Robust Structural Metric Learning

Daryl K. H. Lim DKLIM@UCSD.EDU
Department of Electrical and Computer Engineering, University of California, San Diego, CA 92093 USA

Brian McFee BRM2132@COLUMBIA.EDU
Center for Jazz Studies, Columbia University, New York, NY 10027 USA

Gert Lanckriet GERT@ECE.UCSD.EDU
Department of Electrical and Computer Engineering, University of California, San Diego, CA 92093 USA

Abstract

Metric learning algorithms produce a linear transformation of data which is optimized for a prediction task, such as nearest-neighbor classification or ranking. However, when the input data contains a large portion of non-informative features, existing methods fail to identify the relevant features, and performance degrades accordingly. In this paper, we present an efficient and robust structural metric learning algorithm which enforces group sparsity on the learned transformation, while optimizing for structured ranking output prediction. Experiments on synthetic and real datasets demonstrate that the proposed method outperforms previous methods in both high- and low-noise settings.

1. Introduction

Metric learning algorithms produce a (linear) transformation optimized to yield small distances between *similar* pairs of points, and large distances between *dissimilar* pairs of points (Xing et al., 2003; Weinberger et al., 2006; Davis et al., 2007). The transformation is usually optimized for a specific task, such as visualization or k-nearest-neighbor classification. More generally, *structural metric learning* algorithms optimize for the prediction of *structured outputs* induced by the learned transformation, such as nearest-neighbor rankings (McFee & Lanckriet, 2010) or connectivity graphs (Shaw et al., 2011).

In the usual setting, data is provided as a collection of vectors $x_i \in \mathbb{R}^d$, and the squared distance function $(x_i - x_j)^\mathsf{T} W (x_i - x_j)$ is parameterized by a positive semidefinite matrix W, which is optimized to satisfy a set of pairwise similarity or structural constraints. It is often desirable that W be *low-rank*, as this limits model complexity, exploits correlations between features, and often improves predictive accuracy. When W is low-rank, distances are equivalently computed in a low-dimensional subspace of $\mathbb{R}^d$, effectively providing *output sparsity* (*i.e.*, sparsity after transformation) and allowing for efficient storage and retrieval. Previous work on sparse metric learning focuses on this notion of output sparsity (Ying et al., 2009).

However, *input sparsity* can be equally important to achieving good performance: in general, the input data may contain a significant amount of irrelevant features which should be detected and suppressed by the learning algorithm. In such cases, there will always exist a linear transformation which suppresses the non-informative features (*i.e.*, by setting the corresponding entries of W to 0) and one would hope that a metric learning algorithm should find it. Unfortunately, existing algorithms frequently fail to find such a transformation, and their performance degrades rapidly as the number of noisy features increases.

In this paper, we propose a robust extension to the metric learning to rank (MLR) algorithm (McFee & Lanckriet, 2010). The proposed method imposes a group sparsity penalty on the learned metric to promote input sparsity, and a trace penalty to promote output sparsity. We derive an efficient learning algorithm based upon the 1-slack structural SVM (Joachims et al., 2009) and the alternating direction method of multipliers (ADMM) (Boyd et al., 2011). Our experiments demonstrate that even a very simple noise model can dramatically reduce performance of existing methods, while the proposed method correctly identifies and

Proceedings of the 30^{th} International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

suppresses noisy features.

1.1. Related work

Supervised metric learning is a well-studied problem, of which some representative methods are *information-theoretic metric learning* (ITML) (Davis et al., 2007), *large margin nearest neighbor* (LMNN) (Weinberger et al., 2006), and the method of Xing et al. (2003). However, many of these methods do not explicitly regularize for low rank (*i.e.*, sparsity in the projected space) or sparsity of input features. For example, the log det regularizer of ITML constrains W to be strictly positive definite, which in practice often results in high-rank solutions which necessarily depend on all input features, including noisy ones. LMNN may output a low-rank metric, though it does not explicitly regularize for it.

Ying et al. (2009) proposed a mixed-norm regularized metric learning algorithm to achieve dimensionality reduction (*i.e.*, output sparsity). However, their formulation applies the regularization after a (dense) rotation of the input feature space, and therefore does not promote sparsity with respect to the input features. Low-rank regularization is used in many metric learning algorithms (Shen et al., 2009; McFee & Lanckriet, 2010; Huang et al., 2011), but these methods do not regularize for feature sparsity.

Rosales & Fung (2006) regularize for sparsity in the input space by minimizing $\sum_{i,j} |W_{ij}|$. Their formulation additionally restricts W to the set of diagonally dominant matrices, which allows for an efficient formulation as a linear programming problem, but tends to favor high-rank solutions.

Other robust metric learning formulations have been proposed in which the similarity constraints or labels have been corrupted, rather than the features. Huang et al. (2010) developed an algorithm for the case where a fraction of the similarity constraints are corrupted. Similarly, Zha et al. (2009) leverage auxiliary data to learn a metric when the constraint set is sparse.

1.2. Preliminaries

Let $\mathbb{S}^d$ and $\mathbb{S}^d_+$ denote the sets of $d \times d$, real-valued, symmetric and positive semidefinite matrices. Let $\Pi_S[x]$ denote the orthogonal projection of x onto a convex set S. For matrices A, B, denote the Frobenius inner product by $\langle A, B \rangle_{\mathsf{F}} := \sum_{i,j} A_{ij} B_{ij}$, and norm by $\|A\|_{\mathsf{F}} := \sqrt{\langle A, A \rangle_{\mathsf{F}}}$. Finally, for $x \in \mathbb{R}$, let $[x]_+ := \max(0, x)$.

2. Robust Metric Learning

In this paper, we will build upon the metric learning to rank (MLR) algorithm (McFee & Lanckriet, 2010), a variant of the structural SVM (Tsochantaridis et al., 2005) which optimizes $W \in \mathbb{S}^d_+$ to minimize a ranking loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ (*e.g.*, decrease in mean average precision) over permutations $\mathcal{Y}$ induced by distance.

MLR can be expressed as the following convex optimization problem:

$$\begin{aligned} \min_{W \in \mathbb{S}^d_+} \operatorname{tr}(W) + \frac{C}{n} \sum_{q \in \mathcal{X}} \xi_q \\ \text{s.t.} \, \forall q \in \mathcal{X}, y \in \mathcal{Y} : \\ \langle W, \psi(q, y_q) - \psi(q, y) \rangle_{\mathsf{F}} \geq \Delta(y_q, y) - \xi_q \end{aligned} \tag{1}$$

Here, $\mathcal{X} \subset \mathbb{R}^d$ is the training set of n points; $\mathcal{Y}$ is the set of all permutations over $\mathcal{X}$; $C > 0$ is a slack trade-off parameter; $\psi : \mathbb{R}^d \times \mathcal{Y} \to \mathbb{S}^d$ is a feature encoding of an input-output pair (q, y); and $\Delta(y_q, y) \in [0, 1]$ is the desired margin, *i.e.*, loss incurred for predicting a ranking y rather than the true ranking y_q. The feature map ψ (Joachims, 2005) is designed so that $\langle W, \psi(q, y) \rangle_{\mathsf{F}}$ is large when the ranking of $\mathcal{X}$ induced by distance from q agrees with y, and small otherwise. For a query $q \in \mathbb{R}^d$ with relevant set $\mathcal{X}^+_q \subseteq \mathcal{X}$ and irrelevant set $\mathcal{X}^-_q \subseteq \mathcal{X}$, ψ is defined by

$$\begin{aligned} \psi(q, y) &:= \sum_{i \in \mathcal{X}^+_q} \sum_{j \in \mathcal{X}^-_q} y_{ij} \frac{\phi(q, x_i) - \phi(q, x_j)}{|\mathcal{X}^+_q| \cdot |\mathcal{X}^-_q|}, \\ \phi(q, x) &:= -(q - x)(q - x)^{\mathsf{T}}, \\ y_{ij} &:= \begin{cases} +1 & i \prec_y j \\ -1 & \text{otherwise} \end{cases}. \end{aligned}$$

The regularization term $\operatorname{tr}(W)$ in (1) is used as a convex surrogate for $\operatorname{rank}(W)$ to promote low-rank solutions. However, because it ignores the off-diagonal elements of W, it does not necessarily promote feature sparsity, and performance can degrade with the addition of non-informative features.

2.1. Robust MLR

Ideally, we would like the learning algorithm to produce a metric W which relies only upon informative features. More precisely, if some input dimension i is non-informative, then the corresponding rows and columns of W should suppress the feature, *i.e.*, $W_{i\cdot} = W_{\cdot i} = 0$. In contrast, sparsity should not be enforced for rows corresponding to informative features, as this would limit the ability of the algorithm to exploit correlation between informative features, and reduce output sparsity. This suggests a natural row (or column) grouping

Algorithm 1 Robust metric learning to rank (R-MLR)

1: $\mathcal{A} \leftarrow \{\}$
2: **repeat**
3: $W \leftarrow \operatorname{argmin}_{W \in \mathbb{S}_+^d} f(W)$

$$f(W) := \operatorname{tr}(W) + \lambda \|W\|_{2,1} + C \max_{(\Delta,\Psi)\in\mathcal{A}} [\Delta - \langle W, \Psi \rangle_{\mathsf{F}}]_+ \quad (3)$$

4: $\xi \leftarrow \max_{(\Delta,\Psi)\in\mathcal{A}} [\Delta - \langle W, \Psi \rangle_{\mathsf{F}}]_+$
5: $\widehat{\Delta} \leftarrow 0, \quad \widehat{\Psi} \leftarrow 0$
6: **for** $q = 1, 2, \ldots, n$ **do**
7: $y' \leftarrow \operatorname{argmax}_{y \in \mathcal{Y}} \Delta(y_q, y) + \langle W, \psi(q, y) \rangle_{\mathsf{F}}$
8: $\widehat{\Delta} \leftarrow \widehat{\Delta} + 1/n \Delta(y_q, y')$
9: $\widehat{\Psi} \leftarrow \widehat{\Psi} + 1/n \left(\psi(q, y_q) - \psi(q, y')\right)$
10: **end for**
11: $\mathcal{A} \leftarrow \mathcal{A} \cup \left\{(\widehat{\Delta}, \widehat{\Psi})\right\}$
12: **until** $\widehat{\Delta} - \langle W, \widehat{\Psi} \rangle_{\mathsf{F}} \leq \xi + \epsilon$

of the entries of W when enforcing sparsity, so that rows corresponding to informative features may be dense, but sparsity is enforced over rows to avoid relying upon too many features.

As in the group lasso, row-sparsity can be promoted by mixed-norm regularization (Yuan & Lin, 2006):

$$\|W\|_{2,1} := \sum_{i=1}^{d} \|W_{i\cdot}\|_2.$$

This leads to our *Robust Metric Learning to Rank* (R-MLR) formulation:

$$\min_{W \in \mathbb{S}_+^d} \operatorname{tr}(W) + \lambda \|W\|_{2,1} + \frac{C}{n} \sum_{q \in \mathcal{X}} \xi_q \quad (2)$$
$$\text{s.t. } \forall q \in \mathcal{X}, y \in \mathcal{Y}:$$
$$\langle W, \psi(q, y_q) - \psi(q, y) \rangle_{\mathsf{F}} \geq \Delta(y_q, y) - \xi_q$$

R-MLR balances the trade-off between input and output sparsity through a hyper-parameter $\lambda > 0$, which may be tuned by cross-validation.

As there are a super-exponential number of constraints in 2, we implement the 1-slack cutting-plane method (Joachims et al., 2009) to obtain Algorithm 1. Algorithm 1 approximates (2) by alternately solving a convex optimization problem (step 3) over a small set $\mathcal{A}$ of active constraints, and updating $\mathcal{A}$ with the constraints most violated by the resulting W (steps 5–10). The process repeats until the most-violated constraint (and hence, all other constraints) is satisfied to within some specified $\epsilon > 0$ of the loss on the active set $\mathcal{A}$.

3. Optimization

The original MLR implementation solved for W via projected sub-gradient descent (McFee & Lanckriet, 2010). While this approach could be applied for R-MLR as well, projecting each iterate back onto $\mathbb{S}_+^d$ is computationally expensive: $\mathcal{O}(d^3)$ for each spectral decomposition and thresholding operation. Instead, we will use the alternating direction method of multipliers (Boyd et al., 2011) to efficiently optimize (3).

First, we transform the optimization problem (Algorithm 1, step 3) into an equivalent problem:

$$\min_{W,V,Z} f(W) + g(V) + h(Z) \quad \text{s.t. } W = V = Z$$
$$\text{where } f(W) := \operatorname{tr}(W) + C \cdot \max_{(\Delta,\Psi)\in\mathcal{A}} [\Delta - \langle W, \Psi \rangle_{\mathsf{F}}]_+$$
$$g(V) := \lambda \|V\|_{2,1}, \quad h(Z) := \begin{cases} 0 & Z \in \mathbb{S}_+^d \\ \infty & Z \notin \mathbb{S}_+^d \end{cases}$$

Introducing Lagrange multipliers $\Lambda_V, \Lambda_W \in \mathbb{S}^d$, we obtain the augmented Lagrangian:

$$\begin{aligned} \mathcal{L}_\rho(W, V, Z, \Lambda_W, \Lambda_V) =& f(W) + g(V) + h(Z) \\ &+ \langle \Lambda_W, W - Z \rangle_{\mathsf{F}} + \frac{\rho}{2} \|W - Z\|_{\mathsf{F}}^2 \\ &+ \langle \Lambda_V, V - Z \rangle_{\mathsf{F}} + \frac{\rho}{2} \|V - Z\|_{\mathsf{F}}^2, \end{aligned}$$

where $\rho > 0$ is a scaling parameter. The ADMM algorithm can be written in *scaled form* as follows:

$$\begin{aligned} W^{t+1} &\leftarrow \operatorname{argmin}_{W \in \mathbb{S}^d} f(W) + \frac{\rho}{2} \|W - (Z^t - U_W^t)\|_{\mathsf{F}}^2 \quad (4) \\ V^{t+1} &\leftarrow \operatorname{argmin}_{V \in \mathbb{S}^d} g(V) + \frac{\rho}{2} \|V - (Z^t - U_V^t)\|_{\mathsf{F}}^2 \quad (5) \\ Z^{t+1} &\leftarrow \operatorname{argmin}_{Z \in \mathbb{S}^d} h(Z) \\ &\quad + \rho \left\| Z - \tfrac{1}{2}(W^{t+1} + V^{t+1} + U_W^t + U_V^t) \right\|_{\mathsf{F}}^2 \\ U_W^{t+1} &\leftarrow U_W^t + W^{t+1} - Z^{t+1}; \\ U_V^{t+1} &\leftarrow U_V^t + V^{t+1} - Z^{t+1}. \end{aligned}$$

where $U_W = \frac{1}{\rho}\Lambda_W$, $U_V = \frac{1}{\rho}\Lambda_V$. The optimization algorithm then cycles through each update listed above until convergence, or some maximum number of iterations is exceeded.

3.1. W-update: dual formulation

The W-update (4) is a convex optimization problem similar to (3) with two modifications: 1) the constraint $W \in \mathbb{S}_+^d$ has been removed, and 2) it is strongly convex, due to the quadratic term from $\mathcal{L}_\rho$. In principle, this could be solved directly. However, the active set $\mathcal{A}$ is often quite small: in practical problems, $|\mathcal{A}|$ rarely exceeds 100–200, while the number of parameters is $\mathcal{O}(d^2)$ and can easily number in the tens of thousands.

This suggests that a dual formulation may lead to a more efficient algorithm.

To simplify the following derivation, let $R^t := Z^t - U_W^t$ and $m := |\mathcal{A}|$. The W update (4) can be stated as the following linearly constrained quadratic program:

$$\min_{W,\xi\geq 0} \operatorname{tr}(W) + C\xi + \frac{\rho}{2}\|W - R^t\|_{\mathsf{F}}^2 \qquad (6)$$
$$\text{s.t. } \forall(\Delta_i, \Psi_i) \in \mathcal{A}: \quad \Delta_i - \langle W, \Psi_i\rangle_{\mathsf{F}} - \xi \leq 0.$$

Introducing Lagrange multipliers $\alpha \in \mathbb{R}_+^m$, $\beta \in \mathbb{R}_+$, (6) has the following Lagrangian:

$$\mathcal{L}(W, \xi, \alpha, \beta) = \operatorname{tr}(W) + C\xi + \frac{\rho}{2}\|W - R^t\|_{\mathsf{F}}^2 + \sum_{i=1}^m \alpha_i \left(\Delta_i - \langle W, \Psi_i\rangle_{\mathsf{F}} - \xi\right) - \beta\xi$$

Minimizing over W, we obtain the dual program:

$$\sup_{\substack{\alpha\in\mathbb{R}_+^m \\ \beta\in\mathbb{R}_+}} \inf_{W,\xi} \mathcal{L} = \frac{1}{\rho}\max_{\alpha\in\mathbb{R}_+^m} -\frac{1}{2}\alpha^\mathsf{T} H\alpha - b^\mathsf{T}\alpha \quad \text{s.t.} \quad \sum_i \alpha_i \leq C, \qquad (7)$$

with the structure kernel $H \in \mathbb{S}_+^m$ and cost vector $b \in \mathbb{R}^m$ defined as:

$$H_{ij} := \langle \Psi_i, \Psi_j\rangle_{\mathsf{F}}, \quad b_i := \langle \rho R^t - I, \Psi_i\rangle_{\mathsf{F}} - \rho\Delta_i. \qquad (8)$$

(7) is a linearly constrained quadratic program in m variables, and can easily be solved by off-the-shelf tools. Note that the dual program is independent of both n and d, resulting in significant improvements in efficiency for large problems. After computing a dual optimum $\overline{\alpha}$, a primal optimum $\overline{W}^{t+1}$ can be recovered as follows:

$$\overline{W}^{t+1} = R^t + \frac{1}{\rho}\left(\sum_{i=1}^m \overline{\alpha}_i\Psi_i - I\right). \qquad (9)$$

3.2. V-update

If there was no symmetry constraint in (5), the V-update would take the form of a *prox-operator* $\operatorname{prox}_{\ell_{2,1}}(Z^t - U_V^t, \lambda/\rho)$, where $\operatorname{prox}_{\ell_{2,1}}(A,\lambda) := \operatorname{argmin}_V \lambda\|V\|_{2,1} + \frac{1}{2}\|V - A\|_{\mathsf{F}}^2$. This can be efficiently computed via an element-wise thresholding operation

$$\left[\operatorname{prox}_{\ell_{2,1}}(A,\lambda)\right]_{ij} = A_{ij}\left[1 - \frac{\lambda}{\|A_{i\cdot}\|_2}\right]_+ \qquad (10)$$

where $A_{i\cdot}$ is the ith row of A (Kowalski, 2009). However, this results in an asymmetric matrix, so we compute the $\mathbb{S}^d$-constrained V^{t+1} update via a separate ADMM algorithm, which alternates a $\operatorname{prox}_{\ell_{2,1}}$ update step, a projection $\Pi_{\mathbb{S}^d}(\cdot)$ and an additive dual update, each of which can be computed in linear time.

Algorithm 2 Robust MLR (step 3 of Algorithm 1)

$W^0 \leftarrow 0$, $V^0 \leftarrow 0$, $Z^0 \leftarrow 0$, $U_W^0 \leftarrow 0$, $U_V^0 \leftarrow 0$
for $t = 0, 1, 2, \ldots, T$ (until convergence) **do**
  $\forall i: b_i \leftarrow \langle \rho(Z^t - U_W^t) - I, \Psi_i\rangle_{\mathsf{F}} - \rho\Delta_i$
  $\overline{\alpha} \leftarrow \operatorname{argmax}_\alpha$(7)
  $W^{t+1} \leftarrow Z^t - U_W^t + {}^1\!/\!_\rho\left(\sum_{i=1}^m \overline{\alpha}_i\Psi_i - I\right)$
  $V^{t+1} \leftarrow \operatorname*{argmin}_{V\in\mathbb{S}^d} g(V) + \frac{\rho}{2}\|V - (Z^t - U_V^t)\|_{\mathsf{F}}^2$;
  $Z^{t+1} \leftarrow \Pi_{\mathbb{S}_+^d}\left[\frac{1}{2}(W^{t+1} + V^{t+1} + U_W^t + U_V^t)\right]$
  $U_W^{t+1} \leftarrow U_W^t + W^{t+1} - Z^{t+1}$
  $U_V^{t+1} \leftarrow U_V^t + V^{t+1} - Z^{t+1}$
end for
output W^T

3.3. Z-update

The Z-update simplifies to the orthogonal projection

$$Z^{t+1} \leftarrow \Pi_{\mathbb{S}_+^d}\left[\tfrac{1}{2}(W^{t+1} + V^{t+1} + U_W^t + U_V^t)\right],$$

obtained by thresholding the negative eigenvalues of $\left[\frac{1}{2}(W^{t+1} + V^{t+1} + U_W^t + U_V^t)\right]$ at 0.

After consolidating the update steps, the resulting Robust MLR algorithm is listed as Algorithm 2.

To see the performance gains afforded by the ADMM-based method, we note that by setting $\lambda = 0$ and skipping the V-update, we can obtain an ADMM-based algorithm to solve the MLR problem, which we call MLR-ADMM. This allows us to do a direct comparison between the ADMM-based MLR method and the original solver which used projected sub-gradient descent. Although the ADMM-based algorithm has the same worst-case complexity as projected sub-gradient descent – $O(\epsilon^{-2})$ for ϵ-sub-optimality – it has been observed to yield satisfactory solutions after a small number of steps. Coupled with early stopping (i.e. specifying a maximum number of iterations T for the ADMM algorithm), we were able to obtain significant speedups over projected gradient descent in our experiments.

4. Experiments

To evaluate the proposed method, we conducted three sets of experiments. In the first set of experiments, we augment standard UCI datasets with synthetic correlated noise of varying dimensions to investigate the classification performance of various metric learning algorithms as noisy features are introduced. In the second set of experiments, we evaluate R-MLR on a music similarity task using data from CAL10K (Tingle et al., 2010) and the Million Song Dataset (MSD) (Bertin-Mahieux et al., 2011), and also evaluate the effects of early stopping on training time. In the third set of ex-

periments, we evaluate performance and training time of the various algorithms on an image classification task using image data obtained from ImageNet (Deng et al., 2009).

In all sets of experiments, we also provide comparisons to ℓ_1-MLR, another variant of MLR which imposes a penalty on $\sum_{i,j}|W_{ij}|$ instead of $\|W\|_{2,1}$ in (3), as a similar regularizer has been shown to be effective in promoting feature sparsity by Rosales & Fung (2006). ℓ_1-MLR can be trained via an ADMM algorithm in a similar fashion to R-MLR, and in fact allows for an efficient element-wise thresholding for the corresponding V-update. To train MLR, we used the MLR-ADMM algorithm instead of the original projected sub-gradient descent implementation.

4.1. Classifying noise-augmented UCI data

As a first experiment, we measure classification performance on four standard datasets from the UCI repository: `Balance` ($n = 625, d = 4$), `Ionosphere` ($n = 351, d = 34$), `Iris` ($n = 150, d = 4$) and `Wine` ($n = 178, d = 13$). We compare large-margin nearest neighbor (LMNN) (Weinberger et al., 2006), information-theoretic metric learning (ITML) (Davis et al., 2007), and MLR with both ℓ_1-MLR and R-MLR. Each of the previously proposed algorithms are known to perform comparably well on these datasets, and introducing noise will allow us to carefully measure how performance degrades in higher dimensions relative to a known baseline (the noise-free case).

4.1.1. Experiment setup

To study the effect of noisy features on each learning algorithm, we embedded each dataset into a higher-dimensional space by padding each example x_i with D-dimensional correlated noise x_σ:

$$x_i^\mathsf{T} \mapsto \left(x_i^\mathsf{T}, x_\sigma^\mathsf{T}\right) \in \mathbb{R}^{d+D}, \qquad x_\sigma \sim \mathcal{N}(0, \Sigma). \quad (11)$$

For each dataset and $D \in \{2^5, 2^6, 2^7, 2^8\}$, we sample a covariance matrix $\Sigma \in \mathbb{S}_+^D$ from a unit-scale Wishart distribution. Each example was padded with noise according to (11). Each padded dataset was then split into 25 random 40/30/30 training/validation/test splits, and each split was normalized by coordinate-wise z-scoring with the training set statistics.

Performance was measured by k-nearest-neighbor accuracy using the training set as examples. For ITML, the slack parameter γ was varied over $\{1, 10, \ldots, 10^6\}$ For LMNN, the push-pull parameter μ was varied over $\{0.1, 0.2, \ldots, 0.9\}$. For MLR, the loss function Δ was fixed to mean average precision (MAP), and C was varied over $\{1, 10, \ldots, 10^6\}$. For R-MLR and ℓ_1-MLR we additionally varied $\lambda \in \{0.001, 0.01, 0.1, 1\}$. The number of nearest neighbors used for classification k was also varied in $\{1, 3, 5, 7\}$. For each experiment, the hyper-parameters with the best classification accuracy on the validation set are selected.

4.1.2. Results

Figure 1 displays example Ws produced by ITML, LMNN, MLR, ℓ_1-MLR and R-MLR on each UCI dataset, where the noise dimensionality D is set to 32. In each case, R-MLR correctly identifies and suppresses the noise dimensions by assigning small weights to the corresponding rows and columns of W. In contrast, MLR, ITML and LMNN assign significant weights to the noisy dimensions, degrading classification and retrieval performance. ℓ_1-MLR achieves input sparsity, but only at the expense of increased dimension.

Figure 2 displays the error rates of the various learning algorithms across all datasets and values of D. We observe that R-MLR is able to achieve performance on par with other state-of-the-art algorithms even in the noiseless case. For all datasets and $D \geq 64$, the R-MLR algorithm significantly outperforms MLR, ITML and LMNN under a Bonferroni-corrected Wilcoxon signed rank test ($\alpha = 0.05$). R-MLR also significantly outperforms ℓ_1-MLR for `Ionosphere`, $D = \{128,256\}$ and `Balance` for $D = 128$.

Figure 3 illustrates the effective dimensionality E — the number of dimensions necessary to capture 95% of the spectral mass of W — averaged across all splits of each UCI dataset. Effective dimensionality increases with input dimensionality for ITML and LMNN, but remains low for both MLR and R-MLR. ℓ_1-MLR lies in between, producing metrics of higher rank than MLR or R-MLR.

Across all UCI datasets, R-MLR training time was observed to be on the same order of magnitude as MLR-ADMM, and is consistently shorter than LMNN training time. It is still possible to accelerate our method further by using standard techniques, e.g. parallelizing the constraint generation process, or sampling to approximate the most-violated constraint.

4.2. Music similarity: CAL10K

In the music similarity task, we are provided with vector representations of song recordings, and the goal is to learn a distance (retrieval) function which successfully retrieves relevant songs in response to a query. We use a subset of the CAL10K dataset (Tingle et al., 2010), which is provided as ten 40/30/30 splits of a collection of 5419 songs (McFee et al., 2012). For each song x_i,

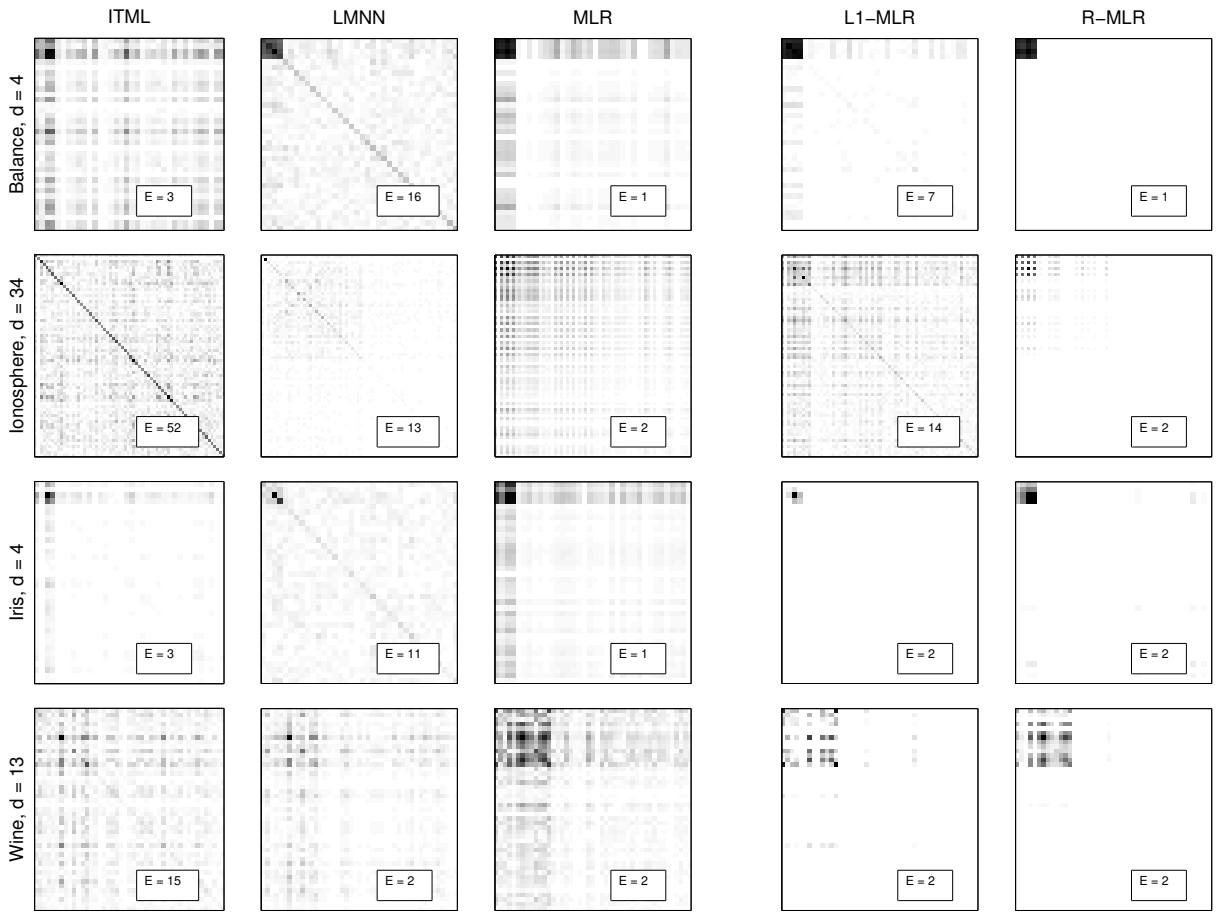

Figure 1. Metrics (W) produced by each algorithm on each UCI dataset padded with 32-dimensional noise generated by (11). For each dataset, d represents the number of features in the original data, and E is the effective dimensionality of W. R-MLR produces sparse, low-rank solutions that correctly identify the informative features (upper-left block), while other algorithms assign significant weight to the noisy dimensions, resulting in dense or high-rank solutions.

a relevant set $\mathcal{X}_i^+ \subset \mathcal{X}_{\text{train}}$ is defined as the subset of songs in the training set performed by the top 10 most similar artists to the performer of x_i, where similarity between artists is measured by the number of shared users in a sample of collaborative filter data (McFee et al., 2012). This top-10 thresholding results in the relevant sets in this data being generally asymmetric and non-transitive, and therefore pair-wise methods such as ITML and classification-based methods like LMNN cannot be applied to the problem. Performance is measured by area under the ROC curve (AUC) of the rankings over the training set induced by distance from a test query.

4.2.1. Experiment setup

We experiment with three song representations, derived from either audio features or lyrical content:

Audio Each song is initially represented as a vector quantization histogram over 1024 acoustic codewords. Histograms were then compressed via PCA to capture 95% of the training set variance. A total of 5419 songs are available in the dataset, and each train/validation/test split retains between 160 and 180 dimensions after PCA.

Lyrics-128, Lyrics-256 1396 of the songs above also have lyrics data available in MSD. We applied Latent Dirichlet Allocation (Blei et al., 2003) on an independent collection of 2000 songs, resulting in $k \in \{128, 256\}$ latent lyrics topics. Each song was then represented by its posterior distribution over topics.

MSD-33 Using the Million Song Dataset, we extracted 33 additional descriptors for each song, including coarse low-level descriptors such as mean timbre, pitch distribution, tempo and loudness, as well as abstract high-level features, such as "dance-

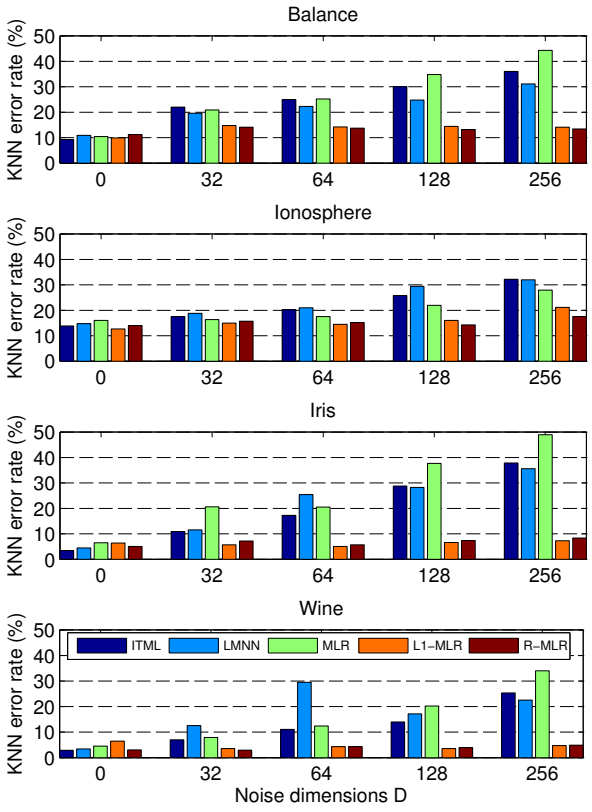

Figure 2. Accuracy of ITML, LMNN, MLR-ADMM and R-MLR as D noisy dimensions are introduced to UCI datasets. Results are averaged across 25 random splits. In all datasets, for $D \geq 64$, R-MLR significantly outperforms MLR, ITML and LMNN.

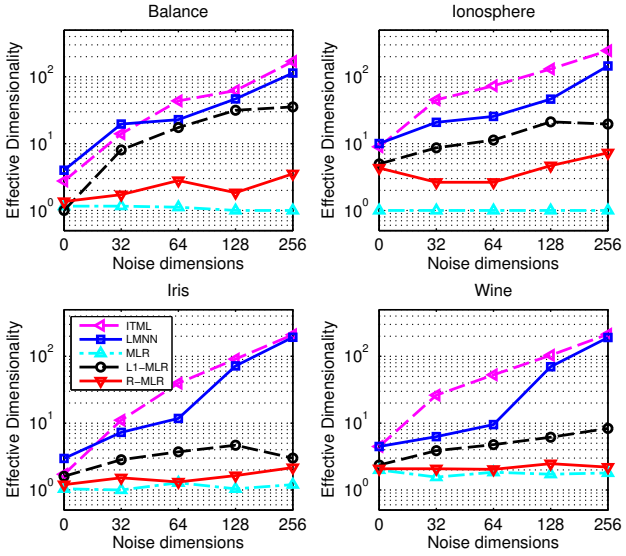

Figure 3. Mean effective dimensionality of learned metrics as dimensionality increases. For all datasets, MLR produces the lowest-dimensionality solutions across all values of D, while R-MLR slightly increases dimensionality due to the tradeoff between the low-rank and robust objectives. For LMNN and ITML, effective dimensionality scales with D across all datasets.

ability", "song hotttnesss", and artist latitude and longitude. These features are generally less pure than the audio and lyrics, but may carry some useful information.

Using the audio and lyrics representations, we compared the performance of MLR, ℓ_1-MLR, and R-MLR, first on the audio and lyrics representations, and then after including the *MSD-33* features. For this experiment, we vary $C \in \{10^{-1}, \cdots, 10^4\}$ and fix Δ to AUC (area under the ROC curve). For R-MLR and ℓ_1-MLR, we additionally vary $\lambda \in \{0.001, 0.01, 0.1, 1\}$. For each experiment, the hyper-parameters with the best AUC performance on the validation set are selected.

To investigate the performance gains afforded by using the ADMM-based algorithm, we compared MLR-ADMM to the original MLR implementation (MLR-Proj) on the *Audio* music similarity task, with the same experimental protocol as above. For MLR-ADMM, the maximum number of iterations T was additionally varied in $\{1, 5, 10, 25, 50, 100\}$. We tracked performance as well as as the number of projection operations onto $\mathbb{S}_+^d$ and calls to the constraint generator, which are the two key bottlenecks during training.

4.2.2. Results

Figure 4 shows the average AUC of each algorithm across 10 folds. When *MSD-33* features are included, R-MLR does significantly better than MLR under a Wilcoxon signed-rank test ($\alpha = 0.05$). Moreover, even in the original audio or lyrics features where the motivating assumption of noisy features may not hold, R-MLR does at least as well or better than MLR.

In these experiments, we noted that the metrics produced by ℓ_1-MLR tend to be strongly diagonal, with very few significant off-diagonal terms (due to the element-wise sparse regularizer). This may over-penalize weakly informative features, and limit the ability to exploit correlations between features. On the other hand, the group sparsity regularizer of R-MLR tends to produce solutions with denser rows, which better enables the algorithm to exploit feature correlations, leading to improved accuracy. In the experiments with MSD-33 features, we observed that the metrics learnt by MLR generally assign large weights to majority of the MSD-33 features, while the solutions learnt

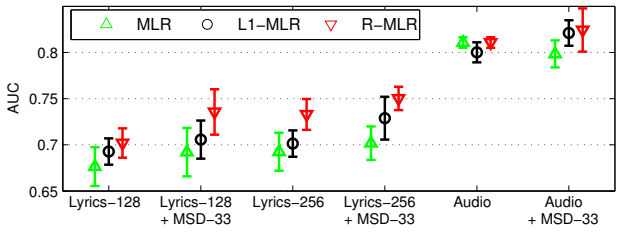

Figure 4. Music similarity performance of each algorithm on the three feature representations *Lyrics-128*, *Lyrics-256* and *Audio*, with and without *MSD-33* features. Performance was measured by AUC and averaged across 10 folds.

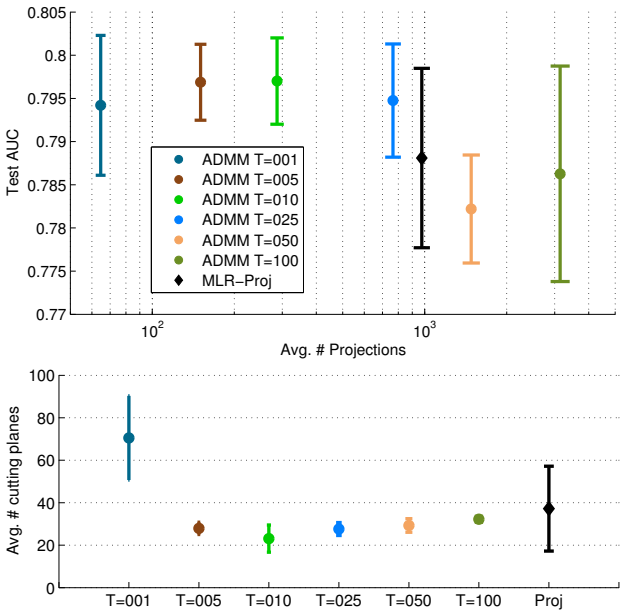

Figure 5. Performance of MLR-ADMM and MLR-Proj for the music similarity task with *Audio* features (best viewed in color). Top: accuracy versus the number of projections onto $\mathbb{S}_+^d$ as the maximum number of ADMM steps T is increased. Bottom: average number of cutting planes required for each value of T. Results are averaged across ten splits; error bars correspond to ± 1 standard deviation.

by R-MLR tended to suppress all but a few MSD-33 features.

The results of the early stopping experiment are presented in Figure 5. MLR-ADMM performs comparably to MLR-Proj across all values of T, and for small values of T, MLR-ADMM requires significantly fewer projection operations than MLR-Proj. For $T = 1$, the W returned at each step can be highly sub-optimal, and as a result, more cutting planes are required to converge. However, for intermediate values of T, the number of cutting planes does not significantly differ from MLR-Proj, and speedup is directly proportional to the decrease in projections.

4.3. Image Classification

In the image classification task, we compare the classification performance and training time of various algorithms. 100 images were chosen from each of 20 categories (classes) from the ImageNet repository, and each image was represented using 1000-dimensional codeword histograms obtained from the ImageNet database. For this experiment, the data set was split into 5 folds of 60/20/20, and the same hyper-parameter values were used as in Section 4.1. The early-stopping parameter T was fixed to 10 for all ADMM-based algorithms. As in the early stopping experiment, we recorded the mean accuracy and training time only for the best hyper-parameter settings for each fold.

4.3.1. Results

The results of the image classification task are shown in Table 1. The ADMM-based algorithms take significantly less time to run and achieve comparable accuracy to the existing methods, in particular MLR-Proj.

Table 1. Mean test set 3-NN accuracy and training time for the image classification task.

Algorithm	Accuracy	Time (h)
ITML	30.5%	4.67
LMNN	36.0%	8.82
MLR-Proj	36.7%	12.62
MLR-ADMM	37.5%	1.32
L1-MLR	37.7%	1.74
R-MLR	38.7%	1.55

5. Conclusion

We proposed a robust extension to the metric learning to rank algorithm, and derived an efficient learning algorithm. Our experiments demonstrate that by regularizing for both input and output sparsity, the R-MLR algorithm detects and suppresses noisy features, and outperforms previous methods in both low- and high-noise settings.

Acknowledgements

The authors acknowledge support from Qualcomm, Inc, Yahoo! Inc., Google, Inc., the Alfred P. Sloan Foundation, and NSF Grants CCF-0830535, IIS-1054960, and EIA-0303622. Daryl Lim was supported by a fellowship from the Agency for Science, Technology and Research (A*STAR), Singapore.

References

Bertin-Mahieux, Thierry, Ellis, Daniel P.W., Whitman, Brian, and Lamere, Paul. The million song dataset. In *International Conference on Music Information Retrieval*, 2011.

Blei, David M., Ng, Andrew Y., and Jordan, Michael I. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3: 993–1022, March 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944937.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and trends in machine learning*, 3(1): 1–122, 2011.

Davis, Jason V., Kulis, Brian, Jain, Prateek, Sra, Suvrit, and Dhillon, Inderjit S. Information-theoretic metric learning. In *ICML*, 2007.

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-jia, Li, Kai, and Li, Fei-fei. Imagenet: A large-scale hierarchical image database. In *In CVPR*, 2009.

Huang, Kaizhu, Jin, Rong, Xu, Zenglin, and Liu, Cheng-Lin. Robust metric learning by smooth optimization. In *UAI*, pp. 244–251, 2010.

Huang, Kaizhu, Ying, Yiming, and Campbell, Colin. Generalized sparse metric learning with relative comparisons. *Knowl. Inf. Syst.*, 28(1):25–45, 2011.

Joachims, T. A support vector method for multivariate performance measures. In *ICML*, 2005.

Joachims, Thorsten, Finley, Thomas, and Yu, Chun-Nam John. Cutting-plane training of structural svms. *Mach. Learn.*, 77(1):27–59, 2009.

Kowalski, M. Sparse regression using mixed norms. *Appl. Comput. Harmon. Anal.*, 27(3):303 – 324, 2009.

McFee, B., Barrington, L., and Lanckriet, G.R.G. Learning content similarity for music recommendation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(8):2207–2218, October 2012.

McFee, Brian and Lanckriet, G.R.G. Metric learning to rank. In *27th annual International Conference on Machine Learning (ICML)*, pp. 775–782, Haifa, Israel, June 2010.

Rosales, Rómer and Fung, Glenn. Learning sparse metrics via linear programming. In *KDD*, pp. 367–373, 2006.

Shaw, Blake, Huang, Bert, and Jebara, Tony. Learning a distance metric from a network. In *Advances in Neural Information Processing Systems 24*. 2011.

Shen, Chunhua, Kim, Junae, Wang, Lei, and van den Hengel, Anton. Positive semidefinite metric learning with boosting. In *Advances in Neural Information Processing Systems 22*. 2009.

Tingle, D., Kim, Y., and Turnbull, D. Exploring automatic music annotation with "acoustically-objective" tags. In *IEEE International Conference on Multimedia Information Retrieval*, 2010.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484, 2005.

Weinberger, Kilian Q., Blitzer, John, and Saul, Lawrence K. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, 2006.

Xing, Eric P., Ng, Andrew Y., Jordan, Michael I., and Russell, Stuart. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pp. 505–512, Cambridge, MA, 2003. MIT Press.

Ying, Yiming, Huang, Kaizhu, and Campbell, Colin. Sparse metric learning via smooth optimization. In *NIPS*, pp. 2214–2222, 2009.

Yuan, Ming and Lin, Yi. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, February 2006. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2005.00532.x. URL http://dx.doi.org/10.1111/j.1467-9868.2005.00532.x.

Zha, Zheng-Jun, Mei, Tao, Wang, Meng, Wang, Zengfu, and Hua, Xian-Sheng. Robust distance metric learning with auxiliary knowledge. In *IJCAI*, pp. 1327–1332, 2009.