Learning Sparse Metrics via Linear Programming

Rómer Rosales and Glenn Fung
Computer-Aided Diagnosis and Therapy Group
Siemens Medical Solutions
51 Valley Stream Parkway
Malvern, PA USA

romer.rosales@siemens.com, glenn.fung@siemens.com

ABSTRACT

Calculation of object similarity, for example through a distance function, is a common part of data mining and machine learning algorithms. This calculation is crucial for efficiency since distances are usually evaluated a large number of times, the classical example being query-by-example (find objects that are similar to a given query object). Moreover, the performance of these algorithms depends critically on choosing a *good* distance function. However, it is often the case that (1) the correct distance is unknown or chosen by hand, and (2) its calculation is computationally expensive (*e.g.,* such as for large dimensional objects). In this paper, we propose a method for constructing relative-distance preserving low-dimensional mappings (sparse mappings). This method allows learning unknown distance functions (or approximating known functions) with the additional property of reducing distance computation time. We present an algorithm that given examples of proximity comparisons among triples of objects (object i is more like object j than object k), learns a distance function, in as few dimensions as possible, that preserves these distance relationships. The formulation is based on solving a linear programming optimization problem that finds an optimal mapping for the given dataset and distance relationships. Unlike other popular embedding algorithms, this method can easily generalize to new points, does not have local minima, and explicitly models computational efficiency by finding a mapping that is sparse, *i.e.,* one that depends on a *small* subset of features or dimensions. Experimental evaluation shows that the proposed formulation compares favorably with an state-of-the art method in several publicly available datasets.

Categories and Subject Descriptors

G.4 [**Mathematical Software**]; H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.4 [**Information Systems Applications**]: Miscellaneous

KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.
Copyright 2006 ACM 1-59593-339-5/06/0008 ...$5.00.

General Terms

Algorithms, measurement, performance

Keywords

Metric learning, dimensionality reduction, linear programming, linear projections, convex optimization, relative distance constraints

1. INTRODUCTION AND RELATED WORK

The notion of a *distance* is essential in many machine learning and data mining concepts. From a practical point of view, the choice of a distance has a direct effect on the performance of many algorithms, both in terms of accuracy and of efficiency. From an accuracy perspective, numerous algorithms rely on the user being able to provide a *good* distance function (*e.g.,* nearest neighbor, clustering methods, SVM, etc). Here, a *good distance function* is roughly one that is small for similar objects and large for dissimilar ones. Clearly, since objects can be similar or dissimilar in many respects, *similarity* is not absolute and depends on the task of interest; thus, different distance functions should in theory be chosen for different tasks. How to best choose an appropriate distance function remains an interesting problem.

From an efficiency perspective, many widely used approaches for data mining, the classical example being query-by-example (find objects that are similar to a given query object), require evaluating a distance function between a large number of points[1]. These calculations often take an important part of the available CPU time and therefore, a relevant problem is how to automatically find accurate approximations to these distance functions but that are efficient to evaluate.

In this paper we present an approach, based on formulating and solving a simple linear programming problem, that allows for automatically building distance functions from examples (1) that are tailored to the problem at hand and (2) that have the additional property of being computationally efficient to evaluate. We are interested in solving the following problem. Let us represent the objects of interest[2] by points $\mathbf{x}_k$ in a D-dimensional space $\Re^D$, with $k = \{1, 2, ..., N\}$. The problem is how to change this representation to points $\hat{\mathbf{x}}_k$ such that distance relationships be-

[1] Other examples include K-means and kernel-based methods in general.

[2] Examples of objects of interest are: database records, user opinions, product characteristics, etc.

tween points are appropriate for the task of interest according to *side information* provided by a user in the form of distance relationships (see Sec. 1.1) and in addition, so that the points lie on a lower dimensional space $\Re^d$.

In order to achieve this, we are interested in finding a transformation $A : \Re^D \rightarrow \Re^d$ that relates any point in the original space to its low dimensional counterpart. When A is a linear transformation, this can be thought of as learning a Mahalanobis distance (*e.g.,* see [18, 12, 16]). Sec. 1.2 describes the connection and differences between the formulation introduced in this paper, the above, and other related methods.

1.1 Specifying distance relationships as side information

In this paper, only distance relationships among a number of points are used to capture the structure of the space, no absolute distances are necessary. These relationships may be provided by a user; thus, it would be beneficial to make the information required from the user (1) simple to obtain and (2) easy to provide. We believe that relationships of the form *object i is closer to j than to k* are both simple to specify and sufficiently informative to capture the properties of the task of interest. In case that an appropriate metric or an algorithm for determining relative similarity were available (but cannot always be used because *e.g.,* it is expensive to evaluate), this information can be obtained more automatically.

In choosing this type of relative relationships, we were inspired by the work in [1]; however, distance relationships of this type were also employed in [15, 12]. Note that we are not interested in preserving absolute distances, which are in general much more difficult to obtain[3].

Another interesting property of this type of distance relationships is that they do not require the concept of class labels (*e.g.,* [16]) or the specification of examples of similar objects vs. dissimilar ones (*e.g.,* [18]). The concepts of similar vs. dissimilar are limited by the fact that a user would need knowledge of at least some (and preferably all) of the rest of the objects in order to determine whether two objects are similar or dissimilar; that is, a reference frame is needed.

1.2 Metric learning with dimensionality reduction

The framework presented in this paper is related to different sets of approaches. A first set can be represented by unsupervised methods that have approached the problem of finding low dimensional representations of the data. Some of these approaches attempt to capture the variance of the data such as Principal Components Analysis [9], while others build low-dimensional embeddings, that is, by transforming a set of data points into a lower dimensional one such that (some or all) distances are preserved. Examples of these approaches include algorithms such as Multidimensional Scaling (MDS) [5], Locally Linear Embeddings (LLE) [11], ISOMAP [13], and low-dimensional embeddings via SDP [17]. These methods can implicitly reduce the amount of computation regarding distance calculations; however due to their purely unsupervised nature, they rely on a distance function to be given and cannot build a function such that the accuracy of certain (*e.g.,* classification) algorithms is improved. In other words, they are not designed to capture the concept of an appropriate distance.

Another set of approaches that are related to ours in a different manner are those that attempt to learn an appropriate distance function from examples. The most closely related approaches are [18, 12, 1]. The first two require some form of supervision and are designed just to learn good distance functions, without explicitly attempting to improve the efficiency of distance calculations. BoostMAP [1] is the most related to our approach in terms of the goals targeted, that is, finding a distance function that is both accurate for the task and efficient to evaluate. Like our method, both [1] and [12] attempt to preserve distance relationships. However, BoostMAP, based on using AdaBoost to combine multiple 1D embeddings to preserve the proximity structure of the data, has the disadvantage that it leads to an iterative, greedy algorithm to optimize the embedding and thus does not have strong optimality guarantees. In contrast, the method in [12] proposes a convex optimization problem based on SVMs. Its disadvantages include the facts that the problem requires quadratic programming and is only designed to find appropriate weights for the different coordinates of the data; thus a very small subset of linear transformations is explored to obtain a solution. When comparing the properties of the solution space, [18] is the most related approach. This is based on finding a square matrix that defines a Mahalanobis distance. The distance is optimized to respect distance constraints represented by two sets, one of similar and one of dissimilar points. The formulation leads to a convex optimization problem.

As we will show, while the general problem formulated in this paper does not appear to accept efficient algorithms, a proposed comparable problem can be solved by using just linear programming.

Table 1 shows a collection of approaches that share some of the goals or motivations of the approach presented in this paper. It also highlights several important distinguishing attributes. In particular we have considered (1) computational efficiency: whether the method attempts to find low dimensional representations for efficient distance evaluation, (2) generalization to new points: whether the method can easily generalized to new unseen points, (3) distance learning capability: whether it can learn a distance function from user examples, and (4) power of learning algorithm: whether the learning algorithm finds local-free optima (such as it is the case for convex formulations). The method proposed in this paper has a unique set of positive attributes that provide significant advantages over recent approaches.

1.3 Notation and background

In the following, vectors will be assumed to be column vectors unless transposed to a row vector by a superscript $\top$. The scalar (inner) product of two vectors $\mathbf{x}$ and $\mathbf{y}$ in the n-dimensional real space $\Re^n$ will be denoted by $\mathbf{x}^\top\mathbf{y}$, for any n. The 2-norm and 1-norm of $\mathbf{x}$ will be denoted by $\|\mathbf{x}\|_2$ and $\|\mathbf{x}\|_1$ respectively. For a matrix $A \in \Re^{m\times n}$, $A_i \in \Re^n$ denotes a column vector formed by the elements of the i-th column of A. A column vector of ones of arbitrary dimension will be denoted by $\vec{e}$, and one of zeros will be denoted by $\vec{0}$. The identity matrix of arbitrary dimension will be denoted by I. For a matrix A, we use $A \succeq 0$ to indicate that this is positive semidefinite.

[3] Absolute distances imply relative distances, but the converse is not true.

Table 1: Related Approaches for Dimensionality Reduction and Metric Learning (Y=Yes, N=No)

Approach	Low dim (efficient evaluat)	Generaliz. to unseen data	Learns from examples	Local-minima free (convexity)	
Athitsos et al. [1]	Y	Y	Y	N	
FastMap [6]	Y	Y	N	N	
MDS [5]	Y	N	N	N	
Schultz-Joachims[12]	N	Y	Y	Y	(QP)
Wagstaff et al. [10]	N	Y	Y	N	
Weinberger et al. [16]	Y	Y	N	Y	(SDP)
Xing et al. [18]	N	Y	Y	Y	(Iterat. Proj.)
This paper	Y	Y	Y	Y	(LP)

2. LEARNING METRICS

Let us say we are given a set of points $\mathbf{x}_k \in \Re^D$ with $k = \{1, ..., N\}$ for which an appropriate distance metric is unknown or expensive to compute. In addition we are given information about a few relative distance comparisons. Formally, we are given a set $\mathcal{T} \subset \{(i,j,k) | f(\mathbf{x}_i, \mathbf{x}_j) < f(\mathbf{x}_i, \mathbf{x}_k)\}$ for some *distance* function f. As indicated above, f may not be known explicitly, instead a user may only be able to provide these distance relationships sparsely or by example. We are interested in finding a linear transformation $A : \Re^D \to \Re^d$ such that:

$$\forall (i,j,k) \in \mathcal{T}, ||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||_2^2 < ||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_k||_2^2 \qquad (1)$$

where $\hat{\mathbf{x}}_k = A\mathbf{x}_k$. Additionally, since we would like to account for efficiency, we will prefer transformations such that $d \ll D$.

In summary, we would like to find a new representation of the original space that uses fewer dimensions and where the L_2 norm respects the desired distance relationships.

For ease of notation, we represent the transformation as a square matrix $A \in \Re^{D \times D}$. It is easy to show that if $A_k = \vec{0}$ (the k-th column of A is equal to the zero vector), then the k-th original dimension (in $\Re^D$) can be ignored when calculating the projection and subsequently when evaluating the implied distance function. If z is the number of columns of A that are zero, then $d = D - z$ (the dimension of the space induced by A clearly depends on z). Because of the relationship between this type of matrix sparsity and efficient metric distance evaluation, we have used the term *sparse metric.*

3. BASE FORMULATION

Putting the above ideas together, the projection matrix A can be formally defined as the optimal solution to the following optimization problem:

$$\max_{A:\Re^D \to \Re^d} \sum_{m=1}^{D} \mathbf{1}(A_m = \vec{0})$$
$$s.t.$$
$$\forall (i,j,k) \in \mathcal{T}, ||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||_2^2 \quad < \quad ||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_k||_2^2, \qquad (2)$$

where the set $\mathcal{T}$ of triples indicates the distance relationships to be satisfied and $\mathbf{1}(\mathcal{E})$ is the indicator function which return the value 1 if the logical expression $\mathcal{E}$ evaluates to true and zero otherwise. The above definition of A is useful at formalizing the desired concept of an optimal projection in the sense presented in this paper. However, as formulated, it is not amenable to practical calculation since it is unclear whether there exist an efficient algorithm for finding A given the set $\mathcal{T}$. We now concentrate on formulating similar problems that can be more efficiently approached.

Note that for $d < D$ the feasible set for the above problem could be empty. In that case, there is no matrix A that can solve the problem and A is undefined. Since in practice we may still be interested in finding a *good* matrix A, even if it does not satisfy all of the constraints, in the following we also study variations of the problem to address these cases.

4. CONVEX FORMULATIONS

Here we concentrate on convex approximations to the problem in Sec.3.

4.1 Optimizing for A

In order to address the discrete nature of the base formulation, we transform the overall problem into one with a continuous cost function. Additionally, to address the case where the feasible set is empty, we relax the constraints by introducing slack variables ϵ_t. The problem can be reformulated as follows:

$$\begin{array}{rcl} \min_{A:\Re^D \to \Re^d} \sum_t \epsilon_t & + & \alpha \sum_{m=1}^{D} ||A_m||_1 \\ s.t. & & \\ \forall (i,j,k) \in \mathcal{T}, ||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||_2^2 & \leq & ||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_k||_2^2 + \epsilon_t \\ \forall t, \epsilon_t & \geq & 0 \end{array} \qquad (3)$$

where t indexes the set $\mathcal{T}$, $\alpha \in \Re$ is a scalar that controls the trade off between the sparsity of A and compliance with the inequalities generated by the triples in $\mathcal{T}$, and $\epsilon_t \in \Re$ represents a slack variable. This new problem is clearly not equivalent to the base formulation; however, the 1-norm tends to suppress terms and to produce sparse solutions. This has been empirically validated, in particular in the SVM framework *e.g.,* [4, 7]. Hence, the expression $\sum_{m=1}^{D} ||A_m||_1$ in formulation (3) will be used to replace the ideal expression $-\sum_{m=1}^{D} \mathbf{1}(A_m = \vec{0})$. The later would lead to a 0-1 Mixed Integer Programming (MIP) problem, known to be NP-hard.

Each distance constraint can be written as:

$$(A\mathbf{x}_i)^\top (A\mathbf{x}_i) - 2(A\mathbf{x}_i)^\top (A\mathbf{x}_j) + (A\mathbf{x}_j)^\top (A\mathbf{x}_j)$$
$$-(A\mathbf{x}_i)^\top (A\mathbf{x}_i) + 2(A\mathbf{x}_i)^\top (A\mathbf{x}_k) - (A\mathbf{x}_k)^\top (A\mathbf{x}_k) \leq \epsilon_t, \quad (4)$$

and by defining $B = A^\top A \in \Re^D \times \Re^D$, Eq. 4 can be further simplified to:

$$\begin{array}{rcl}(\mathbf{x}_j^\top B\mathbf{x}_j) - (\mathbf{x}_k^\top B\mathbf{x}_k) + 2[(\mathbf{x}_i^\top B\mathbf{x}_k) - (\mathbf{x}_i^\top B\mathbf{x}_j)] & \leq & \epsilon_t \\ B & \succeq & 0 \\ B & = & B^\top \end{array} \quad (5)$$

The main advantage of the new equation is that it produces linear constraints in the new variable B, instead of quadratic constraints in A. However, in order for the equivalence to hold, we must have B symmetric and positive semidefinite. Assuming the cost function remains convex in B, this problem is still convex in B. Technically, this formulation is sufficient to approach the problem at hand. However, it becomes a (more expensive to solve) semidefinite programming problem (SDP). Next, we will show how the much more efficient linear programming method (LP) can be employed instead to solve different instances of this formulation.

4.2 Optimizing for $B = A^\top A$

We first focus on finding a cost function equivalent to Eq. 3. For this we note that for $B = A^\top A$:

$$A_k = \vec{0} \Leftrightarrow B_k = \vec{0}, (B^\top)_k = \vec{0}. \quad (6)$$

At the distance calculation level, we can easily note that since:

$$||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||_2^2 = ||A(\mathbf{x}_i - \mathbf{x}_j)||_2^2 = (\mathbf{x}_i - \mathbf{x}_j)^\top B(\mathbf{x}_i - \mathbf{x}_j)$$

we have that:

$B_k = \vec{0} \Rightarrow ||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||_2^2$ does not depend on dimension k as expected.

We can then define a new cost function in terms of B, and arrive to the following problem:

$$\begin{array}{rcl} \multicolumn{3}{l}{\min_{\epsilon,B} \sum_t \epsilon_t + \lambda \sum_{m=1\ldots D} ||B_m||_1} \\ & s.t. & \\ \forall(i,j,k) \in \mathcal{T}, -2(\mathbf{x}_i^\top B\mathbf{x}_j) + (\mathbf{x}_j^\top B\mathbf{x}_j) & & \\ -2(\mathbf{x}_j^\top B\mathbf{x}_k) + (\mathbf{x}_k^\top B\mathbf{x}_k) & \leq & \epsilon_t \\ \forall t, \epsilon_t & \geq & 0 \\ B & = & B^\top \\ B & \succeq & 0, \end{array} \quad (7)$$

where the set of constraints involving B follow from Eqs. 5 This is also a semi-definite programming (SDP) problem which is convex and can be solved using specialized SDP solvers like [14]. However, we will further develop the formulation presented above by restricting our solution space to a subfamily of the PSD matrices: the set of diagonal dominant matrices.

4.2.1 *Imposing diagonal dominance on* B

In order to provide a better understanding of the motivation for our next formulation we present the following theorem stated in [8]:

THEOREM 4.1. **Diagonal Dominance Theorem** *Suppose that* $M \in \Re^{D\times D}$ *is symmetric and that for each* $i = 1, \ldots, n$ *, we have:*

$$M_{ii} \geq \sum_{j\neq i} |M_{ij}|,$$

then M *is positive semi-definite (PSD). Furthermore, if the inequalities above are all strict, then* M *is positive definite.*

Based on the diagonal dominance theorem (for matrices with positive diagonal elements) above, we arrive to the following formulation, which constrains the feasible set to a subset of that in the formulation (7):

$$\begin{array}{rcl} \multicolumn{3}{l}{\min_{\epsilon,B} \quad \sum_t \epsilon_t + \lambda \sum_{m=1\ldots D} |B_{mm}|} \\ & s.t. & \\ \forall(i,j,k) \in \mathcal{T}, -2(x_i^\top B\mathbf{x}_j) + (\mathbf{x}_j^\top B\mathbf{x}_j) & & \\ -2(x_j^\top B\mathbf{x}_k) + (\mathbf{x}_k^\top B\mathbf{x}_k) & \leq & \epsilon_t \\ \forall t, \epsilon_t & \geq & 0 \\ B & = & B^\top \\ \forall(m) B_{mm} & \geq & \sum_n |B_{mn}|, \end{array} \quad (8)$$

where the last constraint is equivalent to diagonal dominance which implies positive semidefiniteness according to theorem 4.1. As it was explained before, the sum of 1-norms in the cost function causes preference for sparse solutions. The second term in the cost function combined with the last constraint has a similar effect. Note also that since $B_{mm} \geq 0$ (as implied by the constraints), the second term is equivalent to Trace(B).

The projection matrix A can be recovered by Cholesky factorization of the symmetric matrix B [8] or by an eigenvalue decomposition. The constraint involving $\mathbf{x}$ and B can be rewritten as follows. For any $\mathbf{x}_i, \mathbf{x}_j \in \Re^D$, define $\tilde{X}_{ij} = \text{vect}(\mathbf{x}_i\mathbf{x}_j^\top)$, where vect() means (column-wise) alignment of all of the matrix elements in a column vector. Define $b = \text{vect}(B)$. The constraint can now be written:

$$\forall(i,j,k) \in \mathcal{T}[\tilde{X}_{jj} + \tilde{X}_{kk} - 2(\tilde{X}_{ij} + \tilde{X}_{jk})]b \leq \epsilon_t \quad (9)$$

Finally, using Eqs. 9 and defining auxiliary variables $S_{mn} \in \Re$ $(m, n \in \{1, ..., D\})$, formulation (8) can be rewritten as a Linear Program in the following way:

$$\begin{array}{rcl} \multicolumn{3}{l}{\min_{\epsilon,B,S} \quad \sum_t \epsilon_t + \lambda \sum_{m=1\ldots D} B_{mm}} \\ & s.t. & \\ \multicolumn{3}{l}{\forall(i,j,k) \in \mathcal{T}} \\ [\tilde{X}_{jj} + \tilde{X}_{kk} - 2(\tilde{X}_{ij} + \tilde{X}_{jk})]b & \leq & \epsilon_t - 1 \\ \forall t, \epsilon_t & \geq & 0 \\ B & = & B^\top \\ \forall(m, n, m \neq n) - S_{mn} & \leq & B_{mn} \leq S_{mn} \\ B_{mm} & \geq & \sum_{\substack{n=1 \\ m\neq n}}^{D} S_{mn} \end{array} \quad (10)$$

Since the matrix B is symmetric, the number of components of B to be found can be reduced to $D(D+1)/2$ instead of D^2, furthermore by doing this, the constraints $B = B^\top$ can be discarded. The right hand side of the first set of inequalities has been changed from ϵ_t to $\epsilon_t - 1$ in order to enforce numerical stability and to avoid obtaining the trivial solution $B = 0$. In order to better understand the motivation for formulation (10) it is important to note that:

(i) Minimizing $\sum_{m=1\ldots D} B_{mm}$ is equivalent to minimizing $\sum_{\substack{n=1 \\ m\neq n}}^{D} S_{mn}$ since $B_{mm} \geq \sum_{\substack{n=1 \\ m\neq n}}^{D} S_{mn}$.

(ii) Since we are implicitly minimizing $\sum_{\substack{n=1 \\ m\neq n}}^{D} S_{mn}$, at the optimal solution $\{B^*, S^*, \epsilon^*\}$ to problem (10), we have that:

$$0 \geq S_{mn}^* = |B_{mn}^*|, \forall(m, n, m \neq n)$$

(*iii*) Combining (*i*) and (*ii*) we obtain:

$$\forall(m) B^*_{mm} \geq \sum_n S^*_{mn} = \sum_n |B^*_{mn}|$$

which implies that B^* is diagonal dominant and hence positive semidefinite.

This last formulation works quite effectively in practice as indicated by the numerical examples presented in the next section.

5. EXPERIMENTAL EVALUATION

This section presents numerical results obtained by applying the above formulation to the problem of learning metrics.

5.1 Experimental Setting

We have tested our approach in a collection of nine publicly available datasets, part of the UCI repository [4]. These datasets are summarized in Table 2. These datasets are commonly used in machine learning tasks as benchmark for performance evaluation. Our choice is majorly motivated by their use to evaluate an alternative approach[18]. This is an state-of-the art method whose code has been made public [5] This method compared well against K-means in the task of finding a distance function that produced *good* clusterings.

This methods finds a linear transformation of the data that respects *similar* vs. *dissimilar* constraints.

The datasets employed in these experiments are generally used for classification since class labels are available for every point. Our method does not require class labels, but instead only relative distance comparisons among a subset of points (clearly class labels provide more information). We use the available class labels to generate a set of triples with distance comparisons that respect the classes. More explicitly, given a randomly chosen set of three points (from the training set), if two of these belong to the same class and a third belongs to a different class, then we place this triple in our set $\mathcal{T}$ (*i.e.,* i and j are the points in the same class, k is the remaining point). In other words, in order to make use of the available class labels, we decided that points in the same class should have smaller pairwise distances between themselves than with points in any of the other classes (after projected by A). For [18], the supervision is in the form of two sets, one called a *similar* set and the other a *dissimilar* set. For this model, we can again use the class labels, now to build a similar set of pairs (likewise for a dissimilar set of pairs). Given this level of supervision, this method attempts to find an optimal Mahalanobis distance matrix to have same-class points closer to each other than different-class points (see [18] for details).

For every triple $(i, j, k) \in \mathcal{T}$ used in our approach for learning, we use $(i, j) \in \mathcal{S}$ and $(i, k) \in \mathcal{D}$ for learning in [18]; where $\mathcal{S}$ and $\mathcal{D}$ are the similar and dissimilar sets required. We believe this provides a fair level of supervision for both algorithms since roughly the same information is provided. It is possible to obtain a superset of $\mathcal{T}$ from $\mathcal{S}$ and $\mathcal{D}$, and by construction $\mathcal{S}$ and $\mathcal{D}$ can be obtained from $\mathcal{T}$.

In order to evaluate performance, we use a 0.85/0.15 split of the data into training and testing. From the training portion, we generate 1500 triples, as explained above, for actual training. This information is provided, in the appropriate representation, to both algorithms. For testing, we randomly choose three points, and if their class labels imply that any two points are closer to each other than to a third (*i.e.,* again if two points have the same class and a third has a different class label), then we check that the correct relationships are satisfied. That is, whether the two points in the same class are closer to each other than any of these points (chosen at random) to the third point. This same measure is used for both algorithms. Thus, we define the *percentage correct* simply as the proportion of points from the test set (sampled at random) that respect the class-implied distance relationship. Both methods compared are attempting to improve this performance measure given the training data and thus we believe this is a valid measure.

5.2 Discussion

Since our method requires setting the balancing parameter λ, we chose it using cross validation (on the training set) by letting λ take values in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2\}$. This indirectly influences d, the optimal number of dimensions the data should be projected to, since a small λ favors low-dimensionality (*i.e.,* a λ close to zero practically ignores the number of non-zero dimensions and concentrate on just fitting the data). However, note that λ does not imply dimensionality since the dimensionality depends on the randomly built training set itself. This automatic choice of dimensionality is a valuable property of the method presented, and to the best of our knowledge, it is not present in the related methods (with the exception, to some degree, of [1]).

Fig. 1 shows the average optimal number of dimensions found by this process in a 10-fold cross validation experiment and the corresponding one-standard-deviation error bars. Note than in some cases the reduction is considerably large, this reduction depends on the properties of the dataset. The number of dimensions was identified by looking at the number of rows in A different than $\vec{0}$; no thresholding was necessary.

Fig. 2 shows the percentage correct averaged over 10 random splits of the data along with one-standard-deviation bars. For each of the 10 splits, 1000 triples from the test set are randomly chosen. When comparing the performance of both methods, we note that, except for dataset 5, our method clearly outperforms the competing approach. Interestingly, for this dataset, the optimal dimension was determined to always (for all randomly chosen data splits) be equal to the original dimensionality (four dimensions).

Since both methods can be seen as trying to learn a Mahalanobis distance so that distance constraints are satisfied, we believe the main reason for a superior performance is related to the possibility to discover projections into lower dimensional spaces. The method presented in this paper always attempt to reduce the dimensionality while the competing method always use all the dimensions. Given the results obtained, this reduction appears to provide an important advantage at the time of generalization. It is generally accepted that a simpler model (*i.e.,* one with less parameters) is preferable (*e.g.,* [3]) and it can reduce overfitting.

Regarding computational at training time. As expected,

[4] http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[5] Data for all experiments and code for [18] was downloaded from http://www.cs.cmu.edu/∼epxing/papers/. The class for dataset 1 was obtained by thresholding the median value attribute to 25K.

Table 2: Benchmark Datasets

Name	Points (N)	Dimensions (D)	Classes
1 Housing-Boston	506	13	2
2 Ionosphere	351	34	2
3 Iris	150	4	3
4 Wine	178	13	3
5 Balance Scale	625	4	3
6 Breast-Cancer Wisconsin	569	30	2
7 Soybean Small	47	35	4
8 Protein	116	20	6
9 Pima Diabetes	768	8	2

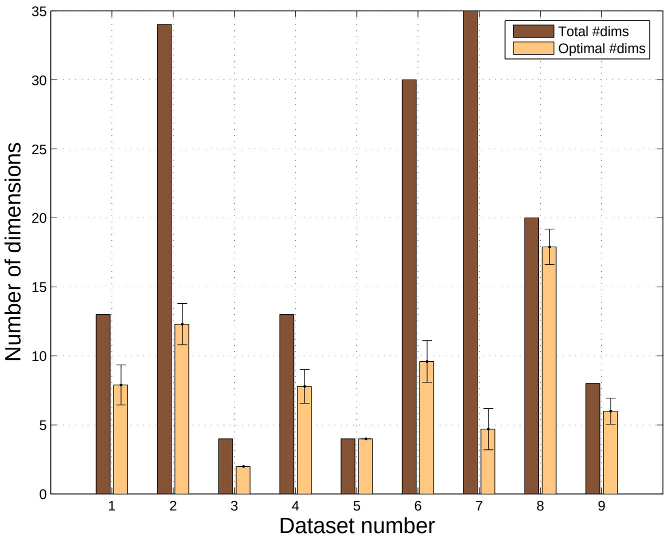

Figure 1: Dimensionality reduction. Total number of dimensions and average number of dimensions (along with one-standard-deviation error bars) found by our algorithm for each dataset. Averages are computed over 10 random splits of training/test points, and 1500 triples per run.

parameter search affects overall run-time (linearly). The cost is proportional to the number of λ values explored. Given λ, training time is comparable with [18]. In terms of memory requirements, since the solution involves finding a full (symmetric) matrix, the parameter space increases quadratically with D.

From a computational efficiency perspective at test time, being able to represent the original data more succinctly is especially advantageous. In particular, when distances can be calculated directly using a low dimensional representation, computational time savings can be significant for on-line applications. The projection step in this approach can be precomputed off-line. In retrieval applications (*e.g.,* query-by-example), the objects can be stored in their low dimensional representation.

From a conceptual point of view, our approach also has the advantage, over other methods, of providing a more effective tool for understanding the data since it can identify whether variables (dimensions) are of high or low relevance for a task of interest.

6. CONCLUSIONS

We have developed a new approach for learning distance functions from a set of relative distance relationships. An important property of this approach is that it targets lower dimensional representations, and the dimensionality is determined automatically depending on the characteristics of the dataset in question and a balancing parameter λ. A key distinction is that, unlike a large number of dimensionality reduction approaches, our approach does not attempt to build an isometry or distance preserving mapping[6], but to respect the proximity relationships between pairs of points. We believe this allows more freedom for finding lower dimensional representations.

We considered the general problem and then designed specific formulations, based on minimizing the norm of the rows of a transformation matrix A, that allowed the use of efficient convex optimization algorithms. In particular, we showed how the diagonal dominance constraint on $B = A^\top A$ leads to a general formulation that can be solved very efficiently using linear programming methods.

Our approach can also be seen as a form of supervised dimensionality reduction, where the supervision comes in the form of distance rankings.

We note that another way to arrive at the formulation (10) is by noticing that $A_k = \vec{0} \Leftrightarrow B_{kk} = 0$, which also justifies the second term of the chosen cost function.

The results from the experimental evaluation show that our method can outperform state-of-the-art approaches in terms of accuracy, and also provides the additional benefit of finding low dimensional representations.

Being able to obtain a distance that depends on a relatively *small* number of features permits to define kernels and/or similarity matrices for classification that may depend on an small number of features. For example, instead of using the standard Gaussian kernel (μ is the Gaussian kernel parameter):

$$(K(X,Y))_{ij} = e^{-\mu\|X_i{}^\top - Y_j\|^2},$$

that depends on all the features, including irrelevant features for classification, we could use a modified Gaussian kernel as follows:

$$(\bar{K}(X,Y))_{ij} = e^{-\mu\|A(X_i{}^\top - Y_j)\|^2},$$

[6] For example by minimizing distortion.

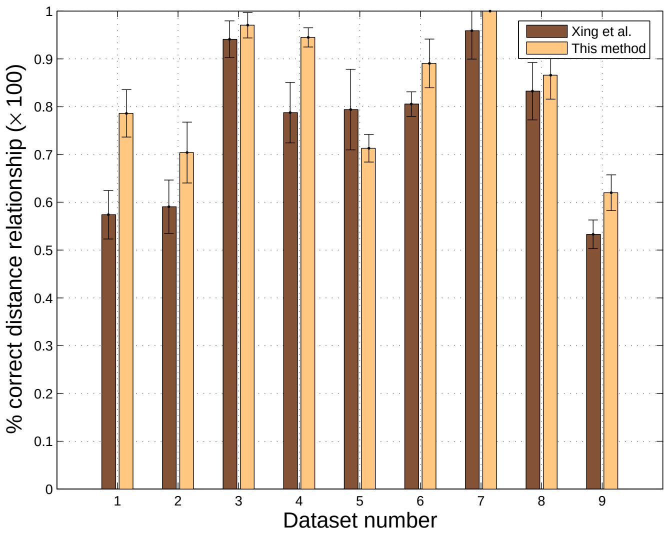

Figure 2: Performance comparison between competing approach and our approach in nine UCI datasets. Bars show performance results on 10 random splits of training/test points. Performance is measured in terms of the percentage of randomly chosen points (1000) from test set whose distance relationship respect the class labels. The number of triples used for training for all runs was 1500. Error bars show one standard deviation.

that would only depend on relevant features. This simple but powerful change may increase classification performance considerably and it is part of our future work.

Another idea worth exploring is the application of LP-boost algorithms [2] to allow our method the handling of datasets in very high dimensional spaces.

The approach followed in this paper have possible implications in other areas. Our results suggest that the technique applied in this paper to approximate an SDP problem by a linear programming problem (which is much easier to solve) has the potential to be applied to other recently proposed machine learning related problems involving SDP formulations.

7. REFERENCES

[1] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. Boostmap: A method for efficient approximate similarity rankings. In *Computer Vision and Pattern Recognition*, 2004.

[2] K. P. Bennett, A. Demiriz, and J. Shawe-Taylor. A column generation algorithm for boosting. In *Proc. 17th International Conf. on Machine Learning*, pages 65–72. Morgan Kaufmann, San Francisco, CA, 2000.

[3] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

[4] P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In J. Shavlik, editor, *Machine Learning Proceedings of the Fifteenth International Conference(ICML '98)*, pages 82–90, San Francisco, California, 1998. Morgan Kaufmann. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-03.ps.

[5] T. Cox and M. Cox. *Multidimensional Scaling.* Chapman & Hall, London, 1994.

[6] C. Faloutsos and K. Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *ACM SIGMOD*, pages 163–174, 1995.

[7] G. Fung, O. L. Mangasarian, and A. Smola. Minimal kernel classifiers. *Journal of Machine Learning Research*, pages 303–321, 2002. University of Wisconsin Data Mining Institute Technical Report 00-08, November 200, ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-08.ps.

[8] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The John Hopkins University Press, Baltimore, Maryland, 3rd edition, 1996.

[9] I. Jolliffe. *Principal Component Analysis.* Springer-Verlag, New York, 1989.

[10] K.Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *International Conference on Machine Learning*, 2001.

[11] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[12] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems*, 2003.

[13] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[14] K. C. Toh, M. J. Todd, and R. Tutuncu. SDPT3 — a Matlab software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1999.

[15] W.Cohen, R.Schapire, and Y. Singer. Learning to order things. In *Advances in Neural Information Processing Systems 10*, 1998.

[16] K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems 18*, 2006.

[17] K. Q. Weinberger, B. D. Packer, and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, Barbados, January 2005.

[18] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side information. In *Advances in Neural Information Processing Systems*, 2002.