Information-Theoretic Metric Learning

Jason V. Davis JDAVIS@CS.UTEXAS.EDU
Brian Kulis KULIS@CS.UTEXAS.EDU
Prateek Jain PJAIN@CS.UTEXAS.EDU
Suvrit Sra SUVRIT@CS.UTEXAS.EDU
Inderjit S. Dhillon INDERJIT@CS.UTEXAS.EDU
Dept. of Computer Science, University of Texas at Austin, Austin, TX 78712

Abstract

In this paper, we present an information-theoretic approach to learning a Mahalanobis distance function. We formulate the problem as that of minimizing the differential relative entropy between two multivariate Gaussians under constraints on the distance function. We express this problem as a particular Bregman optimization problem—that of minimizing the LogDet divergence subject to linear constraints. Our resulting algorithm has several advantages over existing methods. First, our method can handle a wide variety of constraints and can optionally incorporate a prior on the distance function. Second, it is fast and scalable. Unlike most existing methods, no eigenvalue computations or semi-definite programming are required. We also present an online version and derive regret bounds for the resulting algorithm. Finally, we evaluate our method on a recent error reporting system for software called Clarify, in the context of metric learning for nearest neighbor classification, as well as on standard data sets.

1. Introduction

Selecting an appropriate distance measure (or metric) is fundamental to many learning algorithms such as k-means, nearest neighbor searches, and others. However, choosing such a measure is highly problem-specific and ultimately dictates the success—or failure—of the learning algorithm. To this end, there have been several recent approaches that attempt to learn distance functions, e.g., (Weinberger et al., 2005; Xing et al., 2002; Globerson & Roweis, 2005; Shalev-Shwartz et al., 2004). These methods work by exploiting distance information that is intrinsically available in many learning settings. For example, in the problem of semi-supervised clustering, points are constrained to be either similar (i.e., the distance between them should be relatively small) or dissimilar (the distance should be larger). In information retrieval settings, constraints between pairs of distances can be gathered from click-through feedback. In fully supervised settings, constraints can be inferred so that points in the same class have smaller distances to each other than to points in different classes.

While existing algorithms for metric learning have been shown to perform well across various learning tasks, each fails to satisfy some basic requirement. First, a metric learning algorithm should be sufficiently flexible to support the variety of constraints realized across different learning paradigms. Second, the algorithm must be able to learn a distance function that generalizes well to unseen test data. Finally, the algorithm should be fast and scalable.

In this paper, we propose a novel approach to learning a class of distance functions—namely, Mahalanobis distances—that have been shown to possess good generalization performance. The Mahalanobis distance generalizes the standard Euclidean distance by admitting arbitrary linear scalings and rotations of the feature space. We model the problem in an information-theoretic setting by leveraging the relationship between the multivariate Gaussian distribution and the set of Mahalanobis distances. We translate the problem of learning an optimal distance metric to that of learning the optimal Gaussian with respect to an entropic objective. In fact, a special case of our formulation can be viewed as a maximum entropy objective: maximize the differential entropy of a multivariate Gaussian subject to constraints on the associated Mahalanobis distance.

Our formulation is quite general: we can accommodate a range of constraints, including similarity or dissimilarity constraints, and relations between pairs of distances. We can also incorporate prior information regarding the distance function itself. For some problems, standard Euclidean distance may work well. In others, the Mahalanobis distance using the inverse of the sample covariance may yield reasonable results. In such cases, our formulation finds a distance function that is 'closest' to an initial distance function while also satisfying the given constraints.

Appearing in *Proceedings of the 24^{th} International Conference on Machine Learning*, Corvallis, OR, 2007.

We show an interesting connection of our metric learning problem to a recently proposed low-rank kernel learning problem (Kulis et al., 2006). In the latter problem a low-rank kernel K is learned that satisfies a set of given distance constraints by minimizing the LogDet divergence to a given initial kernel K_0. This allows our metric learning algorithm to be kernelized, resulting in an optimization over a larger class of non-linear distance functions. Algorithmically, the connection also implies that the problem can be solved efficiently: it was shown that the kernel learning problem can be optimized using an iterative optimization procedure with cost $O(cd^2)$ per iteration, where c is the number of distance constraints, and d is the dimensionality of the data. In particular, this method does not require costly eigenvalue computations or semi-definite programming. We also present an online version of the algorithm and derive associated regret bounds.

To demonstrate our algorithm's ability to learn a distance function that generalizes well to unseen points, we compare it to existing state-of-the-art metric learning algorithms. We apply the algorithms to Clarify, a recently developed system that classifies software errors using machine learning (Ha et al., 2007). In this domain, we show that our algorithm effectively learns a metric for the problem of nearest neighbor software support. Furthermore, on standard UCI datasets, we show that our algorithm consistently equals or outperforms the best existing methods when used to learn a distance function for k-NN classification.

2. Related Work

Most of the existing work in metric learning relies on learning a Mahalanobis distance, which has been found to be a sufficiently powerful class of metrics that work on many real-world problems. Earlier work by (Xing et al., 2002) uses a semidefinite programming formulation under similarity and dissimilarity constraints. More recently, (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 combination of sub-gradient descent and alternating projections. (Globerson & Roweis, 2005) proceed to learn a metric in the fully supervised setting. Their formulation seeks to 'collapse classes' by constraining within class distances to be zero while maximizing the between class distances. While each of these algorithms was shown to yield excellent classification performance, 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, an operation that is cubic in the dimensionality of the data.

Other notable work wherein the authors present methods for learning Mahalanobis metrics includes (Shalev-Shwartz et al., 2004) (online metric learning), Relevant Components Analysis (RCA) (Shental et al., 2002) (similar to discriminant analysis), locally-adaptive discriminative methods (Hastie & Tibshirani, 1996), and learning from relative comparisons (Schutz & Joachims, 2003).

Non-Mahalanobis based metric learning methods have also been proposed, though these methods usually suffer from suboptimal performance, non-convexity, or computational complexity. Some example methods include neighborhood component analysis (NCA) (Goldberger et al., 2004) that learns a distance metric specifically for nearest-neighbor based classification; convolutional neural net based methods of (Chopra et al., 2005); and a general Riemannian metric learning method (Lebanon, 2006).

3. Problem Formulation

Given a set of n points $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ in $\mathbb{R}^d$, we seek a positive definite matrix A which parameterizes the (squared) Mahalanobis distance.

$$d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T A(\boldsymbol{x}_i - \boldsymbol{x}_j). \qquad (3.1)$$

We assume that prior knowledge is known regarding interpoint distances. Consider relationships constraining the similarity or dissimilarity between pairs of points. Two points are similar if the Mahalanobis distance between them is smaller than a given upper bound, i.e., $d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq u$ for a relatively small value of u. Similarly, two points are dissimilar if $d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \ell$ for sufficiently large ℓ. Such constraints are typically inputs for many semi-supervised learning problems, and can also be readily inferred in a classification setting where class labels are known for each instance: distances between points in the same class can be constrained as similar, and points in different classes can be constrained as dissimilar.

Given a set of interpoint distance constraints as described above, our problem is to learn a positive-definite matrix A that parameterizes the corresponding Mahalanobis distance (3.1). Typically, this learned distance function is used to improve the accuracy of a k-nearest neighbor classifier, or to incorporate semi-supervision into a distance-based clustering algorithm. In many settings, prior information about the Mahalanobis distance function itself is known. In settings where data is Gaussian, parameterizing the distance function by the inverse of the sample covariance may be appropriate. In other domains, squared Euclidean distance (i.e. the Mahalanobis distance corresponding to the identity matrix) may work well empirically. Thus, we regularize the Mahalanobis matrix A to be as close as possible to a given Mahalanobis distance function, parameterized by A_0.

We now quantify the measure of "closeness" between A and A_0 via a natural information-theoretic approach. There exists a simple bijection (up to a scaling function) between the set of Mahalanobis distances and the set of equal-mean multivariate Gaussian distributions (without loss of generality, we can assume the Gaussians have mean $\boldsymbol{\mu}$). Given a Mahalanobis distance parameterized by A, we express its corresponding multivariate Gaussian as $p(\boldsymbol{x}; A) = \frac{1}{Z}\exp\left(-\frac{1}{2}d_A(\boldsymbol{x}, \boldsymbol{\mu})\right)$, where Z is a normalizing constant and A^{-1} is the covariance of the distribution. Using this bijection, we measure the distance between two Mahalanobis distance functions parameterized by A_0 and A by the (differential) relative entropy between their corresponding multivariate Gaussians:

$$\mathrm{KL}(p(\boldsymbol{x}; A_o)\|p(\boldsymbol{x}; A)) = \int p(\boldsymbol{x}; A_0)\log\frac{p(\boldsymbol{x}; A_0)}{p(\boldsymbol{x}; A)}\,d\boldsymbol{x}. \tag{3.2}$$

The distance (3.2) provides a well-founded measure of "closeness" between two Mahalanobis distance functions and forms the basis of our problem given below.

Given pairs of similar points S and pairs of dissimilar points D, our distance metric learning problem is

$$\begin{array}{lll} \min\limits_{A} & \mathrm{KL}(p(\boldsymbol{x}; A_0)\|p(\boldsymbol{x}; A)) & \\ \text{subject to} & d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq u & (i,j) \in S, \\ & d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \ell & (i,j) \in D. \end{array} \tag{3.3}$$

In the above formulation, we consider simple distance constraints for similar and dissimilar points; however, it is straightforward to incorporate other constraints. For example, (Schutz & Joachims, 2003) consider a formulation where the distance metric is learned subject to relative nearness constraints (as in, the distance between i and j is closer than the distance between i and k). Our approach can be easily adapted to handle this setting. In fact, it is possible to incorporate arbitrary linear constraints into our framework in a straightforward manner. For simplicity, we present the algorithm under the simple distance constraints given above.

4. Algorithm

In this section, we first show that our information-theoretic objective (3.3) can be expressed as a particular type of Bregman divergence, which allows us to adapt Bregman's method (Censor & Zenios, 1997) to solve the metric learning problem. We then show a surprising equivalence to a recently-proposed low-rank kernel learning problem (Kulis et al., 2006), allowing kernelization of the algorithm.

4.1. Metric Learning as LogDet Optimization

The LogDet divergence is a Bregman matrix divergence generated by the convex function $\phi(X) = -\log\det X$ defined over the cone of positive-definite matrices, and it equals (for $n \times n$ matrices A, A_0)

$$D_{\ell\mathrm{d}}(A, A_0) = \mathrm{tr}(AA_0^{-1}) - \log\det(AA_0^{-1}) - n. \tag{4.1}$$

It has been shown that the differential relative entropy between two multivariate Gaussians can be expressed as the convex combination of a Mahalanobis distance between mean vectors and the LogDet divergence between the covariance matrices (Davis & Dhillon, 2006). Assuming the means of the Gaussians to be the same, we have,

$$\begin{aligned} \mathrm{KL}(p(\boldsymbol{x}; A_0)\|p(\boldsymbol{x}; A)) &= \frac{1}{2}D_{\ell\mathrm{d}}(A_0^{-1}, A^{-1}) \\ &= \frac{1}{2}D_{\ell\mathrm{d}}(A, A_0), \end{aligned} \tag{4.2}$$

where the second line follows from definition (4.1).

The LogDet divergence is also known as Stein's loss, having originated in the work of (James & Stein, 1961). It can be shown that Stein's loss is the unique *scale invariant* loss-function for which the uniform minimum variance unbiased estimator is also a minimum risk equivariant estimator (Lehmann & Casella, 2003). In the context of metric learning, the scale invariance implies that the divergence (4.1) remains invariant under any scaling of the feature space. The result can be further generalized to invariance under any invertible linear transformation S, since

$$D_{\ell\mathrm{d}}(S^TAS, S^TBS) = D_{\ell\mathrm{d}}(A, B). \tag{4.3}$$

We can exploit the equivalence in (4.2) to express the distance metric learning problem (3.3) as the following LogDet optimization problem:

$$\begin{array}{lll} \min\limits_{A \succeq 0} & D_{\ell\mathrm{d}}(A, A_0) & \\ \text{s.t.} & \mathrm{tr}(A(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T) \leq u & (i,j) \in S, \\ & \mathrm{tr}(A(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T) \geq \ell & (i,j) \in D, \end{array} \tag{4.4}$$

Note that the distance constraints on $d_A(\boldsymbol{x}_i, \boldsymbol{x}_j)$ translate into the above linear constraints on A.

In some cases, there may not exist a feasible solution to (4.4). To prevent such a scenario from occurring, we incorporate *slack variables* into the formulation to guarantee the existence of a feasible A. Let $c(i,j)$ denote the index of the (i,j)-th constraint, and let $\boldsymbol{\xi}$ be a vector of slack variables, initialized to $\boldsymbol{\xi}_0$ (whose components equal u for similarity constraints and ℓ for dissimilarity constraints). Then we can pose the following optimization problem:

$$\begin{array}{lll} \min\limits_{A \succeq 0, \boldsymbol{\xi}} & D_{\ell\mathrm{d}}(A, A_0) + \gamma \cdot D_{\ell\mathrm{d}}(\mathrm{diag}(\boldsymbol{\xi}), \mathrm{diag}(\boldsymbol{\xi}_0)) & \\ \text{s. t.} & \mathrm{tr}(A(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T) \leq \xi_{c(i,j)} & (i,j) \in S, \\ & \mathrm{tr}(A(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T) \geq \xi_{c(i,j)} & (i,j) \in D, \end{array} \tag{4.5}$$

Algorithm 1 Information-theoretic metric learning

```
Input: $X$: input $d \times n$ matrix, $S$: set of similar pairs
    $D$: set of dissimilar pairs, $u, \ell$: distance thresholds
    $A_0$: input Mahalanobis matrix, $\gamma$: slack
    parameter, $c$: constraint index function
Output: $A$: output Mahalanobis matrix
1. $A \leftarrow A_0$, $\lambda_{ij} \leftarrow 0 \; \forall \, i, j$
2. $\xi_{c(i,j)} \leftarrow u$ for $(i,j) \in S$; otherwise $\xi_{c(i,j)} \leftarrow \ell$
3. repeat
    3.1. Pick a constraint $(i,j) \in S$ or $(i,j) \in D$
    3.2. $p \leftarrow (\boldsymbol{x}_i - \boldsymbol{x}_j)^T A (\boldsymbol{x}_i - \boldsymbol{x}_j)$
    3.3. $\delta \leftarrow 1$ if $(i,j) \in S$, $-1$ otherwise
    3.4. $\alpha \leftarrow \min\left(\lambda_{ij}, \frac{\delta}{2}\left(\frac{1}{p} - \frac{\gamma}{\xi_{c(i,j)}}\right)\right)$
    3.5. $\beta \leftarrow \delta\alpha/(1 - \delta\alpha p)$
    3.6. $\xi_{c(i,j)} \leftarrow \gamma\xi_{c(i,j)}/(\gamma + \delta\alpha\xi_{c(i,j)})$
    3.7. $\lambda_{ij} \leftarrow \lambda_{ij} - \alpha$
    3.8. $A \leftarrow A + \beta A(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T A$
4. until convergence
return $A$
```

The parameter γ controls the tradeoff between satisfying the constraints and minimizing $D_{\ell d}(A, A_0)$.

To solve the optimization problem (4.5), we extend the methods from (Kulis et al., 2006). The optimization method which forms the basis for the algorithm repeatedly computes Bregman projections—that is, projections of the current solution onto a single constraint. This projection is performed via the update

$$A_{t+1} = A_t + \beta A_t(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T A_t, \qquad (4.6)$$

where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are the constrained data points, and β is the projection parameter (Lagrange multiplier corresponding to the constraint) computed by the algorithm. Each constraint projection costs $O(d^2)$, and so a single iteration of looping through all constraints costs $O(cd^2)$. We stress that no eigen-decomposition is required in the algorithm. The resulting algorithm is given as Algorithm 1. The inputs to the algorithm are the starting Mahalanobis matrix A_0, the constraint data, and the slack parameter γ. If necessary, the projections can be computed efficiently over a factorization W of the Mahalanobis matrix, such that $A = W^T W$.

4.2. Connection to Low-Rank Kernel Learning

The kernel learning problem posed in (Kulis et al., 2006) seeks to optimize a kernel K subject to constraints on the pairwise distances between points within the kernel. This is quite similar to our proposed metric learning problem, where points between distances are constrained directly. In this section, we present an equivalence between the two models, thus allowing for kernelization of the metric learning problem. The kernel learning problem posed by (Kulis et al., 2006) seeks to optimize a kernel K subject to a set of linear constraints while minimizing the LogDet divergence to a specified kernel K_0. Given $X = [\boldsymbol{x}_1 \, \boldsymbol{x}_2 \, ... \, \boldsymbol{x}_n]$, and the input $n \times n$ kernel matrix $K_0 = X^T A_0 X$, the optimization problem is:

$$\begin{array}{lll} \min\limits_{K} & D_{\ell d}(K, K_0) & \\ \text{subject to} & K_{ii} + K_{jj} - 2K_{ij} \le u & (i,j) \in S, \\ & K_{ii} + K_{jj} - 2K_{ij} \ge \ell & (i,j) \in D, \\ & K \succeq 0. & \end{array} \qquad (4.7)$$

In addition to being convex in the first argument, the LogDet divergence between two matrices is finite if and only if their range spaces are the same (Kulis et al., 2006). Thus, the learned matrix K can be written as a kernel $X^T A X$, for some $(d \times d)$ positive definite matrix A. The results below can be easily generalized to incorporate slack variables in (4.7).

First we establish that the feasible solutions to (3.3) coincide with the feasible solutions to (4.7).

Lemma 1. *Given that $K = X^T A X$, A is feasible for (3.3) if and only if K is feasible for (4.7).*

Proof. The expression $K_{ii} + K_{jj} - 2K_{ij}$ can be written as $(\boldsymbol{e}_i - \boldsymbol{e}_j)^T K (\boldsymbol{e}_i - \boldsymbol{e}_j)$, or equivalently $(\boldsymbol{x}_i - \boldsymbol{x}_j)^T A (\boldsymbol{x}_i - \boldsymbol{x}_j) = d_A(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Thus, if we have a kernel matrix K satisfying constraints of the form $K_{ii} + K_{jj} - 2K_{ij} \le u$ or $K_{ii} + K_{jj} - 2K_{ij} \ge \ell$, we equivalently have a matrix A satisfying $d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \le u$ or $d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \ge \ell$. □

We can now show an explicit relationship between the optimal solution to (3.3) and (4.7).

Theorem 1. *Let A^* be the optimal solution to (3.3) and K^* be the optimal solution to (4.7). Then $K^* = X^T A^* X$.*

Proof. We give a constructive proof for the theorem. The Bregman projection update for (4.4) is expressed as

$$A_{t+1} = A_t + \beta A_t(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T A_t. \qquad (4.8)$$

Similary, the Bregman update for (4.7) is expressed as

$$K_{t+1} = K_t + \beta K_t(\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^T K_t. \qquad (4.9)$$

It is straightforward to prove that the value of β is the same for (4.9) and (4.8). We can inductively prove that at each iteration t, updates K_t and A_t satisfy $K_t = X^T A_t X$. At the first step, $K_0 = X^T A_0 X$, so the base case trivially holds. Now, assume that $K_t = X^T A_t X$; by the Bregman projection update, K_{t+1}

$$\begin{aligned} &= X^T A_t X + \beta X^T A_t X(\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^T X^T A_t X \\ &= X^T A_t X + \beta X^T A_t(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T A_t X \\ &= X^T(A_t + \beta A_t(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T A_t^T)X. \end{aligned}$$

Comparing with (4.8), we see that $K_{t+1} = X^T A_{t+1} X$. Since the method of Bregman projections converges to the optimal A^* and K^* of (3.3) and (4.7), respectively (Censor & Zenios, 1997), we have $K^* = X^T A^* X$. Hence, the metric learning (3.3) and the kernel learning (4.7) problems are equivalent. □

We have proven that the information-theoretic metric learning problem is related to a low-rank kernel learning problem. We can easily modify Algorithm 1 to optimize for K—this is necessary in order to kernelize the algorithm. As input, the algorithm provides K_0 instead of A_0, the value of p is computed as $K_{ii} + K_{jj} - 2K_{ij}$, the projection is performed using (4.9), and the output is K.

4.3. Kernelizing the Algorithm

We now consider kernelizing our metric learning algorithm. In this section, we assume that $A_0 = I$; that is, the maximum entropy formulation that regularizes to the baseline Euclidean distance. Kernelizing for other choices of A_0 is possible, but not presented. If $A_0 = I$, the corresponding K_0 from the low-rank kernel learning problem is $K_0 = X^T X$, the Gram matrix of the inputs. If instead of an explicit representation X of our data points, we have as input a kernel function $\kappa(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y})$, along with the associated kernel matrix K_0 over the training points, a natural question to ask is whether we can evaluate the learned metric on new points in the kernel space. This requires the computation of

$$\begin{aligned} d_A\big(\phi(\boldsymbol{x}), \phi(\boldsymbol{y})\big) &= \big(\phi(\boldsymbol{x}) - \phi(\boldsymbol{y})\big)^T A \big(\phi(\boldsymbol{x}) - \phi(\boldsymbol{y})\big) \\ &= \phi(\boldsymbol{x})^T A \phi(\boldsymbol{x}) - 2\phi(\boldsymbol{x})^T A \phi(\boldsymbol{y}) + \phi(\boldsymbol{y})^T A \phi(\boldsymbol{y}). \end{aligned}$$

We now define a new kernel function $\tilde{\kappa}(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T A \phi(\boldsymbol{y})$. The ability to generalize to unseen data points reduces to the ability to compute $\tilde{\kappa}(\boldsymbol{x}, \boldsymbol{y})$. Note that A represents an operator in a Hilbert space, and its size is equal to the dimensionality of $\phi(\boldsymbol{x})$, which can potentially be infinite (if the original kernel function is the Gaussian kernel, for example).

Even though we cannot explicitly compute A, it is still possible to compute $\tilde{\kappa}(\boldsymbol{x}, \boldsymbol{y})$. As $A_0 = I$, we can recursively "unroll" the learned A matrix so that it is of the form

$$A = I + \sum_{i,j} \sigma_{ij} \phi(\boldsymbol{x}_i) \phi(\boldsymbol{x}_j)^T.$$

This follows by expanding Equation (4.6) down to I. The new kernel function is therefore computed as

$$\tilde{\kappa}(\boldsymbol{x}, \boldsymbol{y}) = \kappa(\boldsymbol{x}, \boldsymbol{y}) + \sum_{i,j} \sigma_{ij} \kappa(\boldsymbol{x}, \boldsymbol{x}_i) \kappa(\boldsymbol{x}_j, \boldsymbol{y}),$$

and is a function of the original kernel function κ and the σ_{ij} coefficients. The σ_{ij} coefficients may be updated while minimizing $D_{\ell \mathrm{d}}(K, K_0)$ without affecting the asymptotic running time of the algorithm; in other words, by optimizing the low-rank kernel learning problem (4.7) for K, we can obtain the necessary coefficients σ_{ij} for evaluation of $\tilde{\kappa}(\boldsymbol{x}, \boldsymbol{y})$. This leads to a method for finding the nearest neighbor of a new data point in the kernel space under the learned metric which can be performed in $O(n^2)$ total time. Details are omitted due to lack of space.

5. Online Metric Learning

In this section, we describe an *online* algorithm for metric learning and prove bounds on the total loss when using LogDet divergence as the regularizer.

5.1. Problem Formulation

As in batch metric learning, the algorithm receives pairs of points; however, unlike in the batch scenario, the online algorithm receives the "target" distance for these points (as opposed to just upper and lower bounds on the target distance). Before receiving the target distance, the algorithm uses its current Mahalanobis matrix to predict the distance between the given pair of points at each step. This formulation is standard in many online regression settings (Kivinen & Warmuth, 1997).

More formally, assume the algorithm receives an instance $(\boldsymbol{x}_t, \boldsymbol{y}_t, d_t)$ at time step t, and predicts $\hat{d}_t = d_{A_t}(\boldsymbol{x}_t, \boldsymbol{y}_t)$ using the current model A_t. The loss associated with this prediction is measured by $l_t(A_t) = (d_t - \hat{d}_t)^2$, where d_t is the "true" (or target) distance between $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$. After incurring the loss, the algorithm updates A_t to the new model A_{t+1} and its total loss is given by $\sum_t l_t(A_t)$.

If there is no correlation between input points and their target distances, then a learning algorithm could incur unbounded loss. Hence, a reasonable goal is to compare the performance of an online learning algorithm with the best possible offline algorithm. Given a T-trial sequence $S = \{(\boldsymbol{x}_1, \boldsymbol{y}_1, d_1), (\boldsymbol{x}_2, \boldsymbol{y}_2, d_2), \ldots, (\boldsymbol{x}_T, \boldsymbol{y}_T, d_T)\}$, let the optimal offline solution be given by

$$A^* = \underset{A \succeq 0}{\operatorname{argmin}} \quad \sum_{t=1}^{T} l_t(A).$$

Typically, the goal of an online algorithm is to bound its total loss in comparison to the loss obtained by A^*.

A common approach for online learning is to solve the following regularized optimization problem at each time step (Kivinen & Warmuth, 1997):

$$\min_{A \succeq 0} \quad f(A) = \overbrace{D(A, A_t)}^{\text{Regularization Term}} \; + \eta_t \overbrace{l_t(A)}^{\text{Loss Term}}, \qquad (5.1)$$

where η_t is the learning rate at the t-th step, and $D(A, A_t)$ is measures the divergence between the new Mahalanobis

Algorithm 2 Online Metric Learning (OML) Algorithm

```
Initialize: $\eta_0 \leftarrow \frac{1}{8}$, $A_0 \leftarrow \frac{1}{n}I$
Prediction: Given $(\boldsymbol{x}_t, \boldsymbol{y}_t)$, predict $\hat{d}_t = d_{A_t}(\boldsymbol{x}_t, \boldsymbol{y}_t)$ .
Update: Upon receiving "true" $d_t$, update model as
    $A_{t+1} \leftarrow A_t - \frac{2\eta_t(\hat{d}_t - d_t)A_t(\boldsymbol{x}_t - \boldsymbol{y}_t)(\boldsymbol{x}_t - \boldsymbol{y}_t)^T A_t}{1 + 2\eta_t(\hat{d}_t - d_t)(\boldsymbol{x}_t - \boldsymbol{y}_t)^T A_t(\boldsymbol{x}_t - \boldsymbol{y}_t)}$,
    where $\eta_t = \eta_0$ if $\hat{d}_t - d_t \geq 0$; otherwise, $\eta_t = \min\left\{\eta_0, \frac{1}{2(d_t - \hat{d}_t)}\left(\frac{1}{(\boldsymbol{x}_t - \boldsymbol{y}_t)^T(I + (A_t^{-1} - I)^{-1})(\boldsymbol{x}_t - \boldsymbol{y}_t)}\right)\right\}$.
```

matrix A and the current matrix A_t. In (5.1), the regularization term favors Mahalanobis matrices that are "close" to the current model A_t, representing a tendency towards conservativeness. On the other hand, the loss term is minimized when A is updated to exactly satisfy the target distance specified at the current time step. Hence, the loss term has a tendency to satisfy target distances for recent examples. The tradeoff between regularization and loss is handled by the learning rate η_t, which is a critical parameter for many online learning problems.

As in batch metric learning, we select $D(A, A_t) = D_{\ell\text{d}}(A, A_t)$ as the regularizer for (5.1):

$$A_{t+1} = \operatorname*{argmin}_{A} D_{\ell\text{d}}(A, A_t) + \eta_t(d_t - \hat{d}_t)^2.$$

To our knowledge, no relative loss bounds have been proven for the above problem. In fact, we know of no existing loss bounds for any LogDet-based online algorithms (Tsuda et al., 2005). We present below a novel algorithm with guaranteed bound on the regret. Our algorithm uses gradient descent, but it adapts the learning rate according to the input data to maintain positive-definiteness of A.

5.2. Algorithm and Analysis

The online metric learning (OML) algorithm is shown as Algorithm 2. Note that $(A_t^{-1} - I)^{-1}$ can be obtained from $(A_{t-1}^{-1} - I)^{-1}$ by using the Sherman-Morrison-Woodbury formula. Thus, the overall complexity of each iteration of the OML algorithm is $O(n^2)$, which is optimal.

We now show that the total loss incurred by OML is bounded with respect to the minimum loss incurred by any batch learning algorithm. Let $L_{\text{OML}} = \sum_t l_t(A_t)$ be the loss obtained by the online algorithm, and let $L_{A^*} = \sum_t l_t(A^*)$ be the loss obtained by the offline algorithm. Without loss of generality we can assume that the input data is scaled, so that $\|\boldsymbol{x} - \boldsymbol{y}\|^2 \leq R^2$ for fixed R and for all $\boldsymbol{x}$ and $\boldsymbol{y}$. One of the key steps in guaranteeing an upper bound on the regret incurred by an online algorithm is to bound the loss incurred at each step in terms of the loss incurred by the optimal offline solution. Below, we state the result—the full proof can be found in (Jain et al., 2007).

Lemma 2 (Loss at one step)**.**

$$a_t(\hat{d}_t - d_t)^2 - b_t(d_t^* - d_t)^2 \leq D_{\ell d}(A^*, A_t) - D_{\ell d}(A^*, A_{t+1}),$$

where A^* is the optimal offline solution, $d_t^* = d_{A^*}(\boldsymbol{x}_t, \boldsymbol{y}_t)$, a_t, b_t are constants s.t. $0 \leq a_t \leq \eta_t$ and $b_t = \frac{\eta_t}{1 - 2\eta_t R^2}$.

A repeated use of Lemma 2 allows us to obtain a loss bound for the overall OML algorithm.

Theorem 2 (Online Metric Learning Loss Bound)**.**

$$L_{\mathit{OML}} \leq \frac{1}{4\eta_{min}R^2} L_{A^*} + \frac{1}{\eta_{min}} D_{\ell d}(A^*, I),$$

where L_{OML} and L_{A^*} are the losses incurred by OML and the optimal batch algorithm, respectively; $\eta_{min} = \min_t \eta_t$.

Proof. Adding the loss bound given by Lemma 2 over all the trials from $1 \leq t \leq T$, we obtain

$$\sum_{t=1}^{T} \eta_t(\hat{d}_t - d_t)^2 \leq \sum_{t=1}^{T} \frac{\eta_t}{1 - 2\eta_t R^2}(d_t^* - d_t)^2 + D_{\ell\text{d}}(A^*, I) - D_{\ell\text{d}}(A^*, A_T).$$

Thus we can conclude that,

$$L_{\text{OML}} \leq \frac{1}{4\eta_{min}R^2} L_{A^*} + \frac{1}{\eta_{min}} D_{\ell\text{d}}(A^*, I). \qquad \square$$

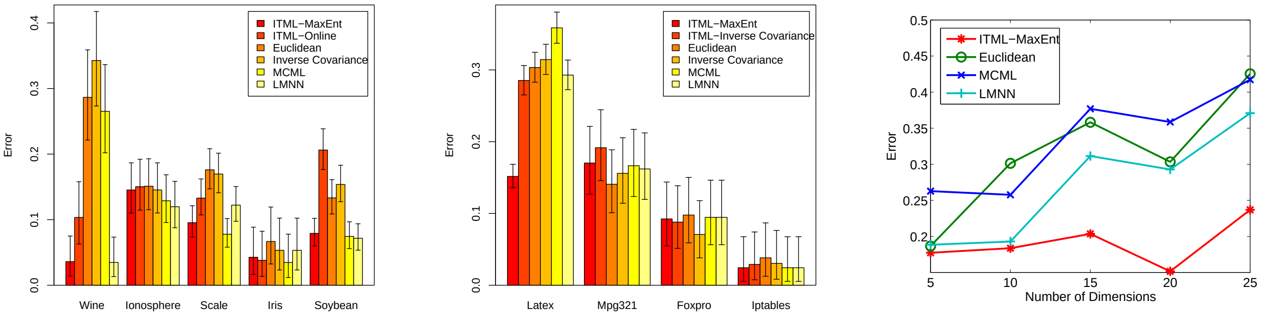
Note that this bound depends on η_{min}, which in turn depends on the input data. If the data is scaled properly then generally η_t will not change much at any time step, an observation that has been verified empirically.

6. Experiments

We compare our Information Theoretic Metric Learning algorithm (ITML) to existing methods across two applications: semi-supervised clustering and k-nearest neighbor (k-NN) classification.

We evaluate metric learning for k-NN classification via two-fold cross validation with $k = 4$. All results presented represent the average over 5 runs. Binomial confidence intervals are given at the 95% level.

To establish the lower and upper bounds of the right hand side of our constraints (ℓ and u in problem (3.3)), we use (respectively) the 5^{th} and 95^{th} percentiles of the observed distribution of distances between pairs of points within the dataset. To determine the constrained point pairs, we randomly choose $20c^2$ pairs, where c is the number of classes in the dataset. Pairs of points in the same class are constrained to be similar, and pairs with differing class labels are constrained to be dissimilar. Overall, we found the algorithm to be robust to these parameters. However, we did



(a) UCI Datasets (b) Clarify Datasets (c) Latex

Figure 1. Classification error rates for k-nearest neighbor classification via different learned metrics. We see in figures (a) and (b) that ITML-MaxEnt is the only algorithm to be optimal (within the 95% confidence intervals) across all datasets. ITML is also robust at learning metrics over higher dimensions. In (c), we see that the error rate for the Latex dataset stays relatively constant for ITML.

find that the variance between runs increased if the number of constraints used was too small (i.e., fewer than $10c^2$). The slack variable parameter, γ, is tuned using cross validation over the values $\{.01, .1, 1, 10\}$. Finally, the online algorithm is run for approximately 10^5 iterations.

In Figure 1(a), we compare ITML-MaxEnt (regularized to the identity matrix) and the online ITML algorithm against existing metric learning methods for k-NN classification. We use the squared Euclidean distance, $d(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^{\mathbf{T}}(\mathbf{x} - \mathbf{y})$ as a baseline method. We also use a Mahalanobis distance parameterized by the inverse of the sample covariance matrix. 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 (Globerson & Roweis, 2005) (MCML), and metric learning via Large Margin Nearest Neighbor (Weinberger et al., 2005) (LMNN). Consistent with existing work (Globerson & Roweis, 2005), we found the method of (Xing et al., 2002) to be very slow and inaccurate. Overall, ITML is the only algorithm to obtain the optimal error rate (within the specified 95% confidence intervals) across all datasets. For several datasets, the online version is competitive with the best metric learning algorithms. We also observed that the learning rate η remained fairly constant, yielding relatively small regret bounds (Theorem 2).

In addition to our evaluations on standard UCI datasets, we also evaluate 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 for (or is capable of) providing an error report. The system works by monitoring a set of predefined program features (the datasets 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 those with 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 datasets shown here are 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).

The dimensionality of the Clarify dataset can be quite large. However, it was shown (Ha et al., 2007) that high classification accuracy can be obtained by using a relatively small subset of available features. Thus, for each dataset, 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 above described procedure. We also evaluate the method ITML-Inverse Covariance, which regularizes to the inverse covariance matrix. Results are given in Figure 1(b). The ITML-MaxEnt algorithm yields significant gains for the Latex benchmark. Note that for datasets where Euclidean distance performs better than using the inverse covariance metric, the ITML-MaxEnt algorithm that normalizes to the standard Euclidean distance yields higher accuracy than that regularized to the inverse covariance matrix (ITML-Inverse Covariance). In general, for the Mpg321, Foxpro, and Iptables datasets, learned metrics yield only marginal gains over the baseline Euclidean distance measure.

Figure 1(c) shows the error rate for the Latex datasets with a varying number of features (the feature sets are again cho-

Table 1. Training time (in seconds) for the results presented in Figure 1(b).

Dataset	ITML-MaxEnt	MCML	LMNN
Latex	**0.0517**	19.8	0.538
Mpg321	**0.0808**	0.460	0.253
Foxpro	**0.0793**	0.152	0.189
Iptables	0.149	**0.0838**	4.19

Table 2. Unsupervised k-means clustering error, along with semi-supervised clustering error with 50 constraints.

Dataset	Unsupervised	ITML	HMRF-KMeans
Ionosphere	0.314	**0.113**	0.256
Digits-389	0.226	**0.175**	0.286

sen using the information gain criteria). We see here that ITML is surprisingly robust. Euclidean distance, MCML, and LMNN all achieve their best error rates for five dimensions. ITML, however, attains its lowest error rate of .15 at $d = 20$ dimensions.

In Table 1, we see that ITML generally learns metrics significantly faster than other metric learning algorithms. The implementations for MCML and LMNN were obtained from their respective authors. The timing tests were run on a dual processor 3.2 GHz Intel Xeon processor running Ubuntu Linux. Time given is in seconds and represents the average over 5 runs.

Finally, we present some semi-supervised clustering results. 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 dataset 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 2 provides results for the baseline k-means error, as well as semi-supervised clustering results with 50 constraints.

Acknowledgments: This research was supported by NSF grant CCF-0431257, NSF Career Award ACI-0093404, and NSF-ITR award IIS-0325116.

References

Basu, S., Bilenko, M., & Mooney, R. J. (2004). A Probabilistic Framework for Semi-Supervised Clustering. *Proc. 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining* (pp. 59–68).

Censor, Y., & Zenios, S. A. (1997). *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press.

Chopra, S., Hadsell, R., & LeCun, Y. (2005). Learning a Similarity Metric Discriminatively, with Application to Face Verification. *IEEE Conf. on Computer Vision and Pattern Recognition*.

Davis, J. V., & Dhillon, I. S. (2006). Differential Entropic Clustering of Multivariate Gaussians. *Adv. in Neural Inf. Proc. Sys. (NIPS)*.

Globerson, A., & Roweis, S. (2005). Metric Learning by Collapsing Classes. *Adv. in Neural Inf. Proc. Sys. (NIPS)*.

Goldberger, J., Roweis, S., Hinton, G., & Salakhutdinov, R. (2004). Neighbourhood Component Analysis. *Adv. in Neural Inf. Proc. Sys. (NIPS)*.

Ha, J., Rossbach, C., Davis, J., Roy, I., Chen, D., Ramadan, H., & Witchel, E. (2007). Improved Error Reporting for Software that Uses Black Box Components. *Programming Language Design and Implementation*. To appear.

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

Jain, P., Kulis, B., & Dhillon, I. S. (2007). *Online linear regression using burg entropy* (Technical Report TR-07-08). The Univ. of Texas at Austin, Dept. of Comp. Sci.

James, W., & Stein, C. (1961). Estimation with quadratic loss. In *Proc. fourth berkeley symposium on mathematical statistics and probability*, vol. 1, 361–379. Univ. of California Press.

Kivinen, J., & Warmuth, M. K. (1997). Exponentiated gradient versus gradient descent for linear predictors. *Inf. Comput.*, *132*, 1–63.

Kulis, B., Sustik, M., & Dhillon, I. S. (2006). Learning Low-rank Kernel Matrices. *Int. Conf. on Machine Learning (ICML)*.

Lebanon, G. (2006). Metric Learning for Text Documents. *Pattern Analysis and Machine Intelligence*, *28*, 497–508.

Lehmann, E. L., & Casella, G. (2003). *Theory of Point Estimation*. Springer. Second edition.

Schutz, M., & Joachims, T. (2003). Learning a Distance Metric from Relative Comparisons. *Adv. in Neural Inf. Proc. Sys. (NIPS)*.

Shalev-Shwartz, S., Singer, Y., & Ng, A. Y. (2004). Online and Batch Learning of Pseudo-Metrics. *Int. Conf. on Machine Learning (ICML)*.

Shental, N., Hertz, T., Weinshall, D., & Pavel, M. (2002). Adjustment learning and relevant component analysis. *Proc. of European Conf. Computer Vision*. Copenhagen, DK.

Tsuda, K., Raetsch, G., & Warmuth, M. K. (2005). Matrix exponentiated gradient updates of online learning and bregman projection. *Journal of Machine Learning Research*, *6*.

Weinberger, K. Q., Blitzer, J., & Saul, L. K. (2005). Distance Metric Learning for Large Margin Nearest Neighbor Classification. *Adv. in Neural Inf. Proc. Sys. (NIPS)*.

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2002). Distance metric learning with application to clustering with side-information. *Adv. in Neural Inf. Proc. Sys. (NIPS)*.