Journal of Machine Learning Research 13 (2012) 519-547 Submitted 2/11; Revised 7/11; Published 3/12

Metric and Kernel Learning Using a Linear Transformation

Prateek Jain PRAJAIN@MICROSOFT.EDU
Microsoft Research India
#9 Lavelle Road
Bangalore 560 003, India

Brian Kulis KULIS@CSE.OHIO-STATE.EDU
599 Dreese Labs
Ohio State University
Columbus, OH 43210, USA

Jason V. Davis JVDAVIS@GMAIL.COM
Etsy Inc.
55 Washington Street, Ste. 512
Brooklyn, NY 11201

Inderjit S. Dhillon INDERJIT@CS.UTEXAS.EDU
The University of Texas at Austin
1 University Station C0500
Austin, TX 78712, USA

Editors: Sören Sonnenburg, Francis Bach and Cheng Soon Ong

Abstract

Metric and kernel learning arise in several machine learning applications. However, most existing metric learning algorithms are limited to learning metrics over low-dimensional data, while existing kernel learning algorithms are often limited to the transductive setting and do not generalize to new data points. In this paper, we study the connections between metric learning and kernel learning that arise when studying metric learning as a linear transformation learning problem. In particular, we propose a general optimization framework for learning metrics via linear transformations, and analyze in detail a special case of our framework—that of minimizing the LogDet divergence subject to linear constraints. We then propose a general regularized framework for learning a kernel matrix, and show it to be *equivalent* to our metric learning framework. Our theoretical connections between metric and kernel learning have two main consequences: 1) the learned kernel matrix parameterizes a linear transformation kernel *function* and can be applied inductively to new data points, 2) our result yields a constructive method for kernelizing most existing Mahalanobis metric learning formulations. We demonstrate our learning approach by applying it to large-scale real world problems in computer vision, text mining and semi-supervised kernel dimensionality reduction.

Keywords: metric learning, kernel learning, linear transformation, matrix divergences, logdet divergence

1. Introduction

One of the basic requirements of many machine learning algorithms (e.g., semi-supervised clustering algorithms, nearest neighbor classification algorithms) is the ability to compare two objects to compute a similarity or distance between them. In many cases, off-the-shelf distance or similarity

functions such as the Euclidean distance or cosine similarity are used; for example, in text retrieval applications, the cosine similarity is a standard function to compare two text documents. However, such standard distance or similarity functions are not appropriate for all problems.

Recently, there has been significant effort focused on task-specific learning for comparing data objects. One prominent approach has been to learn a distance metric between objects given additional side information such as pairwise similarity and dissimilarity constraints over the data. A class of distance metrics that has shown excellent generalization properties is the learned *Mahalanobis distance* function (Davis et al., 2007; Xing et al., 2002; Weinberger et al., 2005; Goldberger et al., 2004; Shalev-Shwartz et al., 2004). The Mahalanobis distance can be viewed as a method in which data is subject to a *linear transformation*, and the goal of such metric learning methods is to learn the linear transformation for a given task. Despite their simplicity and generalization ability, Mahalanobis distances suffer from two major drawbacks: 1) the number of parameters to learn grows quadratically with the dimensionality of the data, making it difficult to learn distance functions over high-dimensional data, 2) learning a linear transformation is inadequate for data sets with non-linear decision boundaries.

To address the latter shortcoming, *kernel learning* algorithms typically attempt to learn a kernel matrix over the data. Limitations of linear methods can be overcome by employing a non-linear input kernel, which implicitly maps the data non-linearly to a high-dimensional feature space. However, many existing kernel learning methods are still limited in that the learned kernels do not generalize to new points (Kwok and Tsang, 2003; Kulis et al., 2006; Tsuda et al., 2005). These methods are therefore restricted to learning in the transductive setting where all the data (labeled and unlabeled) is assumed to be given upfront. There has been some work on learning kernels that generalize to new points, most notably work on hyperkernels (Ong et al., 2005), but the resulting optimization problems are expensive and cannot be scaled to large or even medium-sized data sets. Another approach is multiple kernel learning (Lanckriet et al., 2004), which learns a mixture of base kernels; this approach is inductive but the class of learnable kernels can be restrictive.

In this paper, we explore metric learning with linear transformations over arbitrarily high-dimensional spaces; as we will see, this is equivalent to learning a *linear transformation kernel function* $\phi(\boldsymbol{x})^T W \phi(\boldsymbol{y})$ given an input kernel function $\phi(\boldsymbol{x})^T \phi(\boldsymbol{y})$. In the first part of the paper, we formulate a metric learning problem that uses a particular loss function called the LogDet divergence, for learning the positive definite matrix W. This loss function is advantageous for several reasons: it is defined only over positive definite matrices, which makes the optimization simpler, as we will be able to effectively ignore the positive definiteness constraint on W. Furthermore, the loss function has precedence in optimization (Fletcher, 1991) and statistics (James and Stein, 1961). An important advantage of our method is that the proposed optimization algorithm is scalable to very large data sets of the order of millions of data objects. But perhaps most importantly, the loss function permits efficient kernelization, allowing efficient learning of a linear transformation in kernel space. As a result, unlike transductive kernel learning methods, our method easily handles out-of-sample extensions, that is, it can be applied to unseen data.

We build upon our results of kernelization for the LogDet formulation to develop a general framework for learning linear transformation kernel functions and show that such kernels can be efficiently learned over a wide class of convex constraints and loss functions. Our result can be viewed as a representer theorem, where the optimal parameters can be expressed purely in terms of the training data. In our case, even though the matrix W may be infinite-dimensional, it can be

fully represented in terms of the constrained data points, making it possible to compute the learned kernel function over arbitrary points.

We demonstrate the benefits of a generalized framework for inductive kernel learning by applying our techniques to the problem of inductive kernelized semi-supervised dimensionality reduction. By choosing the trace-norm as a loss function, we obtain a novel kernel learning method that learns *low-rank* linear transformations; unlike previous kernel dimensionality methods, which are either unsupervised or cannot easily be applied inductively to new data, our method intrinsically possesses both desirable properties.

Finally, we apply our metric and kernel learning algorithms to a number of challenging learning problems, including ones from the domains of computer vision and text mining. Unlike existing techniques, we can learn linear transformation-based distance or kernel functions over these domains, and we show that the resulting functions lead to improvements over state-of-the-art techniques for a variety of problems.

2. Related Work

Most of the existing work in metric learning has been done in the Mahalanobis distance (or metric) learning paradigm, which has been found to be a sufficiently powerful class of metrics for a variety of data. In one of the earliest papers on metric learning, Xing et al. (2002) propose a semidefinite programming formulation under similarity and dissimilarity constraints for learning a Mahalanobis distance, but the resulting formulation is slow to optimize and has been outperformed by more recent methods. Weinberger et al. (2005) formulate the metric learning problem in a large margin setting, with a focus on k-NN classification. They also formulate the problem as a semidefinite programming problem and consequently solve it using a method that combines sub-gradient descent and alternating projections. Goldberger et al. (2004) proceed to learn a linear transformation in the fully supervised setting. Their formulation seeks to 'collapse classes' by constraining within-class distances to be zero while maximizing between-class distances. While each of these algorithms was shown to yield improved classification performance over the baseline metrics, their constraints do not generalize outside of their particular problem domains; in contrast, our approach allows arbitrary linear constraints on the Mahalanobis matrix. Furthermore, these algorithms all require eigenvalue decompositions or semi-definite programming, which is at least cubic in the dimensionality of the data.

Other notable works for learning Mahalanobis metrics include Pseudo-metric Online Learning Algorithm (POLA) (Shalev-Shwartz et al., 2004), Relevant Components Analysis (RCA) (Schultz and Joachims, 2003), Neighborhood Components Analysis (NCA) (Goldberger et al., 2004), and locally-adaptive discriminative methods (Hastie and Tibshirani, 1996). In particular, Shalev-Shwartz et al. (2004) provided the first demonstration of Mahalanobis distance learning in kernel space. Their construction, however, is expensive to compute, requiring cubic time per iteration to update the parameters. As we will see, our LogDet-based algorithm can be implemented more efficiently.

Non-linear transformation based metric learning methods have also been proposed, though these methods usually suffer from suboptimal performance, non-convexity, or computational complexity. Examples include the convolutional neural net based method of Chopra et al. (2005); and a general Riemannian metric learning method (Lebanon, 2006).

Most of the existing work on kernel learning can be classified into two broad categories. The first category includes parametric approaches, where the learned kernel function is restricted to be of a

specific form and then the relevant parameters are learned according to the provided data. Prominent methods include multiple kernel learning (Lanckriet et al., 2004), hyperkernels (Ong et al., 2005), and hyper-parameter cross-validation (Seeger, 2006). Most of these methods either lack modeling flexibility, require non-convex optimization, or are restricted to a supervised learning scenario. The second category includes non-parametric methods, which explicitly model geometric structure in the data. Examples include spectral kernel learning (Zhu et al., 2005), manifold-based kernel learning (Bengio et al., 2004), and kernel target alignment (Cristianini et al., 2001). However, most of these approaches are limited to the transductive setting and cannot be used to naturally generalize to new points. In comparison, our kernel learning method combines both of the above approaches. We propose a general non-parametric kernel *matrix* learning framework, similar to methods of the second category. However, based on our choice of regularization and constraints, we show that our learned kernel matrix corresponds to a linear transformation kernel function parameterized by a PSD matrix. As a result, our method can be applied to inductive settings without sacrificing significant modeling power. In addition, our kernel learning method naturally provides kernelization for many existing metric learning methods. Recently, Chatpatanasiri et al. (2010) showed kernelization for a class of metric learning algorithms including LMNN and NCA; as we will see, our result is more general and we can prove kernelization over a larger class of problems and can also reduce the number of parameters to be learned. Furthermore, our methods can be applied to a variety of domains and with a variety of forms of side-information. Independent of our work, Argyriou et al. (2010) recently proved a representer type of theorem for spectral regularization functions. However, the framework they consider is different than ours in that they are interested in sensing an underlying high-dimensional matrix using given measurements.

The research in this paper combines and extends work done in Davis et al. (2007), Kulis et al. (2006), Davis and Dhillon (2008), and Jain et al. (2010). The focus in Davis et al. (2007) and Davis and Dhillon (2008) was solely on the LogDet divergence, while the main goal in Kulis et al. (2006) was to demonstrate the computational benefits of using the LogDet and von Neumann divergences for learning low-rank kernel matrices. In Jain et al. (2010), we showed the equivalence between a general class of kernel learning problems and metric learning problems. In this paper, we unify and summarize the work in the existing conference papers and also provide detailed proofs of the theorems in Jain et al. (2010).

3. LogDet Divergence Based Metric Learning

We begin by studying a particular method, based on the LogDet divergence, for learning metrics via learning linear transformations given pairwise distance constraints. We discuss kernelization of this formulation and present efficient optimization algorithms. Finally, we address limitations of the method when the amount of training data is large, and propose a modified algorithm to efficiently learn a kernel under such circumstances. In subsequent sections, we will take the ingredients developed in this section and show how to generalize them to adapt to a much larger class of loss functions and constraints, which will encompass most of the previously-studied approaches for Mahalanobis metric learning.

3.1 Mahalanobis Distances and Parameterized Kernels

First we introduce the framework for metric and kernel learning that is employed in this paper. Given a data set of objects $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n], \boldsymbol{x}_i \in \mathbb{R}^{d_0}$ (when working in kernel space, the data

matrix will be represented as $\Phi = [\phi(\boldsymbol{x}_1), ..., \phi(\boldsymbol{x}_n)]$, where ϕ is the mapping to feature space, that is, $\phi : \mathbb{R}^{d_0} \to \mathbb{R}^d$), we are interested in finding an appropriate distance function to compare two objects. We consider the Mahalanobis distance, parameterized by a positive definite matrix W; the squared distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is given by

$$d_W(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T W (\boldsymbol{x}_i - \boldsymbol{x}_j). \tag{1}$$

This distance function can be viewed as learning a linear transformation of the data and measuring the squared Euclidean distance in the transformed space. This is seen by factorizing the matrix $W = G^T G$ and observing that $d_W(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|G\boldsymbol{x}_i - G\boldsymbol{x}_j\|_2^2$. However, if the data is not linearly separable in the input space, then the resulting distance function may not be powerful enough for the desired application. As a result, we are interested in working in kernel space; that is, we will express the Mahalanobis distance in kernel space using an appropriate mapping ϕ from input to feature space:

$$d_W(\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)) = (\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j))^T W (\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)).$$

Note that when we choose ϕ to be the identity, we obtain (1); we will use the more general form throughout this paper. As is standard with kernel-based algorithms, we require that this distance be computable given the ability to compute the kernel function $\kappa_0(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{y})$. We can therefore equivalently pose the problem as learning a parameterized kernel function $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T W \phi(\boldsymbol{y})$ given some input kernel function $\kappa_0(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{y})$.

To learn the resulting metric/kernel, we assume that we are given constraints on the desired distance function. In this paper, we assume that pairwise similarity and dissimilarity constraints are given over the data—that is, pairs of points that should be similar under the learned metric/kernel, and pairs of points that should be dissimilar under the learned metric/kernel. Such constraints are natural in many settings; for example, given class labels over the data, points in the same class should be similar to one another and dissimilar to points in different classes. However, our approach is general and can accommodate other potential constraints over the distance function, such as relative distance constraints.

The main challenge is in finding an appropriate loss function for learning the matrix W so that 1) the resulting algorithm is scalable and efficiently computable in kernel space, 2) the resulting metric/kernel yields improved performance on the underlying learning problem, such as classification, semi-supervised clustering etc. We now move on to the details.

3.2 LogDet Metric Learning

The LogDet divergence between two positive definite matrices[1] $W, W_0 \in \mathbb{R}^{d \times d}$ is defined to be

$$D_{\ell \mathrm{d}}(W, W_0) = \mathrm{tr}(WW_0^{-1}) - \log\det(WW_0^{-1}) - d.$$

We are interested in finding W that is closest to W_0 as measured by the LogDet divergence but that satisfies our desired constraints. When $W_0 = I$, we can interpret the learning problem as a maximum entropy problem. Given a set of similarity constraints $\mathcal{S}$ and dissimilarity constraints $\mathcal{D}$, we propose

1. The definition of LogDet divergence can be extended to the case when W_0 and W are rank deficient by appropriate use of the pseudo-inverse. The interested reader may refer to Kulis et al. (2008).

the following problem:

$$\min_{W \succeq 0} D_{\ell d}(W, I), \qquad \text{s.t. } d_W(\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)) \leq u, \ (i,j) \in \mathcal{S}, \\ d_W(\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)) \geq \ell, \ (i,j) \in \mathcal{D}. \tag{2}$$

We make a few remarks about this formulation. The above problem was proposed and studied in Davis et al. (2007). LogDet has many important properties that make it useful for machine learning and optimization, including scale-invariance and preservation of the range space; see Kulis et al. (2008) for a detailed discussion on the properties of LogDet. Beyond this, we prefer LogDet over other loss functions (including the squared Frobenius loss as used in Shalev-Shwartz et al., 2004 or a linear objective as in Weinberger et al., 2005) due to the fact that the resulting algorithm turns out to be simple and efficiently kernelizable, as we will see. We note that formulation (2) minimizes the LogDet divergence to the identity matrix I. This can easily be generalized to arbitrary positive definite matrices W_0. Further, (2) considers simple similarity and dissimilarity constraints over the learned Mahalanobis distance, but other linear constraints are possible. Finally, the above formulation assumes that there exists a feasible solution to the proposed optimization problem; extensions to the infeasible case involving slack variables are discussed later (see Section 3.5).

3.3 Kernelizing the LogDet Metric Learning Problem

We now consider the problem of kernelizing the metric learning problem. Subsequently, we will present an efficient algorithm and discuss generalization to new points.

Given a set of n constrained data points, let K_0 denote the input kernel matrix for the data, that is, $K_0(i,j) = \kappa_0(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$. Note that the squared Euclidean distance in kernel space may be written as $K(i,i) + K(j,j) - 2K(i,j)$, where K is the learned kernel matrix; equivalently, we may write the distance as $\text{tr}(K(\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^T)$, where $\boldsymbol{e}_i$ is the i-th canonical basis vector. Consider the following problem to find K:

$$\min_{K \succeq 0} D_{\ell d}(K, K_0), \qquad \text{s.t. } \text{tr}(K(\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^T) \leq u, \qquad (i,j) \in \mathcal{S}, \\ \text{tr}(K(\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^T) \geq \ell, \qquad (i,j) \in \mathcal{D}. \tag{3}$$

This kernel learning problem was first proposed in the transductive setting in Kulis et al. (2008), though no extensions to the inductive case were considered. Note that problem (2) optimizes over a $d \times d$ matrix W, while the kernel learning problem (3) optimizes over an $n \times n$ matrix K. We now present our key theorem connecting (2) and (3).

Theorem 1 *Let $K_0 \succ 0$. Let W^* be the optimal solution to problem* (2) *and let K^* be the optimal solution to problem* (3)*. Then the optimal solutions are related by the following:*

$$K^* = \Phi^T W^* \Phi, \ \ W^* = I + \Phi S \Phi^T, \\ \text{where } S = K_0^{-1}(K^* - K_0)K_0^{-1}, \quad K_0 = \Phi^T \Phi, \quad \Phi = [\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), \ldots, \phi(\boldsymbol{x}_n)].$$

The above theorem shows that the LogDet metric learning problem (2) can be solved implicitly by solving an equivalent kernel learning problem (3). In fact, in Section 4 we show an equivalence between metric and kernel learning for a general class of regularization functions. The above theorem follows as a corollary to our general theorem (see Theorem 4), which will be proven later.

Next, we generalize the above theorem to regularize against arbitrary positive definite matrices W_0.

Corollary 2 *Consider the following problem:*

$$\min_{W \succeq 0} D_{\ell d}(W, W_0),\ s.t.\ d_W(\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)) \leq u,\ (i,j) \in \mathcal{S},$$
$$d_W(\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)) \geq \ell,\ (i,j) \in \mathcal{D}. \tag{4}$$

Let W^* be the optimal solution to problem (4) *and let K^* be the optimal solution to problem* (3)*. Then the optimal solutions are related by the following:*

$$K^* = \Phi^T W^* \Phi, \qquad W^* = W_0 + W_0 \Phi S \Phi^T W_0,$$
$$\text{where } S = K_0^{-1}(K^* - K_0)K_0^{-1}, \quad K_0 = \Phi^T W_0 \Phi, \quad \Phi = [\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), \ldots, \phi(\boldsymbol{x}_n)].$$

Proof Note that $D_{\ell \mathrm{d}}(W, W_0) = D_{\ell \mathrm{d}}(W_0^{-1/2} W W_0^{-1/2}, I)$. Let $\widetilde{W} = W_0^{-1/2} W W_0^{-1/2}$. Problem (4) is now equivalent to:

$$\begin{aligned} \min_{\widetilde{W} \succeq 0} \quad & D_{\ell \mathrm{d}}(\widetilde{W}, I), \qquad \text{s.t.} \quad d_{\widetilde{W}}(\tilde{\phi}(\boldsymbol{x}_i), \tilde{\phi}(\boldsymbol{x}_j)) \leq u \qquad (i,j) \in \mathcal{S}, \\ & \qquad\qquad\qquad\quad d_{\widetilde{W}}(\tilde{\phi}(\boldsymbol{x}_i), \tilde{\phi}(\boldsymbol{x}_j)) \geq \ell \qquad (i,j) \in \mathcal{D}, \end{aligned} \tag{5}$$

where $\widetilde{W} = W_0^{-1/2} W W_0^{-1/2}$, $\widetilde{\Phi} = W_0^{1/2}\Phi$ and $\widetilde{\Phi} = [\tilde{\phi}(\boldsymbol{x}_1), \tilde{\phi}(\boldsymbol{x}_2), \ldots, \tilde{\phi}(\boldsymbol{x}_n)]$. Now using Theorem 1, the optimal solution $\widetilde{W}^*$ of problem (5) is related to the optimal K^* of problem (3) by $K^* = \widetilde{\Phi}^T \widetilde{W}^* \widetilde{\Phi} = \Phi^T W_0^{1/2} W_0^{-1/2} W^* W_0^{-1/2} W_0^{1/2} \Phi = \Phi^T W^* \Phi$. Similarly, $W^* = W_0^{1/2} \widetilde{W}^* W_0^{1/2} = W_0 + W_0 \Phi S \Phi^T W_0$ where $S = K_0^{-1}(K^* - K_0)K_0^{-1}$. ■

Since the kernelized version of LogDet metric learning is also a linearly constrained optimization problem with a LogDet objective, similar algorithms can be used to solve either problem. This equivalence implies that we can *implicitly* solve the metric learning problem by instead solving for the optimal kernel matrix K^*. Note that using LogDet divergence as objective function has two significant benefits over many other popular loss functions: 1) the metric and kernel learning problems (2), (3) are both equivalent and therefore solving the kernel learning formulation directly provides an out of sample extension (see Section 3.4 for details), 2) projection with respect to the LogDet divergence onto a single distance constraint has a closed-form solution, thus making it amenable to an efficient cyclic projection algorithm (refer to Section 3.5).

3.4 Generalizing to New Points

In this section, we see how to generalize to new points using the learned kernel matrix K^*.

Suppose that we have solved the kernel learning problem for K^* (from now on, we will drop the * superscript and assume that K and W are at optimality). The distance between two points $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{x}_j)$ that are in the training set can be computed directly from the learned kernel matrix as $K(i,i) + K(j,j) - 2K(i,j)$. We now consider the problem of computing the learned distance between two points $\phi(\boldsymbol{z}_1)$ and $\phi(\boldsymbol{z}_2)$ that may not be in the training set.

In Theorem 1, we showed that the optimal solution to the metric learning problem can be expressed as $W = I + \Phi S \Phi^T$. To compute the Mahalanobis distance in kernel space, we see that the inner product $\phi(z_1)^T W \phi(z_2)$ can be computed entirely via inner products between points:

$$\begin{aligned} \phi(z_1)^T W \phi(z_2) &= \phi(z_1)^T (I + \Phi S \Phi^T) \phi(z_2) = \phi(z_1)^T \phi(z_2) + \phi(z_1)^T \Phi S \Phi^T \phi(z_2), \\ &= \kappa_0(z_1, z_2) + k_1^T S k_2, \text{where } k_i = [\kappa_0(z_i, x_1), ..., \kappa_0(z_i, x_n)]^T. \end{aligned} \tag{6}$$

Thus, the expression above can be used to evaluate kernelized distances with respect to the learned kernel function between arbitrary data objects.

In summary, the connection between kernel learning and metric learning allows us to generalize our metrics to new points in kernel space. This is performed by first solving the kernel learning problem for K, then using the learned kernel matrix and the input kernel function to compute learned distances using (6).

3.5 Kernel Learning Algorithm

Given the connection between the Mahalanobis metric learning problem for the $d \times d$ matrix W and the kernel learning problem for the $n \times n$ kernel matrix K, we develop an algorithm for efficiently performing metric learning in kernel space. Specifically, we provide an algorithm (see Algorithm 1) for solving the kernelized LogDet metric learning problem (3).

First, to avoid problems with infeasibility, we incorporate *slack variables* into our formulation. These provide a tradeoff between minimizing the divergence between K and K_0 and satisfying the constraints. Note that our earlier results (see Theorem 1) easily generalize to the slack case:

$$\begin{aligned} \min_{K,\xi} \quad & D_{\ell d}(K, K_0) + \gamma \cdot D_{\ell d}(\text{diag}(\xi), \text{diag}(\xi_0)) \\ \text{s.t.} \quad & \text{tr}(K(e_i - e_j)(e_i - e_j)^T) \leq \xi_{ij} \quad (i,j) \in \mathcal{S}, \\ & \text{tr}(K(e_i - e_j)(e_i - e_j)^T) \geq \xi_{ij} \quad (i,j) \in \mathcal{D}. \end{aligned} \tag{7}$$

The parameter γ above controls the tradeoff between satisfying the constraints and minimizing $D_{\ell d}(K, K_0)$, and the entries of ξ_0 are set to be u for corresponding similarity constraints and ℓ for dissimilarity constraints.

To solve problem (7), we employ the technique of *Bregman projections*, as discussed in the transductive setting (Kulis et al., 2008). At each iteration, we choose a constraint (i, j) from $\mathcal{S}$ or $\mathcal{D}$. We then apply a Bregman projection such that K satisfies the constraint after projection; note that the projection is not an orthogonal projection but is rather tailored to the particular function that we are optimizing. Algorithm 1 details the steps for Bregman's method on this optimization problem. Each update is a rank-one update

$$K \leftarrow K + \beta K(e_i - e_j)(e_i - e_j)^T K,$$

where β is a projection parameter that can be computed in closed form (see Algorithm 1).

Algorithm 1 has a number of key properties which make it useful for various kernel learning tasks. First, the Bregman projections can be computed in closed form, assuring that the projection updates are efficient ($O(n^2)$). Note that, if the feature space dimensionality d is less than n then a similar algorithm can be used directly in the feature space (see Davis et al., 2007). Instead of LogDet, if we use the von Neumann divergence, another potential loss function for this problem,

Algorithm 1 Metric/Kernel Learning with the LogDet Divergence

Input: K_0: input $n \times n$ kernel matrix, $\mathcal{S}$: set of similar pairs, $\mathcal{D}$: set of dissimilar pairs, u, ℓ: distance thresholds, γ: slack parameter

Output: K: output kernel matrix

1. $K \leftarrow K_0$, $\lambda_{ij} \leftarrow 0 \; \forall \; ij$
2. $\xi_{ij} \leftarrow u$ for $(i,j) \in \mathcal{S}$; otherwise $\xi_{ij} \leftarrow \ell$
3. **repeat**
   - 3.1. Pick a constraint $(i,j) \in \mathcal{S}$ or $\mathcal{D}$
   - 3.2. $p \leftarrow (\boldsymbol{e}_i - \boldsymbol{e}_j)^T K (\boldsymbol{e}_i - \boldsymbol{e}_j)$
   - 3.3. $\delta \leftarrow 1$ if $(i,j) \in \mathcal{S}$, -1 otherwise
   - 3.4. $\alpha \leftarrow \min\left(\lambda_{ij}, \frac{\delta\gamma}{\gamma+1}\left(\frac{1}{p} - \frac{1}{\xi_{ij}}\right)\right)$
   - 3.5. $\beta \leftarrow \delta\alpha/(1-\delta\alpha p)$
   - 3.6. $\xi_{ij} \leftarrow \gamma\xi_{ij}/(\gamma + \delta\alpha\xi_{ij})$
   - 3.7. $\lambda_{ij} \leftarrow \lambda_{ij} - \alpha$
   - 3.8. $K \leftarrow K + \beta K(\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^T K$
4. **until** convergence

return K

$O(n^2)$ updates are possible, but are much more complicated and require use of the fast multipole method, which cannot be employed easily in practice. Secondly, the projections maintain positive definiteness, which avoids any eigenvector computation or semidefinite programming. This is in stark contrast with the Frobenius loss, which requires additional computation to maintain positive definiteness, leading to $O(n^3)$ updates.

3.6 Metric/Kernel Learning with Large Data Sets

In Sections 3.1 and 3.3, we proposed a LogDet divergence-based Mahalanobis metric learning problem (2) and an equivalent kernel learning problem (3). The number of parameters involved in these problems is $O(\min(n,d)^2)$, where n is the number of training points and d is the dimensionality of the data. The quadratic dependence affects not only the running time for training and testing, but also requires estimating a large number of parameters. For example, a data set with 10,000 dimensions leads to a Mahalanobis matrix with 100 million entries. This represents a fundamental limitation of existing approaches, as many modern data mining problems possess relatively high dimensionality.

In this section, we present a heuristic for learning structured Mahalanobis distance (kernel) functions that scale linearly with the dimensionality (or training set size). Instead of representing the Mahalanobis distance/kernel matrix as a full $d \times d$ (or $n \times n$) matrix with $O(\min(n,d)^2)$ parameters, our methods use compressed representations, admitting matrices parameterized by $O(\min(n,d))$ values. This enables the Mahalanobis distance/kernel function to be learned, stored, and evaluated efficiently in the context of high dimensionality and large training set size. In particular, we propose a method to efficiently learn an identity plus low-rank Mahalanobis distance matrix and its equivalent kernel function.

Now, we formulate this approach, which we call the high-dimensional identity plus low-rank (IPLR) metric learning problem. Consider a low-dimensional subspace in $\mathbb{R}^d$ and let the columns of U form an orthogonal basis of this subspace. We will constrain the learned Mahalanobis distance

matrix to be of the form:

$$W = I^d + W_l = I^d + ULU^T,$$

where I^d is the $d \times d$ identity matrix, W_l denotes the low-rank part of W and $L \in \mathbb{S}_+^{k \times k}$ with $k \ll \min(n,d)$. Analogous to (2), we propose the following problem to learn an identity plus low-rank Mahalanobis distance function:

$$\begin{aligned} \min_{W,L \succeq 0} \quad & D_{\ell \mathrm{d}}(W, I^d) \qquad \text{s.t.} \quad d_W(\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)) \leq u \quad (i,j) \in \mathcal{S}, \\ & \qquad\qquad\qquad\quad d_W(\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)) \geq \ell \quad (i,j) \in \mathcal{D}, \qquad W = I^d + ULU^T. \end{aligned} \tag{8}$$

Note that the above problem is identical to (2) except for the added constraint $W = I^d + ULU^T$. Let $F = I^k + L$. Now we have

$$\begin{aligned} D_{\ell \mathrm{d}}(W, I^d) &= \mathrm{tr}(I^d + ULU^T) - \log\det(I^d + ULU^T) - d, \\ &= \mathrm{tr}(I^k + L) + d - k - \log\det(I^k + L) - d = D_{\ell \mathrm{d}}(F, I^k), \end{aligned} \tag{9}$$

where the second equality follows from the fact that $\mathrm{tr}(AB) = \mathrm{tr}(BA)$ and Sylvester's determinant lemma. Also note that for all $C \in \mathbb{R}^{n \times n}$,

$$\begin{aligned} \mathrm{tr}(W\Phi C\Phi^T) &= \mathrm{tr}((I^d + ULU^T)\Phi C\Phi^T) = \mathrm{tr}(\Phi C\Phi^T) + \mathrm{tr}(LU^T\Phi C\Phi^T U), \\ &= \mathrm{tr}(\Phi C\Phi^T) - \mathrm{tr}(\Phi' C{\Phi'}^T) + \mathrm{tr}(F\Phi' C{\Phi'}^T), \end{aligned}$$

where $\Phi' = U^T\Phi$ is the reduced-dimensional representation of Φ. Therefore,

$$\begin{aligned} d_W(\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)) &= \mathrm{tr}(W\Phi(\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^T\Phi^T) \\ &= d_I(\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)) - d_I(\phi'(\boldsymbol{x}_i), \phi'(\boldsymbol{x}_j)) + d_F(\phi'(\boldsymbol{x}_i), \phi'(\boldsymbol{x}_j)). \end{aligned} \tag{10}$$

Using (9) and (10), problem (8) is equivalent to the following:

$$\begin{aligned} \min_{F \succeq 0} \quad & D_{\ell \mathrm{d}}(F, I^k), \\ \text{s.t.} \quad & d_F(\phi'(\boldsymbol{x}_i), \phi'(\boldsymbol{x}_j)) \leq u - d_I(\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)) + d_I(\phi'(\boldsymbol{x}_i), \phi'(\boldsymbol{x}_j)), \quad (i,j) \in \mathcal{S}, \\ & d_F(\phi'(\boldsymbol{x}_i), \phi'(\boldsymbol{x}_j)) \geq \ell - d_I(\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)) + d_I(\phi'(\boldsymbol{x}_i), \phi'(\boldsymbol{x}_j)), \quad (i,j) \in \mathcal{D}. \end{aligned} \tag{11}$$

Note that the above formulation is an instance of problem (2) and can be solved using an algorithm similar to Algorithm 1. Furthermore, the above problem solves for a $k \times k$ matrix rather than a $d \times d$ matrix seemingly required by (8). The optimal W^* is obtained as $W^* = I^d + U(F^* - I^k)U^T$.

Next, we show that problem (11) and equivalently (8) can be solved efficiently in feature space by selecting an appropriate basis R ($U = R(R^TR)^{-1/2}$). Let $R = \Phi J$, where $J \in \mathbb{R}^{n \times k}$. Note that $U = \Phi J(J^TK_0J)^{-1/2}$ and $\Phi' = U^T\Phi = (J^TK_0J)^{-1/2}J^TK_0$, that is, $\Phi' \in \mathbb{R}^{k \times n}$ can be computed efficiently in the feature space (requiring inversion of only a $k \times k$ matrix). Hence, problem (11) can be solved efficiently in feature space using Algorithm 1, and the optimal kernel K^* is given by

$$K^* = \Phi^TW^*\Phi = K_0 + K_0J(J^TK_0J)^{-1/2}(F^* - I^k)(J^TK_0J)^{-1/2}J^TK_0.$$

Note that (11) can be solved via Algorithm 1 using $O(k^2)$ computational steps per iteration. Additionally, $O(\min(n,d)k)$ steps are required to prepare the data. Also, the optimal solution W^* (or

K^*) can be stored implicitly using $O(\min(n,d)k)$ memory and similarly, the Mahalanobis distance between any two points can be computed in $O(\min(n,d)k)$ time.

The metric learning problem presented here depends critically on the basis selected. For the case when d is not significantly larger than n and feature space vectors Φ are available explicitly, the basis R can be selected by using one of the following heuristics (see Section 5, Davis and Dhillon, 2008 for more details):

- Using the top k singular vectors of Φ.
- Clustering the columns of Φ and using the mean vectors as the basis R.
- For the fully-supervised case, if the number of classes (c) is greater than the required dimensionality (k) then cluster the class-mean vectors into k clusters and use the obtained cluster centers for forming the basis R. If $c < k$ then cluster each class into k/c clusters and use the cluster centers to form R.

For learning the kernel function, the basis $R = \Phi J$ can be selected by: 1) using a randomly sampled coefficient matrix J, 2) clustering Φ using kernel k-means or a spectral clustering method, 3) choosing a random subset of Φ, that is, the columns of J are random indicator vectors. A more careful selection of the basis R should further improve accuracy of our method and is left as a topic for future research.

4. Kernel Learning with Other Convex Loss Functions

One of the key benefits of our kernel learning formulation using the LogDet divergence (3) is in the ability to efficiently learn a linear transformation (LT) kernel *function* (a kernel of the form $\phi(\boldsymbol{x})^T W \phi(\boldsymbol{y})$ for some matrix $W \succeq 0$) which allows the learned kernel function to be computed over new data points. A natural question is whether one can learn similar kernel functions with other loss functions, such as those considered previously in the literature for Mahalanobis metric learning.

In this section, we propose and analyze a general kernel matrix learning problem similar to (3) but using a more general class of loss functions. As in the LogDet loss function case, we show that our kernel matrix learning problem is equivalent to learning a linear transformation (LT) kernel *function* with a specific loss function. This implies that the learned LT kernel function can be naturally applied to new data. Additionally, since a large class of metric learning methods can be seen as learning a LT kernel function, our result provides a constructive method for kernelizing these methods. Our analysis recovers some recent kernelization results for metric learning, but also implies several new results.

4.1 A General Kernel Learning Framework

Recall that $\kappa_0 : \mathbb{R}^{d_0} \times \mathbb{R}^{d_0} \to \mathbb{R}$ is the input kernel function. We assume that the data vectors in X have been mapped via ϕ, resulting in $\Phi = [\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), \ldots, \phi(\boldsymbol{x}_n)]$. As before, denote the input kernel matrix as $K_0 = \Phi^T \Phi$. The goal is to learn a kernel function κ that is regularized against κ_0 but incorporates the provided side-information. As in the LogDet formulation, we will first consider a transductive scenario, where we learn a kernel matrix K that is regularized against K_0 while satisfying the available side-information.

Recall that the LogDet divergence based loss function in the kernel matrix learning problem (3) is given by:

$$\begin{aligned} D_{\ell d}(K, K_0) &= \mathrm{tr}(KK_0^{-1}) - \log\det(KK_0^{-1}) - n, \\ &= \mathrm{tr}(K_0^{-1/2}KK_0^{-1/2}) - \log\det(K_0^{-1/2}KK_0^{-1/2}) - n. \end{aligned}$$

The kernel matrix learning problem (3) can be rewritten as:

$$\min_{K \succeq 0} \quad f(K_0^{-1/2}KK_0^{-1/2}), \qquad \text{s.t.} \quad \mathrm{tr}(K(\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^T) \leq u, \quad (i,j) \in S,$$
$$\mathrm{tr}(K(\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^T) \geq \ell, \quad (i,j) \in \mathcal{D},$$

where $f(A) = \mathrm{tr}(A) - \log\det(A)$.

In this section, we will generalize our optimization problem to include more general loss functions beyond the LogDet-based loss function f specified above. We also generalize our constraints to include arbitrary constraints over the kernel matrix K rather than just the pairwise distance constraints in the above problem. Using the above specified generalizations, the optimization problem that we obtain is given by:

$$\min_{K \succeq 0} \; f(K_0^{-1/2}KK_0^{-1/2}), \quad \text{s.t. } g_i(K) \leq b_i, \; 1 \leq i \leq m, \tag{12}$$

where f and g_i are functions from $\mathbb{R}^{n \times n} \to \mathbb{R}$. We call f the *loss function* (or regularizer) and g_i the *constraints*. Note that if f and constraints g_i's are all convex, then the above problem can be solved optimally (under mild conditions) using standard convex optimization algorithms (Groschel et al., 1988). Note that our results also hold for unconstrained variants of the above problem, as well as variants with slack variables.

In general, such formulations are limited in that the learned kernel cannot readily be applied to new data points. However, we will show that the above proposed problem is equivalent to learning linear transformation (LT) kernel functions. Formally, an LT kernel function κ_W is a kernel function of the form $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T W \phi(\boldsymbol{y})$, where W is a positive semi-definite (PSD) matrix. A natural way to learn an LT kernel function would be to learn the parameterization matrix W using the provided side-information. To this end, we consider the following generalization of our LogDet based learning problem (2):

$$\min_{W \succeq 0} \; f(W), \quad \text{s.t. } g_i(\Phi^T W \Phi) \leq b_i, \; 1 \leq i \leq m, \tag{13}$$

where, as before, the function f is the loss function and the functions g_i are the constraints that encode the side information. The constraints g_i are assumed to be a function of the matrix $\Phi^T W \Phi$ of learned kernel values over the training data. Note that most Mahalanobis metric learning methods may be viewed as a special case of the above framework (see Section 5). Also, for data mapped to high-dimensional spaces via kernel functions, this problem is seemingly impossible to optimize since the size of W grows quadratically with the dimensionality.

4.2 Analysis

We now analyze the connection between the problems (12) and (13). We will show that the solutions to the two problems are equivalent, that is, by optimally solving one of the problems, the solution

to the other can be computed in closed form. Further, this result will yield insight into the type of kernel that is learned by the kernel learning problem.

We begin by defining the class of loss functions considered in our analysis.

Definition 3 *We say that $f : \mathbb{R}^{n\times n} \to \mathbb{R}$ is a **spectral function** if $f(A) = \sum_i f_s(\lambda_i)$, where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A and $f_s : \mathbb{R} \to \mathbb{R}$ is a real-valued scalar function. Note that if f_s is a convex scalar function, then f is also convex.*

Note that the LogDet based loss function in (3) is a spectral function. Similarly, most of the existing metric learning formulations have a spectral function as their objective function.

Now we state our main result that for a spectral function f, problems (12) and (13) are equivalent.

Theorem 4 *Let $K_0 = \Phi^T\Phi \succ 0$, f be a spectral function as in Definition 3 and assume that the global minimum of the corresponding strictly convex scalar function f_s is $\alpha > 0$. Let W^* be an optimal solution to* (13) *and K^* be an optimal solution to* (12)*. Then,*

$$W^* = \alpha I^d + \Phi S \Phi^T,$$

where $S = K_0^{-1}(K^* - \alpha K_0)K_0^{-1}$. Furthermore, $K^* = \Phi^T W^* \Phi$.

The first part of the theorem demonstrates that, given an optimal solution K^* to (12), one can construct the corresponding solution W^* to (13), while the second part shows the reverse. Note the similarities between this theorem and the earlier Theorem 1. We provide the proof of this theorem below. The main idea behind the proof is to first show that the optimal solution to (13) is always of the form $W = \alpha I^d + \Phi S \Phi^T$, and then we obtain the closed form expression for S using simple algebraic manipulations.

First we introduce and analyze an auxiliary optimization problem that will help in proving the above theorem. Consider the following problem:

$$\min_{W\succeq 0, L} \; f(W), \quad \text{s.t. } g_i(\Phi^T W \Phi) \le b_i,\ 1 \le i \le m, \quad W = \alpha I^d + ULU^T, \tag{14}$$

where $L \in \mathbb{R}^{k\times k}$, $U \in \mathbb{R}^{d\times k}$ is a column orthogonal matrix, and I^d is the $d \times d$ identity matrix. In general, k can be significantly smaller than $\min(n, d)$. Note that the above problem is identical to (13) except for an added constraint $W = \alpha I^d + ULU^T$. We now show that (14) is equivalent to a problem over $k \times k$ matrices. In particular, (14) is equivalent to (15) defined below.

Lemma 5 *Let f be a spectral function as in Definition 3 and let $\alpha > 0$ be any scalar. Then,* (14) *is equivalent to:*

$$\min_{L\succeq -\alpha I^k} \; f(\alpha I^k + L), \quad s.t. \;\; g_i(\alpha \Phi^T\Phi + \Phi^T U L U^T \Phi) \le b_i,\ 1 \le i \le m. \tag{15}$$

Proof The last constraint in (14) asserts that $W = \alpha I^d + ULU^T$, which implies that there is a one-to-one mapping between W and L: given W, L can be computed and vice-versa. As a result, we can eliminate the variable W from (14) by substituting $\alpha I^d + ULU^T$ for W (via the last constraint in (14)). The resulting optimization problem is:

$$\min_{L\succeq -\alpha I^k} \; f(\alpha I^d + ULU^T), \quad \text{s.t. } g_i(\alpha \Phi^T\Phi + \Phi^T U L U^T \Phi) \le b_i,\ 1 \le i \le m. \tag{16}$$

Note that (15) and (16) are the same except for their objective functions. Below, we show that both the objective functions are equal up to a constant, so they are interchangeable in the optimization problem. Let $U' \in \mathbb{R}^{d\times d}$ be an orthonormal matrix obtained by completing the basis represented by U, that is, $U' = [U\ U_\perp]$ for some $U_\perp \in \mathbb{R}^{d\times(d-k)}$ s.t. $U^T U_\perp = 0$ and $U_\perp^T U_\perp = I^{d-k}$. Now, $W = \alpha I^d + ULU^T = U'\left(\alpha I^d + \begin{bmatrix} L & 0 \\ 0 & 0 \end{bmatrix}\right) U'^T$. It is straightforward to see that for a spectral function f, $f(VWV^T) = f(W)$, where V is an orthogonal matrix. Also, $\forall A, B \in \mathbb{R}^{d\times d}$, $f\left(\begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}\right) = f(A) + f(B)$. Using the above observations, we get:

$$f(W) = f(\alpha I^d + ULU^T) = f\left(\begin{bmatrix} \alpha I^k + L & 0 \\ 0 & \alpha I^{d-k} \end{bmatrix}\right) = f(\alpha I^k + L) + (d-k) f(\alpha). \qquad (17)$$

Therefore, the objective functions of (15) and (16) differ by only a constant, that is, they are equivalent with respect to the optimization problem. The lemma follows. ■

We now show that for strictly convex spectral functions (see Definition 3) the optimal solution W^* to (13) is of the form $W^* = \alpha I^d + \Phi S \Phi^T$, for some S.

Lemma 6 *Suppose f, K_0 and α satisfy the conditions given in Theorem 4. Then, the optimal solution to* (13) *is of the form $W^* = \alpha I^d + \Phi S \Phi^T$, where S is a $n \times n$ matrix.*

Proof Note that $K_0 \succ 0$ implies that $d \geq n$. Our results can be extended when $d < n$, that is, $K_0 \succeq 0$, by using the pseudo-inverse of K_0 instead of the inverse. However, for simplicity we only present the full-rank case.

Now, let $W = U\Lambda U^T = \sum_j \lambda_j \boldsymbol{u}_j \boldsymbol{u}_j^T$ be the eigenvalue decomposition of W. Consider a constraint $g_i(\Phi^T W \Phi) \leq b_i$ as specified in (13). Note that if the j-th eigenvector $\boldsymbol{u}_j$ of W is orthogonal to the range space of Φ, that is, $\Phi^T \boldsymbol{u}_j = 0$, then the corresponding eigenvalue λ_j is not constrained (except for the non-negativity constraint imposed by the positive semi-definiteness constraint). Since the range space of Φ is at most n-dimensional, we can assume that $\lambda_j \geq 0, \forall j > n$ are not constrained by the linear inequality constraints in (13).

Since f satisfies the conditions of Theorem 4, $f(W) = \sum_j f_s(\lambda_j)$. Also, $f_s(\alpha) = \min_x f_s(x)$. Hence, to minimize $f(W)$, we can select $\lambda_j^* = \alpha \geq 0, \forall j > n$ (note that the non-negativity constraint is also satisfied here). Furthermore, the eigenvectors $\boldsymbol{u}_j$, $\forall j \leq n$, lie in the range space of Φ, that is, $\forall j \leq n$, $\boldsymbol{u}_j = \Phi \boldsymbol{z}_j$ for some $\boldsymbol{z}_j \in \mathbb{R}^n$. Therefore,

$$W^* = \sum_{j=1}^{n} \lambda_j^* \boldsymbol{u}_j^* \boldsymbol{u}_j^{*T} + \alpha \sum_{j=n+1}^{d} \boldsymbol{u}_j^* \boldsymbol{u}_j^{*T} = \sum_{j=1}^{n} (\lambda_j^* - \alpha) \boldsymbol{u}_j^* \boldsymbol{u}_j^{*T} + \alpha \sum_{j=1}^{d} \boldsymbol{u}_j^* \boldsymbol{u}_j^{*T} = \Phi S \Phi^T + \alpha I^d,$$

where $S = \sum_{j=1}^{n} (\lambda_j^* - \alpha) \boldsymbol{z}_j^* \boldsymbol{z}_j^{*T}$. ■

Now we use Lemmas 5 and 6 to prove Theorem 4.

Proof [Proof of Theorem 4] Let $\Phi = U_\Phi \Sigma V_\Phi^T$ be the singular value decomposition (SVD) of Φ. Note that $K_0 = \Phi^T \Phi = V_\Phi \Sigma^2 V_\Phi^T$, so $\Sigma V_\Phi^T = V_\Phi^T K_0^{1/2}$. Also, assuming $\Phi \in \mathbb{R}^{d\times n}$ to be full-rank and $d > n$, $V_\Phi V_\Phi^T = I$.

Using Lemma 6, the optimal solution to (13) is restricted to be of the form $W = \alpha I^d + \Phi S \Phi^T = \alpha I^d + U_\Phi \Sigma V_\Phi^T S V_\Phi \Sigma U_\Phi^T = \alpha I^d + U_\Phi V_\Phi^T K_0^{1/2} S K_0^{1/2} V_\Phi U_\Phi^T = \alpha I^d + U_\Phi V_\Phi^T L V_\Phi U_\Phi^T$, where $L = K_0^{1/2} S K_0^{1/2}$.

As a result, for spectral functions f, (13) is equivalent to (14), so using Lemma 5, (13) is equivalent to (15) with $U = U_\Phi V_\Phi^T$ and $L = K_0^{1/2} S K_0^{1/2}$. Also, note that the constraints in (15) can be simplified to:

$$g_i(\alpha\Phi^T\Phi + \Phi^T U L U^T \Phi) \leq b_i \equiv g_i(\alpha K_0 + K_0^{1/2} L K_0^{1/2}) \leq b_i.$$

Now, let $K = \alpha K_0 + K_0^{1/2} L K_0^{1/2} = \alpha K_0 + K_0 S K_0$, that is, $L = K_0^{-1/2}(K - \alpha K_0)K_0^{-1/2}$. Theorem 4 now follows by substituting for L in (15). ■

As a first consequence of this result, we can achieve induction over the learned kernels, analogous to (6) for the LogDet case. Given that $K = \Phi^T W \Phi$, we can see that the learned kernel function is a linear transformation kernel; that is, $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T W \phi(\boldsymbol{x}_j)$. Given a pair of new data points $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, we use the fact that the learned kernel is a linear transformation kernel, along with the first result of the theorem ($W = \alpha I^d + \Phi S \Phi^T$) to compute the learned kernel as:

$$\phi(\boldsymbol{z}_1)^T W \phi(\boldsymbol{z}_2) = \alpha \cdot \kappa_0(\boldsymbol{z}_1, \boldsymbol{z}_2) + \boldsymbol{k}_1^T S \boldsymbol{k}_2, \text{where } \boldsymbol{k}_i = [\kappa_0(\boldsymbol{z}_i, \boldsymbol{x}_1), ..., \kappa_0(\boldsymbol{z}_i, \boldsymbol{x}_n)]^T. \tag{18}$$

Since LogDet divergence is also a spectral function, Theorem 4 is a generalization of Theorem 1 and implies kernelization for our metric learning formulation (2). Moreover, many Mahalanobis metric learning methods can be viewed as a special case of (13), so a corollary of Theorem 4 is that we can constructively apply these metric learning methods in kernel space by solving their corresponding kernel learning problem, and then compute the learned metrics via (18). Kernelization of Mahalanobis metric learning has previously been established for some special cases; our results generalize and extend previous methods, as well as provide simpler techniques in some cases. We will further elaborate in Section 5 with several special cases.

4.3 Parameter Reduction

As noted in Section 3.6 that the size of the kernel matrices K and the parameter matrices S are $n \times n$, and thus grow quadratically with the number of data points. Similar to the special case of the LogDet divergence (see Section 3.6), we would like to have a way to restrict our general optimization problem (12) over a smaller number of parameters. So, we now discuss a generalization of (13) by introducing an additional constraint to make it possible to reduce the number of parameters to learn, permitting scalability to data sets with many training points *and* with very high dimensionality.

Theorem 4 shows that the optimal K^* is of the form $\Phi^T W^* \Phi = \alpha K_0 + K_0 S K_0$. In order to accommodate fewer parameters to learn, a natural option is to replace the unknown S matrix with a *low-rank* matrix JLJ^T, where $J \in \mathbb{R}^{n \times k}$ is a pre-specified matrix, $L \in \mathbb{R}^{k \times k}$ is unknown (we use L instead of S to emphasize that S is of size $n \times n$ whereas L is $k \times k$), and the rank k is a parameter of the algorithm. Then, we will explicitly enforce that the learned kernel is of this form.

By plugging in $K = \alpha K_0 + K_0 S K_0$ into (12) and replacing S with JLJ^T, the resulting optimization problem is given by:

$$\min_{L \succeq 0} \; f(\alpha I^n + K_0^{1/2} J L J^T K_0^{1/2}), \quad \text{s.t. } g_i(\alpha K_0 + K_0 J L J^T K_0) \leq b_i, \; 1 \leq i \leq m. \tag{19}$$

Note that the above problem is a strict generalization of our LogDet function based parameter reduction approach (see Section 3.6).

While the above problem involves only $k \times k$ variables, the functions f and g_i's are applied to $n \times n$ matrices and therefore the problem may still be computationally expensive to optimize. Below,

we show that for any spectral function f and linear constraints $g_i(K) = \mathrm{tr}(C_i K)$, (19) reduces to a problem that applies f and g_i's to $k \times k$ matrices only, which provides significant scalability.

Theorem 7 *Let $K_0 = \Phi^T \Phi$ and J be some matrix in $\mathbb{R}^{n \times k}$. Also, let the loss function f be a spectral function (see Definition 3) such that the corresponding strictly convex scalar function f_s has the global minimum at $\alpha > 0$. Then problem* (19) *with $g_i(K) = \mathrm{tr}(C_i K)$ is equivalent to the following alternative optimization problem:*

$$\min_{L \succeq -\alpha (K^J)^{-1}} \; f((K^J)^{-1/2}(\alpha K^J + K^J L K^J)(K^J)^{-1/2}),$$
$$s.t.\ \mathrm{tr}(L J^T K_0 C_i K_0 J) \leq b_i - \mathrm{tr}(\alpha K_0 C_i),\ 1 \leq i \leq m, \tag{20}$$

where $K^J = J^T K_0 J$.

Proof Let $U = K_0^{1/2} J (J^T K_0 J)^{-1/2}$ and let J be a full rank matrix, then U is an orthogonal matrix. Using (17) we get,

$$f(\alpha I^n + U(J^T K_0 J)^{1/2} L (J^T K_0 J)^{1/2} U^T) = f(\alpha I^k + (J^T K_0 J)^{1/2} L (J^T K_0 J)^{1/2}).$$

Now consider a linear constraint $\mathrm{tr}(C_i(\alpha K_0 + K_0 J L J^T K_0)) \leq b_i$. This can be easily simplified to $\mathrm{tr}(L J^T K_0 C_i K_0 J) \leq b_i - \mathrm{tr}(\alpha K_0 C_i)$. Similar simple algebraic manipulations to the PSD constraint completes the proof. ■

Note that (20) is over $k \times k$ matrices (after initial pre-processing) and is in fact similar to the kernel learning problem (12), but with a kernel K^J of smaller size $k \times k$, $k \ll n$.

Similar to (12), we can show that (19) is also equivalent to LT kernel function learning. This enables us to naturally apply the above kernel learning problem in the inductive setting.

Theorem 8 *Consider* (19) *with $g_i(K) = \mathrm{tr}(C_i K)$ and a spectral function f whose corresponding scalar function f_s has a global minimum at $\alpha > 0$. Let $J \in \mathbb{R}^{n \times k}$. Then,* (19) *and* (20) *with $g_i(K) = \mathrm{tr}(C_i K)$ are equivalent to the following linear transformation kernel learning problem (analogous to the connection between* (12) *and* (13)*):*

$$\min_{W \succeq 0, L} \; f(W), \qquad s.t.\ \mathrm{tr}(\Phi^T W \Phi) \leq b_i,\ 1 \leq i \leq m, \quad W = \alpha I^d + \Phi J L J^T \Phi^T. \tag{21}$$

Proof Consider the last constraint in (21): $W = \alpha I^d + \Phi J L J^T \Phi^T$. Let $\Phi = U \Sigma V^T$ be the SVD of Φ. Hence, $W = \alpha I^d + U V^T V \Sigma V^T J L J^T V \Sigma V^T V U^T = \alpha I^d + U V^T K_0^{1/2} J L J^T K_0^{1/2} V U^T$, where we used $K_0^{1/2} = V \Sigma V^T$. For dis-ambiguity, rename L as L' and U as U'. The result now follows by using Lemma 5 where $U = U' V^T$ and $L = K_0^{1/2} J L' J^T K_0^{1/2}$. ■

Note that, in contrast to (13), where the last constraint over W is achieved automatically, (21) requires this constraint on W to be satisfied during the optimization process, which leads to a reduced number of parameters for our kernel learning problem. The above theorem shows that our reduced-parameter kernel learning method (19) also implicitly learns a linear transformation kernel function, hence we can generalize the learned kernel to unseen data points using an expression similar to (18).

5. Special Cases

In the previous section, we proved a general result showing the connections between metric and kernel learning using a wide class of loss functions and constraints. In this section, we consider a few special cases of interest: the von Neumann divergence, the squared Frobenius norm and semi-definite programming. For each of the cases, we derive the required optimization problem and mention the relevant optimization algorithms that can be used.

5.1 Von Neumann Divergence

The von Neumann divergence is a generalization of the well known KL-divergence to matrices. It is used extensively in quantum computing to compare density matrices of two different systems (Nielsen and Chuang, 2000). It is also used in the exponentiated matrix gradient method by Tsuda et al. (2005), online-PCA method by Warmuth and Kuzmin (2008) and fast SVD solver by Arora and Kale (2007). The von Neumann divergence between A and A_0 is defined to be, $D_{\text{vN}}(A,A_0) = \text{tr}(A\log A - A\log A_0 - A + A_0)$, where both A and A_0 are positive definite. Computing the von Neumann divergence with respect to the identity matrix, we get: $f_{\text{vN}}(A) = \text{tr}(A\log A - A + I)$. Note that f_{vN} is a spectral function with corresponding scalar function $f_{\text{vN}}(\lambda) = \lambda\log\lambda - \lambda$ and minima at $\lambda = 1$. Now, the kernel learning problem (12) with loss function f_{vN} and linear constraints is:

$$\min_{K\succeq 0} \; f_{\text{vN}}(K_0^{-1/2} K K_0^{-1/2}), \qquad \text{s.t. } \; \text{tr}(KC_i) \leq b_i, \quad \forall 1 \leq i \leq m. \tag{22}$$

As f_{vN} is an spectral function, using Theorem 4, the above kernel learning problem is equivalent to the following metric learning problem:

$$\min_{W\succeq 0} \; D_{\text{vN}}(W, I), \qquad \text{s.t. } \; \text{tr}(W\Phi C_i \Phi^T) \leq b_i, \quad \forall 1 \leq i \leq m.$$

Using elementary linear algebra, we obtain the following simplified version:

$$\min_{\lambda_1,\lambda_2,\ldots,\lambda_m \geq 0} \; F(\lambda) = \text{tr}(\exp(-C(\lambda)K_0)) + b(\lambda),$$

where $C(\lambda) = \sum_i \lambda_i C_i$ and $b(\lambda) = \sum_i \lambda_i b_i$. Further, $\frac{\partial F}{\partial \lambda_i} = \text{tr}(\exp(-C(\lambda)K_0)C_i K_0) + b_i$, so any first-order smooth optimization method can be used to solve the above dual problem. Alternatively, similar to the method of Kulis et al. (2008), Bregman's projection method can be used to solve the primal problem (22).

5.2 Pseudo Online Metric Learning (POLA)

Shalev-Shwartz et al. (2004) proposed the following batch metric learning formulation:

$$\min_{W\succeq 0} \; \|W\|_F^2, \qquad \text{s.t. } \; y_{ij}(b - d_W(\boldsymbol{x}_i, \boldsymbol{x}_j)) \geq 1, \;\; \forall (i,j) \in \mathcal{P},$$

where $y_{ij} = 1$ if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are similar, and $y_{ij} = -1$ if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are dissimilar. $\mathcal{P}$ is a set of pairs of points with known distance constraints. POLA is an instantiation of (13) with $f(A) = \frac{1}{2}\|A\|_F^2$ and side-information available in the form of pair-wise distance constraints. Note that the regularizer $f(A) = \frac{1}{2}\|A\|^2$ was also employed in Schultz and Joachims (2003) and Kwok and Tsang (2003), and

these methods also fall under our general formulation. In this case, f is once again a strictly convex spectral function, and its global minimum is $\alpha = 0$, so we can use (12) to solve for the learned kernel K as

$$\min_{K \succeq 0} \ \|KK_0^{-1}\|_F^2, \qquad \text{s.t. } g_i(K) \leq b_i,\ 1 \leq i \leq m,$$

The constraints g_i for this problem can be easily constructed by re-writing each of POLA's constraints as a function of $\Phi^T W \Phi$. Note that the above approach for kernelization is much simpler than the method suggested in Shalev-Shwartz et al. (2004), which involves a kernelized Gram-Schmidt procedure at each step of the algorithm.

5.3 SDPs

Weinberger et al. (2005) and Globerson and Roweis (2005) proposed metric learning formulations that can be rewritten as semi-definite programs (SDP), which is a special case of (13) with the loss function being a linear function. Consider the following general semidefinite program (SDP) to learn a linear transformation W:

$$\min_{W \succeq 0} \ \text{tr}(W\Phi C_0 \Phi^T), \qquad \text{s.t. } \text{tr}(W\Phi C_i \Phi^T) \leq b_i, \quad \forall 1 \leq i \leq m. \tag{23}$$

Here we show that this problem can be efficiently solved for high dimensional data in its kernel space, hence kernelizing the metric learning methods introduced by Weinberger et al. (2005) and Globerson and Roweis (2005).

Theorem 9 *Problem* (23) *is kernelizable.*

Proof (23) has a linear objective, that is, it is a non-strict convex problem that may have multiple solutions. A variety of regularizations can be considered that lead to slightly different solutions. Here, we consider the LogDet regularization, which seeks the solution with maximum determinant. To this effect, we add a log-determinant regularization:

$$\min_{W \succeq 0} \ \text{tr}(W\Phi C_0 \Phi^T) - \gamma \log\det W, \qquad \text{s.t. } \text{tr}(W\Phi C_i \Phi^T) \leq b_i, \quad \forall 1 \leq i \leq m. \tag{24}$$

The above regularization was also considered by Kulis et al. (2009b), who provided a fast projection algorithm for the case when each C_i is a one-rank matrix and discussed conditions for which the optimal solution to the regularized problem is an optimal solution to the original SDP. The above formulation also generalizes the metric learning formulation of RCA (Bar-Hillel et al., 2005).

Consider the following variational formulation of (24):

$$\begin{aligned} \min_t \min_{W \succeq 0} \ t - \gamma \log\det W, \qquad & \text{s.t. } \text{tr}(W\Phi C_i \Phi^T) \leq b_i, \quad \forall 1 \leq i \leq m, \\ & \text{tr}(W\Phi C_0 \Phi^T) \leq t. \end{aligned} \tag{25}$$

Note that the objective function of the inner optimization problem of (25) is a spectral function and hence using Theorem 4, (25), or equivalently (24), is kernelizable. ■

5.4 Trace-norm Based Inductive Semi-supervised Kernel Dimensionality Reduction (Trace-SSIKDR)

Finally, we apply our framework to semi-supervised kernel dimensionality reduction, which provides a novel and practical application of our framework. While there exists a variety of methods for kernel dimensionality reduction, most of these methods are unsupervised (e.g., kernel-PCA) or are restricted to the transductive setting. In contrast, we can use our kernel learning framework to learn a low-rank transformation of the feature vectors implicitly that in turn provides a low-dimensional embedding of the data set. Furthermore, our framework permits a variety of side-information such as pair-wise or relative distance constraints, beyond the class label information allowed by existing transductive methods.

We describe our method starting from the linear transformation problem. Our goal is to learn a low-rank linear transformation W whose corresponding low-dimensional mapped embedding of $\boldsymbol{x}_i$ is $W^{1/2}\phi(\boldsymbol{x}_i)$. Even when the dimensionality of $\phi(\boldsymbol{x}_i)$ is very large, if the rank of W is low enough, then the mapped embedding will have small dimensionality. With that in mind, a possible regularizer could be the rank, that is, $r(A) = \text{rank}(A)$; one can easily show that this satisfies the definition of a spectral function. Unfortunately, optimization is intractable in general with the non-convex rank function, so we use the trace-norm relaxation for the matrix rank function, that is, we set $f(A) = \text{tr}(A)$. This function has been extensively studied as a relaxation for the rank function in Recht et al. (2010), and it satisfies the definition of a spectral function (with $\alpha = 0$). We also add a small Frobenius norm regularization for ease of optimization (this does not affect the spectral property of the regularization function). Then using Theorem 4, the resulting relaxed kernel learning problem is:

$$\min_{K \succeq 0} \tau \,\text{tr}(K_0^{-1/2} K K_0^{-1/2}) + \|K_0^{-1/2} K K_0^{-1/2}\|_F^2, \qquad \text{s.t.} \qquad \text{tr}(C_i K) \leq b_i,\ 1 \leq i \leq m, \tag{26}$$

where $\tau > 0$ is a parameter. The above problem can be solved using a method based on Uzawa's inexact algorithm, similar to Cai et al. (2008).

We briefly describe the steps taken by our method at each iteration. For simplicity, denote $\tilde{K} = K_0^{-1/2} K K_0^{-1/2}$; we will optimize with respect to $\tilde{K}$ instead of K. Let $\tilde{K}^t$ be the t-th iterate. Associate variable $z_i^t, 1 \leq i \leq m$ with each constraint at each iteration t, and let $z_i^0 = 0, \forall i$. Let δ_t be the step size at iteration t. The algorithm performs the following updates:

$$U \Sigma U^T \leftarrow K_0^{1/2} C K_0^{1/2},$$

$$\tilde{K}^t \leftarrow U \max(\Sigma - \tau I, 0) U^T, \tag{27}$$

$$z_i^t \leftarrow z_i^{t-1} - \delta \max(\text{tr}(C_i K^{1/2} \tilde{K}^t K^{1/2}) - b_i, 0),\ \forall i, \tag{28}$$

where $C = \sum_i z_i^{t-1} C_i$. The above updates require computation of $K_0^{1/2}$ which is expensive for large high-rank matrices. However, using elementary linear algebra we can show that $\tilde{K}$ and the learned kernel function can be computed efficiently without computing $K_0^{1/2}$ by maintaining $S = K_0^{-1/2} \tilde{K} K_0^{-1/2}$ from step to step.

We first prove a technical lemma to relate eigenvectors U of $K_0^{1/2} C K_0^{1/2}$ and V of the matrix CK_0.

Lemma 10 *Let $K_0^{1/2} C K_0^{1/2} = U_k \Sigma_k U_k^T$, where U_k contains the top-k eigenvectors of $K_0^{1/2} C K_0^{1/2}$ and Σ_k contains the top-k eigenvalues of $K_0^{1/2} C K_0^{1/2}$. Similarly, let $CK_0 = V_k \Lambda_k V_k^{-1}$, where V_k contains*

Algorithm 2 Trace-SSIKDR

Input: K_0, $(C_i, b_i), 1 \leq i \leq m$, τ, δ

1: **Initialize:** $z_i^0 = 0$, $t = 0$
2: **repeat**
3: $t = t + 1$
4: Compute V_k and Σ_k, the top k eigenvectors and eigenvalues of $\left(\sum_i z_i^{t-1} C_i\right) K_0$, where $k = \text{argmax}_j \sigma_j > \tau$
5: $D_k(i,i) \leftarrow 1/\boldsymbol{v}_i^T K_0 \boldsymbol{v}_i$, $1 \leq i \leq k$
6: $z_i^t \leftarrow z_i^{t-1} - \delta \max(\text{tr}(C_i K_0 V_k D_k \Sigma_k D_k V_k^T K_0) - b_i, 0), \forall i$.
7: **until** Convergence
8: **Return** Σ_k, D_k, V_k

the top-k right eigenvectors of CK_0 and Λ_k contains the top-k eigenvalues of CK_0. Then,

$$U_k = K_0^{1/2} V_k D_k, \qquad \Sigma_k = \Lambda_k,$$

where D_k is a diagonal matrix with i-th diagonal element $D_k(i,i) = 1/\boldsymbol{v}_i^T K_0 \boldsymbol{v}_i$. Note that eigenvalue decomposition is unique up to sign, so we assume that the sign has been set correctly.

Proof Let $\boldsymbol{v}_i$ be i-th eigenvector of CK_0. Then, $CK_0\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$. Multiplying both sides with $K_0^{1/2}$, we get $K_0^{1/2} C K_0^{1/2} K_0^{1/2} \boldsymbol{v}_i = \lambda_i K_0^{1/2} \boldsymbol{v}_i$. After normalization we get:

$$(K_0^{1/2} C K_0^{1/2}) \frac{K_0^{1/2} \boldsymbol{v}_i}{\boldsymbol{v}_i^T K_0 \boldsymbol{v}_i} = \lambda_i \frac{K_0^{1/2} \boldsymbol{v}_i}{\boldsymbol{v}_i^T K_0 \boldsymbol{v}_i}.$$

Hence, $\frac{K_0^{1/2}\boldsymbol{v}_i}{\boldsymbol{v}_i^T K_0 \boldsymbol{v}_i} = K_0^{1/2} \boldsymbol{v}_i D_k(i,i)$ is the i-th eigenvector $\boldsymbol{u}_i$ of $K_0^{1/2} C K_0^{1/2}$. Also, $\Sigma_k(i,i) = \lambda_i$. ■

Using the above lemma and (27), we get: $\tilde{K} = K_0^{1/2} V_k D_k \lambda D_k V_k^{-1} K_0^{1/2}$. Therefore, the update for the z variables (see (28)) reduces to:

$$z_i^t \leftarrow z_i^{t-1} - \delta \max(\text{tr}(C_i K_0 V_k D_k \lambda D_k V_k^{-1} K_0) - b_i, 0), \forall i.$$

Lemma 10 also implies that if k eigenvalues of CK_0 are larger than τ then we only need the top k eigenvalues of CK_0. Since k is typically significantly smaller than n, the update should be significantly more efficient than computing the whole eigenvalue decomposition.

Algorithm 2 details an efficient method for optimizing (26) and returns matrices Σ_k, D_k and V_k, all of which contain only $O(nk)$ parameters, where k is the rank of $\tilde{K}^t$, which changes from iteration to iteration. Note that step 4 of the algorithm computes k singular vectors and requires only $O(nk^2)$ computation. Note also that the learned embedding $\boldsymbol{x}_i \rightarrow \tilde{K}^{1/2} K_0^{-1/2} \boldsymbol{k}_i$, where $\boldsymbol{k}_i$ is a vector of input kernel function values between $\boldsymbol{x}_i$ and the training data, can be computed efficiently as $\boldsymbol{x}_i \rightarrow \Sigma_k^{1/2} D_k V_k \boldsymbol{k}_i$, which does not require $K_0^{1/2}$ explicitly.

6. Experimental Results

In Section 3, we presented metric learning as a constrained LogDet optimization problem to learn a linear transformation, and we showed that the problem can be efficiently kernelized to learn

linear transformation kernels. Kernel learning yields two fundamental advantages over standard non-kernelized metric learning. First, a non-linear kernel can be used to learn non-linear decision boundaries common in applications such as image analysis. Second, in Section 3.6, we showed that the kernelized problem can be learned with respect to a reduced basis of size k, admitting a learned kernel parameterized by $O(k^2)$ values. When the number of training examples n is large, this represents a substantial improvement over optimizing over the entire $O(n^2)$ matrix, both in terms of computational efficiency as well as statistical robustness. In Section 4, we generalized kernel function learning to other loss functions. A special case of our approach is the trace-norm based kernel function learning problem, which can be applied to the task of semi-supervised inductive kernel dimensionality reduction.

In this section, we present experiments from several domains: benchmark UCI data, automated software debugging, text analysis, and object recognition for computer vision. We evaluate performance of our learned distance metrics or kernel functions in the context of a) classification accuracy for the k-nearest neighbor algorithm, b) kernel dimensionality reduction. For the classification task, our k-nearest neighbor classifier uses $k = 10$ nearest neighbors (except for Section 6.3 where we use $k = 1$), breaking ties arbitrarily. We select the value of k arbitrarily and expect to get slightly better accuracies using cross-validation. Accuracy is defined as the number of correctly classified examples divided by the total number of classified examples. For the dimensionality reduction task, we visualize the two dimensional embedding of the data using our trace-norm based method with pairwise similarity/dissimilarity constraints.

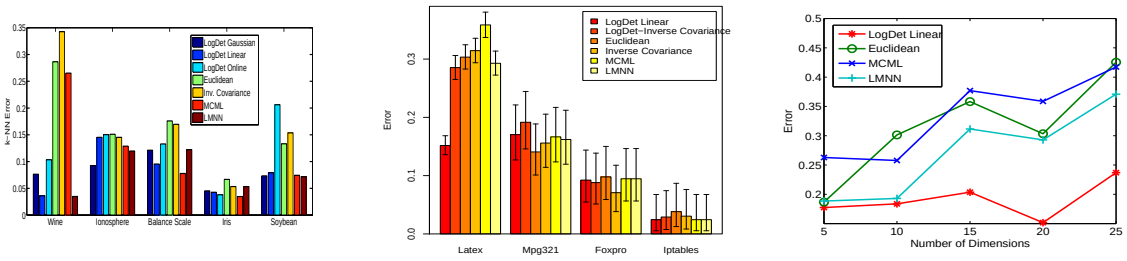
For our proposed algorithms, pairwise constraints are inferred from true class labels. For each class i, 100 pairs of points are randomly chosen from within class i and are constrained to be similar, and 100 pairs of points are drawn from classes other than i to form dissimilarity constraints. Given c classes, this results in $100c$ similarity constraints, and $100c$ dissimilarity constraints, for a total of $200c$ constraints. The upper and lower bounds for the similarity and dissimilarity constraints are determined empirically as the 1^{st} and 99^{th} percentiles of the distribution of distances computed using a baseline Mahalanobis distance parameterized by W_0. Finally, the slack penalty parameter γ used by our algorithms is cross-validated using values $\{.01, .1, 1, 10, 100, 1000\}$.

All metrics/kernels are trained using data only in the training set. Test instances are drawn from the test set and are compared to examples in the training set using the learned distance/kernel function. The test and training sets are established using a standard two-fold cross validation approach. For experiments in which a baseline distance metric/kernel is evaluated (for example, the squared Euclidean distance), nearest neighbor searches are again computed from test instances to only those instances in the training set.

For additional large-scale results, see Kulis et al. (2009a), which uses our parameter-reduction strategy to learn kernels over a data set containing nearly half a million images in 24,000 dimensional space for the problem of human-body pose estimation; we also applied our algorithms on the MNIST data set of 60,000 digits in Kulis et al. (2008).

6.1 Low-Dimensional Data Sets

First we evaluate our LogDet divergence based metric learning method (see Algorithm 1) on the standard UCI data sets in the low-dimensional (non-kernelized) setting, to directly compare with several existing metric learning methods. In Figure 1 (a), we compare LogDet Linear (K_0 equals the linear kernel) and the LogDet Gaussian (K_0 equals Gaussian kernel in kernel space) algorithms



(a)UCI Data Sets (b) Clarify Data Sets (c) Latex

Figure 1: **(a)**: Results over benchmark UCI data sets. LogDet metric learning was run with in input space (LogDet Linear) as well as in kernel space with a Gaussian kernel (LogDet Gaussian). **(b), (c)**: Classification error rates for k-nearest neighbor software support via different learned metrics. We see in figure (b) that LogDet Linear is the only algorithm to be optimal (within the 95% confidence intervals) across all data sets. In (c), we see that the error rate for the Latex data set stays relatively constant for LogDet Linear.

against existing metric learning methods for k-NN classification. We also show results of a recently-proposed online metric learning algorithm based on the LogDet divergence over this data (Jain et al., 2008), called LogDet Online. We use the squared Euclidean distance, $d(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}-\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y})$ as a baseline method (i.e., $W_0 = I$). We also use a Mahalanobis distance parameterized by the inverse of the sample covariance matrix (i.e., $W_0 = \Sigma^{-1}$, where Σ is the sample covariance matrix of the data). This method is equivalent to first performing a standard PCA whitening transform over the feature space and then computing distances using the squared Euclidean distance. We compare our method to two recently proposed algorithms: Maximally Collapsing Metric Learning by Globerson and Roweis (2005) (MCML), and metric learning via Large Margin Nearest Neighbor by Weinberger et al. (2005) (LMNN). Consistent with existing work such as Globerson and Roweis (2005), we found the method of Xing et al. (2002) to be very slow and inaccurate, so the latter was not included in our experiments. As seen in Figure 1 (a), LogDet Linear and LogDet Gaussian algorithms obtain somewhat higher accuracy for most of the data sets. In addition to our evaluations on standard UCI data sets, we also apply our algorithm to the recently proposed problem of nearest neighbor software support for the Clarify system (Ha et al., 2007). The basis of the Clarify system lies in the fact that modern software design promotes modularity and abstraction. When a program terminates abnormally, it is often unclear which component should be responsible or capable of providing an error report. The system works by monitoring a set of predefined program features (the data sets presented use function counts) during program runtime which are then used by a classifier in the event of abnormal program termination. Nearest neighbor searches are particularly relevant to this problem. Ideally, the neighbors returned should not only have the correct class label, but should also represent similar program configurations or program inputs. Such a matching can be a powerful tool to help users diagnose the root cause of their problem. The four data sets we use correspond to the following software: Latex (the document compiler, 9 classes), Mpg321 (an mp3 player, 4 classes), Foxpro (a database manager, 4 classes), and Iptables (a Linux kernel application, 5 classes).

Data Set	n	d	Unsupervised	LogDet Linear	HMRF-KMeans
Ionosphere	351	34	0.314	**0.113**	0.256
Digits-389	317	16	0.226	**0.175**	0.286

Table 1: Unsupervised k-means clustering error using the baseline squared Euclidean distance, along with semi-supervised clustering error with 50 constraints.

Our experiments on the Clarify system, like the UCI data, are over fairly low-dimensional data. Ha et al. (2007) showed that high classification accuracy can be obtained by using a relatively small subset of available features. Thus, for each data set, we use a standard information gain feature selection test to obtain a reduced feature set of size 20. From this, we learn metrics for k-NN classification using the methods developed in this paper. Results are given in Figure 1(b). The LogDet Linear algorithm yields significant gains for the Latex benchmark. Note that for data sets where Euclidean distance performs better than the inverse covariance metric, the LogDet Linear algorithm that normalizes to the standard Euclidean distance yields higher accuracy than that regularized to inverse covariance (LogDet-Inverse Covariance). In general, for the Mpg321, Foxpro, and Iptables data sets, learned metrics yield only marginal gains over the baseline Euclidean distance measure.

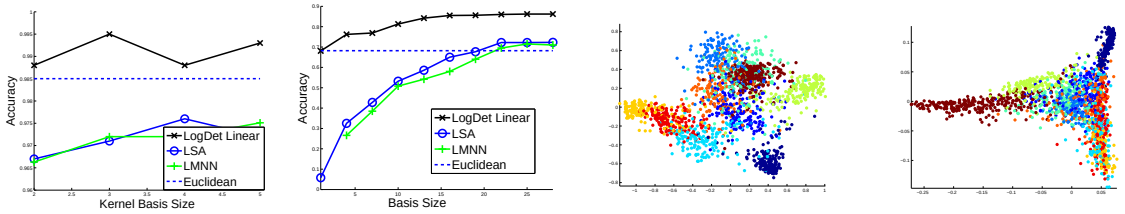
Figure 1(c) shows the error rate for the Latex data sets with a varying number of features (the feature sets are again chosen using the information gain criteria). We see here that LogDet Linear is surprisingly robust. Euclidean distance, MCML, and LMNN all achieve their best error rates for five dimensions. LogDet Linear, however, attains its lowest error rate of .15 at $d = 20$ dimensions.

We also briefly present some semi-supervised clustering results for two of the UCI data sets. Note that both MCML and LMNN are not amenable to optimization subject to pairwise distance constraints. Instead, we compare our method to the semi-supervised clustering algorithm HMRF-KMeans (Basu et al., 2004). We use a standard 2-fold cross validation approach for evaluating semi-supervised clustering results. Distances are constrained to be either similar or dissimilar, based on class values, and are drawn only from the training set. The entire data set is then clustered into c clusters using k-means (where c is the number of classes) and error is computed using only the test set. Table 1 provides results for the baseline k-means error, as well as semi-supervised clustering results with 50 constraints.

6.2 Metric Learning for Text Classification

Next we present results in the text domain. Our text data sets are created by standard bag-of-words Tf-Idf representations. Words are stemmed using a standard Porter stemmer and common stop words are removed, and the text models are limited to the 5,000 words with the largest document frequency counts. We provide experiments for two data sets: CMU 20-Newsgroups Data Set (2008), and Classic3 Data Set (2008). Classic3 is a relatively small 3 class problem with 3,891 instances. The newsgroup data set is much larger, having 20 different classes from various newsgroup categories and 20,000 instances.

Our text experiments employ a linear kernel, and we use a set of basis vectors that is constructed from the class labels via the following procedure. Let c be the number of distinct classes and let k be the size of the desired basis. If $k = c$, then each class mean r_i is computed to form the basis $R = [\boldsymbol{r}_1 \ldots \boldsymbol{r}_c]$. If $k < c$ a similar process is used but restricted to a randomly selected subset of



(a) Classic3 (b) 20-Newsgroups (c)USPS Digits (d)USPS Digits

Figure 2: **(a), (b)**: Classification accuracy for our Mahalanobis metrics learned over basis of different dimensionality. Overall, our method (LogDet Linear) significantly outperforms existing methods. **(c)**: Two-dimensional embedding of 2000 USPS digits obtained using our method Trace-SSIKDR for a training set of just 100 USPS digits. Note that we use the **inductive** setting here and the embedding is color coded according to the underlying digit. **(d)**: Embedding of the USPS digits data set obtained using kernel-PCA.

k classes. If $k > c$, instances within each class are clustered into approximately $\frac{k}{c}$ clusters. Each cluster's mean vector is then computed to form the set of low-rank basis vectors R. Figure 2 shows classification accuracy across bases of varying sizes for the Classic3 data set, along with the newsgroup data set. As baseline measures, the standard squared Euclidean distance is shown, along with Latent Semantic Analysis (LSA) (Deerwester et al., 1990), which works by projecting the data via principal components analysis (PCA), and computing distances in this projected space. Comparing our algorithm to the baseline Euclidean measure, we can see that for smaller bases, the accuracy of our algorithm is similar to the Euclidean measure. As the size of the basis increases, our method obtains significantly higher accuracy compared to the baseline Euclidean measure.

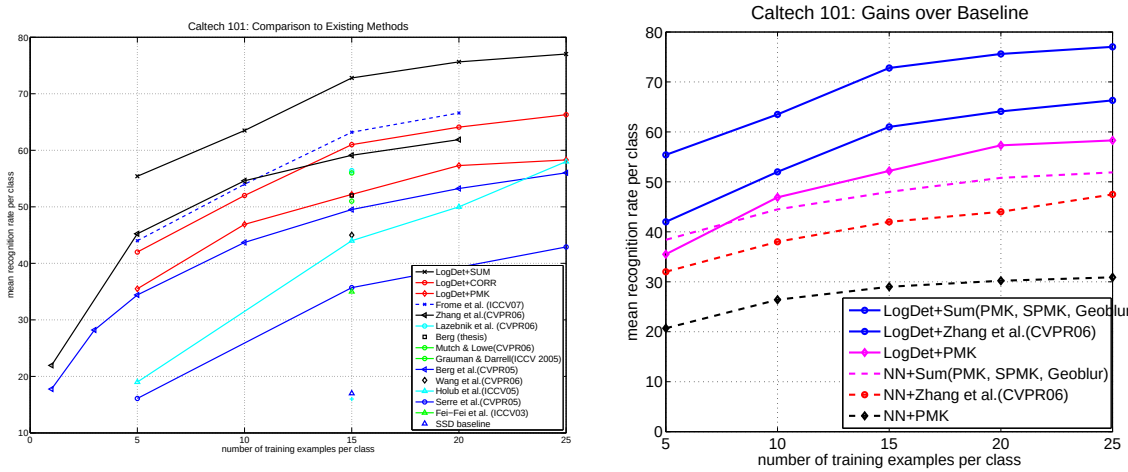
6.3 Kernel Learning for Visual Object Recognition

Next we evaluate our method over high-dimensional data applied to the object-recognition task using Caltech-101 Data Set (2004), a common benchmark for this task. The goal is to predict the category of the object in the given image using a k-NN classifier.

We compute distances between images using learning kernels with three different base image kernels: 1) PMK: Grauman and Darrell's pyramid match kernel (Grauman and Darrell, 2007) applied to SIFT features, 2) CORR: the kernel designed by Zhang et al. (2006) applied to geometric blur features , and 3) SUM: the average of four image kernels, namely, PMK (Grauman and Darrell, 2007), Spatial PMK (Lazebnik et al., 2006), and two kernels obtained via geometric blur (Berg and Malik, 2001). Note that the underlying dimensionality of these embeddings is typically in the millions of dimensions.

We evaluate the effectiveness of metric/kernel learning on this data set. We pose a k-NN classification task, and evaluate both the original (SUM, PMK or CORR) and learned kernels. We set $k = 1$ for our experiments; this value was chosen arbitrarily. We vary the number of training examples T per class for the database, using the remainder as test examples, and measure accuracy in terms of the mean recognition rate per class, as is standard practice for this data set.

Figure 3 (a) shows our results relative to several other existing techniques that have been applied to this data set. Our approach outperforms all existing single-kernel classifier methods when using



(a) (b)

Figure 3: Results on Caltech-101. LogDet+SUM refers to our learned kernel when the base kernel is the average of four kernels (PMK, SPMK, Geoblur-1, Geoblur-2), LogDet+PMK refers to the learned kernel when the base kernel is pyramid match kernel, and LogDet+CORR refers to the learned kernel when the base kernel is correspondence kernel of Zhang et al. (2006). **(a)**: Comparison of LogDet based metric learning method with other state-of-the-art object recognition methods. Our method outperforms all other single metric/kernel approaches. **(b)**: Our learned kernels significantly improve NN recognition accuracy relative to their non-learned counterparts, the SUM (average of four kernels), the CORR and PMK kernels.

the learned CORR kernel: we achieve 61.0% accuracy for $T = 15$ and 69.6% accuracy for $T = 30$. Our learned PMK achieves 52.2% accuracy for $T = 15$ and 62.1% accuracy for $T = 30$. Similarly, our learned SUM kernel achieves 73.7% accuracy for $T = 15$. Figure 3 (b) specifically shows the comparison of the original baseline kernels for NN classification. The plot reveals gains in 1-NN classification accuracy; notably, our learned kernels with simple NN classification also outperform the baseline kernels when used with SVMs (Zhang et al., 2006; Grauman and Darrell, 2007).

6.4 USPS Digits

Finally, we qualitatively evaluate our dimensionality reduction method (see Section 5.4) on the USPS digits data set. Here, we train our method using 100 examples to learn a mapping to two dimensions, that is, a rank-2 matrix W. For the baseline kernel, we use the data-dependent kernel function proposed by Sindhwani et al. (2005) that also accounts for the manifold structure of the data within the kernel function. We then embed 2000 (unseen) test examples into two dimensions using our learned low-rank transformation. Figure 2 (c) shows the embedding obtained by our Trace-SSIKDR method, while Figure 2 (d) shows the embedding obtained by the kernel-PCA algorithm. Each point is color coded according to the underlying digit. Note that our method is able to separate out seven of the digits reasonably well, while kernel-PCA is able to separate out only three of the digits.

7. Conclusions

In this paper, we considered the general problem of learning a linear transformation of the input data and applied it to the problems of metric and kernel learning, with a focus on establishing connections between the two problems. We showed that the LogDet divergence is a useful loss for learning a linear transformation over very high-dimensional data, as the algorithm can easily be generalized to work in kernel space, and proposed an algorithm based on Bregman projections to learn a kernel function over the data points efficiently under this loss. We also showed that our learned metric can be restricted to a small dimensional basis efficiently, thereby permitting scalability of our method to large data sets with high-dimensional feature spaces. Then we considered how to generalize this result to a larger class of convex loss functions for learning the metric/kernel using a linear transformation of the data. We proved that many loss functions can lead to efficient kernel *function* learning, though the resulting optimizations may be more expensive to solve than the simpler LogDet formulation. A key consequence of our analysis is that a number of existing approaches for Mahalanobis metric learning may be applied in kernel space using our kernel learning formulation. Finally, we presented several experiments on benchmark data, high-dimensional vision, and text classification problems as well as a semi-supervised kernel dimensionality reduction problem, demonstrating our method compared to several existing state-of-the-art techniques.

There are several potential directions for future work. To facilitate even larger data sets than the ones considered in this paper, online learning methods are one promising research direction; in Jain et al. (2008), an online learning algorithm was proposed based on LogDet regularization, and this remains a part of our ongoing efforts. Recently, there has been some interest in learning multiple local metrics over the data; Weinberger and Saul (2008) considered this problem. We plan to explore this setting with the LogDet divergence, with a focus on scalability to very large data sets.

Acknowledgments

This research was supported by NSF grant CCF-0728879. We would like to acknowledge Suvrit Sra for various helpful discussions and anonymous reviewers for various helpful suggestions.

References

A. Argyriou, C. A. Micchelli, and M. Pontil. On spectral learning. *Journal of Machine Learning Research (JMLR)*, 11:935–953, 2010.

S. Arora and S. Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 227–236, 2007.

A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research (JMLR)*, 6:937–965, 2005.

S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2004.

Y. Bengio, O. Delalleau, N. Le Roux, J. Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Computation*, 16(10):2197–2219, 2004.

A. C. Berg and J. Malik. Geometric blur for template matching. In *Proccedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 607–614, 2001.

J. Cai, E. J. Candes, and Z. Shen. A singular value thresholding algorithm for matrix completion. Arxiv:0810.3286, 2008.

Caltech-101 Data Set. *http://www.vision.caltech.edu/Image_Datasets/Caltech101/*, 2004.

R. Chatpatanasiri, T. Korsrilabutr, P. Tangchanachaianan, and B. Kijsirikul. A new kernelization framework for Mahalanobis distance learning algorithms. *Neurocomputing*, 73(10–12):1570–1579, 2010.

S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proccedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

Classic3 Data Set. *ftp.cs.cornell.edu/pub/smart*, 2008.

CMU 20-Newsgroups Data Set. *http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.html*, 2008.

N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, 2001.

J. V. Davis and I. S. Dhillon. Structured metric learning for high dimensional problems. In *Proceedings of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 195–203, 2008.

J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proccedings of the International Conference on Machine Learning (ICML)*, pages 209–216, 2007.

S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

R. Fletcher. A new variational result for quasi-Newton formulae. *SIAM Journal on Optimization*, 1 (1), 1991.

A. Globerson and S. T. Roweis. Metric learning by collapsing classes. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, 2005.

J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood component analysis. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, 2004.

K. Grauman and T. Darrell. The Pyramid Match Kernel: Efficient learning with sets of features. *Journal of Machine Learning Research (JMLR)*, 8:725–760, April 2007.

M. Groschel, L. Lovasz, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 1988.

J. Ha, C. J. Rossbach, J. V. Davis, I. Roy, H. E. Ramadan, D. E. Porter, D. L. Chen, and E. Witchel. Improved error reporting for software that uses black-box components. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 101–111, 2007.

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

P. Jain, B. Kulis, I. S. Dhillon, and K. Grauman. Online metric learning and fast similarity search. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, pages 761–768, 2008.

P. Jain, B. Kulis, and I. S. Dhillon. Inductive regularized learning of kernel functions. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, 2010.

W. James and C. Stein. Estimation with quadratic loss. In *Fourth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 361–379. Univ. of California Press, 1961.

B. Kulis, M. Sustik, and I. S. Dhillon. Learning low-rank kernel matrices. In *Proccedings of the International Conference on Machine Learning (ICML)*, pages 505–512, 2006.

B. Kulis, M. Sustik, and I. Dhillon. Low-rank kernel learning with Bregman matrix divergences. *Journal of Machine Learning Research*, 2008.

B. Kulis, P. Jain, and K. Grauman. Fast similarity search for learned metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(12):2143–2157, 2009a.

B. Kulis, S. Sra, and I. S. Dhillon. Convex perturbations for scalable semidefinite programming. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009b.

J. T. Kwok and I. W. Tsang. Learning with idealized kernels. In *Proccedings of the International Conference on Machine Learning (ICML)*, 2003.

G. R. G. Lanckriet, N. Cristianini, P. L. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research (JMLR)*, 5:27–72, 2004.

S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proccedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2169–2178, 2006.

G. Lebanon. Metric learning for text documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(4):497–508, 2006. ISSN 0162-8828.

M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

C. S. Ong, A. J. Smola, and R. C. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research (JMLR)*, 6:1043–1071, 2005.

B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum rank solutions to linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, 2003.

M. Seeger. Cross-validation optimization for large scale hierarchical classification kernel methods. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1233–1240, 2006.

S. Shalev-Shwartz, Y. Singer, and A. Y. Ng. Online and batch learning of pseudo-metrics. In *Proccedings of the International Conference on Machine Learning (ICML)*, 2004.

V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *Proccedings of the International Conference on Machine Learning (ICML)*, pages 824–831, 2005.

K. Tsuda, G. Rätsch, and M. K. Warmuth. Matrix exponentiated gradient updates for on-line learning and Bregman projection. *Journal of Machine Learning Research (JMLR)*, 6:995–1018, 2005.

M. K. Warmuth and D. Kuzmin. Randomized online PCA algorithms with regret bounds that are logarithmic in the dimension. *Journal of Machine Learning Research*, 9:2287–2320, 2008.

K. Q. Weinberger and L. K. Saul. Fast solvers and efficient implementations for distance metric learning. In *Proccedings of the International Conference on Machine Learning (ICML)*, 2008.

K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, 2005.

E. P. Xing, A. Y. Ng, M. I. Jordan, and S. J. Russell. Distance metric learning with application to clustering with side-information. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, pages 505–512, 2002.

H. Zhang, A. C. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *Proccedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2126–2136, 2006.

X. Zhu, J. Kandola, Z. Ghahramani, and J. Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, volume 17, pages 1641–1648, 2005.