifold $\mathcal{P}^{k-1}$.

Learning. We will follow the large margin metric learning approach of (105) and define the optimization problem of learning $\mathbf{L}$ as:

$$\begin{aligned} \min_{\mathbf{L}} \quad & \sum_{ijk \in C(i,j,k)} [\epsilon_{ijk}]_+ + \alpha \sum_{i,j \to i} d_{FIM}(\mathbf{L}\mathbf{p}^i, \mathbf{L}\mathbf{p}^j) \\ s.t. \quad & \mathbf{L} \geq 0 \\ & \sum_i L_{ij} = 1; \ \forall j \\ & \epsilon_{ijk} = d_{FIM}(\mathbf{L}\mathbf{p}^i, \mathbf{L}\mathbf{p}^j) + \gamma - d_{FIM}(\mathbf{L}\mathbf{p}^i, \mathbf{L}\mathbf{p}^k) \end{aligned} \tag{7.16}$$

where α is a parameter that balances the importance of the two terms. Unlike LMNN (105), the margin parameter γ is added in the large margin triplet constraints following the work of (53), since the cosine distance is not linear with $\mathbf{L}^{\mathbf{T}}\mathbf{L}$. The large margin triplet constraints $C(i,j,k)$ for each instance $\mathbf{x}_i$ are generated using its k_1 same-class nearest neighbors and its k_2 different-class nearest neighbors in the $\mathcal{X}$ space and constraining the distance of each instance to its k_2 different class neighbors to be larger than those to its k_1 same class neighbors with γ margin. In the objective function of (7.16) the matrix $\mathbf{L}$ is learned by minimizing the sum of the hinge losses and the sum of the pairwise distances of each instance to its k_1 same-class nearest neighbors.

Optimization. Since the cosine distance defined in equation (7.14) is not convex, the optimization problem (7.16) is not convex. However, the constraints on matrix $\mathbf{L}$ are linear and we can solve this problem using a projected sub-gradient method. At each iteration, the main computation is the sub-gradient computation with complexity $O(mnk)$, where m is the number of large margin triplet constraints. n and k are the dimensions of the $\mathbf{L}$ matrix. The simplex projection operator on matrix $\mathbf{L}$ can be efficiently computed with complexity $O(nk\log(k))$ (27). Note that, learning distance metric on $\mathcal{P}$ has been previously studied by Riemannian Metric Learning (RML) (56) and χ^2-LMNN (53). In χ^2-LMNN, a symmetric χ^2 distance on $\mathcal{P}$ is learned with large margin idea similar to problem 7.16. SBFIML differs from χ^2-LMNN in that it uses the cosine distance to measure the distance on $\mathcal{P}$. As described in section 7.2, the cosine distance is exactly equivalent to the Fisher information distance on $\mathcal{P}$, while the χ^2 distance is only an approximation. In contrast to SBFIML and χ^2-LMNN, the work of RML focuses on unsupervised Fisher information metric learning. More importantly, both RML and χ^2-LMNN can only be applied in problems in which the input data lie on $\mathcal{P}$, while SBFIML can be applied to general data manifolds via the similarity-based differential map. Finally, note that SBFIML can also be applied to problems where we only have access to the pairwise instance similarity matrix, since it needs only the probability mass of finite discrete distributions as its input.

Local Metric Learning View of SBFIML. SBFIML can also be interpreted as a local metric learning algorithm. SBFIML defines the local metric on $\mathcal{X}$ as the pull-back metric of the Fisher information metric induced by the following similarity-based parametric differential map $f_{\mathbf{L}} : \mathcal{X} \longrightarrow \mathcal{P}^{k-1}$:

$$f_{\mathbf{L}}(\mathbf{x}_i) = \mathbf{L} \cdot \mathbf{p}^i, s.t. \quad \mathbf{L} > 0, \textstyle\sum_i L_{ij} = 1, \forall j \tag{7.17}$$

where as before $\mathbf{p}^i$ is the probability mass vector of the finite discrete distribution $p^i(\xi)$ defined in equation (7.6). SBFIML learns the local metric by learning the parameters of $f_{\mathbf{L}}$. The explicit form of the pullback metric $\mathbf{G}^*$ can be computed according to the equation (7.5). Given the pullback metric we can approximate the geodesic distance on $\mathcal{X}$ by assuming that the geodesic curves are formed by straight lines as local metric learning methods (73; 101) do, which would result in a non-metric distance. However, Lemma 7 allows us to approximate the geodesic distance on $\mathcal{X}$ by the Fisher information distance on $\mathcal{P}^{k-1}$. SBFIML follows the latter approach. Compared to the non-metric distance approximation, this new distance is a well defined distance function which has a closed form expression. Furthermore, this new distance approximation has the same asymptotic convergence result as the non-metric distance approximation.

Table 7.1: Mean and standard deviation of 5 times 10-fold CV accuracy results on $\mathbb{R}^d$ datasets. The superscripts $^{+-=}$ next to the accuracies of SBFIML indicate the result of the Student's t-test with SBMML,χ^2 LMNN, LMNN, GLML, PLML, KML and SVM. They denote respectively a significant win, loss or no difference for SBFIML. The **bold** entries for each dataset have no significant difference from the best accuracy for that dataset. The number in the parenthesis indicates the score of the respective algorithm for the given dataset based on the pairwise comparisons of the Student's t-test.

Datasets(#Inst./#Feat./#Class)	SBFIML	SBMML	χ^2 LMNN	LMNN	GLML	PLML	KML	SVM
stk25(208/172/2)	**81.6±1.8**$^{==+++==}$(5.0)	**81.7±3.0**(5.0)	**80.9±1.4**(5.0)	75.7±2.0(1.5)	72.7±1.8(0.5)	74.4±3.6(1.0)	**81.9±2.7**(5.0)	**81.2±1.0**(5.0)
wpbc(198/33/2)	**79.6±1.0**$^{==+++=+}$(5.5)	**79.3±1.1**(5.5)	**78.8±1.7**(4.5)	73.6±1.7(0.5)	76.5±1.6(3.0)	71.7±1.6(0.5)	**79.7±1.2**(5.5)	77.3±0.6(3.0)
wine(178/13/3)	**98.0±1.0**$^{===+===}$(4.0)	**98.3±0.4**(5.0)	97.4±0.3(3.5)	97.3±0.5(3.5)	96.1±1.1(0.5)	**97.5±1.1**(3.5)	**98.1±0.6**(4.0)	**98.1±0.6**(4.0)
sonar(208/60/2)	**87.2±1.3**$^{==+====}$(4.0)	**87.1±2.0**(3.5)	**86.4±2.0**(3.5)	84.8±1.5(2.0)	87.1±0.7(3.5)	86.1±1.4(3.0)	**86.9±2.2**(3.5)	**88.1±0.2**(5.0)
musk(476/166/2)	**96.1±0.4**$^{++=++++}$(6.5)	95.5±0.2(4.5)	94.8±0.5(3.0)	**95.8±0.5**(5.0)	91.3±0.6(0.5)	90.9±0.3(0.5)	95.3±0.2(4.0)	94.9±0.7(4.0)
wdbc(569/30/2)	97.2±0.4$^{-=++=-=}$(3.5)	**97.9±0.3**(6.0)	**97.5±0.5**(4.5)	96.4±0.2(1.0)	96.1±0.4(0.5)	96.8±0.5(3.0)	**97.9±0.3**(6.0)	97.3±0.2(3.5)
balance(625/4/3)	**97.5±0.5**$^{++++++=}$(6.5)	96.6±0.3(4.0)	96.2±0.5(4.0)	90.2±0.8(1.5)	88.8±0.5(0.0)	91.8±2.0(1.5)	96.6±0.3(4.0)	**97.7±0.5**(6.5)
breast(683/10/2)	**96.7±0.3**$^{==+=+==}$(4.5)	**96.4±0.5**(4.0)	**96.9±0.3**(5.0)	95.8±0.4(1.0)	96.4±0.2(3.5)	95.1±0.7(0.5)	**96.5±0.4**(4.5)	**96.9±0.2**(5.0)
australian(690/14/2)	84.6±0.3$^{++++++-}$(6.0)	80.5±0.9(2.0)	83.5±0.5(5.0)	81.2±1.0(2.0)	80.5±0.8(2.0)	80.2±1.0(2.0)	80.8±0.6(2.0)	**85.7±0.9**(7.0)
vehicle(846/18/4)	79.2±0.6$^{+==+-+=}$(4.5)	75.7±1.1(1.0)	78.4±1.3(4.0)	79.6±0.9(4.5)	77.3±0.8(2.5)	**81.3±0.5**(6.5)	76.1±1.2(1.5)	**78.0±7.3**(3.5)
Total Score	50.0	40.5	42.0	22.5	16.5	22.0	40.0	46.5

7.4 Experiments

We will evaluate the performance of SBFIML on ten datasets from the UCI Machine Learning and mldata[1] repositories. The details of these datasets are reported in the first column of Table 7.1. All datasets are preprocessed by standardizing the input features. We compare SBFIML against three metric learning baseline methods: LMNN (105)[2], KML (99)[3], GLML (73), and PLML (101). The former two learn a global Mahalanobis metric in the input feature space $\mathbb{R}^d$ and the RKHS space respectively, and the last two learn smooth local metrics in $\mathbb{R}^d$. In addition, we also compare SBFIML against Similarity-based Mahalanobis Metric Learning (SBMML) to see the difference of pullback metrics $\mathbf{G}^*_{\mathcal{Q}}(\mathbf{x})$, equation 7.10, and $\mathbf{G}^*_{\mathcal{P}}(\mathbf{x})$, equation 7.11. SBMML learns a global Mahalanobis metric in the proximity space $\mathcal{Q}$. Similar to SBFIML, the metric is learned by optimizing the problem 7.16, in which the cosine distance is replaced by Mahalanobis distance. The constraints on $\mathbf{L}$ in problem 7.16 are also removed. To see the difference between the cosine distance used in SBFIML and the χ^2 distance used in χ^2 LMNN, we compare SBFIML against χ^2 LMNN. Note that, both methods solve exactly the same optimization problem 7.16 but with different distance computations. Finally, we also compare SBFIML against SVM for binary classification problems and against multi-class SVMs for multiclass classification problems. In multi-class SVMs, we use the one-against-all strategy to determine the class label.

KML, SBMML and χ^2 LMNN learn a $n \times n$ PSD matrix and are thus computationally expensive for datasets with large number of instances. To speedup the learning process, similar to SBFIML, we can learn a low rank transformation matrix $\mathbf{L}$ of size $k \times n$. For all methods, KML, SBMML, χ^2 LMNN and SBFMIL, we set $k = 0.1n$ in all experiments. The matrix $\mathbf{L}$ in KML and SBMML was initialized by clipping the $n \times n$ identity matrix into the size of $k \times n$. In a similar manner, in χ^2 LMNN and SBFIML the matrix $\mathbf{L}$ was initialized by applying on the initialization matrix $\mathbf{L}$ in KML a simplex projector which ensures the constraints in problem (7.16) are satisfied.

The LMNN has one hyper-parameter μ (105). We set it to its default value $\mu = 1$. As in (73), GLML uses the Gaussian distribution to model the learning instances of a given class. The hyper-parameters of PLML was set following (101). The SBFIML has two hyper-parameters α and γ. Following LMNN (105), we set the α parameter to 1. We select the margin parameter γ from $\{0.0001, 0.001, 0.01, 0.1\}$ using a 4-fold inner Cross

[1]http://mldata.org/.
[2]http://www.cse.wustl.edu/∼kilian/code/code.html.
[3]http://cui.unige.ch/∼wangjun/.

Validation (CV). The selection of an appropriate similarity function is crucial for SBFIML. We choose the similarity function with a 4-fold inner CV from the angular similarity, equation (7.8), and the Gaussian similarity in equation (7.7). We examine two types of Gaussian similarity. In the first we set all σ_k to σ which is selected from $\{0.5\tau, \tau, 2\tau\}$, τ was set to the average of all pairwise distances. In the second we set the σ_k for each anchor point $\mathbf{z}_k$ separately; the σ_k was set by making the entropy of the conditional distribution $p(\mathbf{x}_i|\mathbf{z}_k) = \frac{s_k(\mathbf{x}_i)}{\sum_{i=1}^{n} s_k(\mathbf{x}_i)}$ equal to $\log(nc)$ (40), where n is the number of training instances and c was selected from $\{0.8, 0.9, 0.95\}$.

Since χ^2 LMNN and SBFIML apply different distance parametrizations to solve the same optimization problem, the parameters of χ^2 LMNN are set in exactly the same way as SBFIML, except that the margin parameter γ of χ^2 LMNN was selected from $\{10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}\}$, because χ^2 LMNN uses the squared χ^2 distance (53). The best similarity map for χ^2 LMNN is also selected using a 4-fold inner CV from the same similarity function set as that of SBFIML.

Akin to SBFIML, the performance of KML and SVM depends heavily on the selection of the kernel. We select automatically the best kernel with a 4-fold inner CV. The kernels are chosen from the linear, the set of polynomial (degree 2,3 and 4), the angular similarity, equation (7.8), and the Gaussian kernels with widths $\{0.5\tau, \tau, 2\tau\}$, as in SBFIML τ was set to the average of all pairwise distances. In addition, we also select the margin parameter γ of KML from $\{0.01, 0.1, 1, 10, 100\}$. The C parameter of SVM was selected from $\{0.01, 0.1, 1, 10, 100\}$. SBMML does not have any constraints on the similarity function, thus we select its similarity function with a 4-fold inner CV from a set which includes all kernel and similarity functions used in SBFIML and KML. As in KML, we select the margin parameter γ of SBMML from $\{0.01, 0.1, 1, 10, 100\}$. For all methods, except GLML and SVM which do not involve triplet constraints, the triplet constraints are constructed using three same-class and ten different-class nearest neighbors for each learning instance. Finally, we use the 1-NN rule to evaluate the performance of the different metric learning methods.

To estimate the classification accuracy we used 5 times 10-fold CV. The statistical significance of the differences were tested using Student's t-test with a p-value of 0.05. In order to get a better understanding of the relative performance of the different algorithms for a given dataset we used a simple ranking schema in which an algorithm A was assigned one point if it was found to have a statistically significantly better accuracy than another algorithm B, 0.5 points if the two algorithms did not have a significant difference, and zero points if A was found to be significantly worse than B.

Results. In Table 7.1 we report the accuracy results. We see that SBFIML out-

Table 7.2: Accuracy results on large datasets.

Datasets(#Inst./#Feat./#Class)	SBFIML	SBMML	χ^2 LMNN
German(1000/20/2)	69.40$^{==}$(1.0)	69.30(1.0)	69.10(1.0)
Image(2310/18/2)	98.05$^{==}$(1.0)	98.18(1.0)	97.79(1.0)
Splice(3175/60/2)	**90.93**$^{+=}$(1.5)	90.55(0.5)	90.87(1.0)
Isolet(7797/617/26)	95.45$^{==}$(1.0)	95.19(1.0)	95.70(1.0)
Pendigits(10992/16/10)	**98.08**$^{++}$(2.0)	97.68(0.5)	97.77(0.5)
Total Score	6.5	4.0	4.5

performs in a statistical significant manner the single metric learning method LMNN and the local metric learning methods, GLML and PLML, in seven, eight and six out of ten datasets respectively. When we compare it to KML and SBMML, which learn a Mahalanobis metric in the RKHS and proximity space, respectively, we see that it is significantly better than KML and SBMML in four datasets and significantly worse in one dataset. Compared to χ^2 LMNN, SBFIML outperforms χ^2-LMNN on eight datasets, being statistically significant better on three, and it never loses in statistical significant manner. Finally, compared to SVM, we see that SBFIML is significantly better in two datasets and significantly worse in one dataset. In terms of the total score, SBFIML achieves the best predictive performance with 50 point, followed by SVM ,which scores 46.5 point, and χ^2-LMNN with 42 point. The local metric learning method GLML is the one that performs the worst. A potential explanation for the poor performance of GLML could be that its Gaussian distribution assumption is not that appropriate for the datasets we experimented with.

To provide a better understanding of the predictive performance difference between SBFIML, SBMML, and χ^2 LMNN, we applied them on five large datasets. To speedup the learning process, we use as anchor points 20% of randomly selected training instances. Moreover, the parameter k of low rank transformation matrix $\mathbf{L}$ was reduced to $k = 0.05n$, where n is the number of anchor points. The kernel function and similarity map was selected using 4-fold inner CV. The classification accuracy of Isolet and Pendigits are estimated by the default train and test split, for other three datasets we used 10-fold cross-validation. The statistical significance of difference were tested with McNemar's test with p-value of 0.05.

The accuracy results are reported in Table 7.2. We see that SBFIML achieves statistical significant better accuracy than SBMML on the two datasets, Splice and Pendigits. When compare it to χ^2 LMNN, we see it is statistical significant better on one dataset, Pendigits. In terms of total score, SBFIML achieves the best score, 6.5 points, followed by χ^2 LMNN.

7.5 Conclusion

In this chapter we present a two-stage metric learning algorithm SBFIML. It first maps learning instances onto a statistical manifold via a similarity-based differential map and then defines the distance in the input data space by the Fisher information distance on the statistical manifold. This induces a new family of distance metrics in the input data space with two important properties. First, the induced metrics are robust to density variations in the original data space and second they have largest distance discrimination on the manifold of the anchor points. Furthermore, by learning a metric on the statistical manifold SBFIML can learn distances on different types of input feature spaces. The similarity-based map used in SBFIML is natural and flexible; unlike KML it does not need to be PSD. In addition SBFIML can be interpreted as a local metric learning method with a well defined distance approximation. The experimental results show that it outperforms in a statistical significant manner both metric learning methods and SVM.

Chapter 8

Conclusion and Future Work

8.1 Conclusions

The choice of distance metric is essential for machine learning algorithms. Distance metric learning addresses this challenge by learning a data-dependent distance metric. In this thesis, we focus on distance metric learning for nearest neighbor classification. Four metric learning algorithms are proposed to optimize the nearest neighbor classification.

In the work of Learning Neighborhood for Metric Learning (LNML), we propose a novel formulation of metric learning problem that includes in the learning process the learning of the local target neighborhood relations. This lifts the limitation that most metric learning algorithms predefines the target neighborhood, which are not necessarily optimal, and keep it fixed in the learning process. A two-step iterative optimization algorithm is proposed to learn the target neighborhood and the distance metric alternatively. However, the main limitation of LNML is that it learns a single linear distance metric for a learning problem. As a result, it might not perform well for learning problem with nonlinear complexity. Obviously, to address this limitation, a nonlinear distance metric learning algorithm is necessary.

Kernelized metric learning is one approach to achieve nonlinear distance learning. Its basic idea is to learn a linear distance metric in a nonlinear feature space induced by a kernel function. This results a nonlinear distance metric in the original input data space. However, how to select an appropriate kernel function is crucial for kernelized metric learning. In the work of Metric Learning with Multiple Kernels (MKML), we address the kernel selection problem by multiple kernel learning. It learns together a kernel function and a distance metric in the feature space induced by the learned kernel function. The kernel function we learn is a linear combination of a number of predefined kernels. Similar to LNML, a two-step iterative learning algorithm is proposed to learn alternatively the linear combination weight and distance metric.

Another approach to achieve nonlinear distance metric learning is local metric learning. Unlike single metric learning learns one single distance metric for the whole input data space, local metric learning learns a metric on each neighborhood. However, most previous works learn a number of local unrelated distance metrics. With the increasing of model complexity, these algorithms are easy to overfit the learning problem. In the work of Parametric Local Metric Learning (PLML), we address the overfitting problem by learning a smooth metric tensor function in the input data space. Since learning directly the metric tensor function is computational expensive, we approximate it by learning local metrics as linear combinations of basis metrics defined on anchor points over different regions of input data space; where the linear combination weight is learned by optimizing an approximation error bound. Such that, only the linear combination weight and basis metrics are necessarily learned. An efficient two-stage learning algorithm is proposed to learn first the linear combination weight and then the distance metrics of anchor points.

Finally, in the work of two-stage metric learning, we propose a novel nonlinear Riemannian metric learning algorithm, namely Similarity-Based Fisher Information Metric Learning (SBFIML). It first maps instance from the input data manifold into finite discrete distributions by computing non-negative simialrities to a number of predefined anchor points on the input data manifold. Then, the Fisher information distance on the statistical manifold is used as the nonlinear distance in the input data space. One advantage of SBFIML is that it is robust to the density variation in the original input data space. Furthermore, unlike kernelized metric learning, SBFIML does not require the similarity function being a PSD kernel. Finally, it can be interpreted as a local metric learning algorithm which has a closed form expression for distance approximation.

In these four works, we have continuously improved the state-of-the-art predictive performance of nearest neighbor classification with the help of learning a data-dependent distance metric.

8.2 Future Works

In this section, we discuss the limitations of current distance metric learning algorithms and propose some open issues as future works.

8.2.1 Limitations

We see the following two main limitations of the current distance metric learning algorithms.

Representation Learning. Distance metric learning in fact can be seen as a two-step process. In the first step, we map instance from the input space into another space, i.e. learning a new representation. Then, in the second step, we compute the distance in the new space. For instance, in the approach of MKML, chapter 5, we learn the new representation of instances by multiple kernel learning and then a Mahalanobis distance metric is learned in the new feature space induced by the learned kernel function. However, with few exceptions, so far most distance metric learning algorithms only focus on the second step, i.e. learning a good distance metric in a given data space. Since a good representation is crucial for the latter distance learning step, we see the missing of representation learning in current distance metric learning algorithms is one of the major limitations.

Non-Metric Learning. Distance metric learning algorithms often learn a distance metric in a given input data space to satisfy predefined (dis)similarity or triplet large margin constraints. As a result, it implicitly assumes that there exists an appropriate distance metric that can fit well the relationship between learning instances. However, this assumption might not hold in some learning problems. For example, in the problem of learning a dissimilarity measure of songs, two songs could be labeled as being similar due to one of the following reasons, 1) the two songs are from the same genre, 2) the two songs are in the same album, and 3) two songs are describing the similar story. In this scenario, if the reason of $\mathbf{x}_i$ and $\mathbf{x}_j$ being similar is different from that of $\mathbf{x}_j$ and $\mathbf{x}_k$ being similar. The relationship between $\mathbf{x}_i$ and $\mathbf{x}_k$ might be dissimilar. In these problems, an appropriate dissimilarity measure between learning instances is not any more a valid metric. Therefore, we see the missing of non-metric dissimilarity learning is another limitation of current distance metric learning algorithms.

8.2.2 Representation Learning

To address the challenge of representation learning in distance metric learning, one basic idea is to learn a new instance representation through dictionary learning method (63). Dictionary learning is a matrix factorization based nonlinear representation learning method. Most often, it learns the new representation of learning instances by the following optimization problem,

$$\min_{\boldsymbol{\Theta}} \quad \sum_i \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\Theta}_i\|_2^2 + \lambda\|\boldsymbol{\Theta}\|_1 \tag{8.1}$$

where $\boldsymbol{\Theta}$ is the new representation of learning instances. Its ith column corresponds to representation instance $\mathbf{x}_i$. Matrix $\mathbf{D}$ is the dictionary matrix, where its ith column is the ith basis. Note that, to encourage a sparse representation, L_1 norm regularization is added on matrix $\boldsymbol{\Theta}$. Intuitively, the idea of dictionary learning is to learn a sparse representation for all the learning instances. This is accord with the assumption of manifold data, which assumes data points are sampled from low dimensional manifold and then embedded into a high dimensional space. As a result, dictionary learning is often used to learn a new sparse representation of manifold data. By using dictionary learning to learn new instance representation, distance metric learning algorithms can naturally be extended to learn distance for manifold data.

Another idea is to learn a new instance representation through gradient boosted trees (29). Gradient boosted trees is a nonlinear nonparametric regression method. As a result, it can be used to learn a nonlinear representations of learning instances. Specifically, it learns a new representation $\theta(\mathbf{x}_i)$ for each instance $\mathbf{x}_i$ by learning a number of trees,

$$\theta(\mathbf{x}_i) = \sum_k \gamma_k \theta_k(\mathbf{x}_i)$$

where $\theta_k(\mathbf{x}_i)$ is the output of the kth tree. γ_k is the weight for the output of kth tree. At kth step, the kth tree is learned by fitting the pairs, defined by

$$\{(\mathbf{x}_1, \mathbf{z}_1), \ldots, (\mathbf{x}_n, \mathbf{z}_n)\}$$

where

$$\mathbf{z}_i = \frac{\partial L(\theta(\mathbf{x}_i), y_i)}{\partial \theta(\mathbf{x}_i)}\Big|_{\theta(\mathbf{x}_i)=\sum_{j=1}^{k-1}\theta_j(\mathbf{x}_i)}$$

is the partial derivative of the loss function L with respect to $\theta(\mathbf{x}_i)$. Intuitively, the methodology of gradient boosted tree minimizes the loss function L by at each step learning a weak regression tree to model the partial derivative of each instances and then updating the parameters by a global step size. Comparing to dictionary learning, gradient boosted trees is computational efficient and can easily handle millions of learning instances. Using gradient boosted trees to learn new instance representation, distance metric learning can be extended to learn nonlinear distance for very large scale data.

Learning the representation of instances in fact is learning a nonlinear mapping function whose outputs are the new instance representations. This adds distance metric learning algorithms powerful tool to manipulate data. However, with increasing model complexity, the learned model may easily overfit the data. To prevent from overfitting, various regularization techniques could be useful to constrain the new representations. For instance, the smooth regularization can be used to preserve the local geometry structure of instances and reduce the variance of mapping function. Also, as discussed in dictionary learning, sparse regularization can used to ensure each instance only has few active coordinates, which follows the assumption of manifold data. In short, the representation learning should be well regularized with respected to the structural prior information about data in hand.

8.2.3 Non-Metric Dissimilarity Learning

To address the non-metric dissimilarity learning challenge, the key problem is to learn a family of dissimilarity measure satisfies

$$\begin{aligned} & Dis(\mathbf{x}_i, \mathbf{x}_k) \text{ is large.} \\ s.t. \quad & Dis(\mathbf{x}_i, \mathbf{x}_j) \text{ and } Dis(\mathbf{x}_j, \mathbf{x}_k) \text{ are small.} \end{aligned}$$

where $Dis(\mathbf{x}_i, \mathbf{x}_k)$ is the dissimilarity between instances $\mathbf{x}_i$ and $\mathbf{x}_j$.

One idea is to learn a (p, q) Minkowski dissimilarity, defined as,

$$Dis(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{p}(x_{ik} - x_{jk})^2 - \sum_{k=p+1}^{p+q}(x_{ik} - x_{jk})^2 = (\mathbf{x}_i - \mathbf{x}_j)^T \begin{bmatrix} \mathbf{I}_p & 0 \\ 0 & -\mathbf{I}_q \end{bmatrix} (\mathbf{x}_i - \mathbf{x}_j) \quad (8.2)$$

where $p + q = d$ is the number of input features of learning instances. $\mathbf{I}_p$ is a $p \times p$ identity matrix. It is easy to verify the Minkowski dissimilarity is symmetric. In term of dissimilarity computation, it is very similar to the Euclidean distance computation, but it has negative contribution for the last q component. Thus, a negative dissimilarity is possible between two points. Note that, this is different from the Minkowski distance discussed in section 2.1.

To learn the Minkowski dissimilarity, we can define the parametric Minkowski dissimilarity as

$$Dis_{\mathbf{L}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_j)^T \begin{bmatrix} \mathbf{I}_p & 0 \\ 0 & -\mathbf{I}_q \end{bmatrix} (\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_j) \tag{8.3}$$

Then, similar to Mahalanobis distance metric learning, the Minkowski dissimilarity learning can be learned with respected to similarity and dissimilarity constraints. One of small drawback here is that the signature (p, q) should be selected appropriately.

This methodology is quite general. It can be applied in a similar manner to all learning problems, whose targets are to learn a model with respect to non-metric similarity and dissimilarity constraints. For instance, one potential application is to learn the binary hash function for large scale information indexing (74).

Bibliography

[1] S. Amari and A. Cichocki. Information geometry of divergence functions. *Bulletin of the Polish Academy of Sciences: Technical Sciences*, 58(1):183–195, 2010.

[2] S. Amari and H. Nagaoka. *Methods of information geometry*, volume 191. Amer Mathematical Society, 2007.

[3] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Convex optimization with sparsity-inducing norms. *Optimization for Machine Learning.*

[4] Francis R Bach, Gert RG Lanckriet, and Michael I Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, page 6. ACM, 2004.

[5] Bubacarr Bah, Stephen Becker, Volkan Cevher, and Baron Gozcu. Metric learning with rank and sparsity constraints. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 21–25. Ieee, 2014.

[6] A. Beck and M. Teboulle. Gradient-based algorithms with applications to signal-recovery problems. *Convex Optimization in Signal Processing and Communications*, pages 42–88, 2010.

[7] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

[8] J. Bezdek and R. Hathaway. Some notes on alternating optimization. *Advances in Soft ComputingAFSS 2002*, pages 187–195, 2002.

[9] Wei Bian and Dacheng Tao. Learning a distance metric by empirical loss minimization. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Two*, pages 1186–1191. AAAI Press, 2011.

[10] M. Bilenko, S. Basu, and R.J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *ICML*, page 11, 2004.

[11] Olivier Bousquet and André Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526, 2002.

[12] S.P. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge Univ Pr, 2004.

[13] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[14] Qiong Cao, Zheng-Chu Guo, and Yiming Ying. Generalization bounds for metric and similarity learning. *arXiv preprint arXiv:1207.5437*, 2012.

[15] Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, et al. *Semi-supervised learning*, volume 2. MIT press Cambridge, 2006.

[16] Ratthachat Chatpatanasiri, Teesid Korsrilabutr, Pasakorn Tangchanachaianan, and Boonserm Kijsirikul. A new kernelization framework for mahalanobis distance learning algorithms. *Neurocomputing*, 73(10):1570–1579, 2010.

[17] Jianhui Chen and Jieping Ye. Training svm with indefinite kernels. In *Proceedings of the 25th international conference on Machine learning*, pages 136–143. ACM, 2008.

[18] Yihua Chen, Eric K. Garcia, Maya R. Gupta, Ali Rahimi, and Luca Cazzanti. Similarity-based classification: Concepts and algorithms. *JMLR*, 2009.

[19] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.

[20] Nello Cristianini and John Shawe-Taylor. *An introduction to Support Vector Machines.* Cambridge University Press, 2000.

[21] M. Cuturi and D. Avis. Ground metric learning. *arXiv preprint arXiv:1110.2306*, 2011.

[22] J. Dattorro. *Convex optimization & Euclidean distance geometry.* Meboo Publishing USA, 2005.

[23] J.V. Davis, B. Kulis, P. Jain, S. Sra, and I.S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning.* ACM New York, NY, USA, 2007.

[24] J.V. Davis, B. Kulis, P. Jain, S. Sra, and I.S. Dhillon. Information-theoretic metric learning. In *ICML*, 2007.

[25] H. Do, A. Kalousis, J. Wang, and A. Woznica. A metric learning perspective of svm: on the relation of svm and lmnn. *AISTATS*, 2012.

[26] Harris Drucker, Chris JC Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. *Advances in neural information processing systems*, pages 155–161, 1997.

[27] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *ICML*, 2008.

[28] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[29] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[30] A. Frome, Y. Singer, and J. Malik. Image retrieval and classification using local distance functions. In *Advances in Neural Information Processing Systems*, volume 19, pages 417–424. MIT Press, 2007.

[31] K. Gai, G. Chen, and C. Zhang. Learning kernels with radiuses of minimum enclosing balls. *NIPS*, 2010.

[32] A. Globerson and S. Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2006.

[33] Andrew B Goldberg and Xiaojin Zhu. Seeing stars when there aren't many stars: graph-based semi-supervised learning for sentiment categorization. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, pages 45–52. Association for Computational Linguistics, 2006.

[34] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005.

[35] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011.

[36] L. Grippo and M. Sciandrone. On the convergence of the block nonlinear gauss-seidel method under convex constraints* 1. *Operations Research Letters*, 26(3):127–136, 2000.

[37] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? Metric learning approaches for face identification. In *Proceedings of 12th International Conference on Computer Vision*, pages 498–505, 2009.

[38] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):607–616, 1996.

[39] Sren Hauberg, Oren Freifeld, and Michael Black. A geometric take on metric learning. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2033–2041, 2012.

[40] G. Hinton and S. Roweis. Stochastic neighbor embedding. *Advances in neural information processing systems*, 15:833–840, 2002.

[41] P. Honeine and C. Richard. The angular kernel in machine learning for hyperspectral data classification. In *Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), 2010 2nd Workshop on*, pages 1–4. IEEE, 2010.

[42] K. Huang, Y. Ying, and C. Campbell. Gsml: A unified framework for sparse metric learning. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 189–198. IEEE, 2009.

[43] P. Jain, B. Kulis, J.V. Davis, and I.S. Dhillon. Metric and kernel learning using a linear transformation. *JMLR*, 2012.

[44] P. Jain, B. Kulis, and I. Dhillon. Inductive regularized learning of kernel functions. *NIPS*, 2010.

[45] P. Jain, B. Kulis, and I. Dhillon. Inductive regularized learning of kernel functions. *Advances in Neural Information Processing Systems*, 23:946–954, 2010.

[46] Prateek Jain, Brian Kulis, Jason V Davis, and Inderjit S Dhillon. Metric and kernel learning using a linear transformation. *The Journal of Machine Learning Research*, 13:519–547, 2012.

[47] Tom Jebara, Jun Wang, and Shih-Fu Chang. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 441–448. ACM New York, NY, USA, 2009.

[48] Rong Jin, Shijun Wang, and Yang Zhou. Regularized distance metric learning: Theory and algorithm. In *Advances in neural information processing systems*, pages 862–870, 2009.

[49] Rie Johnson and Tong Zhang. Graph-based semi-supervised learning and spectral kernel design. *Information Theory, IEEE Transactions on*, 54(1):275–288, 2008.

[50] F Jordan and F Bach. Learning spectral clustering. *Advances in neural information processing systems*, 16, 2003.

[51] A. Kalousis, J. Prados, and M. Hilario. Stability of feature selection algorithms: a study on high-dimensional spaces. *Knowledge and information systems*, 12(1):95–116, 2007.

[52] R.E. Kass and P.W. Vos. *Geometrical foundations of asymptotic inference*, volume 908. Wiley-Interscience, 2011.

[53] Dor Kedem, Stephen Tyree, Kilian Weinberger, Fei Sha, and Gert Lanckriet. Non-linear metric learning. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2582–2590, 2012.

[54] Rémi Lajugie, Sylvain Arlot, and Francis Bach. Large-margin metric learning for partitioning problems. *arXiv preprint arXiv:1303.1280*, 2013.

[55] G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M.I. Jordan. Learning the Kernel Matrix with Semidefinite Programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[56] G. Lebanon. Metric learning for text documents. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4):497–508, 2006.

[57] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[58] J.M. Lee. *Introduction to smooth manifolds*, volume 218. Springer, 2002.

[59] S.M. Lee, A.L. Abbott, and P.A. Araman. Dimensionality reduction and clustering on statistical manifolds. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–7. IEEE, 2007.

[60] Daryl Lim and Gert Lanckriet. Efficient learning of mahalanobis metrics for ranking. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1980–1988, 2014.

[61] Daryl Lim, Brian McFee, and Gert R Lanckriet. Robust structural metric learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 615–623, 2013.

[62] Z. Lu, P. Jain, and I.S. Dhillon. Geometry-aware metric learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM New York, NY, USA, 2009.

[63] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Supervised dictionary learning. *arXiv preprint arXiv:0809.3083*, 2008.

[64] B. McFee and G. Lanckriet. Metric learning to rank. In *Proceedings of the 27th annual International Conference on Machine Learning (ICML)*. ACM New York, NY, USA, 2010.

[65] Jiangyuan Mei, Meizhu Liu, Hamid Reza Karimi, and Huijun Gao. Logdet divergence based metric learning using triplet labels. In *ICML*, 2013.

[66] Renqiang Min, David A Stanley, Zineng Yuan, Anthony Bonner, and Zhaolei Zhang. A deep non-linear feature mapping for large-margin knn classification. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 357–366. IEEE, 2009.

[67] Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[68] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[69] N. Nguyen and Y. Guo. Metric Learning: A Support Vector Approach. In *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases-Part II*, page 136. Springer-Verlag, 2008.

[70] M.E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1447–1454. Ieee, 2006.

[71] M.E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on*, pages 722–729. IEEE, 2008.

[72] Jorge Nocedal and Stephen J Wright. *Conjugate gradient methods*. Springer, 2006.

[73] Y.K. Noh, B.T. Zhang, and D.D. Lee. Generative local metric learning for nearest neighbor classification. *NIPS*, 2009.

[74] Mohammad Norouzi and David M Blei. Minimal loss hashing for compact binary codes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 353–360, 2011.

[75] Guo-Jun Qi, Jinhui Tang, Zheng-Jun Zha, Tat-Seng Chua, and Hong-Jiang Zhang. An efficient sparse metric learning in high-dimensional space via l 1-penalized log-determinant regularization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 841–848. ACM, 2009.

[76] C. Radhakrishna Rao. Information and accuracy attainable in the estimation of statistical parameters. *Bulletin of the Calcutta Mathematical Society*, 37(3):81–91, 1945.

[77] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

[78] Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 22:2323–2326, 2000.

[79] Ruslan Salakhutdinov and Geoffrey E Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *International Conference on Artificial Intelligence and Statistics*, pages 412–419, 2007.

[80] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels*. The MIT Press, 2002.

[81] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons Inc, 1998.

[82] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *NIPS*, 2003.

[83] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*, page 41. The MIT Press, 2004.

[84] S. Shalev-Shwartz, Y. Singer, and A.Y. Ng. Online and batch learning of pseudo-metrics. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004.

[85] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming*, 127(1):3–30, 2011.

[86] C. Shen, J. Kim, and L. Wang. A scalable dual approach to semidefinite metric learning. In *CVPR*, 2011.

[87] C. Shen, J. Kim, L. Wang, and A. Hengel. Positive semidefinite metric learning using boosting-like algorithms. *JMLR*, 2012.

[88] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[89] G. Sierksma. *Linear and integer programming: theory and practice*. CRC, 2002.

[90] S. Sonnenburg, G. Ratsch, and C. Schafer. A general and efficient multiple kernel learning algorithm. In *NIPS*, 2006.

[91] Lorenzo Torresani and Kuang-chih Lee. Large margin component analysis. In *Advances in neural information processing systems*, pages 1385–1392, 2006.

[92] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004.

[93] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[94] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 2. Wiley New York, 1998.

[95] J.P. Vert, J. Qiu, and W. Noble. A new pairwise kernel for biological network inference with support vector machines. *BMC bioinformatics*, 8(Suppl 10):S8, 2007.

[96] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[97] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416, 2007.

[98] J. Wang, H. Do, A. Woznica, and A. Kalousis. Metric learning with multiple kernels. In *NIPS*, 2011.

[99] Jun Wang, Huyen T Do, Adam Woznica, and Alexandros Kalousis. Metric learning with multiple kernels. In *Advances in Neural Information Processing Systems*, pages 1170–1178, 2011.

[100] Jun Wang, Alexandros Kalousis, and Adam Woznica. Parametric local metric learning for nearest neighbor classification. In *Advances in Neural Information Processing Systems*, pages 1601–1609, 2012.

[101] Jun Wang, Alexandros Kalousis, and Adam Woznica. Parametric local metric learning for nearest neighbor classification. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1610–1618, 2012.

[102] Jun Wang, Ke Sun, Fei Sha, Stephane Marchand-Maillet, and Alexandros Kalousis. Two-stage metric learning. In *Proceedings of the 31th International Conference on Machine Learning*, 2014.

[103] Jun Wang, Adam Woznica, and Alexandros Kalousis. Learning neighborhoods for metric learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 223–236. Springer, 2012.

[104] K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, volume 18. MIT Press, 2006.

[105] K.Q. Weinberger and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244, 2009.

[106] Lei Wu, Steven CH Hoi, Rong Jin, Jianke Zhu, and Nenghai Yu. Learning bregman distance functions for semi-supervised clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 24(3):478–491, 2012.

[107] E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*. MIT Press, 2003.

[108] Zhixiang Xu, Kilian Q Weinberger, and Olivier Chapelle. Distance metric learning for kernel machines. *arXiv preprint arXiv:1208.3422*, 2012.

[109] Z. Yang and J. Laaksonen. Regularized neighborhood component analysis. *Lecture Notes in Computer Science*, 4522:253–262, 2007.

[110] D.Y. Yeung and H. Chang. Locally smooth metric learning with application to image retrieval. In *ICCV*, 2007.

[111] Y. Ying, C. Campbell, and M. Girolami. Analysis of svm with indefinite kernels. *Advances in Neural Information Processing Systems*, 22:2205–2213, 2009.

[112] Yiming Ying, Kaizhu Huang, and Colin Campbell. Sparse metric learning via smooth optimization. In *Advances in neural information processing systems*, pages 2214–2222, 2009.

[113] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. *NIPS*, 2009.

[114] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. *NIPS*, 2004.

[115] Zheng-Jun Zha, Tao Mei, Jingdong Wang, Zengfu Wang, and Xian-Sheng Hua. Graph-based semi-supervised learning with multiple labels. *Journal of Visual Communication and Image Representation*, 20(2):97–103, 2009.

[116] Xinhua Zhang and Wee S Lee. Hyperparameter learning for graph based semi-supervised learning algorithms. In *Advances in neural information processing systems*, pages 1585–1592, 2006.

[117] Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919, 2003.