Structured Metric Learning for High Dimensional Problems

Jason V. Davis
Dept. of Computer Sciences
University of Texas at Austin
Austin, TX 78712
jdavis@cs.utexas.edu

Inderjit S. Dhillon
Dept. of Computer Sciences
University of Texas at Austin
Austin, TX 78712
inderjit@cs.utexas.edu

ABSTRACT

The success of popular algorithms such as k-means clustering or nearest neighbor searches depend on the assumption that the underlying distance functions reflect domain-specific notions of similarity for the problem at hand. The *distance metric learning* problem seeks to optimize a distance function subject to constraints that arise from fully-supervised or semi-supervised information. Several recent algorithms have been proposed to learn such distance functions in *low* dimensional settings. One major shortcoming of these methods is their failure to scale to *high* dimensional problems that are becoming increasingly ubiquitous in modern data mining applications. In this paper, we present metric learning algorithms that scale linearly with dimensionality, permitting efficient optimization, storage, and evaluation of the learned metric. This is achieved through our main technical contribution which provides a framework based on the log-determinant matrix divergence which enables efficient optimization of structured, low-parameter Mahalanobis distances. Experimentally, we evaluate our methods across a variety of high dimensional domains, including text, statistical software analysis, and collaborative filtering, showing that our methods scale to data sets with tens of thousands or more features. We show that our learned metric can achieve excellent quality with respect to various criteria. For example, in the context of metric learning for nearest neighbor classification, we show that our methods achieve 24% higher accuracy over the baseline distance. Additionally, our methods yield very good precision while providing recall measures up to 20% higher than other baseline methods such as latent semantic analysis.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Algorithms, Experimentation

General Terms

Distance Metric Learning, High Dimensional Learning

KDD'08, August 24–27, 2008, Las Vegas, Nevada, USA.

1. INTRODUCTION

The problem of comparing examples is a fundamental issue as popular algorithms such as k-means clustering and nearest neighbor searches rely on an underlying distance funciton. While common practice has traditionally appealed to off-the-shelf distance functions or hand-tuned metrics, the distance metric learning problem instead seeks to automatically optimize a distance function in either semi-supervised or fully supervised settings. The goal of metric learning is to optimize a distance function that reflects the domain-specific notion for the problem at hand.

Metric learning algorithms typically work by optimizing a target distance under various types of constraints. In semi-supervised clustering applications, constraints are typically either "must-link" (two examples should be in the same cluster), or "cannot-link" (two examples should be in different clusters). In information-retrieval settings, constraints relating pairs of examples can be inferred from click-streams. For example, a click on a second search result without a click on the first indicates that the former result should be closer to the search target than the latter. Finally, constraints can be directly inferred in fully-supervised settings, where examples in the same classes can be constrained to be similar if they share the same class label and dissimilar otherwise.

One class of distance functions that has shown good generalization properties is the Mahalanobis distance [3, 10, 6]. The Mahalanobis distance generalizes the standard squared Euclidean distance commonly used by algorithms such as the k-nearest neighbor classifier. Intuitively, the Mahalanobis distance works by scaling and rotating the feature space, giving certain features more weight while also incorporating correlations between features. Mathematically, the function is defined over a d-dimensional vector space parametrized by a $d \times d$ positive definite matrix. Recently, several papers have proposed methods for learning Mahalanobis matrices subject to a given set of constraints [3, 10, 11, 6]. Overall, algorithms proposed in these papers have resulted in learned distance functions with excellent generalization performance for *low* dimensional problems.

However, in *high* dimensional settings, the problem of learning and evaluating a Mahalanobis distance function with its associated $d \times d$ matrix becomes quickly intractable due to the quadratic dependency on d. This quadratic dependency affects not only the running time for both training and testing, but also poses tremendous challenges in estimating a quadratic number of parameters. For example, a data set with 10,000 dimensions requires estimation of a symmetric positive definite matrix with about 50 million parameters! This represents a fundamental limitation of existing approaches, as many modern data mining problems possess relatively high dimensionality. For example, in text-analysis domains, standard bag-of-words models can reach the size of thousands or even

tens of thousands of features. Statistical software analysis applications that monitor program paths or method counts similarly have features sets with sizes of thousands or more. Finally, in collaborative filtering domains, objects are typically rated by thousands or even millions of reviewers. Methods in these domains typically compare content (e.g. movies, songs, etc.) using a representation in which each reviewer can be viewed as a single feature.

In this paper, we present algorithms for learning structured Mahalanobis distance functions that scale linearly with the dimensionality. Instead of searching for a full $d \times d$ matrix with $O(d^2)$ parameters, our methods search for compressed representations that typically have $O(d)$ parameters. This enables the Mahalanobis distance function to be learned, stored, and evaluated efficiently in the context of high dimensionality.

In particular, the technical contributions of our paper are the problem formulations and resulting algorithms that compute two types of structured low parameter matrices: a low-rank representation, and a diagonal plus low-rank representation. The low-rank representation, HDLR, results in a distance measure which is similar to that used by latent semantic analysis (LSA) [4]. This distance projects data into a low dimensional factor space, and the resulting distance between two examples is the distance between their projections. Our low-rank method can be viewed as a semi-supervised variant of LSA, and is well suited for applications in which higher recall is desired. The second method, HDILR , learns a diagonal plus low-rank matrix, and is well suited for problems where both high recall as well as high precision are important. This is achieved by comparing examples at both the factor level in addition to a component that compares examples at a much finer, feature-level resolution.

Computationally, our algorithms are based on the information-theoretic metric learning method presented in [3]. The problem is formulated as one of learning a "maximum entropy" Mahalanobis distance that satisfies a given set of constraints. Mathematically, this results in a convex programming problem with a matrix-valued objective function called the log-determinant (LogDet) divergence. We provide two new algorithms based on the LogDet divergence that enable learning Mahalanobis distances in high dimensions. Both of these algorithms scale linearly with dimensionality as $O(d)$.

Experimentally, we evaluate our methods in the context of several modern domains, including text, statistical software analysis, and collaborative filtering. We provide experimental evidence to show that existing metric learning algorithms do not scale to high dimensional data sets, while demonstrating that our methods can easily handle data with upwards of ten thousand dimensions. We compare our methods in the context of learning metrics for nearest neighbor searches on the basis of several criteria, including accuracy, precision and recall. As baseline measures, we use the standard Euclidean distance and LSA. Additionally, we compare our methods against a heuristic in which an existing full-rank metric learning algorithm LMNN [10] is used to learn a low-rank distance function. In general, we show that our low-parameter metric learning algorithms can learn high quality distance functions. For example, classification accuracy for the Classic3 data set is improved by 24% over standard Euclidean distance measures. Additionally, our methods achieve precision as high as all other methods while yielding recall values up to 20% higher than a baseline LSA approach.

The paper is organized as follows. Section 2 introduces the problem and provides background on related low-dimensional metric learning methods. Section 3 provides two low-parameter Mahalanobis distance forms for learning in high dimensions. Section 4 formalizes our problems, and we provide efficient and scalable algorithms. Finally, we present experimental results in section 5.

2. BACKGROUND & RELATED WORK

One class of distance measures that has shown good generalization potential is the Mahalanobis distance. This distance function generalizes the standard squared Euclidean distance and is parameterized by a positive (semi)-definite matrix A:

$$d_A = (\boldsymbol{x} - \boldsymbol{y})^T A (\boldsymbol{x} - \boldsymbol{y}). \tag{2.1}$$

Learning such a distance function has been the focus of much recent research [3, 10, 11, 6], and has proven to be quite successful in low dimensional domains. These distance functions are typically learned given supervised or semi-supervised constraint data. For example, the information-theoretic metric learning algorithm presented by Davis et. al. [3] learns a distance function subject to similarity and dissimilarity constraints. The large-margin nearest neighbor method (LMNN) presented by Weinberger et. al. [10] takes a related approach in comparing three examples at a time (i.e. $\boldsymbol{x}$ is more similar to $\boldsymbol{y}$ than to $\boldsymbol{z}$).

A common theme in existing methods is the regularization term found in the problem objective. In Xing et. al., a method is presented in which the learned matrix A is optimized with respect to a sum-of-squares Frobenius objective [11]. LMNN is formulated as a semi-definite programming problem with a linear objective optimizing the trace of the matrix A. The information-theoretic metric learning (ITML) method seeks a matrix that minimizes the differential relative entropy between a baseline Gaussian parametrized by A_0 to a target Gaussian parametrized by A. Mathematically, this entropic objective results in a convex programming problem that minimizes the *log-determinant* (LogDet) divergence with respect to the baseline matrix A_0.

The methods we present in this paper attempt to learn a structured positive semi-definite matrix using the LogDet problem framework, and are similar to ITML, which we now describe in detail. The problem assumes a given set of similarity constraints S and dissimilarity constraints D between pairs of examples. Constraints may be inferred from true labels (where examples in the same class are constrained to be similar and examples from different classes are constrained to be dissimilar), or constraints may be explicitly provided. Other constraints that are linear in the entries of A can also be easily incorporated. Additionally, ITML assumes a baseline Mahalanobis distance function parametrized by a positive definite matrix A_0. The formal goal is to learn a Mahalanobis distance parametrized by A that has minimum LogDet divergence to a given baseline matrix A_0 while satisfying the given constraints:

$$\begin{aligned} \min_{A} \quad & D_{\ell \mathrm{d}}(A|A_0) \\ \text{subject to} \quad & d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq u \qquad (i,j) \in S, \\ & d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \ell \qquad (i,j) \in D. \end{aligned} \tag{2.2}$$

The LogDet objective function $D_{\ell \mathrm{d}}(A|A_0)$ is a non-negative, convex function that in the absence of constraints is minimized when $A = A_0$. It is defined over $d \times d$ positive definite matrices A and A_0:

$$D_{\ell \mathrm{d}}(A|A_0) = \mathrm{tr}(AA_0^{-1}) - \log |AA_0^{-1}| - d,$$

where $|X|$ denotes the determinant of the matrix X. In practice, slack variables for the constraints can be incorporated into the above formulation but are omitted for the sake of clarity.

3. STRUCTURED DISTANCE METRICS

A major shortcoming of ITML and other existing approaches is their quadratic (or even cubic) dependency on the dimensionality. Learning full-rank Mahalanobis matrices for problems where

A) **Metric learning** is an **important problem** in **data mining**.
B) **High dimensional problems** are **common** in **data mining**.
C) **Optimizing distance functions** in **high dimensions** is **challenging**.

Word	A	B	C
metric	1	0	0
learn	1	0	0
important	1	0	0
problem	1	1	0
data	1	1	0
mining	1	1	0
common	0	1	0
high	0	1	1
dimension	0	1	1
optimiz	0	0	1
distance	0	0	1
function	0	0	1
challeng	0	0	1

	A	B	C
A	6	3	0
B	3	6	2
C	0	2	6

Figure 1: The upper left corner lists three sentences. The table on the right shows word counts for each word from the sentences. We can see from the inner product matrix on the lower left that even though all three sentences are about metric learning, the distance between documents A and C is large. This toy example illustrates that term frequency models are quite accurate when inner products are larger, yet can be inaccurate when inner products are small or zero.

dimensionality is large is prohibitive for several reasons. First, optimizing via algorithms with quadratic dependencies on the dimensionality can be quite expensive. Second, learning a full-rank $d \times d$ matrix can be viewed as a statistical inference procedure over $O(d^2)$ parameters. For example, in text analysis applications, data sets with thousands of dimensions result in full-rank Mahalanobis matrices parametrized by millions of values. For even the most robust methods, learning under such conditions is prone to overfitting and requires a very large amount of supervision. Finally, computing the distance between two points with respect to a Mahalanobis distance function parametrized by a dense, full-rank matrix is an $O(d^2)$ operation. To overcome these problems, we now present Mahalanobis distance functions that are parametrized by $O(d)$ values.

3.1 Low-Rank Mahalanobis Distances

Term frequency models represent text documents in terms of individual words and their respective frequencies and are standard representations used in many text analysis applications [1]. These models typically compute the distance between two examples $\boldsymbol{x}$ and $\boldsymbol{y}$ using the cosine similarity, $\cos(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x}^T \boldsymbol{y}}{\|\boldsymbol{x}\| \|\boldsymbol{y}\|}$. Note that when $\boldsymbol{x}$ and $\boldsymbol{y}$ are normalized to have unit L_2 norm, the cosine similarity is equivalent to the standard Euclidean distance: $d_I(\boldsymbol{x}, \boldsymbol{y}) = 2 - 2 * \cos(\boldsymbol{x}, \boldsymbol{y})$. In many high dimensional domains, feature representations tend to be very sparse, and term frequency models are no exception. This poses several problems for standard Euclidean measures. In term frequency models, two documents can have very similar contextual meaning, yet may not necessarily share many of the same words. Hence, the inner product between two documents can be quite small or even zero, resulting in large Euclidean distances. An example of this is shown in Figure 1, where we have three sentences about metric learning. Sentence pairs (A,B) and (B,C) share several common words. Sentences A and C, however, share no common words and the Euclidean distance between them will therefore be quite large. Thus, even though A and C are contextually similar, the model does not reflect their similarity.

Latent factor models work by representing objects in terms of their context or underlying topics [5]. Instead of representing an object $\boldsymbol{x}$ in its original high dimensional space, latent factor models provide a mapping f that transforms $\boldsymbol{x}$ into some lower k-dimensional space. In Figure 1, we saw that if examples A and C are compared using the Euclidean distance via their original full-dimension representations, the resulting distances will be large. This is in spite of the fact that the two examples are in fact quite similar. The goal of a latent factor model is to learn a mapping f such that $f(A)$ and $f(C)$ are close to each other.

A popular class of latent factor models such as latent semantic analysis (LSA) [4] are those that are parametrized by a $d \times k$ projection matrix R. Here, the factor model's mapping function is $f(\boldsymbol{x}) = R^T \boldsymbol{x}$. Consider the Euclidean distance between the latent factors of two points $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\begin{aligned} d_I\left(f(\boldsymbol{x}), f(\boldsymbol{y})\right) &= d_I(R^T\boldsymbol{x}, R^T\boldsymbol{y}) \\ &= (R^T\boldsymbol{x} - R^T\boldsymbol{y})^T(R^T\boldsymbol{x} - R^T\boldsymbol{y}) \\ &= d_{A_\ell}(\boldsymbol{x}, \boldsymbol{y}), \end{aligned} \tag{3.1}$$

where d_{A_ℓ} is a low-rank Mahalanobis distance defined by the low-rank matrix $A_\ell = RR^T$:

$$d_{A_\ell}(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T RR^T (\boldsymbol{x} - \boldsymbol{y}). \tag{3.2}$$

Whereas a full-rank Mahalanobis matrix is parametrized by $O(d^2)$ values, this low-rank matrix is parametrized by the $O(dk)$ parameters in R.

Computationally, we can see from equation (3.1) that low-rank Mahalanobis distances can also be computed efficiently in $O(dk)$ time, as the distance between two d-dimensional instances $\boldsymbol{x}$ and $\boldsymbol{y}$ can be computed by first mapping them to a lower dimensional space by computing $R^T\boldsymbol{x}$ and $R^T\boldsymbol{y}$, and then computing the standard squared Euclidean distance between the low-dimensional points.

3.2 Diagonal Plus Low-Rank Distances

In Figure 1, we saw an example of two sentences that were of similar context but had large Euclidean distances due to zero overlap (i.e. zero inner product) between their respective feature sets. Thus, it would be incorrect to conclude that in the context of term frequency models, if two objects have zero or small inner products, then they are contextually different. However, the converse of this statement is much more likely to hold true. That is, if two documents share many commons words, then they are likely to be contextually similar. However, in traditional low-rank models such as LSA this overlap is largely ignored when data is mapped into a low dimensional space.

We now examine this observation in the context of two standard measures used in information retrieval, precision and recall:

$$recall = \frac{\text{Number of Relevant Documents Returned}}{\text{Total Number of Relevant Documents}}. \tag{3.3}$$

Precision is measured as the number of relevant documents returned, divided by the total number of documents returned:

$$precision = \frac{\text{Number of Relevant Documents Returned}}{\text{Total Number of Documents Returned}}. \tag{3.4}$$

Term frequency models tend to result in higher precision, whereas low-rank factor models provide better recall.

Figure 2 illustrates this behavior for the Classic3 text data set. More details regarding this data set will be presented in Section 5. Precision is plotted for various recall values, comparing nearest neighbor searches using a Tf-Idf Euclidean distance model with that of LSA. We see that for relatively low recall values (i.e. when a relatively small number of documents is returned), the term frequency model significantly outperforms a ten-dimensional LSA factor model in terms of precision. However, as recall increases, the

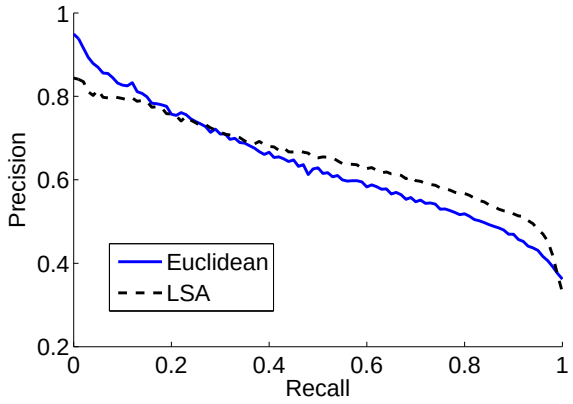


Figure 2: Precision-recall curve for the Classic3 text data set. term frequency models using Euclidean distance yield relatively high precision, while the low-rank LSA method has higher recall.

precision for the Euclidean distance model starts to sharply decrease, after which LSA eventually achieves noticeably higher precision.

In domains where both high recall and high precision are needed, we propose a second Mahalanobis distance that incorporates benefits of both the Euclidean distance as well as a low-rank component. We propose a Mahalanobis distance parametrized by a matrix $I + A_\ell$, where A_ℓ is low-rank:

$$\begin{aligned} d_{I+A_\ell}(\boldsymbol{x},\boldsymbol{y}) &= (\boldsymbol{x}-\boldsymbol{y})^T(I+A_\ell)(\boldsymbol{x}-\boldsymbol{y}) \\ &= (\boldsymbol{x}-\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y}) + (\boldsymbol{x}-\boldsymbol{y})^T A_\ell(\boldsymbol{x}-\boldsymbol{y}) \\ &= d_I(\boldsymbol{x},\boldsymbol{y}) + d_{A_\ell}(\boldsymbol{x},\boldsymbol{y}). \end{aligned} \quad (3.5)$$

Since this proposed measure compares vectors in both the original feature space as well as in a projected low-rank factor feature space, one would expect it to achieve both high recall as well as high precision. Revisiting Figure 1, we can see that the Euclidean component is a good predictor for two of the three distances, resulting in relatively small distances when comparing (A,B) and (B,C). However, the Euclidean distance between A and C is large, and the low-rank component is needed here to effectively compare sentences A and C.

So far, we have proposed and motivated two forms of low-parameter Mahalanobis distances. In the next section, we formalize these two problems and provide efficient algorithms to optimize them.

4. LEARNING STRUCTURED METRICS

4.1 Low-Rank Mahalanobis Distances

We now extend the full-rank ITML algorithm to learn low-rank matrices. Let R be the $d \times k$ factor matrix for the rank-k regularization matrix A_0, i.e. $A_0 = RR^T$. We formulate our high dimensional low-rank (HDLR) metric learning problem as:

$$\begin{aligned} \min_{A} \quad & D_{\ell \mathrm{d}}(A|RR^T) \\ \text{subject to} \quad & d_A(\boldsymbol{x}_i,\boldsymbol{x}_j) \le u \qquad (i,j) \in S, \\ & d_A(\boldsymbol{x}_i,\boldsymbol{x}_j) \ge \ell \qquad (i,j) \in D, \\ & rank(A) \le k. \end{aligned} \quad (4.1)$$

Comparing this to the full-rank ITML formulation (2.2), we see that A_0 here is low-rank, and an additional constraint has been added enforcing the rank of the optimal Mahalanobis matrix A. Recent work by Kulis et. al. [9] considers a related problem of learning low-rank kernel matrices subject to linear constraints on the matrix. In [9], the LogDet divergence was extended to the positive *semi*-definite cone, and it was shown that two matrices have a finite LogDet divergence if and only if they share the same range space. The following was shown in [9]:

LEMMA 1. *The objective $D_{\ell d}(A|RR^T)$ of problem (4.1) is finite if and only if A is positive semi-definite with range space equal to the range space of R.*

So, if the baseline Mahalanobis distance function is parametrized by a rank-k matrix, the optimal solution to the HDLR metric learning problem (4.1) without the rank constraint will necessarily have rank k (assuming that the optimization problem (4.1) is feasible). Therefore, the rank constraint $rank(A) \le k$ need not be explicitly enforced. In section 4.5, we present methods that can be used to choose an appropriate baseline matrix.

4.2 HDLR Algorithm

We now present Algorithm 1 that solves our HDLR formulation (4.1). The algorithm optimizes a slightly modified version of problem (4.1) that incorporates slack variables to allow constraint violation in the case of incorrect or noisy constraints. The slack penalty parameter γ determines the relative weighting given to the LogDet component of the objective as opposed to the slack penalty component of the objective. When γ is large, more weighting is given to the slack terms, and the final solution will more closely satisfy the constraints. When γ is small, more emphasis is given to the LogDet objective, yielding smoother solutions which are closer to the regularization matrix A_0. In practice, γ is chosen via cross-validation.

The algorithm uses the method of cyclic projections and works by iteratively projecting the current solution onto a single constraint. Instead of directly working on the $d \times d$ matrix A, the algorithm instead optimizes its $d \times k$ factor matrix B. The main loop starting in line 2 iterates over each constraint until convergence. In practice, convergence can be checked by monitoring the change in the dual variables λ. Steps 5-10 compute the projection parameter β. In step 11, this parameter is then used to update B via a rank-one update. Each projection can be computed in closed form and requires $O(dk)$ computation, where k is the rank of A_0. Finally, the optimal solution is $A = BB^T$. Note that the latter step may not be needed since as shown in (3.1), the low-rank Mahalanobis distance between two points can be computed in $O(dk)$ time without the need to explicitly compute A.

4.3 Diagonal Plus Low-Rank Distances

We now formulate the high-dimensional identity plus low-rank (HDILR) metric learning problem. As in the formulation presented in Section 4.1, let R be a $d \times k$ factor matrix of our data. We will constrain the low-rank portion of the HDILR formulation to span the columns of R, i.e., the learned low-rank portion of our metric resides in the same range space as the original factor matrix. Whereas the HDLR method has only the low-rank term, the HDILR method has an additional identity matrix term (corresponding to the baseline squared Euclidean distance). Adding a positive semi-definite component to this baseline measure would result in increasing all distances, an undesirable property for enforcing similarity constraints. To overcome this, we offset the low-rank component by subtracting the baseline distance measure projected onto the factor space.

Formally, let U be an orthogonal representation of the columns

Algorithm 1 High Dim. Low-Rank (HDLR) Metric Learning

Require: $A_0 = RR^T$: baseline Mahalanobis matrix, γ: slack penalty, $X = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n]$: set of constrained points, (S, D): similarity / dissimilarity constraints.

$B = R,\ \lambda_{ij} = 0\ \forall\, i, j,\ b_{ij} = 0\ \forall\, i, j$
while not converged **do**
  $(i, j) \leftarrow$ similarity or dissimilarity constraint
  $\delta \leftarrow 1$ if similarity constraint, -1 if dissimilarity constraint
  $d \leftarrow (\boldsymbol{x}_i - \boldsymbol{x}_j)^T B^T B (\boldsymbol{x}_i - \boldsymbol{x}_j)$
  $\eta \leftarrow \min\ \lambda_{ij}, \frac{\delta\gamma}{\gamma+1}\ \ \frac{1}{d} - \frac{1}{b_{ij}}$
  $\alpha \leftarrow \delta\eta/(1 + \delta\eta d)$
  $\lambda_{ij} \leftarrow \lambda_{ij} - \eta$
  $b_{ij} = \gamma b_{ij}/(\gamma + \delta\eta b_{ij})$
  $\beta = \sqrt{1 + \alpha} - 1$
  $B \leftarrow B + \beta(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T B$
end while
$A \leftarrow BB^T$

of R (i.e. $U = R(R^T R)^{-\frac{1}{2}}$, so that $U^T U = I$). We construct the low-rank component A_ℓ of our identity plus low-rank matrix A given in (3.5) as the difference of two matrices, $UU^T AUU^T - UU^T A_0 UU^T = UU^T(A - A_0)UU^T$. The first term is a function of the learned matrix A, and the second term is a low-rank offset provided by the regularization matrix A_0. If $A = A_0$, the low-rank term A_ℓ will always be zero, and the distance function will reduce to that of the standard Euclidean distance. As A diverges from A_0, the low-rank term starts to dominate, giving more emphasis to the underlying factor model.

We will now present the HDILR metric learning problem that learns a full-rank Mahalanobis matrix with the constraint $A = I + UU^T(A - I)UU^T$. The analysis and algorithms presented below can be generalized to arbitrary regularizers A_0 (i.e. $A = I + UU^T(A - A_0)UU^T$). The HDILR problem is:

$$\begin{aligned} \min_A \quad & D_{\ell\mathrm{d}}(A|I) \\ \text{subject to} \quad & d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \le u \qquad (i,j) \in S, \\ & d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \ge \ell \qquad (i,j) \in D, \\ & A = I + UU^T(A - I)UU^T. \end{aligned} \tag{4.2}$$

Consider the distance constraints used in this formulation:

$$\begin{aligned} d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) &= d_{I + A_\ell}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &= d_I(\boldsymbol{x}_i, \boldsymbol{x}_j) + d_{A_\ell}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &= d_I(\boldsymbol{x}_i, \boldsymbol{x}_j) - d_{UU^T}(\boldsymbol{x}_i, \boldsymbol{x}_j) + d_{UU^T AUU^T}(\boldsymbol{x}_i, \boldsymbol{x}_j). \end{aligned}$$

The first two terms are independent of the learned matrix A and are therefore constants in the context of the optimization problem. Moving these to the right hand side, we can rewrite the problem:

$$\begin{aligned} \min_A \quad & D_{\ell\mathrm{d}}(A|I) \\ \text{subject to} \quad & d_{UU^T AUU^T}(\boldsymbol{x}_i, \boldsymbol{x}_j) \le u - c_{ij} \qquad (i,j) \in S, \\ & d_{UU^T AUU^T}(\boldsymbol{x}_i, \boldsymbol{x}_j) \ge \ell - c_{ij} \qquad (i,j) \in D, \\ & A = I + UU^T(A - I)UU^T, \end{aligned} \tag{4.3}$$

where $c_{ij} = d_I(\boldsymbol{x}_i, \boldsymbol{x}_j) - d_{UU^T}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Note that $c_{ij} = 0$ when the training points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ lie in the range space of R (and hence, U). Recall that the original problem constrains the full-rank distance between points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. In (4.3), a low-rank approximation of the learned Mahalanobis distance is constrained. Recall that for the low-rank HDLR formulation presented in the previous section, if A_0 is low-rank, then the optimal solution is also low-rank. The next theorem characterizes the solution for problem (4.3), showing that the optimal solution satisfies $A^* = I + UU^T(A^* - I)UU^T$, thereby obviating the need to explicitly enforce this identity plus low-rank constraint.

THEOREM 1. *Let A^* be the optimal solution to an instance of the information-theoretic metric learning problem (4.3) with similarity constraints S, dissimilarity constraints D, and orthogonal projection matrix U. Then A^* satisfies $I + UU^T(A^* - I)UU^T$.*

PROOF. For appropriately defined constants $\bar{c}_{ij}$, the Lagrangian of problem (4.3) can be written as

$$\begin{aligned} & L(A, \lambda) = \mathrm{tr}(A) - \log|A| \\ & + \sum_{i,j} \delta_{ij}\lambda_{ij} \quad \mathrm{tr}(UU^T AUU^T(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T) - \bar{c}_{ij} \quad , \end{aligned}$$

where λ_{ij} are dual variables with $\lambda_{ij} \ge 0$, $\delta_{ij} = +1$ for similarity constraints and $\delta_{ij} = -1$ for dissimilarity constraints. Using the fact that $\nabla_A \log|A| = A^{-1}$ [2], the gradient of the Lagrangian is,

$$\nabla_A L(A, \lambda) = I - A^{-1} + \sum_{i,j} \delta_{ij}\lambda_{ij} UU^T(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T UU^T.$$

Setting the gradient to zero and solving for A^{-1},

$$\begin{aligned} (A^*)^{-1} &= I + \sum_{i,j} \delta_{ij}\lambda_{ij} UU^T(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T UU^T \\ &= I + UU^T \left(\sum_{i,j} \delta_{ij}\lambda_{ij}(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \right) UU^T \\ &= I + UU^T P UU^T. \end{aligned}$$

Thus, the inverse of the optimal solution has the form identity plus low-rank. To see that the solution A^* also has this form, we can use the Sherman-Morrison-Woodbury formula [7], which states that for any invertible matrix Y and $d \times k$ complex matrix Z:

$$(Y + ZZ^H)^{-1} = Y^{-1} - Y^{-1}Z(I + Z^H Y^{-1} Z)^{-1} Z^H Y^{-1}.$$

Note that Z may be complex, and we denote Z^H as its conjugate transpose. Applying the above equation for $Y = I$, and for $Z = UU^T C$, where C is $d \times d$ and is defined such that $CC^H = P$, we have:

$$\begin{aligned} A^* &= (I + UU^T P UU^T)^{-1} \\ &= I - UU^T C(I + C^H UU^T UU^T C)^{-1} C^H UU^T \\ &= I - UU^T \hat{A} UU^T, \end{aligned}$$

for $\hat{A} = C(I + C^H UU^T UU^T C)^{-1} C^H$. Finally, using the fact that $U^T U = I^k$, we have

$$\begin{aligned} I \ + \ & UU^T(A^* - I)UU^T \\ &= I + UU^T \quad I - UU^T \hat{A} UU^T - I \quad UU^T \\ &= I - UU^T \hat{A} UU^T \\ &= A^*. \end{aligned}$$

□

4.4 HDILR Algorithm

The metric learning problem HDILR formulated in the previous section learns a full-rank $d \times d$ matrix. Using algorithms presented in [3], both running time and storage requirements for this algorithm would still be quadratic in the dimensionality. In this section,

we show that by exploiting the identity plus low-rank structure of formulation (4.3), we can transform the problem to an equivalent problem in k dimensions. Via this transformation, the problem can then be solved in time quadratic in k. The solution can then by mapped back to the optimal solution of the original problem via a simple matrix operation.

Consider the following k-dimensional metric learning problem:

$$\begin{aligned} \min_{M} \quad & D_{\ell d}(M|I^k) \\ \text{subject to} \quad & d_M(U^T\boldsymbol{x}_i, U^T\boldsymbol{x}_j) \leq u - c_{ij} \qquad (i,j) \in S, \\ & d_M(U^T\boldsymbol{x}_i, U^T\boldsymbol{x}_j) \geq \ell - c_{ij} \qquad (i,j) \in D, \end{aligned} \tag{4.4}$$

where the superscript on I^k is used to emphasize the dimensionality of the matrix. We now show how to construct the optimal solution to problem (4.3) given the optimal solution M^* to problem (4.4).

LEMMA 2. *Let M^* be an optimal solution to problem (4.4). Then the optimal solution to problem (4.3) can be constructed as $A^* = I^d + U(M^* - I^k)U^T$.*

PROOF. We first show that problems (4.4) and (4.3) are equivalent. Specifically, we show that for any two matrices M and A satisfying $M = U^T AU$, (a) problem (4.4) is feasible if and only if problem (4.3) is feasible, and (b) the problem objectives differ by at most a constant. To see equivalence with respect to feasibility,

$$\begin{aligned} d_M(U^T\boldsymbol{x}_i, U^T\boldsymbol{x}_j) &= (\boldsymbol{x}_i - \boldsymbol{x}_j)^T UMU^T(\boldsymbol{x}_i - \boldsymbol{x}_j) \\ &= (\boldsymbol{x}_i - \boldsymbol{x}_j)^T UU^T AUU^T(\boldsymbol{x}_i - \boldsymbol{x}_j) \\ &= d_{UU^TAUU^T}(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{aligned}$$

To see that the objective functions of the two problems differ by at most a constant, we consider the trace term:

$$\begin{aligned} \operatorname{tr}(A) &= \operatorname{tr}(I^d + UU^T(A - I^d)UU^T) \\ &= \operatorname{tr}(U^T AUU^T U) + \operatorname{tr}(I^d - UU^T) \\ &= \operatorname{tr}(M) + \operatorname{tr}(I^d - UU^T). \end{aligned}$$

Since the matrix U is orthogonal, $U^T U = I^k$, so $\operatorname{tr}(I^d - UU^T) = d - k$. Next consider the log det term. Let W be the orthogonal complement to U, i.e. a $d \times (d-k)$ matrix such that $I^d - UU^T = WW^T$ and $U^T W = 0$.

$$\begin{aligned} A &= I^d + UU^T(A - I^d)UU^T \\ &= UU^T AUU^T + I^d - UU^T \\ &= UU^T AUU^T + WW^T \\ &= [U\ W] \begin{bmatrix} U^T AU & 0 & U^T \\ 0 & I^{d-k} & W^T \end{bmatrix}. \end{aligned}$$

The determinant of a matrix is invariant under the orthogonal similarity transformation by $[U\ W]$, so

$$\log|A| = \log|U^T AU| + \log|I^{d-k}| = \log|M|.$$

Finally, we have

$$\begin{aligned} D(A|I^d) &= \operatorname{tr}(A) - \log|A| - d \\ &= \operatorname{tr}(M) + d - k - \log|M| - d \\ &= D(M|I^k). \end{aligned}$$

□

Algorithm 2 shows how the above lemma can be used to efficiently solve the HDILR metric learning problem (4.2). Step 1 projects the original d-dimensional data onto a k-dimensional subspace using the low-rank basis U. Next, step 2 solves the (full-rank) ITML problem in this much lower, k-dimensional space, returning a $k \times k$ matrix M^*. The optimal solution can be constructed using Lemma 2 as $A^* = I + UM^*U^T$. Note that this matrix never needs to be explicitly constructed, since a Mahalanobis distance parametrized by A^* can be expressed as the sum of two Mahalanobis distances as shown in equation (3.5). Finally, it is possible that the right hand side of problem (4.3) is negative. This presents problems for the slack variables used in Algorithm 2. However, in practice, the goal is to learn a metric in which constraints are satisfied *relatively*, and this issue can be solved by adding a scalar $c > 0$ to the right hand side of (4.3) in order to ensure positivity.

Algorithm 2 Identity Plus Low-Rank (HDILR) Algorithm

Require: U: low-rank basis, γ: slack penalty, $X = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n]$: set of constrained points, (S, D): similarity / dissimilarity constraints.

1: Form the projected data set $\hat{X} = [U^T\boldsymbol{x}_1, ..., U^T\boldsymbol{x}_n]$
2: Compute optimal solution M^* to full-rank ITML problem (4.4) with constraints D and S over the projected data set $\hat{X}$ using Algorithm 1
3: Return optimal solution $A^* = I + UM^*U^T$

4.5 Choosing an Appropriate Basis

The metric learning algorithms presented in the previous section are parametrized by a low-rank matrix. Algorithms 1 and 2 work by optimizing with respect to a given basis. In order to maximize the quality of the learned metric, an appropriate basis should be chosen.

A standard basis used in unsupervised settings such as latent semantic analysis is the left singular vectors of the singular value decomposition (SVD) [7]. LSA works by taking the SVD of the data matrix. Let X be a $d \times n$ data matrix, where each column represents a d-dimensional instance, and let the SVD of this matrix be UDV^T, where U and V are orthogonal matrices, and D is diagonal. Let U^k denote the first k columns of U. The matrix U^k is typically referred to as the top k principal components, and the matrix $U^k(U^k)^T$ represents a projection from the original space onto a rank-k subspace. LSA uses this projection in computing distances (or cosine similarities) between points, $d_L(\boldsymbol{x}, \boldsymbol{y}) = D_{U^k(U^k)^T}(\boldsymbol{x}, \boldsymbol{y})$. Thus, the baseline matrix that results from this LSA basis is $A_0 = UU^T$.

While the use of the SVD in methods such as LSA has been shown to achieve good results in many information retrieval settings, it is fundamentally an unsupervised method. When used with our metric learning algorithms along with similarity and dissimilarity constraint information, the result is a semi-supervised form of LSA.

In cases where data is fully supervised, we propose a method which chooses a low-rank basis according to the class labels. Whereas LSA chooses vectors based on the SVD, the class-mean method forms vectors directly using the class labels. 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 $\boldsymbol{r}_i$ is computed to form the basis matrix $R = [\boldsymbol{r}_1 \ldots \boldsymbol{r}_c]$. If $k < c$ a similar process is used but restricted to a randomly selected subset of 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 low-rank matrix R.

LSA's use of the SVD results in an orthogonal low-rank basis. The supervised class means method presented here will not generally result in an orthogonal basis. As a final step in forming a class

means basis, we orthogonalize R, resulting in a regularization matrix $A_0 = R(R^TR)^{-1}R^T$. Orthogonalization reduces distortion of the low-rank distance. For example, distances between class means are preserved. Let $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ be two class mean vectors (i.e. columns i and j of R). Then

$$\begin{aligned} d_{A_0}(\boldsymbol{r}_i, \boldsymbol{r}_j) &= (\boldsymbol{r}_i - \boldsymbol{r}_j)^T R(R^TR)^{-1}R^T(\boldsymbol{r}_i - \boldsymbol{r}_j) \\ &= (\boldsymbol{e}_i - \boldsymbol{e}_j)R^TR(R^TR)^{-1}R^TR(\boldsymbol{e}_i - \boldsymbol{e}_j) \\ &= (\boldsymbol{e}_i - \boldsymbol{e}_j)R^TR(\boldsymbol{e}_i - \boldsymbol{e}_j) \\ &= (\boldsymbol{r}_i - \boldsymbol{r}_j)^T(\boldsymbol{r}_i - \boldsymbol{r}_j) = d_I(\boldsymbol{r}_i, \boldsymbol{r}_j), \end{aligned}$$

where $\boldsymbol{e}_i$ is a vector of all zeros with a one in the i^{th} position.

5. EXPERIMENTAL RESULTS

We now present sample results for our methods from a variety of high-dimensional domains: text analysis, statistical software analysis, and collaborative filtering. These datasets can all be characterized by relatively high dimensionality (from 5,000 to more than 100,000 features) and represent a broad sample of modern, high dimensional problems.

We evaluate performance of our learned distance metrics in the context of both classification accuracy for the k-nearest neighbor algorithm, as well as in the context of precision/recall performance for general nearest neighbor searches. The k-nearest neighbor classifier uses $k = 10$ nearest neighbors, breaking ties arbitrarily. Accuracy is defined as the number of correctly classified examples divided by the total number of classified examples. Recall and precision are computed as defined in equations (3.3) and (3.4).

Our experiments compare our two algorithms, HDLR and HDILR, given in Algorithms 1 and 2. We also compare these algorithms to a heuristic based on the Large-Margin Nearest Neighbor (LMNN) metric learning algorithm. In [10], LMNN is presented as a method for learning full-rank Mahalanobis distance matrices. Here, we use a heuristic in which data is first projected onto some low-rank, r-dimensional basis U. LMNN is then run over this r-dimensional problem, yielding a $(r \times r)$ matrix A. Finally, this matrix is transformed back to the original, high-dimensional space as UAU^T. We emphasize that this procedure does not optimize a well-formed global objective, whereas our approaches optimize a log-determinant objective function. The LMNN implementation used is a Matlab implementation provided by Weinberger, one of the authors of [10].

For our proposed algorithms, pairwise constraints are inferred from true labels. For each class 100 pairs of points are randomly chosen from within the class and are constrained to be similar, and 100 pairs of points are drawn from different classes 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 parametrized by A_0. Finally, the slack penalty parameter γ used by Algorithms 1 and 2 is cross-validated using values $\{.01, .1, 1, 10, 100, 1000\}$.

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

5.1 Timing Comparison

We first compare the computational speed of our low-parameter algorithms as compared to existing full-rank methods, LMNN and ITML. Figure 3 shows the time taken to learn metrics of dimensionality 50 to 2000 over a synthetic data set with 900 instances and 3 classes. All implementations are in Matlab and are run on an Intel Pentium processor with 4 GB of RAM. Noting that the time axis is displayed on a log-scale, we can see that our HDLR algorithm scales roughly linearly with dimensionality, whereas existing full-rank methods scale quadratically as dimensionality increases. Further, LMNN ran out of memory when learning a 3000-dimensional metric. Running times of the HDILR method are comparable to those shown for the low-rank HDLR algorithm in Figure 3.

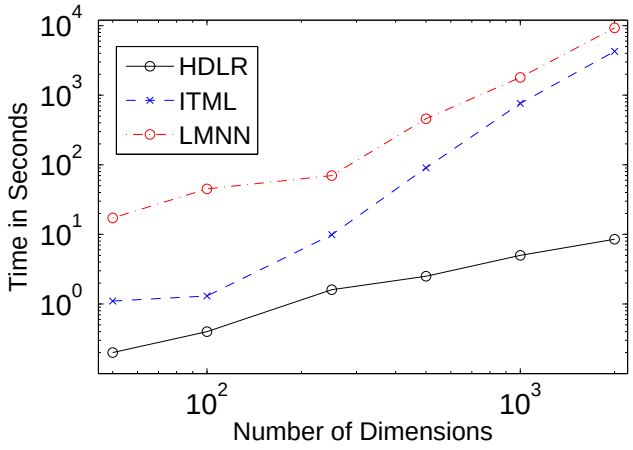


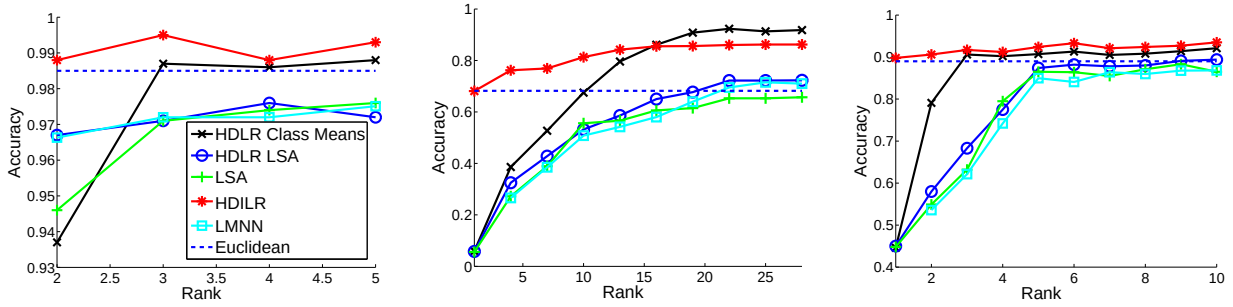
Figure 3: Running times of our high-dimensional algorithms compared to existing full-rank methods. Full-rank Mahalanobis distance learning algorithms do not scale well to high dimensionality, whereas our method HDLR does.

5.2 Text Analysis

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. The text models are limited to the 5,000 words with the largest document frequency counts. We provide experiments over two data sets: CMU's 20-newsgroup data set, and the Classic3 data set. 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.

Figure 4 shows classification accuracy across various ranks for the Classic3 dataset, along with the full newsgroup data set and a subset of the data restricted to the three politics related classes: talk.politics.guns, talk.politics.mideast, and talk.politics.misc. Here, the diagonal plus low-rank method HDILR uses the class means basis as described in section 4.5. Comparing this to the baseline Euclidean measure, we can see that for low dimensionality, the accuracy of the two algorithms is similar, with HDILR having a slightly higher value. For larger ranks, the accuracy of the HDILR method slowly increases, while the accuracy of the low-rank HDLR class means method increases much more quickly. In fact, for the largest 20-class Newsgroup data set (b), we can see that for larger ranks, the HDLR method outperforms the HDILR method. Here, the HDLR method achieves accuracy 27% higher than the baseline Euclidean distance.

The HDILR and HDLR class means methods require full super-



(a) Classic3 (b) 20-Newsgroups (c) Newsgroups Politics Subset

Figure 4: Classification accuracy for Mahalanobis metrics of various ranks. Overall, our methods outperform existing methods.

vision in order to form the low-rank basis. In semi-supervised settings, forming this basis from the similarity and dissimilarity constraints used in our low-rank metric learning algorithms is not possible and a low-rank LSA basis may be used instead. Recall that the LSA basis described in section 4.5 requires no supervision. In Figure 4, we see that our HDLR method outperforms the baseline unsupervised LSA method across all data sets for most dimensions. This is compared to LMNN using the same LSA basis U, which generally performs only comparably to LSA.

Figure 5 shows recall-precision curves for four methods: standard Euclidean distance Tf-Idf measure, LSA, and our two methods using a class means basis. The rank used for LSA and our two methods is ten. We can see that for low recall values, both the Euclidean distance and the HDILR method achieve significantly higher precision values than the other two low-rank methods. As desired recall levels increase, the precision of the Euclidean distance decreases rather quickly, while the HDILR continues to achieve higher precision values that are comparable to or better than the low-rank methods. For the Classic3 data set, HDILR outperforms all other methods for all recall values, marking an improvement over LSA of up to 20%. For the newsgroups Politics subset, the HDILR method outperforms HDLR for low recall values, yet it achieves slightly worse precision for higher recall values.

5.3 Software Analysis

We now present results from the Clarify system [8] which attempt to improve software error messaging via nearest neighbor software support. 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 providing an error report. Clarify works by monitoring a set of predefined program features (the data sets used here represent function counts) during program run-time which are then analyzed in the event of abnormal program termination. In order to troubleshoot problems, Clarify uses nearest neighbor searches to find similar failing executions from other users. Ideally, the neighbors returned should not only have the correct class label, but should also represent those with similar program configurations or program inputs. The four data sets shown here are Mpg321 (an mp3 player, 4 classes, 128 dimensions), Foxpro (a database manager, 4 classes, 12,670 dimensions), and two Linux kernel applications, Iptables and Iproute (having 4-5 classes and 50-250 dimensions). As described in [8], these data sets were created by running each program many times with various inputs exposing each of the error classes. Program features are collected from each program run and each run corresponds to a single training instance.

Figure 6 provides accuracy results for a k-NN classifier used

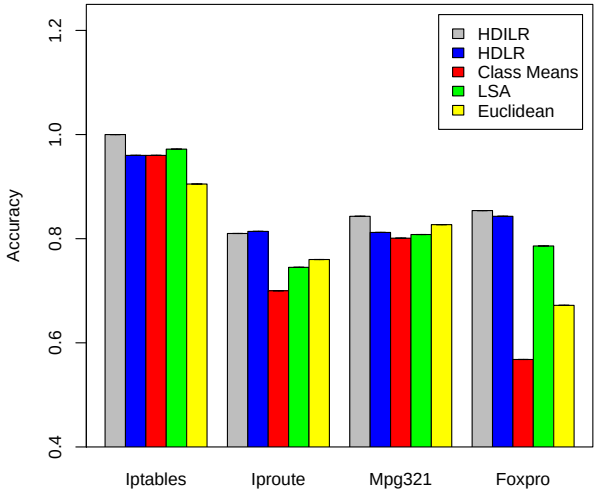


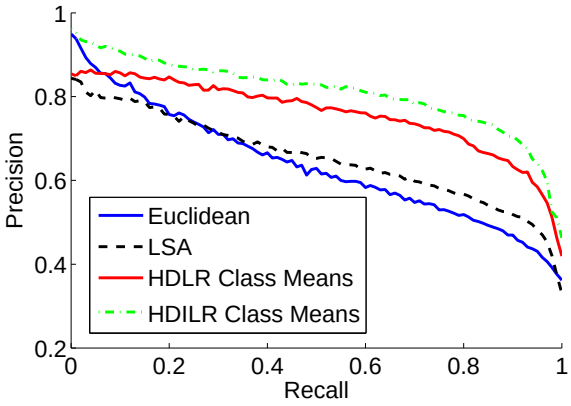
Figure 6: Classification accuracy for four statistical software analysis datasets across six different algorithms: HDLR, HD^{2}LR, a baseline class means method, latent semantic analysis (LSA), and the Euclidean distance.

with our learned rank-10 distance metrics HDLR and HDILR . We compare this against four baseline methods. The class means method is a supervised method in which the class mean basis is used to parametrize a low-rank Mahalanobis distance without performing any additional learning. Confidence intervals intervals shown are computed for the 5^{th} and 95^{th} percentiles. Overall, we see that our HDLR and HDILR methods outperform the other four methods.

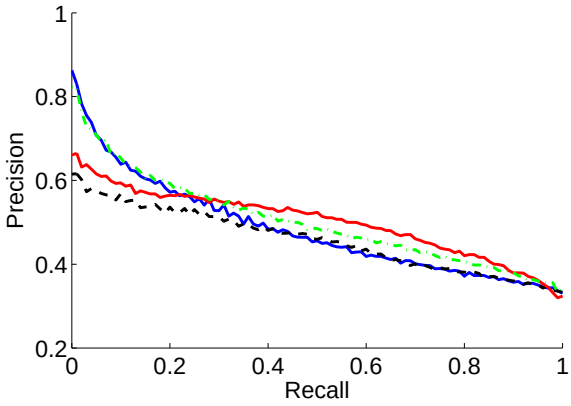
5.4 Collaborative Filtering

Finally, we present experiments over a set of Yahoo song reviews. Here, 14,596 songs are reviewed by a total of 120,397 reviewers. Each review has a score ranging from '1' (the reviewer does not like the song) to '5' (the reviewer liked the song). Further, each song is categorized into one of five genres: Rock, R&B, Pop, Rap, and Country.

Many of today's recommender systems work by performing nearest neighbor searches over such collaborative data in order to help users find similar songs, movies, or products to the ones that he or she already enjoys. Here, we consider the problem of learning a distance function over the Yahoo song data that respects genres. Such a distance function is important as often people prefer songs from a limited number of genres, yet genre information is not known for all songs. In fact, the 14,596 labelled songs used in



(a) Classic3



(a) Newsgroups Politics Subset

Figure 5: Precision-recall curves comparing our low-rank Mahalanobis distance functions to standard LSA and Tf-Idf measures for the Classic3 text data set, and the politics newsgroups subset. Overall, the HDILR method achieves both high precision and high recall that is competitive or better than Euclidean distance and LSA, respectively.

this data set represent a very small subset ($< 1\%$) of the entire set of all songs in the Yahoo music data set, most of which have the genre type 'unknown'.

Figure 7 shows classification accuracy for this data set for four different methods across a varying number of dimensions. Here, we see that the low-rank method HDLR (64.5% accuracy) performed significantly better than HDILR (53.9% accuracy). This data set is extremely sparse, with an average of 33 reviews per song (99.97% sparse).

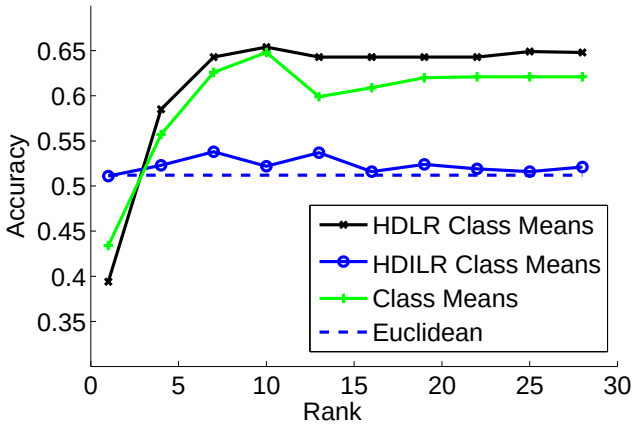


Figure 7: Classification accuracy for distance functions of various ranks for the Yahoo music data set. Here, the HDLR method significantly outperforms the HDILR method. This suggests that HDLR method may yield better results in problems that are extremely sparse.

6. CONCLUSIONS

As seen in the previous section, the quality of a nearest neighbor search can be greatly improved by learning an appropriate distance measure. In this paper, we have proposed formulations and algorithms that are well suited for learning metrics in high dimensional settings. Nearest neighbor searches are very simple algorithms, yet at the same time are quite flexible as they can be used for both classification tasks as well as for standard information retrieval applications. In this paper, we presented algorithms that provide both better classification performance in terms of accuracy, precision and recall over existing baseline methods.

Acknowledgements We would like to thank Yahoo! for the use of their collaborative filtering data set. This research was supported by NSF grant CCF-0431257, NSF-ITR award IIS-0325116 and NSF grant IIS-0713142.

7. REFERENCES

[1] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.

[3] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic Metric Learning. In *ICML*, June 2007.

[4] 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.

[5] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

[6] A. Globerson and S. Roweis. Metric Learning by Collapsing Classes. In *NIPS*, 2005.

[7] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, second edition, 1989.

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

[9] B. Kulis, M. Sustik, and I. S. Dhillon. Learning Low-rank Kernel Matrices. In *ICML*, 2006.

[10] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance Metric Learning for Large Margin Nearest Neighbor Classification. In *NIPS*, 2005.

[11] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *NIPS*, 2002.