2012 IEEE 12th International Conference on Data Mining

Metric Learning From Relative Comparisons by Minimizing Squared Residual

Eric Yi Liu*†, Zhishan Guo*†, Xiang Zhang*‡, Vladimir Jojic*† and Wei Wang*†§
* {liuyi, zsguo}@cs.unc.edu, xiang.zhang@case.edu, vjojic@cs.unc.edu, weiwang@cs.ucla.edu
† Department of Computer Science, University of North Carolina at Chapel Hill
‡ Department of Electrical Engineering and Computer Science, Case Western Reserve University
§ Department of Computer Science, University of California at Los Angeles

***Abstract*—Recent studies [1]–[5] have suggested using constraints in the form of relative distance comparisons to represent domain knowledge: $d(a, b) < d(c, d)$ where $d(\cdot)$ is the distance function and a, b, c, d are data objects. Such constraints are readily available in many problems where pairwise constraints are not natural to obtain. In this paper we consider the problem of learning a Mahalanobis distance metric from supervision in the form of relative distance comparisons. We propose a simple, yet effective, algorithm that minimizes a convex objective function corresponding to the sum of squared residuals of constraints. We also extend our model and algorithm to promote sparsity in the learned metric matrix. Experimental results suggest that our method consistently outperforms existing methods in terms of clustering accuracy. Furthermore, the sparsity extension leads to more stable estimation when the dimension is high and only a small amount of supervision is given.**

***Keywords*-metric learning; Mahalanobis metric; relative comparisons;**

I. Introduction

In many machine learning problems such as clustering and nearest-neighbor classification, distance metrics are essential in measuring the distance or similarity between objects. Recent studies [1], [6]–[8] have suggested that task-specific metric learning can help understand the relationships among features and the resulting metrics outperform off-the-shelf distance functions such as the Euclidean distance in many problems. Such task-specific metrics are often learned from examples or side knowledge in the form of constraints. For example, many existing methods (see, for survey, [9]) focus on learning metric from pairwise constraints: "a and b are similar", "c and d are not similar". In clustering problems, these pairwise constraints are often translated into Must-Link (a and b are in the same cluster) and Cannot-Link (c and d are in different clusters) constraints.

Here, we consider the problem of learning a distance metric from constraints in the form of relative comparisons: "the distance between two objects are greater (or smaller) than the distance between another two objects". Denoting the four objects by a, b, c, d, we can write the comparison as:

$$d(a, b) < d(c, d)$$

where $d(\cdot)$ is the distance function. When the comparison is defined only on three objects, it is of the form: $d(a, b) < d(a, c)$. Unlike pairwise constraints that are absolute qualitative feedback, relative comparisons are more natural in representing vague domain knowledge. For instance, a domain expert usually has some general sense of the closeness (similarity) among a subset of data objects. Such knowledge of closeness may be vague and only in a relative sense, that is, we can only determine, between two objects, which one is closer to a given third object, but we cannot determine an absolute closeness measure.

Knowledge of relative comparisons are readily available in many real world problems. One example is the ranking problem in search engine (ranking search results or relevance based advertisements). Query and user-click logs can effectively provide knowledge in relative comparisons but not easily expressed in pairwise constraints [1]. A metric learned from these comparisons can reveal feature importance and feature-feature interactions. Thus it can provide valuable guidance in feature-engineering of ranking problems (e.g., to design cross-features). Another example is clustering with structural knowledge as supervision. We have previously shown that paired relative comparisons can be used to express structural knowledge unambiguously [5]. Existing structural knowledge sources such as taxonomical databases can be effectively translated into a minimum set of relative comparisons. For instance, one can conduct clustering or classification analysis on gene-expression data with comparisons extracted from partial hierarchical knowledge in the Gene Ontology database.

In this paper, we present a simple, yet effective, algorithm that learns a Mahalanobis metric from a given set of relative comparisons. This is done by formulating and minimizing a convex loss function that corresponds to the sum of squared hinge loss of violated constraints. We also give an extension to our model and algorithm to incorporate entry-wise ℓ_1 penalty that encourages sparsity in the learned metric matrix. We evaluate the performance of our algorithm and two existing methods in supervised clustering problems using real-world datasets. To the best of our knowledge, this is the first direct evaluation of methods that learn metrics from relative comparisons and provides insights into practical applications. Experiments with metrics of various dimensions suggest that our algorithm consistently outperforms existing methods in terms of clustering accuracy. With sparsity imposed, our algorithm can achieve more stable estimation when the dimension is high and only a small amount of supervision is given.

Throughout the paper, we denote matrices by uppercase letters and vectors by lowercase letters in bold font. $\mathbf{x}_{(i)}$ denotes the i-th entry in vector $\mathbf{x}$ and $M_{(i,j)}$ denotes the (i,j)-th entry of the matrix M. A positive-semidefinite matrix M is denoted as $M \succeq 0$ and we denote the set of $m \times m$ symmetric positive-semidefinite matrices by S_+^m. The trace and Frobenius norm of matrix M are denoted as $\mathbf{tr}(M)$ and $\|M\|_F$.

II. Problem Formulation

Let $X = [\mathbf{x}_1, ...\mathbf{x}_n]$ be the set of data objects where each data object $\mathbf{x}_i \in R^m$. A set of relative comparisons is given in

1550-4786/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDM.2012.38

the form of:

$$\mathcal{C} = \{(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d) : d(\mathbf{x}_a, \mathbf{x}_b) < d(\mathbf{x}_c, \mathbf{x}_d)\}$$

Here we consider the distance function

$$d_M(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T M (\mathbf{x}_i - \mathbf{x}_j)}$$

where M is the Mahalanobis matrix we wish to learn. In order for M to be a proper metric matrix, M has to be a $m \times m$ positive-semidefinite matrix and we consider M to be also a symmetric matrix that can be written as $M = AA^T$ where A is a $m \times m$ matrix that represents a linear transformation from the original feature space to a learned feature space.

A. Loss Function of Constraints

We propose to optimize over the sum of squared residuals in satisfying constraints. Given a Mahalanobis metric $M = AA^T$, the residual is zero for satisfied constraints and is the difference of the M-defined distances (i.e., the Euclidean distances in the A-transformed space) for violated constraints. This gives the loss function of each constraint $d(\mathbf{x}_a, \mathbf{x}_b) < d(\mathbf{x}_c, \mathbf{x}_d)$:

$$L(d(\mathbf{x}_a, \mathbf{x}_b) < d(\mathbf{x}_c, \mathbf{x}_d)) = H(d_M(\mathbf{x}_a, \mathbf{x}_b) - d_M(\mathbf{x}_c, \mathbf{x}_d))$$

where $H(\cdot)$ is the squared hinge function defined as:

$$H(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x^2 & \text{if } x > 0. \end{cases}$$

The rationale behind the loss function is analogous to minimizing squared residual in regression. If most information in constraints are correct, we should expect that the learned M would perform reasonably well so that most constraints are satisfied or close to the satisfaction boundary. Some of the constraints may not be satisfied due to noise or expressibility of Mahalanobis distance. But we prefer them not to deviate much from the satisfaction boundary by imposing the squared penalty. With squared loss upon violation, we obtain more information than in the traditional way of casting relative comparisons into linear constraints [1], [3], [4]. This is also conceptually different from maximizing the number of constraints satisfied as in [2].

The summed loss function for all constraints is:

$$\begin{aligned} L(\mathcal{C}) &= \sum_{(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d) \in \mathcal{C}} w_{a,b,c,d} H(d_M(\mathbf{x}_a, \mathbf{x}_b) - d_M(\mathbf{x}_c, \mathbf{x}_d)) \\ &= \sum_{(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d) \in \mathcal{C}} w_{a,b,c,d} H(\sqrt{(\mathbf{x}_a - \mathbf{x}_b)^T M (\mathbf{x}_a - \mathbf{x}_b)} - \\ &\qquad \sqrt{(\mathbf{x}_c - \mathbf{x}_d)^T M (\mathbf{x}_c - \mathbf{x}_d)}) \end{aligned}$$

$w_{a,b,c,d}$ is the weight associated with each constraint. Such weights can be derived from the confidence or probability of each constraint or can simply be uniform when there is no extra knowledge.

B. Regularized Objective Function

We also impose a regularization term on the overall loss function so that the learned metric M conforms to prior knowledge, especially when the number of constraints is small. This is done by adding a "closeness" measure between M and a given Mahalanobis metric (usually the identity matrix or the precision matrix). Here we choose the LogDet divergence [6]. The overall objective function we wish to minimize is then:

$$\begin{aligned} &\mathcal{D}_{ld}(M, M_0) + \\ &\sum_{(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d) \in \mathcal{C}} w_{a,b,c,d} H(d_M(\mathbf{x}_a, \mathbf{x}_b) - d_M(\mathbf{x}_c, \mathbf{x}_d)) \end{aligned} \quad (1)$$

where M_0 is a given prior metric matrix, and

$$\mathcal{D}_{ld}(M, M_0) = \mathbf{tr}(MM_0^{-1}) - \log \det(MM_0^{-1}) - d$$

Since M_0 is a constant matrix, we replace the above definition by:

$$\mathcal{D}_{ld}(M, M_0) = \mathbf{tr}(MM_0^{-1}) - \log \det(M)$$

which leads to the same solution.

C. Convexity of the Objective Function

Despite the usually unwanted square root operation, the objective function to minimize is in fact a convex function of M over the set of positive-semidefinite matrices.

Lemma 2.1: The loss function of each constraint is a convex function of M over the set of positive-semidefinite matrices. Proof of Lemma 2.1 is omitted due to space limit.

Since $\mathcal{D}_{ld}(M, M_0)$ is also a convex function of M, the objective function is a sum of non-negatively weighted convex functions and is thus a convex function of M over the set of positive-semidefinite matrices. By doing some calculus, we can see that the objective function is smooth and its derivative exists for all $M \succeq 0$.

D. The Optimization Problem

The optimization problem we wish to solve can be formally written as:

$$\begin{aligned} &\underset{M}{\text{minimize}} \ (\mathcal{D}_{ld}(M, M_0) + \\ &\qquad \sum_{(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d) \in \mathcal{C}} w_{a,b,c,d} H(d_M(\mathbf{x}_a, \mathbf{x}_b) - d_M(\mathbf{x}_c, \mathbf{x}_d))) \\ &\text{subject to } \ M \succeq 0. \end{aligned} \quad (2)$$

III. Learning Algorithm

We use a simple projected gradient method [10] to ensure that M stays positive-semidefinite. Let $F(M)$ be the objective function (1) parameterized by M, $\mathbf{v}_{cd} = \mathbf{x}_c - \mathbf{x}_d$ and $\mathbf{v}_{ab} = \mathbf{x}_a - \mathbf{x}_b$. The gradient of the objective function is given by:

$$\frac{\partial F(M)}{\partial M_{(i,j)}} = M^{-1}_{0(i,j)} + M^{-1}_{(j,i)} + \sum_{\mathcal{C}} w_{a,b,c,d} \ g_M(\mathbf{v}_{a,b}, \mathbf{v}_{c,d})$$

where $g_M(\mathbf{v}_{a,b}, \mathbf{v}_{c,d}) = 0$ if $\mathbf{v}_{a,b}^T M \mathbf{v}_{a,b} < \mathbf{v}_{c,d}^T M \mathbf{v}_{c,d}$, otherwise:

$$\begin{aligned} &g_M(\mathbf{v}_{a,b}, \mathbf{v}_{c,d}) = \mathbf{v}_{a,b(i)} \mathbf{v}_{a,b(j)} + \mathbf{v}_{c,d(i)} \mathbf{v}_{c,d(j)} \\ &- \frac{\mathbf{v}_{a,b(i)} \mathbf{v}_{a,b(j)} \times \mathbf{v}_{c,d}^T M \mathbf{v}_{c,d} + \mathbf{v}_{c,d(i)} \mathbf{v}_{c,d(j)} \times \mathbf{v}_{a,b}^T M \mathbf{v}_{a,b}}{\sqrt{\mathbf{v}_{a,b}^T M \mathbf{v}_{a,b} \mathbf{v}_{c,d}^T M \mathbf{v}_{c,d}}}. \end{aligned}$$

Below we present our algorithm LSML (Least Squared-residual Metric Learning):

Algorithm *LSML*

Input: M_0, $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$: set of data objects,
$\mathcal{C} = \{(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d)\}$: set of relative comparisons
Output: a Mahalanobis metric M

1) Repeat until M converges:
2) $\nabla M \leftarrow \frac{\partial F(M)}{\partial M}$
3) $M_{best} \leftarrow 0$, $s_{best} \leftarrow F(M)$
4) For each step size l,
5) $M' \leftarrow M - l\nabla M$
6) Decompose M' by $M' = Q\ \Lambda\ Q^T$
7) $M' \leftarrow Q\ \max(\Lambda, 0)\ Q^T$
8) if $F(M') < s_{best}$
9) $M_{best} \leftarrow M', s_{best} \leftarrow F(M')$
10) $M \leftarrow M_{best}$
11) end

As long as the prior matrix M_0 is symmetric, updates in iterations will be symmetric and the resulting metric M stays in S_+^m. Lines (6-7) conduct the projection to S_+^m by enforcing nonnegative eigen values. The convergence is warranted since this is a special case of the projected subgradient method [11].

A. Time Complexity

Two types of operations are performed for a constant number of times in each iteration.

- the constraint loss and gradient calculations
- matrix inversions and eigen decompositions

The first type has complexity $O(|\mathcal{C}| \times m^2)$ and the second type has complexity $O(m^3)$. Thus the total time complexity is $O(\max(|\mathcal{C}|, m) \times m^2)$ per iteration.

IV. An Extension to Incorporate Sparsity

We propose an extension, named SpLSML, to our LSML algorithm to promote sparsity in the learned metric matrix. This can be beneficial when the dimensionality is high and the supervision knowledge (number of constraints) is very limited. The metric learning problem itself can be extremely data hungry as the number of fitted parameters (number of entries in M here) grows quadratically with the number of features. Promoting sparsity in the learned metric matrix could suppress unimportant M entries and lead to more stable estimation with small amount of constraints. It also brings computational benefits in evaluating similarity.

To encourage sparsity in M, instead of optimizing (2), we try to optimize:

$$\begin{aligned} &\underset{M}{\text{minimize}}\ \ (\mathcal{D}_{ld}(M, M_0)\ +\ \lambda \left\|M\right\|_1\ + \\ &\qquad \sum_{(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d) \in \mathcal{C}} w_{a,b,c,d} H(d_M(\mathbf{x}_a, \mathbf{x}_b) - d_M(\mathbf{x}_c, \mathbf{x}_d))) \\ &\text{subject to}\ \ M \succeq 0 \end{aligned} \tag{3}$$

where $\left\|M\right\|_1 = \sum_{i,j} |M_{(i,j)}|$ is the entry-wise ℓ_1 norm of matrix M.

Note that, though the objective function in (3) is still convex of M, its gradient is not smooth over all $M \succeq 0$ due to the ℓ_1-norm term. As a result, with direct gradient descent, convergence to zero is not warranted for entries that should shrink to zero. To deal with this issue, here we propose to use the alternating direction method of multipliers (ADMM) [12], which is intended to blend the decomposability of dual ascent with the superior convergence properties of the method of multipliers. To apply ADMM, we re-write (3) as:

$$\begin{aligned} &\underset{M,P}{\text{minimize}}\ \ (\mathcal{D}_{ld}(M, M_0)\ +\ \lambda \left\|P\right\|_1\ + \delta_{S_+^m}(M)\ + \\ &\qquad \sum_{(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d) \in \mathcal{C}} w_{a,b,c,d} H(d_M(\mathbf{x}_a, \mathbf{x}_b) - d_M(\mathbf{x}_c, \mathbf{x}_d))) \\ &\text{subject to}\ \ M = P \end{aligned} \tag{4}$$

where $\delta_{S_+^m}(M)$ is a characteristic function of the set of symmetric positive-semidefinite matrices:

$$\delta_{S_+^m}(M) = \begin{cases} 0 & \text{if } M \in S_+^m \\ +\infty & \text{otherwise.} \end{cases}$$

The augmented Lagrangian function is then given by:

$$\begin{aligned} L_\rho(U, M, P) \ &= \ \mathbf{tr}(MM_0^{-1}) - \log\det(M) + \delta_{S_+^m}(M) \\ &+ \ \sum_{\mathcal{C}} w_{a,b,c,d} H(d_M(\mathbf{x}_a, \mathbf{x}_b) - d_M(\mathbf{x}_c, \mathbf{x}_d)) \\ &+ \ \lambda \left\|P\right\|_1 + \langle U, M - P\rangle + \frac{\rho}{2}\left\|M - P\right\|_F^2 \end{aligned}$$

where $\rho \in R$ is the penalty parameter that only affects the convergence rate and $U \in R^{m \times m}$ is a self-updated matrix that links M and P. $\langle U, M - P\rangle$ is the inner product between matrices U and $M - P$ which is defined as $\mathbf{tr}(U(M - P))$. Note that minimizing $\langle U, M - P\rangle + \frac{\rho}{2}\left\|M - P\right\|_F^2$ over M or P is equivalent to minimizing $\frac{\rho}{2}\left\|M - P + U/\rho\right\|_F^2$. We thus obtain the ADMM updates as follows:

$$\begin{aligned} M^{k+1} \ &= \ \underset{M}{\text{argmin}}\ \mathbf{tr}(MM_0^{-1}) - \log\det(M) + \delta_{S_+^m}(M) \\ &+ \ \sum_{\mathcal{C}} w_{a,b,c,d} H(d_M(\mathbf{x}_a, \mathbf{x}_b) - d_M(\mathbf{x}_c, \mathbf{x}_d)) \\ &+ \ \frac{\rho}{2}\left\|M - P^k + U^k/\rho\right\|_F^2 \end{aligned} \tag{5}$$

$$P^{k+1} \ = \ \underset{P}{\text{argmin}} \lambda \left\|P\right\|_1 + \frac{\rho}{2}\left\|M^{k+1} - P + U^k/\rho\right\|_F^2 \tag{6}$$

$$U^{k+1} \ = \ U^k + \rho(M^{k+1} - P^{k+1}). \tag{7}$$

Since $\frac{\rho}{2}\left\|M - P + U/\rho\right\|_F^2$ is also a smooth convex function of M, we can easily adapt our previous algorithm to solve (5). In addition, an inexact solution from a relatively small number of iterations in solving (5) would suffice the need of convergence [13] as long as the total error across iterations is bounded. The optimization problem in (6) decomposes into entry-wise optimization problems which can be solved in isolation. We can thus optimize over each entry $P_{(i,j)}$ separately and see that it has a closed-form solution:

$$P_{(i,j)} = \text{sgn}(M_{(i,j)}^{k+1} + \frac{U_{(i,j)}^k}{\rho}) \times \max(|M_{(i,j)}^{k+1} + \frac{U_{(i,j)}^k}{\rho}| - \frac{\lambda}{\rho}, 0).$$

Here M and P are updated in an alternating fashion, and the updates of U keep narrowing the distance between M and P, so the states get nearer and nearer to the feasible region of the original constrained optimization problem (4). We also applied a simple updating scheme of ρ as in [12] to make the convergence less dependent on the initial choice of ρ. In each iteration, solving (5) is equivalent to a full run of LSML if solved to high precision. But here we only require an inexact approximation which can be obtained in few tens of gradient moves.

V. EXPERIMENTS

In this section, we show the effect of our model and algorithm in supervised clustering setting on datasets from the UCI repository [14]. For each dataset, we supply our algorithm sets of relative comparisons as supervision to learn a distance metric. We then use the distance metric to conduct K-Means clustering and measure the clustering accuracy in pairwise F-measure as defined in [5]. We also include two existing metric learning algorithms ([1], [2]) and two baseline methods without metric-adaption in comparison.

A. Datasets and Constraints

Here we give results for six real-world datasets from the UCI repository [14]. Table I lists the numbers of objects, dimensions and clusters of these datasets. The *Digits-389* and *Letters-IJLT* datasets are subsets from the original *Digits* and *Letters* handwritten character recognition datasets. The two subsets contain the character classes that are considered difficult to distinguish.

Dataset	#objects	#dimensions	#clusters
Iris	150	4	3
Wine	178	13	3
Digits-389	3165	16	3
Letters-IJLT	3059	16	4
Ionosphere	351	34	2
Sonar	208	60	2

Table I
DATASETS USED IN EXPERIMENTAL EVALUATION

For each dataset, we randomly generated relative comparisons in the form of $d(\mathbf{x}_a, \mathbf{x}_b) < d(\mathbf{x}_a, \mathbf{x}_c)$, where $\mathbf{x}_a, \mathbf{x}_b$ belong to the same class and $\mathbf{x}_c$ is in a different class based on the class labels associated with the dataset. The learning algorithms evaluated in the experiments only learn from the relative comparisons and do not make use of class label information. The size of constraint sets ranges from $0.2 \times |D|$ to $2 \times |D|$ where $|D|$ is the number of objects in a dataset. For each input size, we repeated with 50 randomly generated sets and take the average to ensure consistent observation.

We chose not to generate all possible constraints in a selected percentage of data objects, as adopted by several studies on pairwise constraints and also in [2]. We feel that this design is less suitable for relative comparisons whose constraint space ($|D|^3$) is much larger than that of pairwise constraints($|D|^2$).

B. Methods Evaluated

For the present evaluation, we regularize our algorithm towards the identity matrix and the inverse of covariance matrix. We assume a constant weight for all constraints and the weight is inversely proportional to the number of constraints ($w = \frac{\mu}{|C|}$). Thus, in our LSML algorithm, there is only one parameter μ to determine. We did a simple grid search to choose the μ that attains the best overall accuracy for each dataset. For our sparsity extension SpLSML, one more parameter (λ) is added in the grid search. The other two implicit parameters (ρ, U) only affects the convergence rate but not the converging point. Also the two are self-updating and do not rely much on the initial values.

Two existing metric learning approaches [1], [2] are included in our evaluation. We refer to the method in [1] as "*SVM-Metric*", as it learns metrics from constraints by optimizing a SVM-like quadratic function. In implementing this method, we set the regularization parameter $A = I$ as in [1] and did a grid search for C for each dataset. The other existing method included is the SSSVAD algorithm in [2]. This algorithm does K-Means clustering and metric learning simultaneously. We implemented the algorithm based on our understanding of the paper and tuned the parameters to attain its best performance. The parameter space $(\delta, \gamma_T, \eta, \rho)$ is however much larger than that of the above methods. We tried to fix the learning rates η, ρ first using a subset of datasets and then did grid search for the best combination of δ, γ_T. Unlike in SpLSML, the learning rate parameters in SSSVAD does affect the final results. SSSVAD's outcome is also affected by its initialization. Thus, we repeat the learning process five times for each parameter setting and input data (the accuracy measured for each dataset and constraint size is the average of 50×5 runs).

We also include two baseline methods that use fixed Mahalanobis metrics (the identity matrix and the inverse of covariance matrix). They correspond to Euclidean distance and Euclidean distance after a standard PCA whitening transform. Below we list all methods included in experiments:

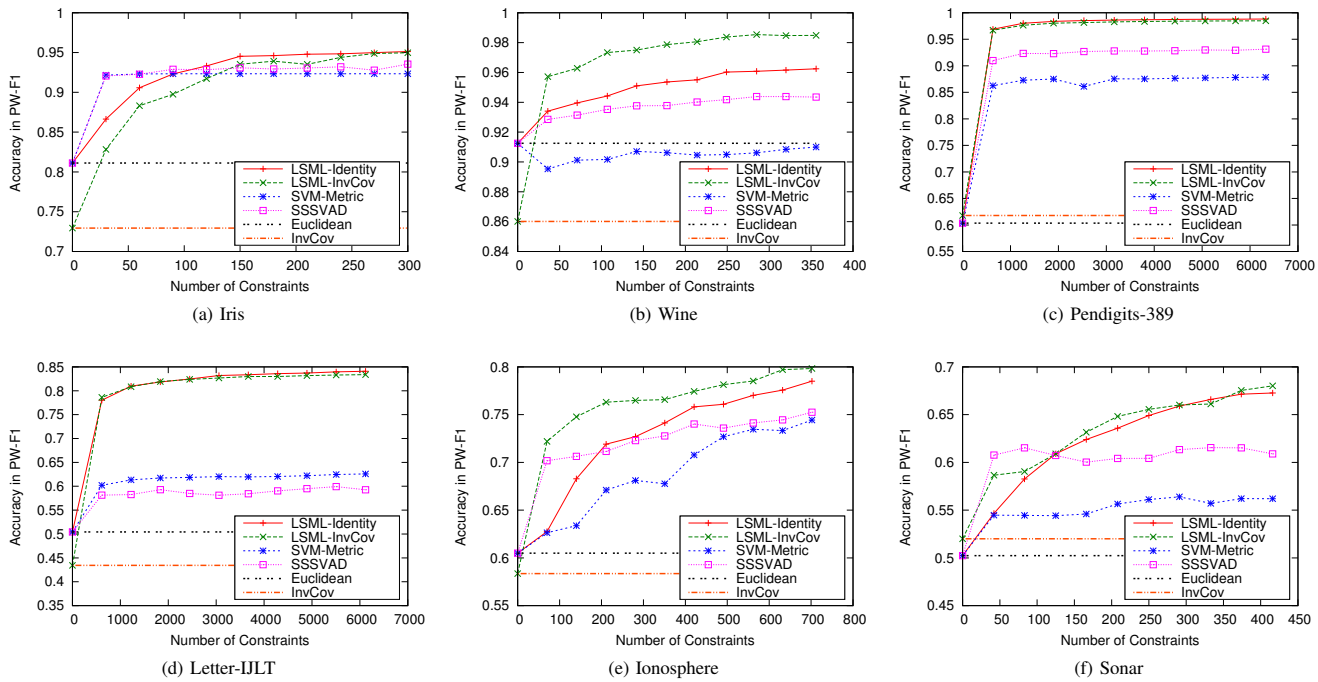
- LSML / SpLSML
  regularized towards both the identity matrix and the inverse of covariance matrix (InvCov)
- SVM-Metric in [1]
- SSSVAD in [2]
- Euclidean
- InvCov

C. Clustering Accuracy with Learned Metrics

The clustering accuracy results for all six datasets are presented in Figure 1. All four metric learning methods have shown noticeable improvements in clustering over the corresponding baselines in most experiments. Our algorithm LSML regularized towards the two initial matrices (the identity matrix and inverse of covariance matrix) consistently outperforms the two existing methods by a large margin in almost all experiments. This suggests that regularizing towards the identity matrix or inverse of covariance matrix can work well empirically when the data distribution is unknown. The difference between using the two initial matrices depends on the specific datasets and does not correspond to the baseline performance without constraint-based learning.

The accuracy curves of all four metric learning methods are relatively flat when datasets have smaller number of dimensions. This indicates that they all tend to saturate early with small number of constraints in these datasets (Figure 1 a,b,c,d). For the datasets with larger number of features (Figure 1 e,f), the accuracy of LSML keeps increasing with the number of constraints fed. This is expected since the number of entries in M grows quadratically with the number of dimensions. The more entries to optimize, the more information we need to extract from constraints. Under the same dimension, the SVM-Metric, which parameterizes through only a diagonal matrix (with A fixed), reaches its accuracy plateau with much fewer constraints compared to LSML.

We also measured the accuracy variance across different sets of constraints (data not shown due to space limit). The variance is generally small compared to the accuracy gain with constraints. All datasets but one have standard deviation below



(a) Iris (b) Wine (c) Pendigits-389

(d) Letter-IJLT (e) Ionosphere (f) Sonar

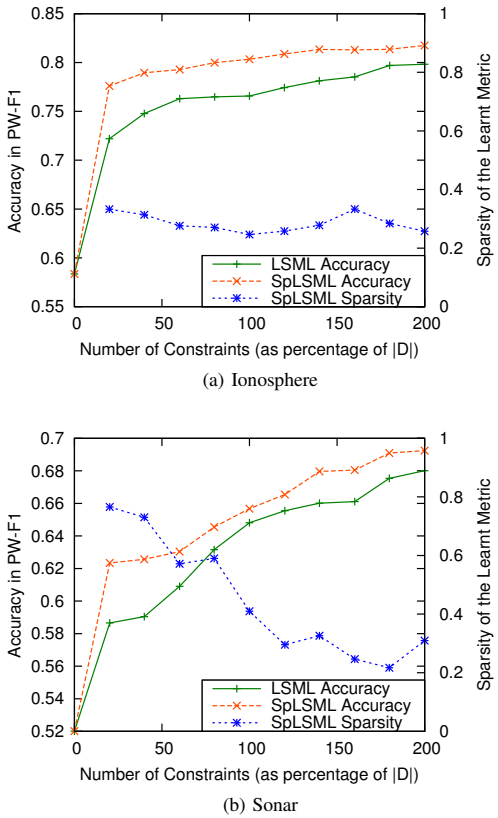
Figure 1. The clustering accuracy for six UCI datasets with randomly generated relative comparisons. For each dataset, the number of constraints ranges from 0 to two times the number of objects in the dataset. Each measure is the average of 50 runs with independently sampled sets of constraints.

0.05 with $|D|$ constraints fed. The four datasets with relatively low dimension have standard deviation below 0.02. Though the variance is affected by many intrinsic properties of each dataset, we can clearly conclude that: 1) The variance decreases with the number of constraints fed. 2) The variance increases with number of dimensions. The SVM-Metric has comparable variance with LSML in low-dimensional cases and moderately lower variance in higher dimensional cases. The variance of SSSVAD is considerably larger than that of the other methods due to its clustering-based learning behavior.

D. Sparsity Extension

The extension SpLSML aims to promote sparsity in the learned metric matrix which is beneficial with high dimensional data and limited supervision (small number of constraints). This is because promoting sparsity could suppress relatively unimportant entries in M and lead to more stable estimation with limited information. In our sparsity-promoted model, the λ parameter controls the weight of the sparsity term (ℓ_1-norm of M). Obviously, when the weight is minimal, this reduces to the basic LSML where all entries in M are being optimized. When the weight is too high, a resulting metric that is highly sparse could neglect important features or feature-feature interactions and lead to a degenerated similarity measure. Thus we seek an optimal trade-off point between fitting too many and too few parameters by tuning the λ parameter together with w, the weight of the constraint loss.

We find that in lower dimensional datasets (iris, wine, letter-IJLT and pendigits-389), SpLSML does not exhibit significant difference from the basic LSML. This is more or less expected as the number of constraints is not small compared to the



(a) Ionosphere

(b) Sonar

Figure 2. Accuracy (left y-axis) and sparsity (right y-axis) achieved by SpLSML in comparison with LSML (both regularized towards the inverse of covariance matrix).

number of fitted parameters (m^2). For the two datasets of higher dimensionality, SpLSML can achieve noticeable gain by suppressing relatively unimportant entries in M. The sparsity achieved is more pronounced in dataset sonar which has approximately three times more parameters to be fitted and less objects and constraints than ionosphere.

Figure 2 shows the accuracy and sparsity achieved by our sparsity extension (SpLSML) on sonar and ionosphere compared with the basic LSML algorithm. For both regularization matrices, SpLSML attains higher accuracy than the basic LSML. The advantage is more significant with small number of constraints as we have expected. We notice that stable gain can be achieved even with only moderate sparsity. Also the sparsity decreases with the number of constraints fed. This is because the algorithm has more information to fit more parameters. Thus keeping the same sparsity level may not necessarily help improving accuracy. The entries that are "zeroed-out" even with large number of constraints can be interpreted as features or feature-pairs that are unlikely to contribute to the similarity measure.

VI. Related Work

The problem of learning a metric from relative comparisons is first introduced in [1] with motivating examples from search data. Schultz and Joachims proposed to model this as a SVM-like quadratic programming problem. The distance or (dis)similarity between objects is parameterized through a real matrix A and a nonnegative diagonal matrix W. In the learning process, A is usually pre-specified and W is to be fitted. A simple choice for A such as I limits the expressibility of the metric while determining a more suitable A beforehand can be a non-trivial problem. Similar to the formulation in [1], Rosales and Fung [3] and Huang et al. [4] have casted relative comparisons into linear constraints with slack variables and optimize on the sum of slack variables. They further promote other properties such as sparsity by imposing different types of penalty terms. The major difference between these models and ours is that they cast relative comparisons into linear constraints to satisfy while we use squared hinge function to minimize "loss" from relative comparisons.

Kumar and Kummamuru proposed a very different approach to learn metric from relative comparisons [2]. Their method splits the whole data space into K partitions by clustering where K is a pre-specified parameter corresponding to the number of clusters. A weighted Euclidean distance is learned within each partition or cluster. Though allowing much flexibility, this model has several drawbacks: 1) The learned metric corresponds to a K-clustering result and cannot be easily generalized to unobserved cluster(s). 2) As a clustering-based learning method, its results are sensitive to the initialization of the clustering process and hidden parameters.

All the methods listed above compared their work against studies on learning from pairwise constraints. However, relative comparisons are only in relative sense and often obtained without labeling objects (e.g., from a ranking list or structure knowledge). The knowledge of relative comparisons and pairwise constraints cannot be translated into each other's form equivalently. In this paper, we try to compare methods in the same class under a consistent setting.

Besides learning from relative comparisons, metric learning is a much broader area with many efforts on learning from pairwise constraints. Representative studies include [6]–[8]. In [6], the authors proposed a LogDet divergence based learning framework that does iterative projection onto pairwise constraints. Though it is mentioned that the framework can be generalized to arbitrary linear constraints, we find it non-trivial to extend it to handle relative comparisons: 1) Relative comparisons cannot be written as rank-1 constraint matrices which allows closed-form updates in each projection. 2) The zone consistency assumption of Bregman's projection algorithm [15] is not guaranteed even with the presence of slack variables.

VII. Conclusion

In this paper, we propose a formulation for metric learning from relative comparisons by minimizing the sum of squared hinge loss. A simple yet effective gradient-descent algorithm is presented to solve the corresponding convex optimization problem. We also give a sparsity-promoted extension which improves the metric estimation when the number of constraints is small compared to the number of features. Our algorithms require low effort in parameter tuning: the basic LSML algorithm has only one parameter representing the weight of constraints; the sparsity extension SpLSML adds one more parameter to indicate the preference of sparsity. In the future we plan to explore applications on large-size biological datasets.

Acknowledgement

We would like to thank Yunchao Gong for help on conducting experiments. This work was partially supported by NSF IIS-0812464 and NIH R01HG006703.

References

[1] M. Schultz and T. Joachims, "Learning a distance metric from relative comparisons," in *NIPS*, 2003.

[2] N. Kumar and K. Kummamuru, "Semisupervised clustering with metric learning using relative comparisons," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 4, pp. 496–503, 2008.

[3] K. Huang, Y. Ying, and C. Campbell, "Generalized sparse metric learning with relative comparisons," *Knowledge and Information Systems*, vol. 28, pp. 25–45, 2011.

[4] R. Rosales and G. Fung, "Learning sparse metrics via linear programming," in *KDD*, 2006, pp. 367–373.

[5] E. Y. Liu, Z. Zhang, and W. Wang, "Clustering with relative constraints," in *KDD*, 2011, pp. 947–955.

[6] J. Davis *et al.*, "Information-theoretic metric learning," in *ICML*, 2007.

[7] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *NIPS*, 2006.

[8] E. Xing *et al.*, "Distance metric learning with application to clustering with side-information," in *NIPS*, 2003, pp. 505–512.

[9] L. Yang and A. R. Jin, "Contents distance metric learning: A comprehensive survey," 2006.

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

[11] N. Z. Shor, "Minimization methods for non-differentiable functions," *Series in Computational Mathematics*, 1985.

[12] S. Boyd *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[13] J. Eckstein and D. P. Bertsekas, "On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators," *Mathematical Programming*, vol. 55, pp. 293–318, 1992.

[14] A. Asuncion and D. Newman, "UCI machine learning repository," 2007.

[15] Y. Censor and S. Zenios., *Parallel optimization: Theory, algorithms, and applications*. Oxford University Press, 1997.