The Dissertation Committee for Chunhua Shen
certifies that this is the approved version of the following dissertation:

Metric Learning with Convex Optimization

Committee:

Prof. Alan Welsh, Supervisor

Metric Learning with Convex Optimization

by

Chunhua Shen

A thesis submitted for the degree of

Master of Philosophy at

The Australian National University

October 2009

This thesis contains no material that has been accepted for the award of any other degree or diploma in any University, and to the best of my knowledge and belief, contains no material published or written by another person, except where due reference is made in the thesis.

沈春華. 14/10/09

CHUNHUA SHEN

The Australian National University

October 2009

Acknowledgments

I would like to thank my supervisor Prof. Alan Welsh for the advice, guidance, support and time.
This would not have been possible without the support and understanding of my family.

CHUNHUA SHEN

The Australian National University
October 2009

Metric Learning with Convex Optimization

Chunhua Shen
The Australian National University, 2009

Supervisor: Prof. Alan Welsh

In machine learning, pattern recognition and statistics, many algorithms significantly depends on an appropriate metric over the input vectors. Euclidean distance might be the most common (also simplest) metric. Nevertheless, the Euclidean distance does not utilize any information that could be available and helpful for the learning tasks. In theory, given a particular classification task, one should learn a metric by using as much information as possible. It has been an extensively sought-after goal to learn an appropriate distance metric for classification. In this thesis, two approaches to metric learning based on convex optimization for classification tasks are proposed.

The first algorithm uses sequential semidefinite programming to solve a trace quotient problem. It is shown that many dimensionality reduction (or metric learning) problems can be written into a trace quotient formulation. This new convex optimization based method can also accommodate other constraints like

sparsity constraints. The second algorithm tries to learn a quadratic Mahalanobis distance from proximity comparisons. The learning problem can be formulated as a semidefinite program, which does not scale well on large-size problems. A new matrix-generation method, termed PSDBoost, is proposed. PSDBoost is inspired by boosting algorithms in machine learning. At each iteration, a linear program needs to be solved, which is computationally much cheaper. Numerical experiments are presented.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter, we present an overview of metric learning and convex optimization. Most relevant work to our algorithms will be given in each chapter. Here we focus on a brief introduction to general metric methods. We also provide fundamental backgrounds on convex optimization, in particular, linear programming and semidefinite programming.

1.1 Distance Metric Learning

Many classical machine learning algorithms, such as k-nearest neighbor (k-NN), usually rely upon the distance metric over the input data vectors. Distance Metric learning is to learn a distance metric for the input space of data from a set of pair of similar/dissimilar points that preserves the distance relation among the training data. Previous work has demonstrated that a learned metric can indeed greatly improve the performance in classification, clustering and other learning tasks [53, 31, 12, 15, 41, 22, 48, 42].

This topic has been extensively researched in the recent years. Depending on the training data's information, algorithms can be divided into two categories:

unsupervised metric learning and supervised metric learning. For example, principal component analysis (PCA) and multidimensional scaling (MDS) that do not utilize any label information are unsupervised learning. In this thesis, we mainly focus on supervised learning for classification; *i.e.*, learning a distance metric from side information that is usually presented in a set of pairwise (or distance comparison) constraints. The optimal distance metric is typically sought by preserving these constraints and at the same time, optimizing a certain regularized term.

In the following, we will review a few classical methods and those mostly related to ours. Spectral methods are a class of traditional algorithms used to discover informative linear projections of the input space. The linear projections can be viewed as learning Mahalanobis distance metrics. Note that typically the learned Mahalanobis metrics are rank deficient in this case. We review some widely-used spectral methods here. These linear methods can usually be *kernelized* using the kernel trick. The kernel versions of these methods are beyond the scope of this thesis.

Principal component analysis and linear discriminant analysis (LDA) are two classical dimensionality reduction techniques. PCA finds the subspace that has maximum variance of the input data. LDA tries to project the data onto a subspace by maximizing the between-class distance and minimizing the within-class variance. Essentially both methods compute the linear transformation P such that the original data $\boldsymbol{x}$ are projected to a low-dimensional space by $P^\top \boldsymbol{x}$. The covariance matrix of the input data can be written as

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^\top, \tag{1.1}$$

Here $\boldsymbol{\mu}$ is the sample mean. The linear projection matrix P can be learned by maximizing the variance in the projected space under the constraint that P is

orthogonal. Mathematically it is

$$\max_{P} \mathbf{Tr}(P^{\top}\mathbf{C}P), \text{ subject to } P^{\top}P = \mathbf{I}. \tag{1.2}$$

This problem can be solved by eigen-decomposition of $\mathbf{C}$. As discussed, PCA does not use any class label information and it is an unsupervised method. In practice, PCA can be used to pre-process the input data. It has some de-noise functionality. By only keeping a few top eigenvectors of the decomposition, PCA projects the data into a low-dimensional space and thus reduces the computational complexity. In contrast, LDA finds the linear projection matrix P that maximizes the between-class variance and at the same time minimizes the within-class variance. If S_w and S_b denote the within-class scatter matrix and between-class matrix respectively, the optimization problem we want to solve is

$$\max_{P} \mathbf{Tr}\left((P^{\top}S_wP)^{-1}(P^{\top}S_bP)\right), \text{ subject to } P^{\top}P = \mathbf{I}. \tag{1.3}$$

This problem can again be solved using generalized eigen-decomposition [23]. LDA has been widely used in many application due to its simplicity and effectiveness. There are many other variants of LDA, for example [52, 27]. In some scenarios, these variants perform better than LDA.

Until recently, work has been done on learning a metric using the pairwise constraints (*a.k.a.*, equivalence constraints), which are formed by the relationship of pairs of data. Given two training points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, if they belong to the same class, we ought to minimize the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$; otherwise we maximize their distance. In the context of learning a Mahalanobis distance, the problem can be written as

$$\min_{X} \sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j)\in\mathcal{S}} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\mathrm{X}}^2, \text{ subject to } \sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j)\in\mathcal{D}} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\mathrm{X}}^2 \geq 1, X \succcurlyeq 0. \tag{1.4}$$

Here sets $\mathcal{S}$ and $\mathcal{D}$ denotes the similarity set and dissimilarity set respectively. Note that the positive semidefiniteness constraint $X \succcurlyeq 0$ is needed to ensure a valid Mahalanobis matrix. $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\mathrm{X}}^2 = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top X(\boldsymbol{x}_i - \boldsymbol{x}_j)$ is the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ with the Mahalanobis metric matrix X. The relationship between the Mahalanobis metric matrix X and the linear projection matrix P is:

$$X = PP^\top.$$

This is why methods like PCA or LDA can be viewed as metric learning algorithms. (1.4) always produces a rank-one solution [48]. To avoid this problem, the first constraint in (1.4) is changed to $\sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}} \sqrt{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\mathrm{X}}^2} \geq 1$ in [48]. Although the resulting problem is a convex program and hence the global optimum is guaranteed, it may not be trivial to solve. It is in a general form and the semidefiniteness constraint makes the problem not scale well. (1.4) was advocated to improve the performance of clustering algorithms like k-means. It may not be appropriate for learning a distance metric for k-NN classification. Neighborhood component analysis (NCA) is proposed to learn a Mahalanobis distance for k-NN classification by minimizing the leave-one-out error [18]. Given a training point $\boldsymbol{x}_i$, the probability that $\boldsymbol{x}_j$ being a neighbor of $\boldsymbol{x}_i$ is p_{ij}:

$$p_{ij} = \frac{\exp(-\|P^\top \boldsymbol{x}_i - P^\top \boldsymbol{x}_j\|^2)}{\sum_{k \neq i} \exp(-\|P^\top \boldsymbol{x}_i - P^\top \boldsymbol{x}_k\|^2)}. \tag{1.5}$$

If we denote the set of data that have the class label with $\boldsymbol{x}_i$ as $\mathcal{S}_i$, then the probability that $\boldsymbol{x}_i$ being correctly classified is $p_i = \sum_{j \in \mathcal{S}_i} p_{ij}$. The purpose is to maximize the expected number of correctly classified data points. That is, to maximize $f(P) = \sum_i p_i$. The optimization can be solved using gradient descent algorithms. However, the problem is non-convex and hence no global optimum is guaranteed. NCA seems to over-fit the training data when one does not have a

large number of training data, or the dimensionality of the data is high. NCA does not scale well because the number of parameters in the projection matrix is quadratic in the dimensionality. It becomes computationally intractable when the dimensionality is large.

A related algorithm, termed metric learning by collapsing classes (MLCC), was proposed in [17]. Unlike NCA, MLCC can be formulated as a convex optimization over the space of positive semidefinite matrices. A drawback of MLCC is that it implicitly assumes that the data points in each class have a unimodal distribution such that they can be collapsed to a single point in the transformed space.

In [46], a large margin nearest neighbor method is designed to learn a Mahalanobis metric. The goal is to make the k-nearest neighbors always belong to the same class while examples from different classes are separated by a large margin. As in support vector machines, the margin criterion leads to a convex optimization using the hinge loss. It can handle multi-class problems naturally. The problem is formulated as a standard semidefinite program, which can be solved using off-the-shelf solvers like SeDemi [43], CSDP [5], or SDPT3 [45].

Given that convex optimization becomes more and more important in machine learning and recent advances in metric learning have proved its usefulness, we review some fundamental concepts here. In particular, we give an overview of linear programming and semidefinite programming, which are our tools for the algorithms developed in the next two sections.

1.2 Convex Optimization

A mathematical programming problem has the form

$$\begin{aligned} &\min\ f_0(\boldsymbol{x}) \\ &\text{s.t. } f_i(\boldsymbol{x}) \leq b_i, i = 1, \cdots, m. \end{aligned} \tag{1.6}$$

The variable to be optimized is a vector $\boldsymbol{x}$, f_0 is the objective function and $f_i, i = 1, \cdots, m$, are the constraints. When all the functions involved in (1.6) are convex, the program is a special case, for which the global optimum is usually able to be efficiently found (in polynomial time). This class of problems is generally referred to as convex optimization problems. There is in general no analytical formula for the solution of convex optimization problems. However, there are effective methods for solving some special cases. The interior-point method is one of the methods [6]. We focus on two special cases of (1.6) here, namely linear programs and semidefinite programs.

1.2.1 Linear programming

In this section, we overview techniques of linear programs (LP). LP has a linear objective function, and a bunch of linear equality and linear inequality constraints. The standard form of LP writes:

$$\begin{aligned} &\max \ \boldsymbol{c}^\top \boldsymbol{x} \\ &\text{s.t. } A\boldsymbol{x} \leq \boldsymbol{b}. \end{aligned} \tag{1.7}$$

$\boldsymbol{x}$ is the vector of variables and $\boldsymbol{c}$ and $\boldsymbol{b}$ are vectors of known coefficients and the matrix A is also known. Many practical problems in operations research can be expressed as LPs.

Given an LP, referred to as a primal problem, there is a corresponding dual problem, which provides an upper bound to the optimal value of the primal problem. In matrix form, we can express the primal problem as:

$$\begin{aligned} &\max \ \boldsymbol{c}^\top \boldsymbol{x} \\ &\text{s.t. } A\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq 0. \end{aligned} \tag{1.8}$$

The corresponding dual problem is

$$\begin{aligned} &\min \boldsymbol{b}^\top \boldsymbol{y} \\ &\text{s.t. } A^\top \boldsymbol{y} \geq \boldsymbol{c}, \boldsymbol{y} \geq 0, \end{aligned} \tag{1.9}$$

where $\boldsymbol{y}$ is the dual variable.

The duality theory states that (1) the dual of a dual LP is the original primal LP; (2) Every feasible solution for an LP gives a bound on the optimal value of the objective function of its dual; (3) The strong duality theorem states the optimum of the dual and the optimum of the primal coincide. Therefore one can always solve an LP by solving its dual. See [6] for details about the weak and strong duality theorems and their conditions.

There are two main algorithms for solving LPs: simplex and interior point algorithms. The simplex algorithm solves LP problems by constructing an admissible solution at a vertex of the polyhedron and then walking along edges of the polyhedron to vertices with successively higher values of the objective function until the optimum is reached. The simplex method is usually fast in practice. However, the theoretical worst complexity is exponential time. In contrast, interior point methods finds the optimal solution by progressing along points on the boundary of a polyhedral set. Interior point methods start from the interior of the feasible region and converge to a vertex. There are many efficient off-the-self LP solvers, *e.g.*, CPlex [26], Mosek [34]. CPlex can solve large-scale problems up to 10^6 variables and constraints.

1.2.2 Semidefinite programming

Semidefinite programming (SDP) can be viewed as generalization of LP in the sense that the variable is now a matrix X and the nonnegativeness constraint is replaced by a matrix cone constraint $X \succcurlyeq 0$, which means X lies in a positive

semidefinite cone. In general, an SDP has the form:

$$\begin{aligned} &\min \langle C, X\rangle \\ &\text{s.t. } \langle A_i, X\rangle = b_i, i = 1, \cdots, m, \\ &\quad X \succcurlyeq 0. \end{aligned} \tag{1.10}$$

The variable we want to optimize is X. Here $\langle A, B\rangle = \sum_{ij} A_{ij}B_{ij}$ calculates the inner product of two matrices.

We can also derive its dual problem as in the case of LP,

$$\begin{aligned} &\max \langle b, y\rangle \\ &\text{s.t. } \sum_{i=1}^{m} y_i A_i \preccurlyeq C. \end{aligned} \tag{1.11}$$

The dual variable is y. Here $A \preccurlyeq B$ means $A - B \preccurlyeq 0$ or $B - A \succcurlyeq 0$.

Like LP, the weak duality and strong duality hold for SDPs under some conditions. Usually the weak duality always holds even for non-convex problems. The weak duality can be used to find nontrivial lower bounds for difficult problems. The strong duality usually holds for convex problems. The conditions that guarantee strong duality in convex problems are called constraint qualifications. See [6] for details. Unlike LP where one can always recover the primal solution from the dual solution by solving a linear system, one may not always be able to find the primal solution of an SDP from its dual. However, interior point methods solve the primal and dual at the same time.

In convex optimization [6], many problems such as quadratic programs (QP), quadratically constrained quadratic programs (QCQP), and second-oder cone programs (SOCP) can be viewed as special cases of SDP. In machine learning, SDPs have gained more and more research interests because many problems, such

as metric learning and kernel learning, can be formulated as SDPs. The desirable properties of these formulations are: (1) The global optimum is guaranteed. In other words, it is local-optimum-free; (2) For reasonable-scale problems (up to a few thousand variables), interior point methods work efficiently. Nevertheless, as mentioned before, current solvers do not scale very well, largely due to an inverse of the Hessian matrix needs to be calculated and stored, which requires cubic complexity. It is an active research topic to design first-order methods for solving SDPs, in which no Hessian is involved [6].

The next two chapters present our main algorithms and the thesis is concluded in the last chapter.

Chapter 2

Supervised Dimensionality Reduction via Sequential SDP

Many dimensionality reduction problems end up with a trace quotient formulation, for example, the linear discriminant analysis method. Since it is difficult to directly solve the trace quotient problem, traditionally the trace quotient cost function is replaced by an approximation such that generalized eigen-decomposition can be applied. By contrast, we directly optimize the trace quotient in this work. It is reformulated as a quasi-linear semidefinite optimization problem, which can be solved globally and efficiently using standard off-the-shelf semidefinite programming solvers. Also this optimization strategy allows one to enforce additional constraints (for example, sparseness constraints) on the projection matrix. We apply this optimization framework to a novel dimensionality reduction algorithm. The performance of the proposed algorithm is demonstrated in experiments on several UCI machine learning benchmark examples, USPS handwritten digits as well as ORL and Yale face data.

2.1 Introduction

In pattern recognition and computer vision, techniques for dimensionality reduction have been extensively studied [44, 46, 48, 42, 49]. Many of the dimensionality reduction methods, such as linear discriminant analysis (LDA) and its kernel version, end up with solving a trace quotient problem

$$W^{\circ} = \underset{W^{\top}W=\mathbf{I}_{d\times d}}{\operatorname{argmax}} \frac{\mathbf{Tr}(W^{\top}S_bW)}{\mathbf{Tr}(W^{\top}S_vW)}, \tag{2.1}$$

where S_b, S_v are two positive semidefinite (p.s.d.) matrices ($S_b \succcurlyeq 0, S_v \succcurlyeq 0$), $\mathbf{I}$ the $d \times d$ identity matrix (sometimes the dimension of $\mathbf{I}$ is omitted when it can be inferred from the context) and $\mathbf{Tr}(\cdot)$ denoting the matrix trace. $W \in \mathbb{R}^{D\times d}$ is target projection matrix for dimensionality reduction (typically $d \ll D$). In the supervised learning framework, usually S_b represents the distance of different classes while S_v is the distance between data points in the same class. For example, S_b is the inter-class scatter matrix and S_v is the intra-class scatter matrix for LDA. By formulating the problem of dimensionality reduction in a general setting and constructing S_b and S_v in different ways, we can implement many different methods in the above mathematical framework.

Despite the importance of the trace quotient problem, to date it lacks a direct and globally optimal solution. Usually, as an approximation, the quotient trace cost $\mathbf{Tr}((W^{\top}S_vW)^{-1}(W^{\top}S_bW))$ is instead used such that the generalized eigen-decomposition (GEVD) can be applied and a close-form solution is readily available. It is easy to check that when $\mathbf{rank}(W) = 1$, i.e., W is a vector, then Equation (2.1) is actually a Rayleigh quotient problem. It can be solved by GEVD [19, 49]. The eigenvector corresponding to the eigenvalue of largest magnitude of the matrix $S_v^{-1}S_b$ gives the optimal W°. Unfortunately, when $\mathbf{rank}(W) > 1$, the problem becomes much more complicated. Heuristically, the dominant eigenvectors

of $S_v^{-1}S_b$ corresponding to the largest eigenvalues are used to form the optimal W°. It is believed that the largest eigenvalue contains more useful information. Nevertheless such a GEVD approach cannot produce an optimal solution to the original optimization problem (2.1) [49]. Furthermore, the GEVD approach does not yield an orthogonal projection matrix. It is shown in [8, 25] that orthogonal basis functions preserve the metric structure of the data better and they have more discriminating power. Orthogonal LDA (OLDA) is proposed to compute a set of orthogonal discriminant vectors via the simultaneous diagonalization of the scatter matrices [52]. In [51] it is shown that solely optimizing the Fisher criterion does not necessarily yield optimal discriminant vectors. It is better to include correlation constraints into optimization. The features produced by the classical LDA could be highly correlated (because they are not orthogonal), leading to high redundancy of information. Including correlation constraints such as

$$W^\top(S_v + S_b)W = \mathbf{0}$$

could be beneficial for classification [51].

Recently semidefinite programming (SDP) or more general convex programming [6, 3] has been attracting more and more interests in machine learning due to its flexibility and desirable global optimality [47, 28, 31]. Moreover, there exist interior-point algorithms to efficiently solve SDPs in polynomial time. Large margin nearest neighbor (LMNN) [46] is an example of using SDP to learn a metric. LMNN learns a metric by maintaining consistency in data's neighborhood and keeping a large margin at the boundaries of different classes. It has been shown in [46] that LMNN delivers the state-of-the-art performance among most distance metric learning algorithms.

In this chapter, we proffer a novel SDP based method for solving the trace quotient problem *directly*. It has the following appealing properties:

- The low target dimension is selected by the user and the algorithm guarantees a globally optimal solution using fractional programming. In other words, it is local-optima-free. Moreover, the fractional programming can be efficiently solved by a sequence of SDPs;
- The projection matrix is orthonormal naturally;
- Unlike the GEVD approach to LDA, using our proposed algorithm, the data are not restricted to be projected to at most $c-1$ dimensions. Here c is the number of classes.

To our knowledge, this is the first attempt that directly solves the trace quotient problem and meanwhile, a global optimum is deterministically guaranteed. We then develop methods to design S_b and S_v. The traditional LDA is only optimal when all the classes follow single Gaussian distributions that share the same covariance matrix. Our new S_b and S_v are not confined by this assumption.
The remaining content is organized as follows. In Section 2.2, we describe our algorithm in detail. Section 2.3 applies this optimization framework to dimensionality reduction. In Section 2.4, we briefly review relevant work in the literature. The experiment results are presented in Section 2.5. We discuss new extensions in Section 2.6. Finally concluding remarks are discussed in Section 2.7.

2.2 Solving the Trace Quotient Problem Using SDP

In this section, we show how the trace quotient is reformulated into an SDP problem.

2.2.1 SDP formulation

By introducing an auxiliary variable δ, the problem (2.1) is equivalent to

$$\begin{aligned} \underset{\delta,W}{\text{maximize}} \quad & \delta && \text{(2.2a)} \\ \text{subject to} \quad & \mathbf{Tr}(W^\top S_b W) \geq \delta \cdot \mathbf{Tr}(W^\top S_v W) && \text{(2.2b)} \\ & W^\top W = \mathbf{I}_{d\times d} && \text{(2.2c)} \\ & W \in \mathbb{R}^{D\times d}. && \text{(2.2d)} \end{aligned}$$

The variables we want to optimize here are δ and W. But we are only interested in W with which the value of δ is maximized. This problem is clearly not convex because the constraint (2.2b) is not convex, and in addition (2.2d) is actually a non-convex rank constraint. (2.2c) is quadratic in W. It is obvious that δ must be positive.

Let us define a new variable $Z \in \mathbb{R}^{D\times D}, Z = WW^\top$, and now the constraint (2.2b) is converted to $\mathbf{Tr}((S_b - \delta S_v)Z) \geq 0$ under the fact that $\mathbf{Tr}(W^\top SW) = \mathbf{Tr}(SWW^\top) = \mathbf{Tr}(SZ)$. Because Z is a matrix production of W and its transpose, it must be p.s.d. In terms of Z, the cost function (2.1) is a linear fraction, therefore it is quasi-convex (More precisely, it is also quasi-concave, hence quasi-linear [6]). The standard technique for solving quasi-concave maximization (or quasi-convex minimization) problems is bisection search which involves solving a sequence of SDPs for our problem. The following theorem due to [36] serves as a basis for converting the non-convex constraint (2.2d) into a linear one.

Theorem 2.2.1. *Define sets* $\Omega_1 = \{WW^\top : W^\top W = \mathbf{I}_{d\times d}\}$ *and* $\Omega_2 = \{Z : Z = Z^\top, \mathbf{Tr}(Z) = d, 0 \preccurlyeq Z \preccurlyeq \mathbf{I}\}$. *Then* Ω_1 *is the set of extreme points of* Ω_2.

See [36] for the proof. Theorem 2.2.1 states, as constraints, Ω_1 is more strict than

Ω_2. Therefore constraints (2.2c) and (2.2d) can be relaxed into $\mathbf{Tr}(Z) = d$ and $0 \preccurlyeq Z \preccurlyeq \mathbf{I}$, which are both convex. When the cost function is linear and it is subject to Ω_2, the solution will be at one of the extreme points [37]. Consequently, for linear cost functions, the optimization problems subject to Ω_1 and Ω_2 are exactly equivalent.

With respect to Z and δ, (2.2b) is still non-convex: the problem may have locally optimal points. But still the global optimum can be efficiently computed via a sequence of convex feasibility problems. By observing that the constraint is linear if δ is known, we can convert the optimization problem into a set of convex feasibility problems. A bisection search strategy is adopted to find the optimal δ. This technique is widely used in fractional programming [6, 1]. Let δ° denote the unknown optimal value of the cost function. Given $\delta^* \in \mathbb{R}$, if the convex feasibility problem[1]

$$\begin{aligned} \text{find} \quad & Z && \text{(2.3a)} \\ \text{subject to} \quad & \mathbf{Tr}((S_b - \delta^* S_v)Z) \geq 0 && \text{(2.3b)} \\ & \mathbf{Tr}(Z) = d && \text{(2.3c)} \\ & 0 \preccurlyeq Z \preccurlyeq \mathbf{I} && \text{(2.3d)} \end{aligned}$$

is feasible, then we have $\delta^\circ \geq \delta^*$. Otherwise, if the above problem is infeasible, then we can conclude $\delta^\circ < \delta^*$. This way we can check whether the optimal value δ° is smaller or larger than a given value δ^*. This observation motivates a simple algorithm for solving the fractional optimization problems using bisection search, which solves an SDP feasibility problem at each step. Algorithm 1 shows how it works.

Thus far, a question remains unanswered: are constraints (2.3c) and (2.3d)

[1] A feasibility problem has no cost function. The objective is to check whether the intersection of the convex constraints is empty.

Algorithm 1 Bisection search.

Require: δ_l: Lower bounds of δ; δ_u: Upper bound of δ and the tolerance $\sigma > 0$.
  while $\delta_u - \delta_l > \sigma$ **do**
    $\delta = \frac{\delta_l + \delta_u}{2}$.
    Solve the convex feasibility problem described in (2.3a)–(2.3d).
    if feasible **then**
      $\delta_l = \delta$;
    else
      $\delta_u = \delta$.
    end if
  end while

equivalent to constraints (2.2c) and (2.2d) for the feasibility problem? Essentially the feasibility problem is equivalent to

$$\begin{aligned} \text{maximize} \quad & \mathbf{Tr}((S_b - \delta^* S_v)Z) && \text{(2.4a)} \\ \text{subject to} \quad & \mathbf{Tr}(Z) = d && \text{(2.4b)} \\ & 0 \preccurlyeq Z \preccurlyeq \mathbf{I}. && \text{(2.4c)} \end{aligned}$$

If the maximum value of the cost function is non-negative, then the feasibility problem is feasible. Conversely, it is infeasible. Because this cost function is linear, we know that Ω_1 can be replaced by Ω_2, i.e., constraints (2.3c) and (2.3d) are equivalent to (2.2c) and (2.2d) for the optimization problem.

Note that constraint (2.3d) is not in the standard form of SDP. It can be rewritten into the standard form as

$$\begin{bmatrix} Z & \mathbf{0} \\ \mathbf{0} & Q \end{bmatrix} \succcurlyeq 0, \qquad \text{(2.5a)}$$

$$Z + Q = \mathbf{I}, \qquad \text{(2.5b)}$$

where the matrix Q acts as a slack variable. Now the problem can be solved using standard SDP packages such as CSDP [5] and SeDuMi [43]. We use CSDP in all of

our experiments.

2.2.2 Estimating bounds of δ

The bisection search procedure requires a low bound and an upper bound of δ. The following theorem from [36] is useful for estimating the bounds.

Theorem 2.2.2. *Let $S \in \mathbb{R}^{D\times D}$ be a symmetric matrix, and $\varphi_{S,1} \geq \varphi_{S,2} \geq \cdots \geq \varphi_{S,D}$ be the sorted eigenvalues of S from largest to smallest, then*

$$\max_{W^\top W=\mathbf{I}_{d\times d}} \mathbf{Tr}(W^\top SW) = \sum_{i=1}^{d} \varphi_{S,i}.$$

Refer to [36] for the proof. This theorem can be extended to obtain the following corollary (following the proof for Theorem 2.2.2):

Corollary 2.2.1. *Let $S \in \mathbb{R}^{D\times D}$ be a symmetric matrix, and $\psi_{S,1} \leq \psi_{S,2} \leq \cdots \leq \psi_{S,D}$ be its sorted eigenvalues from smallest to largest, then*

$$\min_{W^\top W=\mathbf{I}_{d\times d}} \mathbf{Tr}(W^\top SW) = \sum_{i=1}^{d} \psi_{S,i}.$$

Therefore, we estimate the upper bound of δ:

$$\delta_u = \frac{\sum_{i=1}^{d} \varphi_{S_b,i}}{\sum_{i=1}^{d} \psi_{S_v,i}}. \tag{2.6}$$

In the trace quotient problem, both S_b and S_v are p.s.d. That is to say, all of their eigenvalues are non-negative. Be aware that the denominator of (2.6) could be zeros and $\delta_u = +\infty$. This occurs when the d smallest eigenvalues of S_v are all zeros. In this case, $\mathbf{rank}(S_v) \leq D - d$. In the case of LDA, $\mathbf{rank}(S_v) = \min(D, N)$. Here N is the number of training points. When $N \leq D - d$, which is termed the *small sample problem*, δ_u is invalid.

A principle component analysis (PCA) preprocessing can always be performed to remove the null space of the covariance matrix of the data, such that δ_u becomes valid.

A lower bound of δ is then

$$\delta_l = \frac{\sum_{i=1}^{d} \psi_{S_b,i}}{\sum_{i=1}^{d} \varphi_{S_v,i}}. \tag{2.7}$$

Clearly $\delta_l \geq 0$.

The bisection algorithm converges in $\lceil \log_2(\frac{\delta_u - \delta_l}{\sigma}) \rceil$ iterations, and obtains the global minimum within the predefined accuracy of σ. The bisection procedure is intuitive to understand. Next we describe another algorithm—Dinkelbach's algorithm—less intuitive but faster, for fractional programming.

2.2.3 Dinkelbach's algorithm

Dinkelbach algorithm [39] proposes an iterative procedure for solving the fractional program $\text{maximize}_{\boldsymbol{x}} f(\boldsymbol{x})/g(\boldsymbol{x})$, where $\boldsymbol{x}$ is constrained on a convex set and $f(\boldsymbol{x})$ is concave, $g(\boldsymbol{x})$ is convex. It considers the parametric problem,

$$\text{maximize}_{\boldsymbol{x}} f(\boldsymbol{x}) - \delta g(\boldsymbol{x}),$$

where δ is a constant. Here we need to solve

$$\text{maximize}_{Z} \mathbf{Tr}((S_b - \delta S_v)Z). \tag{2.8}$$

The algorithm generates a sequence of values of δ's that converge to the global optimum function value. The bisection search converges linearly while the Dinkelbach's algorithm converges super-linearly (better than linearly and worse than quadratically).

Dinkelbach's iterative algorithm for our trace quotient problem is described in Algorithm 2. We omit the convergence analysis of the algorithm which can be

Algorithm 2 Dinkelbach algorithm.

Require: An initialization $Z^{(0)}$ which satisfies constraints (2.4b) and (2.4c).

Set

$$\delta = \frac{\mathbf{Tr}(S_b Z^{(0)})}{\mathbf{Tr}(S_v Z^{(0)})}$$

and $k = 0$.

($\star$) $k = k + 1$. Solve the SDP (2.8) subject to constraints (2.4b) and (2.4c) to get the optimal $Z^{(k)}$, given δ.

if $\mathbf{Tr}((S_b - \delta S_v)Z^{(k)}) = 0$ **then**

stop and the optimal $Z^{\circ} = Z^{(k)}$.

else

Set

$$\delta = \frac{\mathbf{Tr}(S_b Z^{(k)})}{\mathbf{Tr}(S_v Z^{(k)})}$$

and go to step ($\star$).

end if

found in [39]. In Algorithm 2, note that: (1) A test of the form $\mathbf{Tr}((S_b - \delta S_v)Z^{(k)}) \geq 0$ is unnecessary since for any fixed k, $\mathbf{Tr}((S_b - \delta S_v)Z^{(k)}) = \max_Z \mathbf{Tr}((S_b - \delta S_v)Z) \geq \mathbf{Tr}((S_b - \delta S_v)Z^{(k-1)}) = 0$. However due to computer's numerical accuracy limit, in implementation it is possible that the value of $\mathbf{Tr}((S_b - \delta S_v)Z^{(k)})$ is a negative value which is very close to zero; (2) To find the initialization $Z^{(0)}$ that must reside in the set defined by the constraints, in general, one might solve a feasibility problem. In our case, it is easy to find out that a square matrix $Z^{(0)} \in \mathbb{R}^{D \times D}$ with d diagonal entries being 1 and all the other entries being 0 satisfies (2.4b) and (2.4c). This initialization is used in our experiments and it works well. Dinkelbach's algorithm needs no parameters and it converges faster. In contrast, bisection needs one to estimate the bounds for the cost function.

2.2.4 Computing W from Z

From the covariance matrix Z learned by SDP, we can calculate the projection matrix W by eigen-decomposition. Let V_i denote the i^{th} eigenvector, with eigenvalue λ_i. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$ be the sorted eigenvalues. It is straightforward to see that $W = \text{diag}(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \cdots, \sqrt{\lambda_D})V^\top$, where $\text{diag}(\cdot)$ is a square matrix with the input as its diagonal elements. To obtain a $D \times d$ projection matrix, the smallest $D - d$ eigenvalues are simply truncated. The projection matrix obtained in this way is not the same as the projection corresponding to maximizing the cost function subject to a rank constraint. However this approach is a reasonable approximation. Moreover, like PCA, dropping the eigenvectors corresponding to small eigenvalues may de-noise the input data, which is desirable in some cases.

This is the general treatment for recovering a low dimensional projection from a covariance matrix. In our case, this procedure is precise. This is obvious: λ_i, the eigenvalues of $Z = WW^\top$, are the same as the eigenvalues of $W^\top W = \mathbf{I}_{d\times d}$. That means, $\lambda_1 = \lambda_2 = \cdots = \lambda_d = 1$ and the remaining $D - d$ eigenvalues are all zeros. Hence in our case we can simply stack the first d leading eigenvectors to obtain W.

2.3 Application to Dimensionality Reduction

There are various strategies to construct the matrix S_b and S_v, which represent the inter-class and intra-class scatter matrices respectively. In general, we have a set of data $\{\boldsymbol{x}_p\}_{p=1}^M \in \mathbb{R}^D$ and we are given a *similarity* set $\mathcal{S}$ and a *dissimilarity* set $\mathcal{D}$. Formally, $\{\mathcal{S} : (\boldsymbol{x}_p, \boldsymbol{x}_q) \in \mathcal{S} \quad \text{if } \boldsymbol{x}_p \text{ and } \boldsymbol{x}_q \text{ are similar}\}$ and $\{\mathcal{D} : (\boldsymbol{x}_p, \boldsymbol{x}_q) \in \mathcal{D} \quad \text{if } \boldsymbol{x}_p \text{ and } \boldsymbol{x}_q \text{ are dissimilar}\}$. We want to maximize the distance

$$\sum_{(p,q)\in\mathcal{D}} \text{dist}^2_W(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sum_{(p,q)\in\mathcal{D}} \|W^\top \boldsymbol{x}_p - W^\top \boldsymbol{x}_q\|^2$$

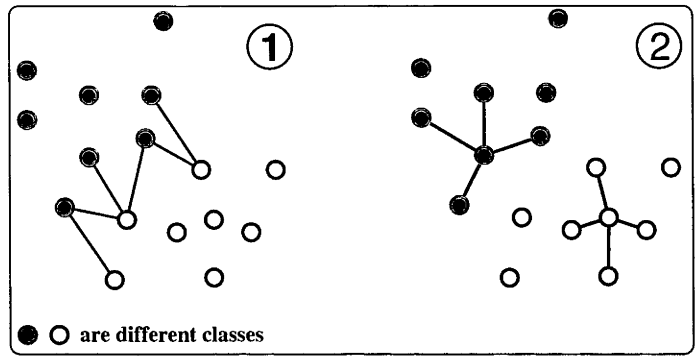


Figure 2.1: The connected edges in (1) define the dissimilarity set $\mathcal{D}$ and the connections in (2) define the similarity set $\mathcal{S}$. As shown in (1), the inter-class marginal samples are connected while in (2), each sample is connected to its k' nearest neighbors in the same class. For clarity only few connections are shown.

$$= \sum_{(p,q)\in\mathcal{D}} (\boldsymbol{x}_p - \boldsymbol{x}_q)^\top Z(\boldsymbol{x}_p - \boldsymbol{x}_q)$$
$$= \mathbf{Tr}\,(S_b Z),$$

where $S_b = \sum_{(p,q)\in\mathcal{D}}(\boldsymbol{x}_p - \boldsymbol{x}_q)(\boldsymbol{x}_p - \boldsymbol{x}_q)^\top$. This distance measures the inter-class distance. We also want to minimize the intra-class compactness distance:

$$\sum_{(p,q)\in\mathcal{S}} \mathbf{dist}^2_W(\boldsymbol{x}_p, \boldsymbol{x}_q) = \mathbf{Tr}\,(S_v Z),$$

with $S_v = \sum_{(p,q)\in\mathcal{S}}(\boldsymbol{x}_p - \boldsymbol{x}_q)(\boldsymbol{x}_p - \boldsymbol{x}_q)^\top$.

Inspired by the marginal fisher analysis (MFA) algorithm proposed in [50], we construct similar graphs for building S_b and S_v. Figure 2.1 demonstrates the basic idea. For each class, assuming $\boldsymbol{x}_p$ is in this class, and if the pair (p, q) belongs to the k closest pairs that have different labels, then $(p, q) \in \mathcal{D}$. The intra-class set $\mathcal{S}$ is easier: we connect each sample to its k' nearest neighbors in the same class. k and k' are parameters defined by the user.

This strategy avoids certain drawbacks of LDA. We do not force all the pairwise samples in the same class to be close (This might be a too strict requirement). Instead, we are more interest in driving neighboring samples as closely as possible. We do not assume any special distribution on the data. The set $\mathcal{D}$ characterizes the margin information between classes. For non-Gaussian data, it is expected to better represent the separability of different classes than the inter-class covariance of LDA. Therefore we maximize the margins while condensing individual classes simultaneously. For ease of presentation, we refer this algorithm as SDP_1, whose S_b and S_v are calculated by the above-mentioned strategy.

[16] defines a 1-nearest-neighbor margin based on the concept of the nearest neighbor to a point $\boldsymbol{x}$ with the same and different label. Motivated by their work, we can slightly modify MFA's inter-class distance graph. The similarity set $\mathcal{S}$ remains unchanged as described previously. But to create the dissimilarity set $\mathcal{D}$, a simpler way is that, for each $\boldsymbol{x}_p$ we connect it to its k differently-labeled neighbors $\boldsymbol{x}_q$'s ($\boldsymbol{x}_p$ and $\boldsymbol{x}_q$ have different labels). The algorithm that implements this concept is referred to as SDP_2. It is difficult to analyse which one is better. Indeed the experiments indicate for different data sets, no single method is consistently better than the other one. One may also use support vector machines (SVMs) to find the boundary points of the separation plane and then create $\mathcal{D}$ (and then S_b) based on those boundary points [15].

2.4 Related Work

The closest work to ours is [49] in the sense that it also proposes a method to solve the trace quotient directly. [49] finds the projection matrix W in the Grassmann manifold. Compared with optimization in the Euclidean space, the main advantage of optimization on the Grassmann manifold is fewer variables. Thus the scale of the problem is smaller. There are major differences between [49] and our

method: (1) [49] optimizes $\mathbf{Tr}(W^\top S_b W - \delta \cdot W^\top S_v W)$ and they do not have a principled way to determine the optimal value of δ. In contrast, we optimizes the trace quotient function itself and a deterministic bisection search or the Dinkelbach's iteration guarantees the optimal δ; (2) The optimization in [49] is non-convex (difference of two quadratic functions). Therefore it is likely to become trapped into a local maximum, while our method is globally optimal.

[32] simply replaces LDA's cost function with $\mathbf{Tr}(W^\top S_b W - W^\top S_v W)$, i.e., setting $\delta = 1$. Then GEVD is used to obtain the low rank projection matrix. Obviously this optimization is not equivalent to the original problem, although it avoids the matrix inversion problem of LDA.

[48] proposes a convex programming approach to maximize the distances between classes and simultaneously to clip (but not to minimize) the distances within classes. Unlike our method, in their approach the rank constraint is not considered. Hence it is metric learning but not necessary a dimensionality reduction method. Furthermore, although the formulation of [48] is convex, it is not an SDP. It is more computationally expensive to solve and general-purpose SDP solvers are not applicable. SDP (or general convex programming) is also used in [46, 17] for learning a distance metric. [46] learns a metric that shrinks distances of neighboring similarly-labeled points and repels points in different classes by a large margin. [17] also learns a metric using convex programming.

We borrow the idea from [50] to construct the similarity and dissimilarity sets. The MFA algorithm in [50] optimizes a different cost function. It originates from graph embedding. Note that there is a kernel version of MFA. It is straightforward to kernelize our problem since it is still a trace quotient for the kernel version. We leave this topic for future research.

2.5 Experiments

In all our experiments, the Bisection Algorithm 1 and Dinkelbach's Algorithm 2 output almost identical results but Dinkelbach converges as twice faster as Bisection does.

We observe that the direct solution indeed yields larger trace quotient than the quotient trace using GEVD because that is what we maximize.

Data visualization. As an intuitive demonstration, we run the proposed SDP algorithms on an artificial concentric circles data set [18], which consists of four classes (shown in different colors). The first two dimensions follow concentric circles while the remaining eight dimensions are all Gaussian noise. When the scale of the noise is large, PCA is distracted by the noise. LDA also fails because the data set is not linearly separable and each class' center overlaps in the same point. Both of our algorithms find the informative features (Figure 2.2-(3)(4)). Ideally we should optimize the projected neighborhood relationship as in [18]. Unfortunately it is difficult. [18] utilizes softmax nearest neighbors to model the neighborhood relationships before the projection is known. However the cost is non-convex. As an approximation, one usually calculates the neighborhood relationships in the input space. Laplacian eigenmap [2] is an example. When the noise is large enough, the neighborhood obtained in this way may not *faithfully* represent the true data structure. We deliberately set the noise of the concentric data set very large, which breaks our algorithms (Figure 2.2-(5)(6)). Nevertheless useful prior information can be used to define a meaningful $\mathcal{D}$ and $\mathcal{S}$ whenever it is available. As an example, we use the sets $\mathcal{D}$ and $\mathcal{S}$ of Figure 2.2-(3)(4) and then calculate S_b and S_v with the highly noisy data, our algorithms are still able to find the first two useful dimensions perfectly, as shown in Figure 2.2-(7)(8).

Classification. In the first classification experiment, we evaluate our algorithm on different data sets and compare it with PCA, LDA and large margin nearest

neighbor classifier (LMNN)[2]. Note that our algorithm is much faster than LMNN in [46], especially when the number of training data is large. That is because the complexity of our algorithm is independent of the number of data while in [46] more data produce more SDP constraints that slow down the SDP solver. A description of the data sets is in Table 2.1.

PCA is used to reduce the dimensionality of image data (USPS handwritten digits[3] and ORL face data[4]) as a preprocessing procedure for accelerating the computation. For five data sets the results are reported over 50 random 70/30 splits of the data. USPS has a predefined training and testing sets.

In the experiments, we did not carefully tune the parameters (k, k') associated with our proposed SDP approaches due to computational burden. However, we find that the parameters are not sensitive in a wide range. They can be optimally determined by cross-validation. We report a 3-NN (nearest neighbor) classifier's testing error. The result is shown in Table 2.2, where the baseline is obtained by directly apply 3-NN classification on the original data. Next we present details of tests.

UCI data sets: Iri, Wine and Bal. These are small data sets with only 3 classes, which are from UCI machine learning repository [35]. Except the Wine data, which are well separated and LDA performs best, for the other two data, our SDP algorithms present competitive results.

USPS digit recognition. Two tests are conducted on the USPS handwriting digit data set. In the first test, we use all the 10 digits. USPS has predefined training and testing subsets. The training subset has 7291 digits. We randomly split the training subset: 20% for training and 80% for testing. The dimensionality of these 16×16 images are reduced to 55D by PCA. 90.14% of the variance is preserved.

[2]The codes are obtained from the authors' website `http://www.weinbergerweb.net/Downloads/LMNN.html`

[3]`http://www.gaussianprocess.org/gpml/data/`

[4]`http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`

	Iris	Wine	Bal	USPS1	USPS2	ORL1	ORL2
train samples	105	125	438	1459	2394	280	200
test samples	45	53	187	5832	628	120	200
classes	3	3	3	10	3	40	40
input dimensions	4	13	4	256	256	644	644
dimensions after PCA	4	13	4	55	50	42	42
parameters (k, k') for SDP_1	(100,5)	(50,3)	(220,3)	(300,3)	(300,3)	(300,3)	(200,3)
parameters (k, k') for SDP_2	(3,3)	(1,5)	(3,5)	(3,5)	(1,5)	(2,3)	(2,3)
runs	50	50	50	10	1	50	50

Table 2.1: Description of data sets and experimental parameters k and k'.

LMNN gives the best result with an error rate 4.22%. Our SDPs have similar performance. For the second test, it is only run once with the predefined training subset and test subset. The digits $1, 2$ and 3 are used. On this data set, our two SDPs deliver lowest test errors. It is worth noting that LDA performs even worse than PCA. This is likely due to the data's non-Gaussian distribution.

ORL face recognition. This data set consists of 400 faces of 40 individuals: 10 per each. The image size is 56×46. We down-sample them by a factor of 2. Then PCA is applied to obtain 42D eigenfaces, which captures about 81% of the variance. Again two tests are conducted on this set. The training and testing sets are obtained by 7/3 and 5/5 sampling for each person respectively. For both tests, LMNN performs best, and SDP_2 is the second best one. Also note that for each method, its performance on ORL1 is better than its corresponding result on ORL2. This is expected since ORL1 contains more training examples.

For all the tests, our algorithms are consistently better than PCA and LDA. The state-of-the-art LMNN outperforms ours on tasks with many classes such as USPS1, ORL1 and ORL2. It might be due to the fact that, inspired by SVM, LMNN enforces constraints on each training point. These constraints ensure that the learned metric correctly classifies as many training points as possible. The price is that LMNN's SDP optimization problem involves many constraints. With a large amount of training data, the required computational demand could be prohibitive. This is because the number of variables of LMNN is linear in the number of training data points. Therefore as SVM, it is difficult to scale it to large size problems. In contrast, our SDP formulation is independent of the amount of training data. The complexity is entirely determined by the dimension of the input data.

Because we have observed that for the data sets with few classes, our SDP approaches usually are better than LMNN, we now verify this observation with

more experiments. We run SDP_2 and LMNN on the data set ORL2. We vary the number of classes c from 5 to 32. The first c individuals' images are used. The parameters of SDP_2 remain unchanged: $k = 2$ and $k' = 3$. For each value of c, the experiment is run 10 times. We report the classification result in Table 2.3. This result confirms that our SDPs perform well for tasks with few classes. It also explains why LMNN outperforms our SDPs for data sets having many classes. It might also be possible to include constraints as LMNN does in our SDP formulation.

The second classification experiment we have conducted is to compare our methods with two LDA's variations, namely, uncorrelated linear discriminant analysis (ULDA) [27] and orthogonal linear discriminant analysis (OLDA) [52]. ULDA was proposed for extracting feature vectors with uncorrelated attributes. The crucial property of OLDA is that the discriminant vectors of OLDA are orthogonal to each other (In other words, the transformation matrix of OLDA is orthogonal). The Yale face database[5] is used here. The Yale database contains 165 gray-scale images of 15 individuals. There are 11 images per subject. The images demonstrate variations in lighting condition, facial expression (normal, happy, sad, sleepy, surprised, and wink). The face images are manually aligned and cropped into 32×32 pixels, with 256 gray levels per pixel. The 11 faces for each individual is randomly split into training and testing sets by 4/7, 5/6 and 6/5 sampling. PCA is performed to reduce 1024D into 50D, which contains above 98% of the total variation.

An important parameter for most subspace learning based face recognition methods is dimensionality estimation. Usually the classification accuracy varies in the number of dimensions. Cross validation is often needed to estimate the best dimensionality. We simply set the dimensionality to $c - 1$, where c is the number

[5] `http://cvc.yale.edu/projects/yalefaces/yalefaces.html`

	Iris	Wine	Bal	USPS1	USPS2	ORL1	ORL2
baseline	4.57(2.50), 4	29.48(5.48), 13	18.10(2.02), 4	6.36(0.21), 256	2.39, 256	-	-
PCA	4.80(2.51), 2	29.28(5.53), 8	12.46(2.06), 3	5.60(0.27), 55	2.07, 50	5.11(1.91), 42	8.68(2.01), 42
LDA	4.23(2.56), 2	**1.52**(1.64), 2	13.26(2.66), 3	7.43(0.42), 9	3.03, 2	4.85(1.92), 39	7.63(1.93), 39
LMNN	4.40(2.75), 4	**4.04**(2.02), 13	**12.17**(2.37), 3	**4.22**(0.35), 55	1.91, 50	**2.23**(1.39), 42	**4.69**(1.98), 42
SDP_1	**3.02**(2.19), 3	4.83(3.05), 8	13.38(2.05), 3	5.28(0.25), 50	**1.75**, 40	3.47(1.56), 39	6.64(2.21), 39
SDP_2	**3.60**(2.61), 3	12.83(5.63), 8	**9.70**(1.99), 3	**4.73**(0.10), 50	**1.91**, 40	**3.32**(1.24), 39	**5.87**(1.98), 39

Table 2.2: Classification error of a 3-NN classifier on each date set (except USPS2) in the format of **mean(std)**%. The final dimension for each method is shown after the error. The best and second best results are shown in bold.

# of classes	5	8	10	14	18	25	32
LMNN	0(0)	5.00(3.33)	3.80(1.99)	1.57(1.60)	3.49(2.08)	**2.34**(1.38)	**3.12**(2.05)
SDP_2	0(0)	**0.13**(0.56)	**1.20**(1.20)	**1.29**(1.05)	**3.11**(1.72)	3.20(1.77)	4.63(1.36)

Table 2.3: Classification error of a 3-NN classifier v.s. number of classes on the Face data (ORL2). Classification error is in the format of **mean(std)**%. Our SDP algorithm has increasing error when number of classes increases. This might be because we optimize a cost function based on global distances summation (the trace calculation). For LMNN it is not the case. Clearly on data sets with few classes, SDP is better than LMNN.

of classes. That means, on the Yale dataset, the final dimensions for all algorithms are 14. As in the first experiment, we also fix the parameters of SDP_2: $k = 2$ and $k' = 3$.

	4 Train	5 Train	6 Train
baseline	47.76(4.18)	44.40(3.61)	40.87(5.12)
LDA	39.38(8.34)	24.17(2.88)	21.40(3.07)
ULDA	29.14(5.17)	25.61(2.85)	22.80(4.12)
OLDA	27.57(5.55)	24.61(3.35)	**20.53**(3.33)
SDP_2	**27.05**(5.65)	**23.17**(3.31)	20.73(2.85)

Table 2.4: Classification error of a 3-NN classifier on the Yale face database in the format of **mean(std)**%. Each case is run 20 times to calculate the mean and standard deviation. SDP_2 performs slightly better than OLDA.

Table 2.4 summarizes the classification results. We see that ULDA performs similarly with the traditional LDA. OLDA achieves higher accuracies than ULDA and LDA. The proposed SDP algorithm is slightly better than OLDA. Since both OLDA and the proposed SDP algorithm produces orthogonal transformation matrix, we may conclude that orthogonality does benefit subspace based face recognition.

As mentioned, for the LDA algorithm and its variations, the data are restricted to be mapped to at most $c - 1$ dimensions. Our SDP algorithms do not have this restriction. We have compared the final classification results on Yale when the final dimensionality varies using the SDP_2 algorithm in Table 2.5. It can be observed that $c - 1$ is not the best dimensionality for SDP_2 in this case.

final dimensions	14	20	24	30
SDP_2	27.05(5.65)	26.43(5.14)	26.86(4.18)	28.62(5.27)

Table 2.5: Classification error of a 3-NN classifier on the Yale face database with 4 training examples. Each case is run 20 times.

A disadvantage of the proposed SDP algorithms is that it is computationally more expensive than spectral methods. In the above experiment, the Dinkelbach

algorithm needs around 80 seconds to converge. In contrast, LDA, ULDA or OLDA needs about 2 seconds[6].

2.6 Extension: Explicitly Controlling Sparseness of W

In this section, we show that with the flexible optimization framework, it is straightforward to enforce additional constraints on the projection matrix. We consider the sparseness constraints here.

Sparseness builds one type of feature selection mechanism. It has many applications in pattern analysis and image processing [20, 21, 24, 11]. Mathematically, we want the projection matrix W to be sparse. That is, $\mathbf{Card}(W) < \Theta$ $(0 < \Theta \leq Dd)$. Here Θ is a predefined parameter. $\mathbf{Card}(W)$ denotes the cardinality of the matrix W, i.e., the number of non-zero entries in the matrix W. Since $Z = WW^{\top}$, we rewrite $\mathbf{Card}(W) \leq \Theta$ as $\mathbf{Card}(Z) \leq \Theta^2$. The discrete non-convex cardinality constraint can be relaxed into a weaker convex one using the technique discussed in [11].

For any $u \in \mathbb{R}^D$, $\mathbf{Card}(u) = \Theta$ means the following inequality holds: $\|u\|_1 \leq \sqrt{\Theta}\,\|u\|_2$. We can then replace the non-convex constraint $\mathbf{Card}(Z) \leq \Theta^2$ by a convex constraint: $\|Z\|_1 \leq \Theta\,\|Z\|_{\mathrm{F}}$. $\|A\|_{\mathrm{F}} = \sqrt{\sum_{ij} A_{ij}^2}$ stands for the Frobenius norm. Since $\|Z\|_{\mathrm{F}} = \left\|WW^{\top}\right\|_{\mathrm{F}} = \left\|W^{\top}W\right\|_{\mathrm{F}} = \|\mathbf{I}_{d\times d}\|_{\mathrm{F}} = \sqrt{d}$, now the sparseness constraint becomes convex (it is easy to rewrite it into a sequence of linear constraints)

$$\|Z\|_1 \leq \Theta\sqrt{d}. \tag{2.9}$$

By inserting the constraint (2.9) into Algorithm 1 or 2, we obtain a sparse projection. Note that (2.9) is a convex constraint,[7] which can be viewed as a

[6]The computation environment is: Matlab 7.4 on a desktop with a P4 3.4GHz CPU and 1G memory. The SDP solver used is CSDP 6.0.1.

[7]There is a standard trick from mathematical programming for expressing the ℓ_1-norm as a linear function. By decomposing the variable $Z = Z_+ - Z_-$ into positive and negative parts respectively,

convex lower bound on the function $\mathbf{Card}(Z)$. It can be decomposed into $O(D^2)$ linear constraints. For a large D, the memory requirements of Newton's method in interior-point algorithms could be prohibitive.

We first run a simple experiment on artificial data to show how the sparseness of the projection matrix W changes as the value of $\Theta\sqrt{d}$ varies. For simplicity, we set $d = 1$; i.e., W is a 1D vector. We randomly generate the matrices S_b and S_v in this way: $S = U^\top U + 16w^\top w$. Here S means both S_b and S_v but $S_b \neq S_v$. $U \in \mathbb{R}^{10\times 10}$ is a random matrix with all its elements following a uniform distribution in $[0, 1]$ and

$$w = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0].$$

We sample 40 different pairs of matrices S_b and S_v. We then input S_b and S_v into the Dinkelbach algorithm with the additional sparseness constraint (2.9). For each Θ between 1 and 10, we solve the SDP. W is extracted by computing the first eigenvector of Z. The cardinality of W as a function of Θ is illustrated in Figure 2.3[8]. We can see that Θ is indeed a good indicator of the cardinality. Note that when $\Theta = 1$, one always gets a W with a single element being one and all others being zeros in this example. We also plot an example of the obtained W with $\Theta = 3$ and W without sparseness constraints for an intuitive comparison in Figure 2.4.

The second experiment is conducted on the Wine data described in Table 2.1. S_b and S_v are constructed using SDP_1 using the same parameters shown in Table 2.1. The final projected dimension is 8. We want each column of W to be sparse. In other words, only a subset of features are selected. We compare our performance against the simple thresholding method [7]. Table 2.6 reports the classification

(2.9) is written into $\mathbf{1}^\top(Z_+ + Z_-)\mathbf{1} \leq \Theta\sqrt{d}$ and $Z_+ \geq 0$, $Z_- \geq 0$ (element-wise non-negative). Here $\mathbf{1}$ is a column vector with all elements being ones.

[8]For calculating cardinality, an element is regarded as non-zero if its absolute magnitude is larger than 10% of the vector's maximum absolute magnitude.

error. As expected, the proposed algorithm performs better than the simple thresholding method.

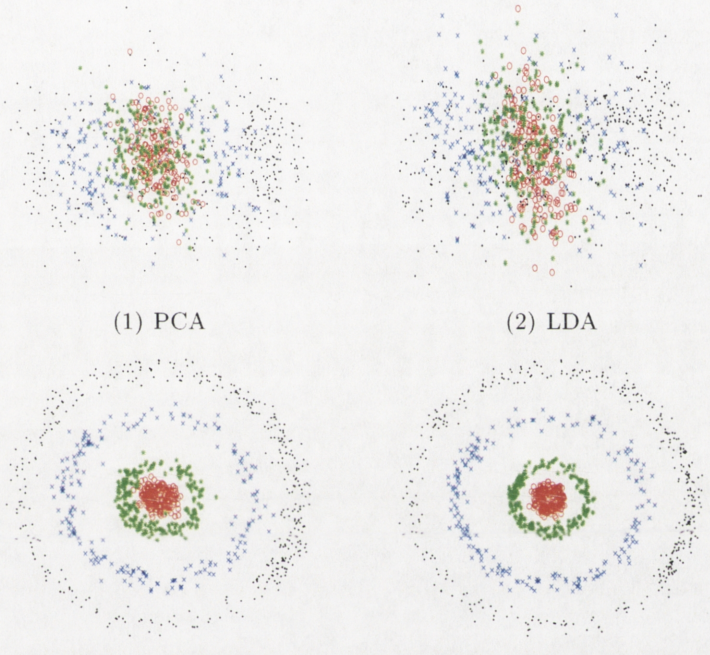
cardinality	4	5	6
SDP with sparseness	6.98(3.63)	5.47(2.44)	5.09(2.74)
simple thresholding	7.55(2.52)	7.55(3.72)	8.02(3.35)

Table 2.6: Classification error of a 3-NN classifier on the Wine dataset w.r.t. the cardinality of each row of W. Each case is run 20 times.

2.7 Conclusion

In this work we have presented a new supervised dimensionality reduction algorithm. It has two key components: a global optimization strategy for solving the trace quotient problem; and a new trace quotient cost function specifically designed for linear dimensionality reduction. The proposed algorithms are consistently better than LDA. Experiments show that our algorithms' performance is comparable to the LMNN algorithm but with computational advantages. Future work will be focused on the following directions. First, we have confined ourself to linear dimensionality reduction in this chapter. We will explore the kernel approach. We already know that some nonlinear dimensionality reduction algorithms like kernel LDA also need to solve trace quotient problems. Second, new strategies will be devised to define an optimal discriminative set $\mathcal{D}$. [15] might be a direction. Third, SDP's computational complexity is heavy. New efficient methods are desirable to make it scalable to large-size problems.

Figure 2.2: Subfigures (1)(2) show the data projected into 2D using PCA and LDA. Both fail to recover the data structure. Subfigures (3)(4) show the results obtained by the two SDPs proposed in this chapter. The local structure of the data is preserved after projected by SDPs. Subfigures (5)(6) are the results when the rear eight dimensions are extremely noisy. In this case the neighboring relationships based on the Euclidean distance in the input space are completely meaningless. Subfigures (7)(8) successfully recover data's underlying structure given user-provided neighborhood graphs.



(1) PCA (2) LDA

(3) SDP_1, $k = 230, k' = 5$ (4) SDP_2, $k = 5, k' = 5$

(5) SDP_1, $k = 230, k' = 5$ (6) SDP_2, $k = 5, k' = 5$

(7) SDP_1, $k = 230, k' = 5$ (8) SDP_2, $k = 5, k' = 5$

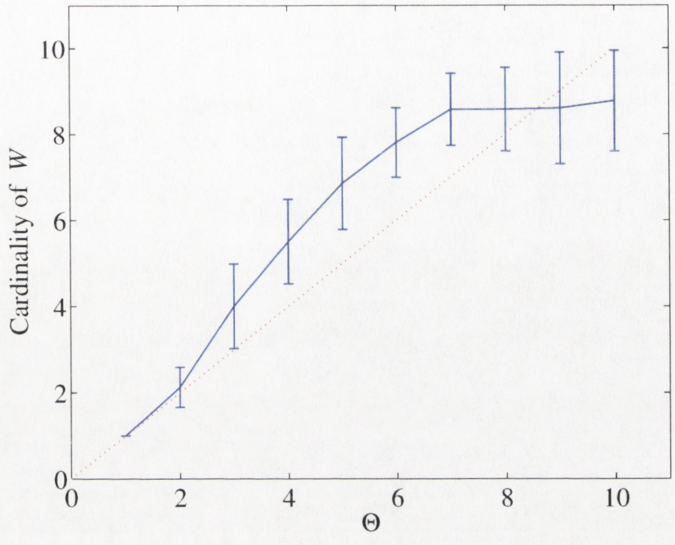


Figure 2.3: Cardinality of W v.s. Θ. The error bar shows the standard deviation averaged on 40 runs.

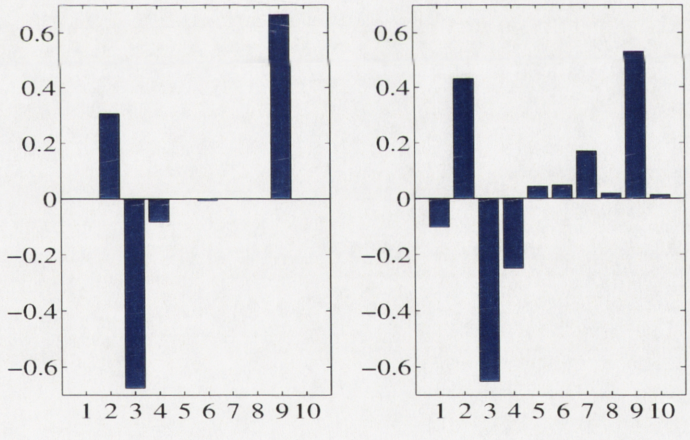


Figure 2.4: The projection vector W obtained with sparseness constraints $\Theta = 3$ (left) and no sparseness constraints (right). Clearly the sparseness constraints do produce a sparse W while most of W's elements are active without sparseness constraints.

Chapter 3

PSDBoost: Matrix-Generation Linear Programming for Mahalanobis Metric Learning

In this chapter, we consider the problem of learning a positive semidefinite matrix. The critical issue is how to preserve positive semidefiniteness during the course of learning. Our algorithm is mainly inspired by LPBoost [13] and the general greedy convex optimization framework of Zhang [54]. We demonstrate the essence of the algorithm, termed PSDBoost (positive semidefinite Boosting), by focusing on a few different applications in machine learning. The proposed PSDBoost algorithm extends traditional Boosting algorithms in that its parameter is a positive semidefinite matrix with trace being one instead of a classifier. PSDBoost is based on the observation that any trace-one positive semidefinite matrix can be decomposed into linear convex combinations of trace-one rank-one matrices, which serve as base learners of PSDBoost. Numerical experiments are presented.

3.1 Introduction

Column generation (CG) [33] is a technique widely used in linear programming (LP) for solving large-sized problems. Thus far it has mainly been applied to solve problems with linear constraints. The proposed work here—which we dub matrix generation (MG)—extends the column generation technique to non-polyhedral semidefinite constraints. In particular, as an application we show how to use it for solving a semidefinite metric learning problem. The fundamental idea is to rephrase a bounded semidefinite constraint into a polyhedral one with infinitely many variables. This construction opens possibilities for use of the highly developed linear programming technology. Given the limitations of current semidefinite programming (SDP) solvers to deal with large-scale problems, the work presented here is of importance for many real applications.

The choice of a metric has a direct effect on the performance of many algorithms such as the simplest k-NN classifier and some clustering algorithms. Much effort has been spent on learning a good metric for pattern recognition and data mining. Clearly a good metric is task-dependent: different applications should use different measures for (dis)similarity between objects. We show how a Mahalanobis metric is learned from examples of proximity comparison among triples of training data. For example, assuming that we are given triples of images $\mathbf{a}_i$, $\mathbf{a}_j$ and $\mathbf{a}_k$ ($\mathbf{a}_i$, $\mathbf{a}_j$ have same labels and $\mathbf{a}_i$, $\mathbf{a}_k$ have different labels, $\mathbf{a}_i \in \mathbb{R}^D$), we want to learn a metric between pairs of images such that the distance from $\mathbf{a}_j$ to $\mathbf{a}_i$ ($\mathbf{dist}_{ij}$) is smaller than from $\mathbf{a}_k$ to $\mathbf{a}_i$ ($\mathbf{dist}_{ik}$). Triplets like this are the input of our metric learning algorithm. By casting the problem as optimization of the inner product of the linear transformation matrix and its transpose, the formulation is based on solving a semidefinite program. The algorithm finds an *optimal* linear transformation that maximizes the margin between distances $\mathbf{dist}_{ij}$ and $\mathbf{dist}_{ik}$.

A major drawback of this formulation is that current SDP solvers utilizing

interior-point (IP) methods do not scale well to large problems with computation complexity roughly $O(n^{4.5})$ (n is the number of variables). On the other hand, linear programming is much better in terms of scalability. State-of-the-art solvers like CPLEX [26] can solve large problems up to millions of variables and constraints. This motivates us to develop an LP approach to solve our SDP metric learning problem.

We define some notations here. A bold lower case letter $\boldsymbol{x}$ represents a column vector and an upper case letter X is a matrix. We denote the space of $D \times D$ symmetric matrices by $\mathbb{S}^D$, and positive semidefinite matrices by $\mathbb{S}_+^D$. $\mathbf{Tr}(\cdot)$ is the trace of a square matrix and $\langle X, Z \rangle = \mathbf{Tr}(XZ^\top) = \sum_{ij} X_{ij} Z_{ij}$ calculates the inner product of two matrices. An element-wise inequality between two vectors writes $\boldsymbol{u} \leq \boldsymbol{v}$, which means $\boldsymbol{u}_i \leq \boldsymbol{v}_i$ for all i.

We use $X \succcurlyeq 0$ to indicate that matrix X is positive semidefinite. For a matrix $X \in \mathbb{S}^D$, the following statements are equivalent: (1) $X \succcurlyeq 0$ ($X \in \mathbb{S}_+^D$); (2) All eigenvalues of X are nonnegative ($\lambda_i(X) \geq 0$, $i = 1, \cdots, D$); and (3) $\forall \boldsymbol{u} \in \mathbb{R}^D$, $\boldsymbol{u}^\top X \boldsymbol{u} \geq 0$.

3.2 Related Work

We overview some relevant work in this section.

Column generation was first proposed by Dantzig and Wolfe [10] for solving some special structured linear programs with extremely large number of variables. [33] has presented a comprehensive survey on this technique. The general idea of CG is that, instead of solving the original large-scale problem (master problem), one works on a restricted master problem with a reasonably small subset of variables at each step. The dual of the restricted master problem is solved by the simplex method, and the optimal dual solution is used to find the new column to be included into the restricted master problem. LPBoost [13] is a direct application of

CG in Boosting. For the first time, LPBoost shows that in an LP framework, unknown weak hypotheses can be learned from the dual although the space of all weak hypotheses is infinitely large. This is the highlight of LPBoost, which has directly inspired our work.

Metric learning using convex optimization has attracted a lot of attention recently [48, 46, 38]. These works have made it possible to learn distance functions that are more appropriate for a specific task, based on partially labeled data or proximity constraints. These techniques improve classification or clustering accuracy by taking advantage of prior information. There is plenty of work reported. We list a few that are most relevant to ours. [48] learns a Mahalanobis metric for clustering using convex optimization to minimize the distance between examples belonging to the same class, while at the same time restricting examples in difference classes not to be too close. The work in [46] also learns a Mahalanobis metric using SDP by optimizing a modified k-NN classifier. They have used first-order alternating projection algorithms, which are faster than generic SDP solvers. The authors in [38] learns a Mahalanobis by considering proximity relationships of training examples. The final formulation is also an SDP. They replace the positive semidefinite (p.s.d.) conic constraint using a sequence of linear constraints under the fact that a diagonal dominance matrix must be p.s.d. (but not *vice versa*). In other words the conic constraint is replaced by a more strict one. The feasibility set shrinks and the solution obtained is not necessarily a solution of the original SDP.

3.3 Preliminaries

We begin with some basic definitions that will be useful.

3.3.1 Extreme points of trace-one semidefinite matrices

Before we present our main results, we prove an important theorem that serves the basis of the proposed algorithm.

Definition 3.3.1. *For any positive integer M, given a set of points $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_M\}$ in a real vector or matrix space* Sp, *the* convex hull *of* Sp *spanned by M elements in* Sp *is defined as:*

$$\mathbf{conv}_M(\mathrm{Sp}) = \left\{ \sum_{i=1}^{M} \theta_i \boldsymbol{x}_i \middle| \theta_i \geq 0, \sum_{i=1}^{M} \theta_i = 1, \boldsymbol{x}_i \in \mathrm{Sp} \right\}.$$

Define the convex hull[1] *of* Sp *as:*

$$\begin{aligned} \mathbf{conv}(\mathrm{Sp}) &= \bigcup_M \mathbf{conv}_M(\mathrm{Sp}) \\ &= \left\{ \sum_{i=1}^{M} \theta_i \boldsymbol{x}_i \middle| \theta_i \geq 0, \sum_{i=1}^{M} \theta_i = 1, \boldsymbol{x}_i \in \mathrm{Sp}, M \in \mathbb{Z}_+ \right\}. \end{aligned}$$

Here $\mathbb{Z}_+$ denotes the set of all positive integers.

Definition 3.3.2. *Let us define Γ_1 to be the space of all positive semidefinite matrices $X \in \mathbb{S}_+^D$ with trace equaling one:*

$$\Gamma_1 = \{X \mid X \succcurlyeq 0, \mathbf{Tr}(X) = 1\};^2$$

and Ω_1 to be the space of all positive semidefinite matrices with both trace and rank equaling one:

$$\Omega_1 = \{Z \mid Z \succcurlyeq 0, \mathbf{Tr}(Z) = 1, \mathbf{rank}(Z) = 1\}.$$

[1] Strictly, the union of convex hulls may not be a convex hull in general. It is a linear convex span.

[2] Such a matrix X is called a density matrix, which is one of the main concepts in quantum physics. A density matrix of rank one is called a pure state, and a density matrix of rank higher than one is called a mixed state.

We also define Γ_2 *as the convex hull of* Ω_1, i.e.,

$$\Gamma_2 = \mathbf{conv}(\Omega_1).$$

Lemma 3.3.3. *Let* Ω_2 *be a convex polytope defined as* $\Omega_2 = \{\boldsymbol{\lambda} \in \mathbb{R}^D | \, \lambda_k \geq 0, \forall k = 1, \cdots, D, \sum_{k=1}^{D} \lambda_k = 1\}$, *then the points with only one element equaling one and all the others being zeros are the extreme points (vertexes) of* Ω_2. *All the other points cannot be extreme points.*

Proof: Without loss of generality, let us consider such a point $\boldsymbol{\lambda}' = \{1, 0, \cdots, 0\}$. If $\boldsymbol{\lambda}'$ is not an extreme point of Ω_2, then it must be expressed as an convex combination of a few *other* points in Ω_2: $\boldsymbol{\lambda}' = \sum_{i=1}^{M} \theta_i \boldsymbol{\lambda}^i$, $\theta_i > 0$, $\sum_{i=1}^{M} \theta_i = 1$ and $\boldsymbol{\lambda}^i \neq \boldsymbol{\lambda}'$. Then we have equations: $\sum_{i=1}^{M} \theta_i \lambda_k^i = 0$, $\forall k = 2, \cdots, D$. It follows that $\lambda_k^i = 0$, $\forall i$ and $k = 2, \cdots, D$. That means, $\lambda_1^i = 1$ $\forall i$. This is inconsistent with $\boldsymbol{\lambda}^i \neq \boldsymbol{\lambda}'$. Therefore such a convex combination does not exist and $\boldsymbol{\lambda}'$ must be an extreme point. It is trivial to see that any $\boldsymbol{\lambda}$ that has more than one active element is an convex combination of the above-defined extreme points. So they cannot be extreme points. □

Theorem 3.3.4. Γ_1 *equals to* Γ_2; i.e., Γ_1 *is also the convex hull of* Ω_1. *In other words, all* $Z \in \Omega_1$, *forms the set of extreme points of* Γ_1.

Proof: It is easy to check that any convex combination $\sum_i \theta_i Z^i$, such that $Z^i \in \Omega_1$, resides in Γ_1, with the following two facts: (1) a convex combination of p.s.d. matrices is still a p.s.d. matrix; (2) $\mathbf{Tr}\big(\sum_i \theta_i Z^i\big) = \sum_i \big(\theta_i \, \mathbf{Tr}(Z^i)\big) = 1$. By denoting $\lambda_1 \geq \cdots \geq \lambda_D \geq 0$ the eigenvalues of a $Z \in \Gamma_1$, we know that $\lambda_1 \leq 1$ because $\sum_{i=1}^{D} \lambda_i = \mathbf{Tr}(Z) = 1$. Therefore, all eigenvalues of Z must satisfy: $\lambda_i \in [0, 1]$, $\forall i = 1, \cdots, D$ and $\sum_i^D \lambda_i = 1$. By looking at the eigenvalues of Z and using Lemma 3.3.3, it immediately follows that a matrix Z such that $Z \succcurlyeq 0$, $\mathbf{Tr}(Z) = 1$ and $\mathbf{rank}(Z) > 1$ cannot be an extreme point of Γ_1. The only

candidates for extreme points are those rank-one matrices ($\lambda_1 = 1$ and $\lambda_{2,\cdots,D} = 0$). Moreover, it is not possible that some rank-one matrices are extreme points and others are not because the other two constraints $Z \succcurlyeq 0$ and $\mathbf{Tr}(Z) = 1$ do not distinguish between different rank-one matrices.
Hence, all $Z \in \Omega_1$ forms the set of extreme points of Γ_1. Furthermore, Γ_1 is a convex and compact set, which must have extreme points. Krein-Milman Theorem [29] tells us that a convex and compact set is equal to the convex hull of its extreme points. □

This theorem is a special case of the results from [36] in the context of eigenvalue optimization. A different proof for the above theorem's general version can also be found in [14]. In the context of SDP optimization, what is of interest about Theorem 3.3.4 is as follows: it tells us that a bounded p.s.d. matrix constraint $X \in \Gamma_1$ can be equivalently replaced with a set of constrains which belong to Γ_2. At first glance, this is a highly counterintuitive proposition because Γ_2 involves many more complicated constraints. Both θ_i and Z^i ($\forall i = 1, \cdots, M$) are unknown variables. Even worse, M could be extremely (or even indefinitely) large.

3.3.2 Boosting

Boosting is an example of ensemble learning, where multiple learners are trained to solve the same problem. Typically a boosting algorithm [40] creates a single strong learner by incrementally adding base (weak) learners to the final strong learner. The base learner has an important impact on the strong learner. In general, a boosting algorithm builds on a user-specified base learning procedure and runs it repeatedly on modified data that are outputs from the previous iterations.
The inputs to a boosting algorithm are a set of training example $\boldsymbol{x}$, and their

corresponding class labels y. The final output strong classifier takes the form

$$F_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{i=1}^{M} \theta_i f_i(\boldsymbol{x}). \tag{3.1}$$

Here $f_i(\cdot)$ is a base learner. From Theorem 3.3.4, we know that a matrix $X \in \Gamma_1$ can be decomposed as

$$X = \sum_{i=1}^{M} \theta_i Z^i, Z^i \in \Omega_1. \tag{3.2}$$

By observing the similarity between Equations (3.1) and (3.2), we may view Z^i as a weak classifier and the matrix X as the strong classifier we want to learn. This is exactly the problem that boosting methods have been designed to solve. This observation inspires us to solve a special type of SDPs using boosting techniques. A sequential greedy approximation algorithm proposed by Zhang [54] is an efficient way of solving a class of convex problems, which provides fast convergence rates. It is shown in [54] that boosting algorithms can be interpreted within the general framework of [54]. The main idea of sequential greedy approximation is as follows. Given an initialization $\boldsymbol{u}^0 \in \mathbb{V}$, $\mathbb{V}$ can be a subset of a linear vector space, a matrix space or a functional space. The algorithm finds $\boldsymbol{u}^i \in \mathbb{V}$, $i = 1, \cdots$, and $0 \leq \lambda \leq 1$ such that the cost function $F((1-\lambda)\boldsymbol{u}^{i-1} + \lambda \boldsymbol{u}^i)$ is approximately minimized; Then the solution $\boldsymbol{u}^i$ is updated as $\boldsymbol{u}^i = (1-\lambda)\boldsymbol{u}^{i-1} + \lambda \boldsymbol{u}^i$ and the iteration goes on.

3.4 Large-margin Semidefinite Metric Learning

We consider the Mahalanobis metric learning problem as an example although the proposed technique can be applied to many other problems in machine learning such as nonparametric kernel matrix learning [30].

We are given a set of training examples $\mathbf{a}_i \in \mathbb{R}^D$, $i = 1, 2, \cdots$. The task is to learn a distance metric such that with the learned metric, classification or clustering will achieve better performance on testing data. The information available is a bunch

of relative distance comparisons. Mathematically we are given a set $\mathcal{S}$ which contains the training triplets: $\mathcal{S} = \{(\mathbf{a}_i, \mathbf{a}_j, \mathbf{a}_k) | \, \mathbf{dist}_{ij} < \mathbf{dist}_{ik}\}$, where $\mathbf{dist}_{ij}$ measures distance between $\mathbf{a}_i$ and $\mathbf{a}_j$ with a certain metric. In this work we focus on the case that **dist** calculates the Mahalanobis distance. Equivalently we are learning a linear transformation $P \in \mathbb{R}^{D \times d}$ such that **dist** is the Euclidean distance in the projected space: $\mathbf{dist}_{ij} = \left\| P^\top \mathbf{a}_i - P^\top \mathbf{a}_j \right\|_2^2 = (\mathbf{a}_i - \mathbf{a}_j)^\top P P^\top (\mathbf{a}_i - \mathbf{a}_j)$. It is not difficult to see that the inequalities in the set $\mathcal{S}$ are non-convex because a difference of quadratic terms in P is involved. In order to *convexify* the inequalities in $\mathcal{S}$, a new variable $X = PP^\top$ is instead used. This is a typical technique for modeling an SDP problem [6]. We wish to maximize the margin that is defined as the distance between $\mathbf{dist}_{ij}$ and $\mathbf{dist}_{ik}$. That is, $\rho = \mathbf{dist}_{ik} - \mathbf{dist}_{ij} = (\mathbf{a}_i - \mathbf{a}_k)^\top X (\mathbf{a}_i - \mathbf{a}_k) - (\mathbf{a}_i - \mathbf{a}_j)^\top X (\mathbf{a}_i - \mathbf{a}_j)$. Also one may use soft margin to tolerate noisy data. Putting these thoughts together, the final convex program we want to optimize is:

$$
\begin{aligned}
\max_{\rho, X, \boldsymbol{\xi}} \;\; & \rho - C \sum\nolimits_{r=1}^{|\mathcal{S}|} \xi_r \\
\text{s.t.} \;\; & X \succcurlyeq 0, \mathbf{Tr}(X) = 1, \boldsymbol{\xi} \geq 0, \\
& (\mathbf{a}_i - \mathbf{a}_k)^\top X (\mathbf{a}_i - \mathbf{a}_k) - (\mathbf{a}_i - \mathbf{a}_j)^\top X (\mathbf{a}_i - \mathbf{a}_j) \geq \rho - \xi_r, \\
& \forall (\mathbf{a}_i, \mathbf{a}_j, \mathbf{a}_k) \in \mathcal{S}.
\end{aligned}
\tag{3.3}
$$

Here r indexes the training set $\mathcal{S}$. $|\mathcal{S}|$ denotes the size of $\mathcal{S}$. C is a trade-off parameter that balances the training error and the margin. Same as in support vector machine, the slack variable $\boldsymbol{\xi} \geq 0$ corresponds to the soft-margin hinge loss. Note that the constraint $\mathbf{Tr}(X) = 1$ removes the scale ambiguity because the distance inequalities are scale invariant.

To simplify our exposition, we write

$$A^r = (\mathbf{a}_i - \mathbf{a}_k)(\mathbf{a}_i - \mathbf{a}_k)^\top - (\mathbf{a}_i - \mathbf{a}_j)(\mathbf{a}_i - \mathbf{a}_j)^\top. \tag{3.4}$$

The last constraint in (3.3) is then written

$$\langle A^r, X \rangle \geq \rho - \xi_r, \forall A^r \text{ built from } \mathcal{S}; r = 1, \cdots |\mathcal{S}|. \tag{3.5}$$

Problem (3.3) is a typical SDP since it has a linear cost function and linear constraints plus a p.s.d. conic constraint. Therefore it can be solved using off-the-shelf SDP solvers like CSDP [5]. As mentioned general interior-point SDP solvers do not scale well to large-sized problems. Current solvers can only solve problems up to a few thousand variables, which makes many applications intractable. For example, in face recognition if the inputs are 30×30 images, then $D = 900$ and there would be 0.41 million variables. Next we show how we reformulate the above SDP into an LP.

3.5 Boosting via Matrix-Generation Linear Programming

Using Theorem 3.3.4, we can replace the p.s.d. conic constraint in (3.3) with a linear convex combination of rank-one unitary p.s.d. matrices: $X = \sum_{i=1}^{M} \theta_i Z^i$. Substituting X in Problem (3.3), we obtain

$$\begin{aligned} \max_{\rho,\boldsymbol{\theta},\boldsymbol{\xi},Z} \quad & \rho - C \sum\nolimits_{r=1}^{|\mathcal{S}|} \xi_r \\ \text{s.t.} \quad & \boldsymbol{\xi} \geq 0, \\ & \left\langle A^r, \textstyle\sum_{i=1}^{M} \theta_i Z^i \right\rangle = \sum_{i=1}^{M} \langle A^r, Z^i \rangle \theta_i \geq \rho - \xi_r, \\ & \forall A^r \text{ built from } \mathcal{S}; r = 1, \cdots |\mathcal{S}|, \end{aligned} \tag{P_1}$$

$$\sum_{i=1}^{M} \theta_i = 1, \boldsymbol{\theta} \geq 0,$$
$$Z^i \in \Omega_1, i = 1, \cdots, M.$$

This above problem is still very hard to solve since it has non-convex rank constraints and an indefinite number of variables (M is indefinite because there are an indefinite number of rank-one matrices). However if we somehow know matrices Z^i $(i = 1, \cdots)$ *a priori*, we can then drop all the constraints imposed on Z^i $(i = 1, \cdots)$ and the problem becomes a linear program; or more precisely a semi-infinite linear program (SILP) because it has an infinitely large set of variables $\boldsymbol{\theta}$.

Column generation is a state-of-the-art method for optimally solving difficult large-scale optimization problems. It is a method to avoid considering all variables of a problem *explicitly*. If an LP has extremely many variables (columns) but much fewer constraints, CG can be very beneficial. The crucial insight behind CG is: for an LP problem with many variables, the number of non-zero variables of the optimal solution is equal to the number of constraints, hence although the number of possible variables may be large, we only need a small subset of these in the optimal solution. It works by only considering a small subset of the entire variable set. Once it is solved, we ask the question: "Are there any other variables that can be included to improve the solution?". So we must be able to solve the *subproblem*: given a set of dual values, one either identifies a variable that has a favorable reduced cost, or indicates that such a variable does not exist. In essence, CG finds the variables with negative reduced costs without explicitly enumerating all variables. For a general LP, this may not be possible. But for some types of problems it is possible.

We now consider Problem (P_1) as if all Z^i, $(i = 1, \cdots)$ were known. The dual of

(P_1) is easily derived:

$$\begin{aligned} \min_{\pi, \boldsymbol{w}} \quad & \pi \\ \text{s.t.} \quad & \textstyle\sum_{r=1}^{|\mathcal{S}|} \langle A^r, Z^i \rangle w_r \leq \pi, \; i = 1, \cdots, M, \\ & \textstyle\sum_{r=1}^{|\mathcal{S}|} w_r = 1, \\ & 0 \leq w_r \leq C, \; r = 1, \cdots, |\mathcal{S}|. \end{aligned} \tag{D_1}$$

For convex programs with strong duality, the dual gap is zeros, which means the optimal value of the primal and dual problems coincide. For LPs and SDPs, strong duality holds under very mild conditions (almost always satisfied by LPs and SDPs considered here) [6].

We now only consider a small subset of the variables in the primal; *i.e.*, only a subset of Z (denoted by $\tilde{Z}$)[3] is used. The LP solved using $\tilde{Z}$ is usually termed *restricted master problem* (RMP). Because the primal variables correspond to the dual constraints, solving RMP is equivalent to solving a relaxed version of the dual problem. With a finite $\tilde{Z}$, the first set of constraints in (D_1) are finite, and we can solve the LP that satisfies all the existing constraints.

If we can prove that among all the constraints that we have not added to the dual problem, no single constraint is violated, then we can conclude that solving the restricted problem is equivalent to solving the original problem. Otherwise, there exists at least one constraint that is violated. The violated constraints correspond to variables in primal that are not in RMP. Adding these variables to RMP leads to a new RMP that needs to be re-optimized. In our case, by finding the violated constraint, we generate a rank-one matrix Z'. Hence, as in LPBoost [13] we have a

[3]We also use $\tilde{\boldsymbol{\theta}}$, $\tilde{\pi}$ and $\tilde{\boldsymbol{w}}$ *etc.* to denote the solution of the current RMP and its dual.

base learning algorithm as an oracle that either finds a new Z' such that

$$\sum_{r=1}^{|\mathcal{S}|}\langle A^r, Z'\rangle w_r > \tilde{\pi},$$

where $\tilde{\pi}$ is the solution of the current restricted problem, or a guarantee that such a Z' does not exist. To make convergence fast, we find the one that has largest deviation. That is,

$$Z' = \text{argmax}_Z \left\{\sum_{r=1}^{|\mathcal{S}|}\langle A^r, Z\rangle \tilde{w}_r, \text{ s.t. } Z \in \Omega_1\right\}. \tag{B$_1$}$$

Again here $\tilde{w}_r$ $(r = 1, \cdots, |\mathcal{S}|)$ are obtained by solving the current restricted dual problem (D$_1$). Let us denote Opt(B$_1$) the optimal value of the optimization problem in (B$_1$). We now have a criterion that guarantees the optimal convex combination over all Z's satisfying the constraints in Γ_2 has been found. If Opt(B$_1$) $\leq \tilde{\pi}$, then we are done—we have solved the original problem.
The presented algorithm is a variant of the CG technique. At each iteration, a new matrix is generated, hence the name *matrix generation.*

3.5.1 Base learning algorithm

In this section, we show that the optimization problem (B$_1$) can be exactly and efficiently solved using eigen-decomposition.
From $Z \succcurlyeq 0$ and $\mathbf{rank}(Z) = 1$, we know that Z has the format: $Z = \boldsymbol{u}\boldsymbol{u}^\top$, $\boldsymbol{u} \in \mathbb{R}^D$; and $\mathbf{Tr}(Z) = 1$ means $\|\boldsymbol{u}\|_2 = 1$. We have

$$\sum_{r=1}^{|\mathcal{S}|}\langle A^r, Z\rangle \tilde{w}_r = \langle \sum_{r=1}^{|\mathcal{S}|} \tilde{w}_r A^r, Z\rangle = \boldsymbol{u}\big(\sum_{r=1}^{|\mathcal{S}|} \tilde{w}_r A^r\big)\boldsymbol{u}^\top.$$

By denoting

$$\tilde{\mathbf{H}} = \sum_{r=1}^{|\mathcal{S}|} \tilde{w}_r A^r, \tag{3.6}$$

the optimization in (B_1) equals:

$$\max_{\boldsymbol{u}} \ \boldsymbol{u}^\top \tilde{\mathbf{H}} \boldsymbol{u}, \ \text{subject to} \|\boldsymbol{u}\|_2 = 1. \tag{3.7}$$

It is clear that the largest eigenvalue of $\tilde{\mathbf{H}}$, $\lambda_{\max}(\tilde{\mathbf{H}})$, and its corresponding eigenvector $\boldsymbol{u}_1$ give the solution to the above problem. Note that $\tilde{\mathbf{H}}$ is symmetric. Therefore we have the solution of the original problem (B_1): $\text{Opt}(B_1) = \lambda_{\max}(\tilde{\mathbf{H}})$ and $Z' = \boldsymbol{u}_1 \boldsymbol{u}_1^\top$.

There are approximate eigenvalue solvers, which guarantee that for a symmetric matrix U and any $\varepsilon > 0$, a vector $\boldsymbol{v}$ is found such that $\boldsymbol{v}^\top U \boldsymbol{v} \geq \lambda_{\max} - \varepsilon$. To approximately find the largest eigenvalue and eigenvector can be very efficient using Lanczos or power method. We use the MATLAB function *eigs* to calculate the largest eigenvector, which calls mex files of ARPACK. ARPACK is a collection of Fortran subroutines designed to solve large scale eigenvalue problems. When the input matrix is symmetric, this software uses a variant of the Lanczos process called the implicitly restarted Lanczos method [9].

Putting all the above analysis together, we summarize our PSDBoost algorithm for metric learning in Algorithm 3. Note that, in practice, we can relax the convergence criterion by setting a small positive threshold $\varepsilon' > 0$ in order to obtain a good approximation quickly. Namely the convergence criterion is $\text{Opt}(B_1) \leq \pi + \varepsilon'$.

The algorithm has some appealing properties. Each iteration the solution is provably better than the preceding one, and has rank at most one larger. Hence after M iterations the algorithm attains a solution with rank at most M. The algorithm preserves CG's property that each iteration improves the quality of the solution. The bounded rank follows the fact that $\mathbf{rank}(A + B) \leq \mathbf{rank}(A) + \mathbf{rank}(B)$, $\forall$ matrices A and B.

An advantage of the proposed PSDBoost algorithm over standard boosting

schemes is the totally-corrective weight update in each iteration, which leads faster convergence. The coordinate descent optimization employed by standard boosting algorithms is known to have a slow convergence rate in general. However, the price of this totally-corrective update is obvious. PSDBoost spans the space of the parameter X incrementally. The computational cost for solving the subproblem grows with the number of linear constraints, which increases by one at each iteration. Also it needs more and more memory to store the generated base learner Z^i as represented by a series of unit vectors. To alleviate this problem, one can use a *selection and compression mechanism* as the aggregation step of bundle methods [4]. When the size of of the bundle becomes too large, bundle methods select columns to be discarded and the selected information is aggregated into a single one. It can be shown that as long as the aggregated column is introduced in the bundle, the bundle algorithm remains convergent, although different selection of discarded columns may lead to different convergence speeds. See [4] for details.

3.6 Experiments

In the first experiment, we have artificially generated 600 points in 24 dimensions. Therefore the learned metric is of size 24×24. The triplets are obtained in this way: For a point $\mathbf{a}_i$, we find its nearest neighbor in the same class $\mathbf{a}_j$ and its nearest neighbor in the different class $\mathbf{a}_k$. We subsample to have 550 triplets for training. To show the convergence, we have plotted the optimal values of the dual problem (D_1) at each iteration in Figure 3.1. We see that PSDBoost quickly converges to the near-optimal solution. We have observed the so-called *tailing-off effect* of CG on large datasets. While a near-optimal solution is approached considerably fast, only little progress per iteration is made close to the optimum. Stabilization techniques have been introduced to partially alleviate this problem [33]. However, approximate solutions are sufficient for most machine learning

tasks. Moreover, we usually are not interested in the numerical accuracy of the solution but the test error for many problems such as metric and kernel learning. The second experiment uses the Pendigits data from the UCI repository that contains handwritten samples of digits 1, 5, 7, 9. The data for each digits are 16-dimensional. 80 samples for each digit are used for training and 500 for each digit for testing. The results show that PSDBoost converges quickly and the learned metric is very similar to the results obtained by a standard SDP solver. The classification errors on testing data with a 1-nearest neighbor are identical using the metrics learned by PSDBoost and a standard SDP solver. Both are 1.3%.

3.7 Conclusion

We have presented a new boosting algorithm, PSDBoost, for learning a positive semidefinite matrix. In particular, as an example, we use PSDBoost to learn a distance metric for classification. PSDBoost can also be used to learn a kernel matrix, which is of interest in machine learning. We are currently exploring new applications with PSDBoost. Also we want to know what kind of SDP optimization problems can be approximately solved by PSDBoost.

Algorithm 3 PSDBoost for semidefinite metric learning.

Input: Training set triplets $(\mathbf{a}_i, \mathbf{a}_j, \mathbf{a}_k) \in \mathcal{S}$; Calculate A^r, $r = 1, \cdots$ from $\mathcal{S}$ using Equation (3.4).

Initialization:

1. $M = 1$ (no bases selected);
2. $\boldsymbol{\theta} = \mathbf{0}$ (all primal coefficients are zeros);
3. $\pi = 0$;
4. $w_r = \frac{1}{|\mathcal{S}|}$, $r = 1, \cdots, |\mathcal{S}|$ (uniform dual weights).

while true **do**

1. Find a new base Z' by solving Problem (B_1), *i.e.*, eigen-decomposition of $\tilde{\mathbf{H}}$ in (3.6);
2. **if** $\mathrm{Opt}(\mathrm{B}_1) \leq \pi$ **then** break (problem solved);
3. Add Z' to the restricted master problem, which corresponds to a new constraint in Problem (D_1);
4. Solve the dual (D_1) to obtain updated π and w_r ($r = 1, \cdots, |\mathcal{S}|$);
5. $M = M + 1$ (base count).

end

Output:

1. Calculate the primal variable $\boldsymbol{\theta}$ from the optimality conditions and the last solved dual LP;
2. The learned p.s.d. matrix $X \in \mathbb{R}^{D \times D}$, $X = \sum_{i=1}^{M} \theta_i Z^i$.

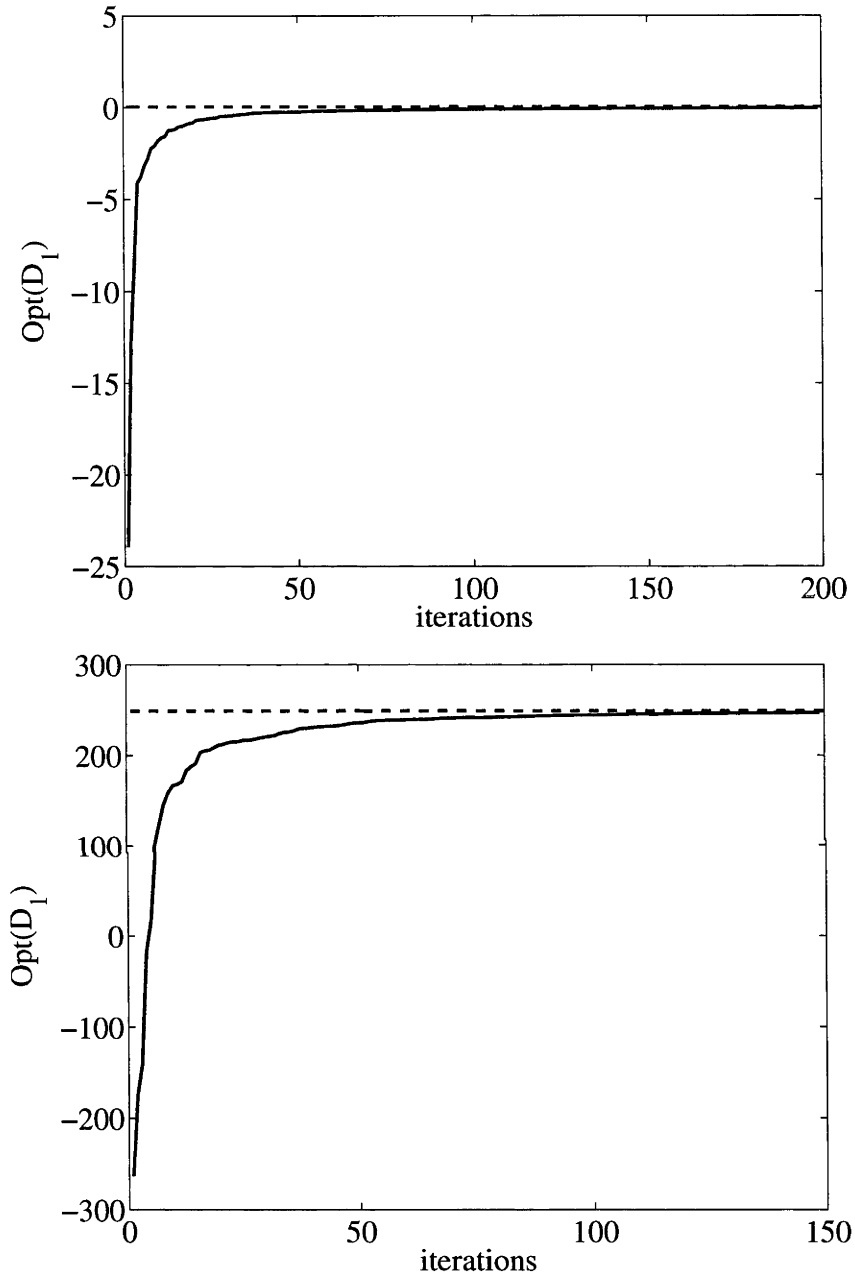


Figure 3.1: The objective value of the dual problem (D_1) on the first (top) and second (bottom) experiment. The dashed line shows the ground truth obtained by directly solving the original primal SDP (3.3) using interior-point methods.

Chapter 4

Conclusion

We have studied distance metric learning in this thesis. Two methods are proposed for learning a Mahalanobis metric. The first algorithm uses sequential semidefinite programming to solve a trace quotient problem. It is claimed that many dimensionality reduction (or metric learning) problems can be written into a trace quotient formulation. We have also shown that within this new convex optimization framework, other constraints like sparsity constraints can be easily accommodated.

The second algorithm tries to learn a quadratic Mahalanobis distance from proximity comparisons. It does not directly use the label information. The learning problem can be formulated as a semidefinite program, which does not scale well on large-size problems. In order to improve its scalability, a new matrix-generation method, termed PSDBoost, is proposed. PSDBoost is inspired by boosting algorithms in machine learning. At each iteration, a linear program needs to be solved, which is computationally much cheaper.

We have only covered a small part of the metric learning topic.

Although a large body of work has been done on this topic in the literature, many issues are not truly solved so far. We list a few in the following.

1. Unsupervised distance metric learning. Dimension reduction has been extensively studied, unsupervised distance metric learning is a more general problem. There is no general principle framework to learn a distance metric without pairwise constraints and side-information, although unsupervised distance metric learning has many potential applications.

2. Efficiency and scalability. As discussed before, many existing metric learning techniques have difficulties to handle large-scale problems, *e.g.*, [48, 46]. Efficient and simple methods are needed. A research direction is how to simplify the learning problem while improving the performance with less training samples and larger dimensions.

3. Nonlinear distance metric learning. We have only discussed linear metric learning problems in this thesis. However, most real problems are nonlinear and experiments show nonlinear methods usually perform better. Although many metric learning formulation can be kernelized using the so-called kernel trick, it is an interesting topic to learn an explicit nonlinear distance metric.

Conclusively, as an important technique in machine learning and statistics, there are still many open problems in distance metric learning.

Bibliography

[1] AGARWAL, S., CHANDRAKER, M., KAHL, F., BELONGIE, S., AND KRIEGMAN, D. J. Practical global optimization for multiview geometry. In *Proc. Eur. Conf. Comp. Vis.* (Graz, Austria, 2006), vol. 1, pp. 592–605.

[2] BELKIN, M., AND NIYOGI, P. Laplacian Eigenmaps for dimensionality reduction and data representation. *Neural Comp. 15*, 6 (2003), 1373–1396.

[3] BEN-TAL, A., AND NEMIROVSKI, A. S. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications.* SIAM, Philadelphia, PA, USA, 2001.

[4] BONNANS, J. F., GILBERT, J. C., LEMARÉCHAL, C., AND SAGASTIZÁBAL, C. A. *Numerical Optimization: Theoretical and Practical Aspects (1st edition).* Springer-Verlag, Berlin, 2003.

[5] BORCHERS, B. CSDP, a C library for semidefinite programming. *Optim. Methods and Softw. 11*, 1 (1999), 613–623.

[6] BOYD, S., AND VANDENBERGHE, L. *Convex Optimization.* Cambridge University Press, 2004.

[7] CADIMA, J., AND JOLLIFFE, I. T. Loading and correlations in the interpretation of principle compenents. *J. Appl. Statistics 22*, 2 (1995), 203–214.

[8] CAI, D., HE, X., HAN, J., AND ZHANG, H.-J. Orthogonal Laplacianfaces for face recognition. *IEEE Trans. Image Process. 15*, 11 (Nov. 2006), 3608–3614.

[9] CALVETTI, D., REICHEL, L., AND SORENSEN, D. C. An implicitly restarted Lanczos method for large symmetric eigenvalue problems. *Elec. Trans. Numer. Anal 2* (Mar 1994), 1–21. `http://etna.mcs.kent.edu`.

[10] DANTZIG, G. B., AND WOLFE, P. Decomposition principle for linear programs. *Operation Res. 8*, 1 (1960), 101–111.

[11] D'ASPREMONT, A., GHAOUI, L. E., JORDAN, M. I., AND LANCKRIET, G. R. G. A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*.

[12] DAVIS, J. V., KULIS, B., JAIN, P., SRA, S., AND DHILLON, I. S. Information-theoretic metric learning. In *Proc. Int. Conf. Mach. Learn.* (Corvalis, Oregon, 2007), ACM Press, pp. 209–216.

[13] DEMIRIZ, A., BENNETT, K., AND SHAWE-TAYLOR, J. Linear programming boosting via column generation. *Mach. Learn. 46*, 1-3 (2002), 225–254.

[14] FILLMORE, P. A., AND WILLIAMS, J. P. Some convexity theorems for matrices. *Glasgow Math. Journal 12* (1971), 110–117.

[15] FRANSENS, R., DE PRINS, J., AND VAN GOOL, L. SVM-based nonparametric discriminant analysis, an application to face detection. In *Proc. IEEE Int. Conf. Comp. Vis.* (Nice, France, 2003), vol. 2, pp. 1289–1296.

[16] GILAD-BACHRACH, R., NAVOT, A., AND TISHBY, N. Margin based feature selection–theory and algorithms. In *Proc. Int. Conf. Mach. Learn.* (Banff, Alberta, Canada, 2004).

[17] GLOBERSON, A., AND ROWEIS, S. Metric learning by collapsing classes. In *Proc. Adv. Neural Inf. Process. Syst.* (2005).

[18] GOLDBERGER, J., ROWEIS, S., HINTON, G., AND SALAKHUTDINOV, R. Neighbourhood component analysis. In *Proc. Adv. Neural Inf. Process. Syst.* (2004), MIT Press.

[19] GOLUB, G. H., AND VAN LOAN, C. *Matrix Computations*, 3rd ed. Johns Hopkins University Press, 1996.

[20] HEILER, M., AND SCHNORR, C. Learning non-negative sparse image codes by convex programming. In *Proc. IEEE Int. Conf. Comp. Vis.* (Beijing, China, 2005), pp. 1667–1674.

[21] HEILER, M., AND SCHNÖRR, C. Controlling sparseness in non-negative tensor factorization. In *Proc. Eur. Conf. Comp. Vis.* (Graz, Austria, 2006), vol. 1, pp. 56–67.

[22] HOI, S., JIN, R., AND LYU, M. R. Learning nonparametric kernel matrices from pairwise constraints. In *Proc. Int. Conf. Mach. Learn.* (Corvalis, Oregon, 2007), ACM Press, pp. 361–368.

[23] HORN, R. A., AND JOHNSON, C. R. *Matrix Analysis.* Cambridge University Press, 1985.

[24] HOYER, P. O. Non-negative matrix factorization with sparseness constraints. *J. Mach. Learn. Res. 5* (2004), 1457–1469.

[25] HUA, G., VIOLA, P. A., AND DRUCKER, S. M. Face recognition using discriminatively trained orthogonal rank one tensor projections. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (Minneapolis, MN, 2007).

[26] ILOG, INC. CPLEX 11.1, 2008. `http://www.ilog.com/products/cplex/`.

[27] JIN, Z., YANG, J.-Y., HU, Z.-S., AND LOU, Z. Face recognition based on the uncorrelated discriminant transformation. *Pattern Recogn. 34*, 7 (2001), 1405–1416.

[28] KEUCHEL, J., SCHNÖRR, C., SCHELLEWALD, C., AND CREMERS, D. Binary partitioning, perceptual grouping, and restoration with semidefinite programming. *IEEE Trans. Pattern Anal. Mach. Intell. 25*, 11 (2003), 1364–1379.

[29] KREIN, M., AND MILMAN, D. On extreme points of regular convex sets. *Studia Mathematica 9* (1940), 133–138.

[30] KULIS, B., SUSTIK, M., AND DHILLON, I. Learning low-rank kernel matrices. In *Proc. Int. Conf. Mach. Learn.* (Pittsburgh, Pennsylvania, 2006), pp. 505–512.

[31] LANCKRIET, G. R. G., CRISTIANINI, N., BARTLETT, P., GHAOUI, L. E., AND JORDAN, M. I. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res. 5* (2004), 27–72.

[32] LI, H., JIANG, T., AND ZHANG, K. Efficient and robust feature extraction by maximum margin criterion. *IEEE Trans. Neural Netw. 17*, 1 (2006), 157–165.

[33] LÜBBECKE, M. E., AND DESROSIERS, J. Selected topics in column generation. *Operation Res. 53*, 6 (2005), 1007–1023.

[34] MOSEK APS. The MOSEK optimization toolbox for matlab manual, version 5.0, revision 93, 2008. `http://www.mosek.com/`.

[35] NEWMAN, D., HETTICH, S., BLAKE, C., AND MERZ, C. UCI repository of machine learning databases, 1998. `http://archive.ics.uci.edu/ml/`.

[36] OVERTON, M. L., AND WOMERSLEY, R. S. On the sum of the largest eigenvalues of a symmetric matrix. *SIAM J. Matrix Anal. Appl. 13*, 1 (1992), 41–45.

[37] OVERTON, M. L., AND WOMERSLEY, R. S. Optimality conditions and duality theory for minimizing sums of the largest eigenvalues of symmetric matrices. *Math. Program. 62*, 1-3 (1993), 321–357.

[38] ROSALES, R., AND FUNG, G. Learning sparse metrics via linear programming. In *Proc. ACM Int. Conf. Knowledge Discovery & Data Mining* (Philadelphia, PA, USA, 2006), pp. 367–373.

[39] SCHAIBLE, S. Fractional programming. II on Dinkelbach's algorithm. *Management Sci. 22*, 8 (1976), 868–873.

[40] SCHAPIRE, R. E. Theoretical views of boosting and applications. In *Proc. Int. Conf. Algorithmic Learn. Theory* (London, UK, 1999), Springer-Verlag, pp. 13–25.

[41] SCHULTZ, M., AND JOACHIMS, T. Learning a distance metric from relative comparisons. In *Proc. Adv. Neural Inf. Process. Syst.* (2003), MIT Press.

[42] SHEN, C., WELSH, A., AND WANG, L. PSDBoost: Matrix-generation linear programming for positive semidefinite matrices learning. In *Proc. Adv. Neural Inf. Process. Syst.* (Vancouver, Canada, 2008), D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., pp. 1473–1480.

[43] STURM, J. F. Using SeDuMi 1.02, a matlab toolbox for optimization over symmetric cones (updated for version 1.05). *Optim. Methods and Softw. 11-12* (1999), 625–653.

[44] TORRESANI, L., AND LEE, K. C. Large margin component analysis. In *Proc. Adv. Neural Inf. Process. Syst.* (Vancouver, Canada, 2006), pp. 1385–1392.

[45] TÜTÜNCÜ, R. H., TOH, K. C., AND TODD, M. J. Solving semidefinite-quadratic-linear programs using SDPT3. *Math. Program. 95*, 2 (2003), 189–217.

[46] WEINBERGER, K. Q., BLITZER, J., AND SAUL, L. K. Distance metric learning for large margin nearest neighbor classification. In *Proc. Adv. Neural Inf. Process. Syst.* (2005), pp. 1473–1480.

[47] WEINBERGER, K. Q., AND SAUL, L. K. Unsupervised learning of image manifolds by semidefinite programming. *Int. J. Comp. Vis. 70*, 1 (2006), 77–90.

[48] XING, E., NG, A., JORDAN, M., AND RUSSELL, S. Distance metric learning, with application to clustering with side-information. In *Proc. Adv. Neural Inf. Process. Syst.* (2002), MIT Press.

[49] YAN, S., AND TANG, X. Trace quotient problems revisited. In *Proc. Eur. Conf. Comp. Vis.* (2006), Springer-Verlag, pp. 232–244.

[50] YAN, S., XU, D., ZHANG, B., ZHANG, H., YANG, Q., AND LIN, S. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Trans. Pattern Anal. Mach. Intell. 29*, 1 (2007), 40–51.

[51] YANG, J., YANG, J., AND ZHANG, D. What's wrong with Fisher criterion? *Pattern Recogn. 35*, 11 (2002), 2665–2668.

[52] YE, J., AND XIONG, T. Null space versus orthogonal linear discriminant analysis. In *Proc. Int. Conf. Mach. Learn.* (Pittsburgh, Pennsylvania, 2006), pp. 1073–1080.

[53] YU, J., AMORES, J., SEBE, N., RADEVA, P., AND TIAN, Q. Distance learning for similarity estimation. *IEEE Trans. Pattern Anal. Mach. Intell. 30*, 3 (2008), 451–462.

[54] ZHANG, T. Sequential greedy approximation for certain convex optimization problems. *IEEE Trans. Inf. Theory 49*, 3 (2003), 682–691.

Glossary of Terms

Term	Explanation
hinge loss	The hinge loss function is defined as $\text{hinge}(z) = \max(0, 1 - z)$, which is a typical loss function for classification.
convex cone	A set X is called a convex cone if for any $x, y \in X$ and any scalars $a \geq 0$ and $b \geq 0$, $ax + by \in X$.
trace quotient problem	For a given pair (A, B) of real symmetric positive semidefinite matrices, and a given non-zero matrix X, the trace quotient problem in this thesis is defined as $$f(A, B; X) = \frac{\mathbf{Tr}(X^\top AX)}{\mathbf{Tr}(X^\top BX)}.$$
Rayleigh quotient problem	For a given pair (A, B) of real symmetric positive semidefinite matrices, and a given non-zero vector $\boldsymbol{x}$, the generalized Rayleigh quotient problem is defined as $$f(A, B; \boldsymbol{x}) = \frac{\boldsymbol{x}^\top A\boldsymbol{x}}{\boldsymbol{x}^\top B\boldsymbol{x}}.$$ When B is an identity matrix, the problem is called the standard Rayleigh quotient problem.

fractional programming	Let $f(\cdot), g(\cdot)$ and $h(\cdot)$ be real-valued functions defined on a subset of $\mathbb{R}^n$. Fractional programming solves the following problem: $$\min_{\boldsymbol{x}} \frac{f(\boldsymbol{x})}{h(\boldsymbol{x})}, \text{ s.t. } g(\boldsymbol{x}) \leq 0.$$
Lanczos power method	It is an iterative algorithm invented by C. Lanczos that is an adaptation of power methods to find eigenvalues and eigenvectors of a square matrix or the singular value decomposition of a rectangular matrix.
p.s.d.	positive semidefinite
PCA	principal component analysis
LDA	linear discriminant analysis
NCA	neighborhood component analysis
LMNN	large margin nearest neighbor
CG	column generation
GEVD	generalized eigenvalue decomposition
RMP	restricted master problem
Frobenius norm	The Frobenius norm of a matrix X is calculated as $\|X\|_{\mathrm{F}} = \sqrt{\sum_{ij} X_{ij}^2}$.
$\langle A, B \rangle$	$\langle A, B \rangle = \sum_{ij} A_{ij} B_{ij}$ is the inner product of two matrices.
$A \succcurlyeq B$	$A \succcurlyeq B$ means $A - B$ is positive semidefinite, $A - B \succcurlyeq 0$.

This thesis was typeset with LaTeX 2$_\varepsilon$[1] by the author.

[1] LaTeX 2$_\varepsilon$ is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society.