Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Fixed-Rank Supervised Metric Learning on Riemannian Manifold

Yadong Mu
AT&T Labs Research
Middletown, NJ 07748, U.S.A.
Email: myd@research.att.com

Abstract

Metric learning has become a critical tool in many machine learning tasks. This paper focuses on learning an optimal Mahalanobis distance matrix (parameterized by a positive semi-definite matrix $\mathbf{W}$) in the setting of supervised learning. Recently, particular research attention has been attracted by low-rank metric learning, which requires that matrix $\mathbf{W}$ is dominated by a few large singular values. In the era of high feature dimensions, low-rank metric learning effectively reduces the storage and computation overheads. However, existing low-rank metric learning algorithms usually adopt sophisticated regularization (such as LogDet divergence) for encouraging matrix low-rankness, which unfortunately incurs iterative computations of matrix SVD. In this paper, we tackle low-rank metric learning by enforcing fixed-rank constraint on the matrix $\mathbf{W}$. We harness the Riemannian manifold geometry of the collection of fixed-rank matrices and devise a novel second-order Riemannian retraction operator. The proposed operator is efficient and ensures that $\mathbf{W}$ always resides on the manifold. Comprehensive numerical experiments conducted on benchmarks clearly suggest that the proposed algorithm is substantially superior or on par with the state-of-the-art in terms of k-NN classification accuracy. Moreover, the proposed manifold retraction operator can be also naturally applied in generic rank-constrained machine learning algorithms.

Introduction

Metric learning methods (Yang 2006; Kulis 2013; Bellet, Habrard, and Sebban 2013) strive to effectively gauge the pairwise distance between two data objects. In the past few years, metric learning has become ubiquitous and stimulated by a large spectrum of applications in the research fields such as machine learning and information retrieval. For a number of fundamental machine learning algorithms (such as k-means clustering and k-NN classification) and applications (such as image search), the naive Euclidean distance is often insufficient for describing domain-specific data affinities. Learning a well-defined metric (typically in the form of Mahalanobis distance metric) plays a crucial role in obtaining state-of-the-art performance and a vast literature of metric learning algorithms under various problem settings has been created, including semi-supervised learning (Hoi, Liu, and Chang 2008; Liu et al. 2010), fully supervised learning (Xing et al. 2003; Ying and Li 2012; Hu, Lu, and Tan 2014), transfer learning (Luo et al. 2014) or structural learning (Lim, Lanckriet, and McFee 2013). Prominent real-world applications of metric learning techniques include face verification (Hu, Lu, and Tan 2014), image retrieval (Hoi, Liu, and Chang 2010), image annotation (Verma and Jawahar 2012), and bioinformatics (Kato and Nagano 2010), etc.

Instead of using nonlinear distance functions (Weinberger and Saul 2006; Kedem et al. 2012), the main scope of this paper is to optimize the (squared) Mahalanobis distance (Xing et al. 2003) between two data points, which is mathematically expressed as below

$$\mathcal{D}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{W} (\mathbf{x}_i - \mathbf{x}_j), \tag{1}$$

where the parametric matrix $\mathbf{W}$ is often assumed to be positive semi-definite in order to ensure a well-defined metric.

In supervised metric learning, the matrix $\mathbf{W}$ is critically optimized based on pre-specified similar/dissimilar constraints. Profiting from advances in numerical optimization techniques (particularly sophisticated schemes on PSD matrices (Shen et al. 2009)), supervised metric learning is making big strides towards elevating the accuracies of machine learning systems. Importantly, the number of free parameters in $\mathbf{W}$ is quadratic with regard to the feature dimension, which indicates a huge feasible region in many real-world scenarios. Avoiding the over-fitting is therefore crucial for attaining good accuracy on unseen data points. To this end, a large body of metric learning literature has actively explored the regularization imposed on $\mathbf{W}$ (Jin, Wang, and Zhou 2009; Liu and Vemuri 2012; Chen et al. 2012). For example, the information-theoretic approaches, exemplified by sparse distance metric learning (**SDML**) (Qi et al. 2009) and sparse semi-supervised metric Learning (**S3ML**) (Liu et al. 2010), propose to use LogDet divergence and L_1 regularization on $\mathbf{W}$. The LogDet divergence effectively aligns the learned $\mathbf{W}$ with our empirical belief of $\mathbf{W}$. The justification for the sparse regularization primarily is a theoretical result in covariance matrix estimation (Ravikumar et al. 2011).

Matrix low-rankness is another widespreadly adopted regularization in machine learning, such as in robust PCA (Candès et al. 2011) and subspace segmentation (Liu, Lin, and Yu 2010). A number of low-rank metric learning

algorithms (Zhong, Huang, and Liu 2011; Bi et al. 2011; Kulis, Sustik, and Dhillon 2009; Liu et al. 2015) have been proposed. A major merit of low-rank $\mathbf{W}$ is having the factorization $\mathbf{W} = \mathbf{L}\mathbf{L}^\top$ ($\mathbf{L}$ is low-rank and non-unique), such that the distance computation can be greatly expedited by $(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{W}(\mathbf{x}_i - \mathbf{x}_j) = (\mathbf{L}^\top \mathbf{x}_i - \mathbf{L}^\top \mathbf{x}_j)^\top (\mathbf{L}^\top \mathbf{x}_i - \mathbf{L}^\top \mathbf{x}_j)$. For example, the work in (Kulis, Sustik, and Dhillon 2009) designed a rank-1 update rule, picking a single constraint for processing at each iteration. The authors prove that the algorithm preserves matrix rank, so it can learn low-rank metric matrices when the input parameters have low rank. However, it requires iterating many cycles through all constraints to reach convergence. Existing low-rank metric learning approaches suffer from two intrinsic limitations. First, since most low-rankness encouraging norms can hardly eliminate small singular values of $\mathbf{W}$, they cannot bound the rank of intermediate solutions a priori, which makes the compact factorization $\mathbf{W} = \mathbf{L}\mathbf{L}^\top$ infeasible. Second, a full SVD of an intermediate $\mathbf{W}$ is often routinely conducted after each update, in order to casting it back to the PSD matrix cone, as seen in the work of (Liu et al. 2015). For high-dimensional data, the cost of iterative full SVD is computationally unaffordable.

Inspired by recent advances in optimization over the manifold of low-rank matrices (Absil, Mahony, and Sepulchre 2007; Meyer, Bonnabel, and Sepulchre 2011; Shalit, Weinshall, and Chechik 2010), in this paper we propose a novel low-rank metric learning algorithm, which is essentially characterized by a smooth objective function and a fixed-rank constraint on $\mathbf{W}$. The smoothness of the objective function enables an efficient gradient descent optimization scheme. The rank constraint explicitly ensures a low-rank solution.By virtue of the geometry of the Riemannian manifold formed by low-rank matrices, we expose a novel retraction operator defined on this Riemannian manifold, which serves as our key technical contribution. The operator enjoys both conceptual elegance and lower complexity. On the one hand, it is rigorously proved to be a second-order Riemannian retraction. In other words, it concurrently accomplishes both the gradient descent and fixed-rank projection in a unified computational procedure. On the other hand, the computation does not rely on full SVD of $\mathbf{W}$, which is a significant improvement in comparison with existing works. We conduct comprehensive evaluations on seven benchmarks for corroborating the effectiveness of the proposed algorithm.

Problem Specification

Notations: We here clarify the notations used throughout this paper. Let $\mathcal{S}_+^d, \mathcal{S}^d$ denote the sets of positive semidefinite or generic symmetric matrices of size $d \times d$ respectively. For any matrix $\mathbf{M}$, let $\|\mathbf{M}\|_F$ represent the Frobenius norm (the sum of squared singular values). $tr(\mathbf{M})$ denotes the trace of a square matrix, namely $\sum_i \mathbf{M}(i,i)$ over all valid index i. Let $[\mathbf{x}]_+ = \max(\mathbf{x}, \mathbf{0})$ be an element-wise filter function.

Our problem setting is virtually compatible with conventional supervised metric learning. A dataset $\mathcal{X} = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times [0, 1, \ldots, \ell]\}_{i=1}^n$ is provided for training purpose, where d is the dimensionality of observations. Any integer from $[1 : \ell]$ represents a unique valid label. For un-annotated samples, their label indicators are simply set to 0. For the task of Mahalanobis-style metric learning, the goal is to seek for a distance matrix $\mathbf{W}^* \in \mathcal{S}_+^d$, such that the distance value between two feature vectors $\mathbf{x}_i, \mathbf{x}_j$ can be well described by $(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{W}^* (\mathbf{x}_i - \mathbf{x}_j)$. Our formulation is comprised of two criteria:

Criterion-I: Rank Constraint and Matrix Complexity. As exhibited by the achievement in metric learning and subspace learning (Wang and Tang 2004), a budgeted number of new dimensions can be sufficient for distinguishing similar or dissimilar data pairs. In light of this observation, we aggressively fix the rank of $\mathbf{W}$ to be equal or below a prespecified constant of k. The value of k, which reflects our belief of the intrinsic dimension of the data, is often far smaller compared with the feature dimension. The optimal choice of k can hardly be identified in theory. Nonetheless, as later revealed by our experiments, the k-NN classification accuracy is stably excellent for a large range of k. Therefore an empirical guess of the value of k often works in practice.

To mitigate the risk of over-fitting in tackling out-of-sample data, we leverage (squared) matrix Forbenius norm to regularize $\mathbf{W}$, which is essentially the sums of all squared elements in $\mathbf{W}$. Formally it can be expressed as $tr(\mathbf{W}^\top \mathbf{W})$. Note that the popularly-used matrix nuclear norm (Liu et al. 2015) is not favorable in our setting, owing to the fixed-rank constraint. We now have the regularization terms imposed on $\mathbf{W}$, as following:

$$\mathcal{R}(\mathbf{W}) = \tfrac{1}{2}\lambda tr(\mathbf{W}^\top \mathbf{W}) + f_b(rank(\mathbf{W}) \leq k), \qquad (2)$$

where $f_b(\cdot)$ represents the barrier function. It returns $+\infty$ when the statement is false, otherwise 0. $\lambda > 0$ is a regularization parameter to balance the relative importance of $tr(\mathbf{W}^\top \mathbf{W})$ and other loss terms.

Criterion-II: Margin Optimization. Supervision information can be utilized in various forms for the task of metric learning. Let us simply adopt the pairwise form owing to its simplicity. A piece of pairwise annotation clearly specifies whether two samples share the same label. To universally encode the supervision information, let us define a *target matrix* $\mathbf{T}$:

$$\mathbf{T}(i,j) = \begin{cases} -1, & y_i = y_j,\ y_i, y_j \neq 0, \\ 1, & y_i \neq y_j,\ y_i, y_j \neq 0, \\ 0, & \text{otherwise.} \end{cases} \qquad (3)$$

To simplify the statement, let us re-define the pairwise distance among $\mathbf{x}_i, \mathbf{x}_j$:

$$\mathcal{D}(\mathbf{x}_i, \mathbf{x}_j) = 2(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{W}(\mathbf{x}_i - \mathbf{x}_j) - 1. \qquad (4)$$

The above distance is expected to be close to the values specified in $\mathbf{T}$. Directly regressing $\mathcal{D}(\mathbf{x}_i, \mathbf{x}_j)$ to $\mathbf{T}(i,j)$ in least-squares is problematic, since it triggers penalties when the resultant distance is "better" than $\mathbf{T}$ (*e.g.*, $\mathcal{D}(\mathbf{x}_i, \mathbf{x}_j) < -1$ for $\mathbf{T}(i,j) = -1$). To mitigate the over penalty, we adopt a soft margin based loss term, namely $[1 - \mathcal{D}(\mathbf{x}_i, \mathbf{x}_j)]_+$ for $\mathbf{T}(i,j) = 1$ or $[\mathcal{D}(\mathbf{x}_i, \mathbf{x}_j) - (-1)]_+$ for $\mathbf{T}(i,j) = -1$.

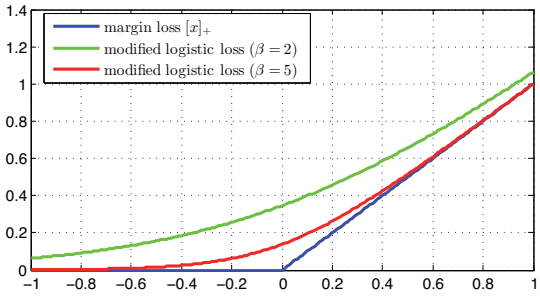

Figure 1: Smoothing margin loss via modified logistic losses.

The margin loss can then be presented in a unified form below:

$$\mathcal{L}_{margin}(\mathbf{W}) = \sum_{i,j}[\mathbf{T}(i,j)\cdot(\mathbf{T}(i,j) - \mathcal{D}(\mathbf{x}_i,\mathbf{x}_j))]_+, \quad (5)$$

where multiplying $\mathbf{T}(i,j)$ excludes any un-annotated data from contributing to the margin loss.

The soft margin function $[x]_+$ has a non-differentiable point at $x = 0$ and thus only sub-gradient is available. To improve the scalability to gigantic data, we expect the objective function to be smooth at every point in order to employ efficient gradient descent optimization. To eliminate the irregularity of $[x]_+$, we here adopt the *modified logistic loss* (Zhang and Oles 2001; Zhang et al. 2003) to smooth the margin loss. Let $\sigma(x) = 1/(1+\exp(-x))$ denote the logistic function. The smoothing trick relies on the approximation below:

$$[x]_+ = \lim_{\beta\to+\infty} -1/\beta \cdot \log\left(\sigma(-\beta x)\right). \quad (6)$$

The r.h.s of Eqn. (6) asymptotically converges to the l.h.s. with increasingly larger β. A few exemplar approximate losses are displayed in Figure 1. We set $\beta = 5$ throughout this work. A smooth margin loss can be obtained by a marriage of Eqn. (6) with Eqn. (5), namely

$$\mathcal{L}_{margin}(\mathbf{W};\beta) = -\tfrac{1}{\beta}\sum_{i,j}\log\left(\sigma(-\beta\xi_{ij})\right), \quad (7)$$

where ξ_{ij} is an auxiliary variable defined by

$$\xi_{ij} = \mathbf{T}(i,j)\cdot(\mathbf{T}(i,j) - \mathcal{D}(\mathbf{x}_i,\mathbf{x}_j)). \quad (8)$$

Putting all above considerations together, we obtain the final problem formulation of supervised metric learning:

$$\begin{aligned} \arg\min_{\mathbf{W}} \quad & \mathcal{L}_{margin}(\mathbf{W}) + \tfrac{1}{2}\lambda tr(\mathbf{W}^\top\mathbf{W}) \quad (9)\\ s.t. \quad & rank(\mathbf{W}) \le k,\ \mathbf{W}\in\mathcal{S}_+^d. \end{aligned}$$

The Optimization Method

Our algorithm intelligently optimizes Problem (9) over the Riemannian manifold of low-rank matrices. A gradient descent scheme is adopted for progressively reducing the objective value. The prominent complication in the optimization stems from the fixed-rank and PSD constraints, namely the new solution is required to stay rank-k and a PSD matrix after moving along the descending direction. Direct projection onto the rank-k PSD cone is problematic since the projected solution is not guaranteed to still decrease the objective. Therefore, an efficient line-search procedure is critical.

Algorithm 1 The Proposed Optimization Method on Riemannian Manifold

Input: data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$, target matrix $\mathbf{T} \in \{-1,0,1\}^{n\times n}$.
Output: optimal solutions $\mathbf{W}^*$;
Parameter: step sizes η_0, η_{min}, linear search attenuation factor $s \in (0,1)$, rank constraint k;

Initialize $\mathbf{W}$ with random numbers;
Perform partial eig-decomposition on $\mathbf{W}$ to extract k largest singular pairs, obtaining $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$;
$\eta = \eta_0$;
while not converged **do**
  Calculate $\mathbf{G}$ by Equation (13);
  Project $\mathbf{G}$ to obtain $\mathbf{P} \leftarrow \mathbf{U}^\top\mathbf{G}\mathbf{U}$;
  $\eta \leftarrow \eta/s$;
  while $\eta \ge \eta_{min}$ **do**
    $\mathbf{A} \leftarrow \eta\mathbf{U}\mathbf{\Lambda} + \mathbf{G}\mathbf{U}$;
    Perform economic QR-decomposition on matrix $\mathbf{A}$, obtaining $(\mathbf{Q}_A, \mathbf{R}_A) \leftarrow qr(\mathbf{A})$;
    $\mathbf{B} \leftarrow \mathbf{R}_A(\eta\mathbf{P} + \mathbf{\Lambda})\mathbf{R}_A^\top$;
    Perform eigen-decomposition on matrix $\mathbf{B}$, obtaining $(\mathbf{U}_B, \mathbf{S}_B) \leftarrow eig(\mathbf{B})$;
    $\widehat{\mathbf{U}} \leftarrow \mathbf{Q}_A\mathbf{U}_B$, $\widehat{\mathbf{\Lambda}} \leftarrow [\mathbf{S}_B]_+$, $\mathbf{W}' = \widehat{\mathbf{U}}\widehat{\mathbf{\Lambda}}\widehat{\mathbf{U}}^\top$;
    if $tr((\mathbf{W}' - \mathbf{W})\mathbf{G}) < 0$ **then**
      $\eta \leftarrow s\eta$; **continue**;
    end if
    if $\mathbf{W}'$ satisfies the Armijo rule **then**
      $\mathbf{W} \leftarrow \mathbf{W}'$; $\mathbf{U} \leftarrow \widehat{\mathbf{U}}$; $\mathbf{\Lambda} \leftarrow \widehat{\mathbf{\Lambda}}$; **break**;
    end if
    $\eta \leftarrow s\eta$;
  end while
end while
$\mathbf{W}^* \leftarrow \mathbf{W}$;

Gradient Calculation

For notational brevity, let us introduce an auxiliary variable $\mathbf{\Sigma}_\xi \in \mathbb{R}^{n\times n}$ with $\mathbf{\Sigma}_\xi(i,j) = \sigma(\beta\xi_{ij})$, and use $\mathbf{X}\in\mathbb{R}^{d\times n}$ to represent the data matrix attained by piling all feature vectors in a column-wise fashion.

The gradient of $\lambda tr(\mathbf{W}^\top\mathbf{W})$ is trivially obtained as

$$\tfrac{1}{2}\partial\lambda tr(\mathbf{W}^\top\mathbf{W})/\partial\mathbf{W} = \lambda\mathbf{W}. \quad (10)$$

Regarding the margin loss term, utilizing the standard calculus rule $\partial\log(\sigma(x))/\partial x = \sigma(-x)$, its gradient with regard to $\mathbf{W}$ can be computed as

$$\begin{aligned} & \tfrac{1}{4}\cdot\partial\mathcal{L}_{margin}(\mathbf{W})/\partial\mathbf{W}\\ =\ & \tfrac{1}{4}\textstyle\sum_{i,j\in[1:n]}\sigma(\beta\xi_{ij})\cdot\partial\xi_{ij}\partial\mathbf{W}\\ =\ & -\tfrac{1}{2}\textstyle\sum_{i,j}\sigma(\beta\xi_{ij})\cdot\mathbf{T}(i,j)\cdot(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^\top\\ =\ & -\textstyle\sum_{i,j}\sigma(\beta\xi_{ij})\cdot\mathbf{T}(i,j)\cdot(\mathbf{x}_i\mathbf{x}_i^\top - \mathbf{x}_i\mathbf{x}_j^\top) \quad (11)\\ =\ & -(\mathbf{X}\mathbf{D}\mathbf{X}^\top - \mathbf{X}\left(\mathbf{T}\circ\mathbf{\Sigma}_\xi\right)\mathbf{X}^\top)\\ =\ & -\mathbf{X}\left(\mathbf{D} - \mathbf{T}\circ\mathbf{\Sigma}_\xi\right)\mathbf{X}^\top, \quad (12) \end{aligned}$$

where $\circ$ represents the Hadamard product for element-wise multiplication. Eqn. (11) holds owing to the symmetry of $\mathbf{T}\circ\mathbf{\Sigma}_\xi$ in our problem setting. $\mathbf{D}$ is a diagonal matrix whose diagonal elements are the sums of corresponding rows of $\mathbf{T}\circ\mathbf{\Sigma}_\xi$. It is verified that $\mathbf{D} - \mathbf{T}\circ\mathbf{\Sigma}_\xi$ is the Laplacian matrix calculated under current estimations of $\mathbf{W}$.

Riemannian gradient retraction for updating $\mathbf{W}$

We use a variable $\mathbf{G}$ to denote the *negative gradient*, which represents the best possible descending direction. $\mathbf{G}$ can be computed by combining Equations (10)(12):

$$\mathbf{G} = 4\mathbf{X}\left(\mathbf{D} - \mathbf{T} \circ \mathbf{\Sigma}_{\xi}\right)\mathbf{X}^{\top} - \lambda\mathbf{W}. \tag{13}$$

The complication of optimizing $\mathbf{W}$ primarily stems from the non-convex rank constraint $rank(\mathbf{W}) \leq k$. Given a step size η, a move along the negative gradient direction, which brings $\mathbf{W} + \eta\mathbf{G}$, nearly always violates the rank-k constraint. One may advocate projecting $\mathbf{W} + \eta\mathbf{G}$ back to be rank-k PSD cone by keeping k largest singular values of $\mathbf{W} + \eta\mathbf{G}$. It can be computationally done through SVD. This strategy is widely known as *projected gradient method* in the field of numerical optimization. However, we find in practice that the new solution obtained by this simple projection operation tends to heavily deviate from the original descent direction $\mathbf{G}$ and results in a worse solution. Intuitively, a better tactic for ensuring a better solution is concurrently considering gradient descent and any post-processing (such as the SVD-based projection) followed.

The step size η plays a key role for a gradient-descent based optimization method. Conventionally, fine tuning the step size can be done by sophisticated schemes, such as *Armijo* or *Goldstein* conditions (Nocedal and Wright 2006). For low-rank metric learning methods, heuristic choice of step size is often used. For example, the work in (Davis et al. 2007) adopted the Lagrange multiplier calculated from a single supervisory similar/dissimilar constraint as the step size. As a crucial caveat at the computational aspect, one shall definitely avoid iteratively performing SVD of $\mathbf{W} + \eta\mathbf{G}$ for each candidate η. As shown later, the above consideration has motivated our development of an algorithm which only requires one-time SVD of $\mathbf{W}$ during the entire optimization.

Now we proceed to introduce the proposed Riemannian retraction operator. The new method is based on the concept of low-rank matrix manifold. Importantly, all $d \times d$ symmetric matrices of rank k lie on a manifold embedded in the *ambient space* $\mathbb{R}^{d\times d}$. Denote the manifold as

$$\mathcal{M}_{d,k} = \{\mathbf{W} \;:\; \mathbf{W} \in \mathcal{S}^d,\;\; rank(\mathbf{W}) = k\}. \tag{14}$$

Each point $\mathbf{W}$ on $\mathcal{M}_{d,k}$ defines a *tangent space*, denoted $T_{\mathbf{W}}\mathcal{M}$. A typical Riemannian gradient descent-and-retraction procedure consists of two consecutive operations:

1. *Projection* that casts the gradient $\mathbf{G}$ from the ambient space into $T_{\mathbf{W}}\mathcal{M}$, the tangent space at current solution;
2. *Retraction* that maps a point in the tangent space back to the manifold of low-rank matrices.

The procedure is intuitively presented in Figure 2, where $\mathbf{G} \mapsto \widehat{\mathbf{G}}$, $\widehat{\mathbf{G}} \mapsto \mathbf{W}'$ correspond to the projection and retraction operations respectively.

Let $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\top}$ be the thin SVD of the rank-k matrix $\mathbf{W}$, where $\mathbf{U} \in \mathbb{R}^{d\times k}$ defines the bases of $\mathbf{W}$'s row/column subspace and $\mathbf{\Lambda} \in \mathbb{R}^{k\times k}$ is a diagonal matrix. To define the complemental subspace induced by $\mathbf{U}$, we introduce the notation $\mathbf{R} \in \mathbb{R}^{d\times r}$ $(r = d-k)$ with $\mathbf{R}\mathbf{U}^{\top} = \mathbf{0}, \mathbf{R}^{\top}\mathbf{R} = \mathbf{I}_r$ and $\mathbf{U}\mathbf{U}^{\top} + \mathbf{R}\mathbf{R}^{\top} = \mathbf{I}_d$. According to a standard argument in (Absil, Mahony, and Sepulchre 2007), for any

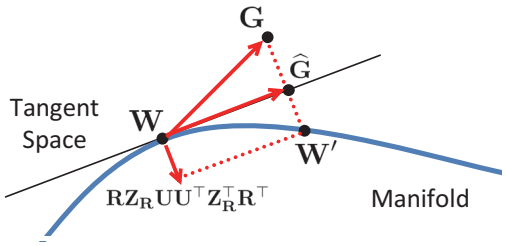

Figure 2: Illustration of Riemannian projection ($\mathbf{G} \mapsto \widehat{\mathbf{G}}$) and retraction ($\widehat{\mathbf{G}} \mapsto \mathbf{W}'$).

symmetric matrix $\mathbf{G}$, the orthogonal projection to the tangent space $T_{\mathbf{H}}\mathcal{M}$ can be accomplished by $P_{T_{\mathbf{W}}\mathcal{M}}(\mathbf{G})$: $\mathbf{G} \mapsto \widehat{\mathbf{G}} \triangleq \mathbf{P}_{\mathbf{U}}\mathbf{G}\mathbf{P}_{\mathbf{U}} + \mathbf{P}_{\mathbf{U}}\mathbf{G}\mathbf{P}_{\mathbf{R}} + \mathbf{P}_{\mathbf{R}}\mathbf{G}\mathbf{P}_{\mathbf{U}}$, where $\mathbf{P}_{\mathbf{U}} = \mathbf{U}\mathbf{U}^{\top}, \mathbf{P}_{\mathbf{R}} = \mathbf{R}\mathbf{R}^{\top}$ are known as *projection matrices*. In fact, $\widehat{\mathbf{G}}$ is only used for algorithmic analysis and need not be explicitly computed.

To ensure the new $\mathbf{W}'$ reside on $\mathcal{M}_{d,k}$, a natural solution is explicitly expressing it as the product of two rank-k matrices, namely $\mathbf{W}' \triangleq \mathbf{K}\mathbf{K}^{\top}$ with $\mathbf{K} \in \mathbb{R}^{d\times k}$. We here adopt a constructive methodology for computing $\mathbf{K}$. Specifically, let us further assume that $\mathbf{K} = (\mathbf{W}^{\frac{1}{2}} + \mathbf{Z})\mathbf{U}$ with $\mathbf{Z} \in \mathbb{R}^{d\times d}$. Recall that $(\mathbf{U}, \mathbf{R})$ are basis matrices complementary to each other, any $\mathbf{Z}$ can thus be equivalently represented as $\mathbf{U}\mathbf{Z}_{\mathbf{U}} + \mathbf{R}\mathbf{Z}_{\mathbf{R}}$, where $\mathbf{Z}_{\mathbf{U}} \in \mathbb{R}^{k\times d}, \mathbf{Z}_{\mathbf{R}} \in \mathbb{R}^{r\times d}$. The real descent direction in practice is

$$\begin{aligned}\mathring{\mathbf{G}} &\triangleq \mathbf{W}' - \mathbf{W} \\ &= (\mathbf{W}^{\frac{1}{2}} + \mathbf{Z})\mathbf{U}\mathbf{U}^{\top}(\mathbf{W}^{\frac{1}{2}} + \mathbf{Z})^{\top} - \mathbf{W} \\ &= \mathbf{Z}\mathbf{U}\mathbf{U}^{\top}\mathbf{Z}^{\top} + \mathbf{W}^{\frac{1}{2}}\mathbf{U}\mathbf{U}^{\top}\mathbf{Z}^{\top} + \mathbf{Z}\mathbf{U}\mathbf{U}^{\top}\mathbf{W}^{\frac{1}{2}}\end{aligned} \tag{15}$$

Now the retraction operation reduces to optimizing $\mathbf{Z}$ (or $\mathbf{Z}_{\mathbf{U}}, \mathbf{Z}_{\mathbf{R}}$) under specific criterion. We choose to optimize $\mathbf{Z}_{\mathbf{U}}, \mathbf{Z}_{\mathbf{R}}$, such that $\mathring{\mathbf{G}}$ approximates $\mathbf{G}$ as accurate as possible in a least-squares sense[1]. In fact, letting $\mathbf{U}^{\top}\mathring{\mathbf{G}}\mathbf{U} = \mathbf{U}^{\top}\mathbf{G}\mathbf{U}$ and $\mathbf{U}^{\top}\mathring{\mathbf{G}}\mathbf{R} = \mathbf{U}^{\top}\mathbf{G}\mathbf{R}$ obtains two equations:

$$\mathbf{Z}_{\mathbf{U}}\mathbf{U}\mathbf{U}^{\top}\mathbf{Z}_{\mathbf{U}}^{\top} + \mathbf{\Lambda}^{\frac{1}{2}}\mathbf{U}^{\top}\mathbf{Z}_{\mathbf{U}}^{\top} + \mathbf{Z}_{\mathbf{U}}\mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}} = \mathbf{U}^{\top}\mathbf{G}\mathbf{U} \tag{16}$$

$$\mathbf{Z}_{\mathbf{U}}\mathbf{U}\mathbf{U}^{\top}\mathbf{Z}_{\mathbf{R}}^{\top} + \mathbf{\Lambda}^{\frac{1}{2}}\mathbf{U}^{\top}\mathbf{Z}_{\mathbf{R}}^{\top} = \mathbf{U}^{\top}\mathbf{G}\mathbf{R}, \tag{17}$$

Solving them obtains

$$\mathbf{Z}_{\mathbf{U}}\mathbf{U} = (\mathbf{U}^{\top}\mathbf{G}\mathbf{U} + \mathbf{\Lambda})^{1/2} - \mathbf{\Lambda}^{1/2} \tag{18}$$

$$\mathbf{Z}_{\mathbf{R}}\mathbf{U} = (\mathbf{R}^{\top}\mathbf{G}\mathbf{U})(\mathbf{U}^{\top}\mathbf{G}\mathbf{U} + \mathbf{\Lambda})^{-1/2}. \tag{19}$$

Plugging them into $\mathbf{K} = (\mathbf{W}^{\frac{1}{2}} + \mathbf{Z})\mathbf{U}$ further obtains

$$\mathbf{K} = (\mathbf{U}\mathbf{\Lambda} + \mathbf{G}\mathbf{U})(\mathbf{U}^{\top}\mathbf{G}\mathbf{U} + \mathbf{\Lambda})^{-1/2}, \tag{20}$$

and based on the assumption $\mathbf{W}' = \mathbf{K}\mathbf{K}^{\top}$ we have

$$\mathbf{W}' = (\mathbf{U}\mathbf{\Lambda} + \mathbf{G}\mathbf{U})([\mathbf{U}^{\top}\mathbf{G}\mathbf{U} + \mathbf{\Lambda}]_{+})^{\dagger}(\mathbf{U}\mathbf{\Lambda} + \mathbf{G}\mathbf{U})^{\top}, \tag{21}$$

which is the new point after retraction. Note that $\mathbf{U}^{\top}\mathbf{G}\mathbf{U} + \mathbf{\Lambda}$ is not necessarily a PSD $k \times k$ matrix, which may lead to a violation of $\mathbf{W}' \in \mathcal{S}_{+}^{d}$. We thus abuse $[\cdot]_{+}$ in Eqn. (21) to denote a function which abandons all negative eigen-values.

[1] In what follows, we ignore the step size η for conciseness.

Complexity and Theoretic Results

We avoid iterative SVD of $\mathbf{W}$ by maintaining the latest k eigen-pairs $(\mathbf{U}, \mathbf{\Lambda})$ (lines #13 and #18 in Algorithm 1). This can be accomplished by an economic QR-decomposition of $\mathbf{U}\mathbf{\Lambda} + \mathbf{G}\mathbf{U}$ (line #10). In Algorithm 1, the most expensive operation in Eqn.(21) stems from the computation of $\mathbf{U}^\top \mathbf{G}\mathbf{U}$. Other operations have time complexity of $\mathcal{O}(k^3)$ (eigen-decomposition in line #12) or $\mathcal{O}(dk^2)$ (matrix multiplication such as in lines #9 and #11). To sum up, in comparison with conventional SVD-based projected gradient method with $\mathcal{O}(d^2k)$, our algorithm only computes $\mathbf{U}^\top \mathbf{G}\mathbf{U}$ once for all possible step sizes and the other parts enjoy largely reduced complexity in $\mathcal{O}(nk^2 + k^3)$.

It is interesting to analyze the difference between $\mathring{\mathbf{G}}$ and $\widehat{\mathbf{G}}$. In fact, we have

Theorem 1 *$\mathring{\mathbf{G}}$ has the following properties: 1) the residual $\mathring{\mathbf{G}} - \widehat{\mathbf{G}}$ is in the normal direction of the tangent plane $T_{\mathbf{W}}\mathcal{M}$; 2) $tr(\widehat{\mathbf{G}}^\top \mathring{\mathbf{G}}) \geq 0$.*

As argued in gradient descent method (Nocedal and Wright 2006), $\mathring{\mathbf{G}}$ is a descent direction only if $tr(\mathbf{G}^\top \mathring{\mathbf{G}}) \geq 0$, Theorem 1 ensures that $\mathring{\mathbf{G}}$ is always positively correlated with $\widehat{\mathbf{G}}$, which is the projection of $\mathbf{G}$ on the tangent space. In practice, we track the sign of $tr(\mathbf{G}^\top \mathring{\mathbf{G}})$ for quickly filtering out non-descending directions, as seen in Algorithm 1.

Importantly, as our main theoretic observation, the proposed computation in Eqn. (21) rigorously defines a *second-order Riemannian retraction* operator:

Theorem 2 *Let $R_{\mathbf{W}} : \mathcal{T}_{\mathbf{W}}\mathcal{M} \mapsto \mathcal{M}$ denote the retraction function which describes the calculation of $\widehat{\mathbf{G}} \mapsto \mathbf{W}'$ in Figure 2. $R_{\mathbf{W}}$ defines a second-order retraction on Riemannian manifold. Specifically, it satisfies: 1) $R_{\mathbf{W}}(0) = \mathbf{W}$; 2) Local rigidity, namely the curve defined by $\gamma_{\widehat{\mathbf{G}}}(\tau) = R_{\mathbf{W}}(\tau\widehat{\mathbf{G}})$ satisfies $\dot{\gamma}_{\widehat{\mathbf{G}}}(0) = \widehat{\mathbf{G}}$; and 3) $P_{T_{\mathbf{W}}\mathcal{M}}\left(\frac{dR_{\mathbf{W}}(\tau\widehat{\mathbf{G}})}{d\tau^2}|_{\tau=0}\right) = \mathbf{0}$, namely the second order derivatives are all normal to the manifold.*

All proofs of above theorems are deferred to the supplemental material due to space limit.

Experiments

In this experimental section we denote the proposed method as *fixed-rank metric learning* (FRML).

Dataset Description: We adopt seven machine learning benchmarks. Table 1 summarizes the important information of the experimental data. We download three datasets, *DNA, Splice, Vowel*, from the data repository of LibSVM.[2] *KDDCup04*[3] represents the Quantum Physics dataset used for KDD data mining competition. *CIFAR10*[4] is comprised of images from ten semantic categories, such as "airplane" and "horse". *COIL20*[5] is another multi-view image object recognition benchmark established by Columbia University. *HAR* stands for *Human Activity Recognition*, which is part of UCI collection[6] and contains sensor recordings from smart phone accelerometer and gyroscope. For most benchmarks, we directly use the features provided by the repository organizers. We extract visual features on CIFAR10 using the output of some intermediate layer in deep networks (Jia et al. 2014), and concatenate raw pixels as features for COIL20.

Dataset	#Sample	#Feature	#Class	k
DNA	3,186	180	3	20
Splice	3,175	60	2	20
Vowel	990	10	11	5
KddCup04	50,000	65	2	20
CIFAR10	60,000	2,048	10	100
COIL20	1,440	1,024	20	100
HAR	10,299	561	6	100

Table 1: Summary of the benchmarks used in the experiments.

Baseline Algorithms: We carry out quantitative comparisons with seven baseline algorithms, including 1) *Euclidean* **(EU)**: the standard Euclidean distance; 2) *Inverse Covariance* **(InvCov)**: a Mahalanobis metric using the inverse covariance matrix as its metric matrix; 3) *Principal Component Analysis* **(PCA)**: a classic statistical tool designed for data that are corrupted by Gaussian noises; 4) *Supervised Locality Preserving Projections* **(SLPP)** (He 2005): it solves a variational problem that optimally preserves the neighborhood structure of the data set. We use its supervised variant. 3) *Large Margin Nearest Neighbor* **(LMNN)** (Weinberger and Saul 2009): the intuition underlying this work is to simultaneously attract same-label instances into the neighborhood and push away those with different labels; 4) *Information-Theoretic Metric Learning* **(ITML)** (Davis et al. 2007): a work which popularizes LogDet regularization in metric learning. It is fed by pairwise constraints and updates the parameters from a single constraint at each iteration; 5) *Sparse Distance Metric Learning* **(SDML)** (Qi et al. 2009): it also adopt LogDet divergence for regularizing the Mahalanobis metric. Meanwhile, an L_1 sparsity term is included to zero most off-diagonal elements in the metric matrix, improving the robustness. For SDML, we implement two efficient solvers using either graphicalLasso (nearly identical to the original solver in (Qi et al. 2009)) or ADMM. After comparing the efficacy of these two versions, we adopt ADMM-based solver in the evaluations. For other baseline algorithms, we use standard built-in routines in Matlab or source codes obtained from the authors.

Accuracy and Speed: In all experiments, we randomly sample 50% data as the training set, and the rest for testing purpose. 1,000 random samples in the test set (or the entire test set for Vowel and COIL20) are treated as queries. A leave-one-out scheme is utilized for evaluation. Specifically, we retrieve the 50 nearest neighbors for each query from the test set, and calculate the percentage of same-label samples among them. The precision is averaged over all queries, ob-

[2] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[3] http://osmot.cs.cornell.edu/kddcup/datasets.html

[4] http://www.cs.toronto.edu/~kriz/cifar.html

[5] http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

[6] https://archive.ics.uci.edu/ml/datasets

	DNA	Splice	Vowel	KDDCup04	CIFAR10	COIL20	HAR
EU	57.27 ± 0.66	61.67 ± 0.45	26.63 ± 0.30	60.84 ± 0.62	71.16 ± 1.38	45.72 ± 0.29	82.37 ± 0.62
InvCov	36.07 ± 0.24	56.90 ± 0.44	22.41 ± 0.45	60.38 ± 0.56	22.23 ± 1.25	7.22 ± 0.14	28.99 ± 0.68
PCA	70.37 ± 0.50	66.79 ± 0.64	26.73 ± 0.43	60.46 ± 0.46	71.87 ± 0.99	46.53 ± 0.40	82.47 ± 0.39
SLPP	74.24 ± 0.71	70.33 ± 0.84	28.42 ± 0.48	59.64 ± 0.62	71.19 ± 1.19	50.48 ± 1.13	91.50 ± 0.75
LMNN	86.31 ± 0.65	74.66 ± 0.70	32.44 ± 0.64	61.47 ± 0.54	75.04 ± 0.97	63.95 ± 0.37	$\mathbf{97.05 \pm 0.23}$
ITML	88.13 ± 1.54	76.10 ± 1.04	36.09 ± 1.68	62.73 ± 0.81	$--/--$	7.28 ± 0.14	94.94 ± 0.49
SDML	86.76 ± 0.65	72.78 ± 1.15	28.28 ± 0.63	62.74 ± 0.69	72.66 ± 1.23	46.90 ± 0.60	88.01 ± 0.57
FRML	$\mathbf{91.88 \pm 0.58}$	$\mathbf{77.06 \pm 0.58}$	$\mathbf{39.01 \pm 1.07}$	$\mathbf{64.17 \pm 0.82}$	$\mathbf{79.19 \pm 1.01}$	$\mathbf{67.66 \pm 0.61}$	94.26 ± 1.02

Table 2: 50-NN accuracies on the test set. All evaluations are conducted on four shared machines in a private large-scale cluster. Each is equipped with 64 CPU cores and 760GB physical memory. All implementations are based on carefully optimized Matlab code. On each benchmark the best accuracy is highlighted in bold. $--/--$ implies the program does not converge within one day.

	Training Time (in seconds)			
	KDDCup04	**CIFAR10**	**COIL20**	**HAR**
PCA	0.07	13.78	4.50	1.16
SLPP	5.73	7.01	0.35	1.07
LMNN	272.79	2880.48	249.31	543.53
ITML	399.14	$--$	89662.11	10054.13
SDML	72.76	63952.49	14595.21	4338.01
FRML	18.47	403.82	426.67	105.66

Table 3: Training time on four most time-consuming benchmarks. Note that ITML does not converge on CIFAR10 in one day.

taining average precision scores. 10 independent trials are conducted to reduce the effect of randomness.

Regarding the parameter tuning, we empirically set the target dimensions of SLPP, LMNN and our proposed FRML (namely the fixed rank k), which are found in the last column of Table 1. SDML is unable to specify the rank constraint. The optimization threads of FRML and SDML terminate when the relative improvement of objective values between two consecutive iterations is below 10^{-5}, or they reach the maximum iteration count of 200 / 100 respectively. ITML terminates after at most 100 passes over all constraints or reaches its default solution precision. For LogDet based formulations, one can choose either identity matrix or covariance matrix as the prior. As suggested by (Qi et al. 2009), we apply covariance matrix for SDML. We also empirically compare both options for ITML, and report the better accuracies achieved with identity matrix. In Problem (9), λ is set to be 10^{-4} in all experiments. The L_1 sparsity parameter is set as 10^{-3} for SDML. Moreover, both SDML and ITML rely on k-NN affinity graphs, whose construction takes unaffordable time on KDDCup04 and CIFAR10. We thus draw 10,000 random samples for approximate graph construction.

The recorded 50-NN accuracies and training time are seen in Tables 2 and 3 respectively. Our proposed FRML dominates by significant margins on 6 out of 7 benchmarks in terms of accuracy and in par with other algorithms on HAR. Regarding the training time, we find that ITML and SDML are significantly slower than others, particulary on high-dimensional data (such as CIFAR10). We arguably attribute their low efficacy to the slow convergence rate of

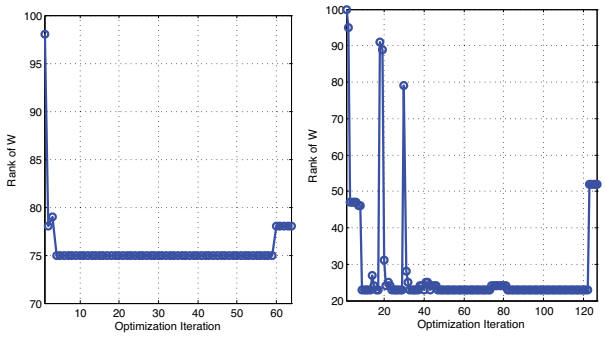

Figure 3: Illustration of $rank(\mathbf{W})$ at different optimization iterations for FRML. The figures correspond to CIFAR10 (left) and HAR (right) respectively. On both benchmarks FRML converges in fewer than 200 iterations.

ITML (it computes a stochastic gradient from a single constraint each time, which is noisy) and iterative full eigen-decomposition of SDML respectively. In comparison, our proposed FRML only requires a one-time $\mathcal{O}(n^2k)$ SVD for all optimization and $\mathcal{O}(n^2k)$ matrix product at each linear-search iteration. The core computation enjoys $\mathcal{O}(nk^2 + k^3)$ complexity, which explains the fast training of FRML in Table 3.

Parameter Sensitivity: One may argue that FRML is sensitive to the choice of rank k. In fact, we find in practice that FRML is able to adaptively pursue a low-rank solution, even a high value of k is specified. Figure 3 shows the evolutions of $rank(\mathbf{W})$ at each iteration, on CIFAR10 and HAR respectively. Though k is initially set to be 100 on both benchmarks, the algorithm intelligently adjust the rank of current solution. The ranks of final solutions stop at 78 and 52 for CIFAR10/HAR respectively. The implications are two-folds: 1) the intrinsic discriminative dimension for supervised metric learning is often very low compared with the feature dimensions, 2) though the optimal k in FRML is unable to be determined in theory, we can empirically assign a relatively high value and are still able to expect a low-rank solution eventually.

Conclusions

We present a novel fixed-rank formulation for supervised metric learning and propose a Riemannian manifold based optimization method. The proposed FRML clearly distinguishes itself by elegant theoretic analysis and reduced computational complexity. Our evaluations exhibit significant improvements in terms of both accuracy and training time. It is also potentially applicable for other rank-constrained machine learning problems.

References

Absil, P.-A.; Mahony, R.; and Sepulchre, R. 2007. *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ, USA: Princeton University Press.

Bellet, A.; Habrard, A.; and Sebban, M. 2013. A survey on metric learning for feature vectors and structured data. *CoRR* abs/1306.6709.

Bi, J.; Wu, D.; Lu, L.; Liu, M.; Tao, Y.; and Wolf, M. 2011. Adaboost on low-rank PSD matrices for metric learning. In *CVPR*, 2617–2624.

Candès, E. J.; Li, X.; Ma, Y.; and Wright, J. 2011. Robust principal component analysis? *J. ACM* 58(3):11:1–11:37.

Chen, X.; Tong, Z.; Liu, H.; and Cai, D. 2012. Metric learning with two-dimensional smoothness for visual analysis. In *CVPR*.

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

He, X. 2005. *Locality Preserving Projections*. Ph.D. Dissertation, Chicago, IL, USA. AAI3195015.

Hoi, S.; Liu, W.; and Chang, S. 2008. Semi-supervised distance metric learning for collaborative image retrieval. In *CVPR*.

Hoi, S. C.; Liu, W.; and Chang, S.-F. 2010. Semi-supervised distance metric learning for collaborative image retrieval and clustering. *ACM Trans. Multimedia Comput. Commun. Appl.* 6(3):18:1–18:26.

Hu, J.; Lu, J.; and Tan, Y.-P. 2014. Discriminative deep metric learning for face verification in the wild.

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; and Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*.

Jin, R.; Wang, S.; and Zhou, Y. 2009. Regularized distance metric learning: Theory and algorithm. In *NIPS*.

Kato, T., and Nagano, N. 2010. Metric learning for enzyme active-site search. *Bioinformatics* 26(21):2698–2704.

Kedem, D.; Tyree, S.; Weinberger, K. Q.; Sha, F.; and Lanckriet, G. R. G. 2012. Non-linear metric learning. In *NIPS*.

Kulis, B.; Sustik, M. A.; and Dhillon, I. S. 2009. Low-rank kernel learning with bregman matrix divergences. *Journal of Machine Learning Research* 10:341–376.

Kulis, B. 2013. Metric learning: A survey. *Foundations and Trends in Machine Learning* 5(4):287–364.

Lim, D.; Lanckriet, G. R. G.; and McFee, B. 2013. Robust structural metric learning. In *ICML*.

Liu, M., and Vemuri, B. C. 2012. A robust and efficient doubly regularized metric learning approach. In *ECCV*, 646–659.

Liu, W.; Ma, S.; Tao, D.; Liu, J.; and Liu, P. 2010. Semi-supervised sparse metric learning using alternating linearization optimization. In *SIGKDD*.

Liu, W.; Mu, C.; Ji, R.; Ma, S.; Smith, J.; and Chang, S. 2015. Low-rank similarity metric learning in high dimensions. In *AAAI*.

Liu, G.; Lin, Z.; and Yu, Y. 2010. Robust subspace segmentation by low-rank representation. In *ICML*.

Luo, Y.; Liu, T.; Tao, D.; and Xu, C. 2014. Decomposition-based transfer distance metric learning for image classification. *Image Processing, IEEE Transactions on* 23(9):3789–3801.

Meyer, G.; Bonnabel, S.; and Sepulchre, R. 2011. Linear regression under fixed-rank constraints: A riemannian approach. In *ICML*.

Nocedal, J., and Wright, S. J. 2006. *Numerical Optimization, second edition*. World Scientific.

Qi, G.; Tang, J.; Zha, Z.; Chua, T.; and Zhang, H. 2009. An efficient sparse metric learning in high-dimensional space via l_1-penalized log-determinant regularization. In *ICML*.

Ravikumar, P.; Wainwright, M. J.; Raskutti, G.; and Yu, B. 2011. High-dimensional covariance estimation by minimizing l1-penalized log-determinant divergence. *Electron. J. Statist.* 5:935–980.

Shalit, U.; Weinshall, D.; and Chechik, G. 2010. Online learning in the manifold of low-rank matrices. In *NIPS*.

Shen, C.; Kim, J.; Wang, L.; and van den Hengel, A. 2009. Positive semidefinite metric learning with boosting. In *NIPS*.

Verma, Y., and Jawahar, C. 2012. Image annotation using metric learning in semantic neighbourhoods. In *ECCV*.

Wang, X., and Tang, X. 2004. A unified framework for subspace face recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 26(9):1222–1228.

Weinberger, K. Q., and Saul, L. K. 2006. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *AAAI*.

Weinberger, K., and Saul, L. 2009. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research* 10:207–244.

Xing, E. P.; Ng, A.; Jordan, M.; and Russell, S. 2003. Distance metric learning, with application to clustering with side-information. In *NIPS*.

Yang, L. 2006. Distance metric learning: A comprehensive survey.

Ying, Y., and Li, P. 2012. Distance metric learning with eigenvalue optimization. *J. Mach. Learn. Res.* 13(1):1–26.

Zhang, T., and Oles, F. 2001. Text categorization based on regularized linear classification methods. *Infomation Retrieval* 4(1):5–31.

Zhang, J.; Jin, R.; Yang, Y.; and Hauptmann, A. 2003. Modified logistic regression: An approximation to SVM and its applications in large-scale text categorization. In *ICML*.

Zhong, G.; Huang, K.; and Liu, C.-L. 2011. Low rank metric learning with manifold regularization. In *ICDM*.