Information-theoretic Semi-supervised Metric Learning via Entropy Regularization*

Gang Niu
Tokyo Institute of Technology, Japan
gang@sg.cs.titech.ac.jp

Bo Dai
Georgia Institute of Technology, USA
bohr.dai@gmail.com

Makoto Yamada
Yahoo Labs, USA
makotoy@yahoo-inc.com

Masashi Sugiyama
Tokyo Institute of Technology, Japan
sugi@cs.titech.ac.jp

Abstract

We propose a general information-theoretic approach to semi-supervised metric learning called SERAPH *(SEmi-supervised metRic leArning Paradigm with Hyper-sparsity)* that does not rely upon the manifold assumption. Given the probability parameterized by a Mahalanobis distance, we maximize its entropy on labeled data and minimize its entropy on unlabeled data following *entropy regularization*. For metric learning, entropy regularization improves *manifold regularization* by considering the dissimilarity information of unlabeled data in the unsupervised part, and hence it allows the supervised and unsupervised parts to be integrated in a natural and meaningful way. Moreover, we regularize SERAPH by *trace-norm regularization* to encourage low-dimensional projections associated with the distance metric. The non-convex optimization problem of SERAPH could be solved efficiently and stably by either a gradient projection algorithm or an EM-like iterative algorithm whose M-step is convex. Experiments demonstrate that SERAPH compares favorably with many well-known metric learning methods, and the learned Mahalanobis distance possesses high discriminability even under noisy environments.

Keywords: metric learning, semi-supervised learning, information-theoretic learning, entropy regularization

*This article is an extended version of Niu et al. (2012) that has appeared in the Proceedings of the 29th International Conference on Machine Learning. A Matlab implementation of the proposed SERAPH is available from the authors' website: http://sugiyama-www.cs.titech.ac.jp/~gang/software.html.

1 Introduction

How to learn a good distance metric for the input data domain is a crucial issue for many distance-based learning algorithms. The goal of metric learning is to find a new metric under which "similar" data are close and "dissimilar" data are far apart (Xing et al., 2003). The great majority of metric learning methods developed in the last decade fall into three types:

(a) Supervised type requiring *class labels* (e.g., Chiaromonte and Cook, 2002; Sugiyama, 2007; Fukumizu et al., 2009).[1] Two data points with the same label are regarded similar, and those with different labels are regarded dissimilar;

(b) Supervised type requiring *weak labels*, that is, $\{\pm 1\}$-valued labels that indicate the similarity and dissimilarity of data pairs directly (e.g., Xing et al., 2003; Goldberger et al., 2005; Weinberger et al., 2006; Globerson and Roweis, 2006; Torresani and Lee, 2007; Davis et al., 2007). See the illustration in Figure 1;

(c) Unsupervised type that requires no label information (e.g., Roweis and Saul, 2000; Tenenbaum et al., 2000; Belkin and Niyogi, 2002). Unlike previous types, the similarity and dissimilarity here are extracted from data instead of being given as supervision.

The second type has been extensively studied, since weak labels are much cheaper than class labels when the number of classes is fairly large. That being said, supervised metric learning based on weak labels still has a strict limitation: Algorithms of this type need each data point be involved in at least one weak label, otherwise these algorithms cannot see that data point as it never exists. This limitation is often problematic for real-world applications and needs to be fixed.

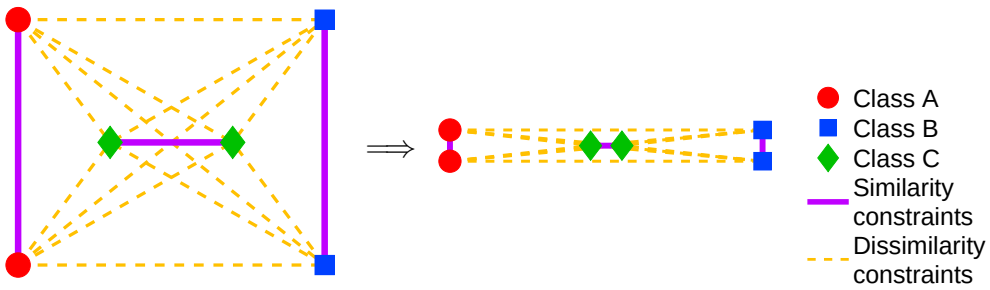
Based on the belief that preserving the geometric structure of all labeled and unlabeled data in an unsupervised manner can be better than strongly relying on the limited labeled data, semi-supervised metric learning has emerged. To the best of our knowledge, all previous semi-supervised methods that extend types (a) and (b) employ *off-the-shelf* unsupervised techniques in type (c). For example,

- Principal component analysis (e.g., Yang et al., 2006; Sugiyama et al., 2010);
- Manifold regularization or embedding (e.g., Hoi et al., 2008; Baghshah and Shouraki, 2009; Zha et al., 2009; Liu et al., 2010).

More specifically, they rely upon the *manifold assumption* and implement the following:

- If two data points are close under the original metric, pull them to make them not far away under the new metric;
- If two data points are far away under the original metric, do nothing.

[1] Learning a Mahalanobis distance (which will be defined in Eq. (1)) in the scenario of metric learning is equivalent to learning a projection in the scenario of dimensionality reduction, since the Mahalanobis distance of the original data equals the Euclidean distance of the projected data.



(a) Original data (b) Projected data

Figure 1: Illustration of supervised metric learning based on weak labels. In this figure, we have three classes, each with two labeled data. The goal is to find a new metric under which data in the same class are close and data from different classes are far apart. Note that the original class labels will not be revealed to metric learning algorithms, and we show the projected data here, since the Mahalanobis distance of the original data equals the Euclidean distance of the projected data.

In the second case, we should not push the two data points to make them far away under the new metric, since they may be connected by the data manifold and should be close under the new metric even though they are originally far away. By implementing these two cases, those semi-supervised methods successfully extract the similarity information of unlabeled data.

However, there remain two issues. First, those methods ignore the dissimilarity information of unlabeled data. This can be a huge waste of information, since most unlabeled data pairs would be dissimilar if the number of underlying classes is large and the classes are balanced. To this end, an appealing semi-supervised metric learning method should be able to make use of the dissimilarity information of unlabeled data. Second, similarity of unlabeled data extracted by those methods is measured by closeness under the original metric, and it is inconsistent with similarity of labeled data. Recall that metric learning aims at finding a new metric, and weak labels indicating similar but far away data pairs are in principle the most informative ones. Therefore, under the original metric, closeness is not the reason for similarity of labeled data, whereas it is the reason for similarity of unlabeled data. In contrast, similarity and closeness generally imply each other for both labeled and unlabeled data under the new metric. To this end, an appealing method should focus on the new metric when extracting the similarity information of unlabeled data. As a matter of fact, the unsupervised parts of the existing methods that rely upon the manifold assumption and implement the aforementioned two cases are inconsistent with their supervised parts in terms of these two issues. Simply putting them together works in practice, but this paradigm is conceptually neither natural nor unified.

In this paper, we propose a general information-theoretic approach to semi-supervised metric learning called SERAPH *(SEmi-supervised metRic leArning Paradigm with Hyper-*

sparsity), in order to address these issues. It extracts not only the similarity information but also the dissimilarity information of unlabeled data, and to do so it accesses the new metric rather than the original one. Our idea is to optimize a new Mahalanobis distance metric through optimizing a conditional probability parameterized by that metric. We maximize the entropy of this probability on labeled data pairs, and minimize the entropy of this probability on unlabeled data pairs following *entropy regularization* (Grandvalet and Bengio, 2005), which can achieve the sparsity of the posterior distribution (Graça et al., 2009; Gillenwater et al., 2011), i.e., unlabeled data pairs can be classified with high confidence. Furthermore, we employ *mixed-norm regularization* (Argyriou et al., 2007) to encourage the sparsity of projection matrices associated with the new metric in terms of their singular values (Ying et al., 2009), and the new metric can carry out dimensionality reduction implicitly and adaptively. Unifying the posterior sparsity and the projection sparsity brings to us the *hyper-sparsity*. Thanks to this hyper-sparsity, the new metric learned by SERAPH possesses high discriminability even under noisy environments.

Our contributions can be summarized as three folds. Firstly, we formulate supervised metric learning based on weak labels as an instance of the generalized maximum entropy distribution estimation (Dudík and Schapire, 2006). Secondly, we propose an extension of this estimation to semi-supervised metric learning via entropy regularization (Grandvalet and Bengio, 2005). It is able to consider the dissimilarity information of unlabeled data based on the Mahalanobis distance being learned. Thirdly, we develop two ways to solve the non-convex optimization problem involved in this extension, namely, a direct gradient projection algorithm and an indirect EM-like iterative algorithm.

The rest of this paper is organized as follows. The model of SERAPH is formulated in Section 2, and then two algorithms are developed in Section 3 to solve the optimization problem involved in the model. In Section 4, we discuss three sparsity mentioned above and two additional justifications of the model. A comparison with related works is made in Section 5. Experimental results are reported in Section 6. Then in Section 7, we offer two extensions to SERAPH. Finally, we give concluding remarks and future prospects in Section 8.

2 SERAPH, the Model

In this section, we formulate the model of SERAPH. We first propose the supervised part, and then introduce its regularization terms.

2.1 Problem setting

Suppose that we have a training set $\mathcal{X} = \{x_i \mid x_i \in \mathbb{R}^m\}_{i=1}^n$ which contains n points each with m features. Let the set of similar data pairs be

$$\mathcal{S} = \{(x_i, x_j) \mid x_i \text{ and } x_j \text{ are similar}\},$$

and the set of dissimilar data pairs be

$$\mathcal{D} = \{(x_i, x_j) \mid x_i \text{ and } x_j \text{ are dissimilar}\}.$$

With some abuse of terminology, we refer to $\mathcal{S} \cup \mathcal{D}$ as the labeled data, and

$$\mathcal{U} = \{(x_i, x_j) \mid i \neq j, (x_i, x_j) \notin \mathcal{S} \cup \mathcal{D}\}$$

as the unlabeled data. A weak label $y_{i,j}$ is assigned to (x_i, x_j) such that

$$y_{i,j} = \begin{cases} +1 & \text{if } (x_i, x_j) \in \mathcal{S}, \\ -1 & \text{if } (x_i, x_j) \in \mathcal{D}, \\ \text{undefiend} & \text{if } (x_i, x_j) \in \mathcal{U}. \end{cases}$$

We abbreviate $\sum_{(x_i,x_j)\in\mathcal{S}\cup\mathcal{D}}$, $\sum_{(x_i,x_j)\in\mathcal{U}}$ and $\sum_{y\in\{-1,+1\}}$ to $\sum_{\mathcal{S}\cup\mathcal{D}}$, $\sum_{\mathcal{U}}$ and $\sum_y$ respectively for simplicity. Consider learning a Mahalanobis distance metric for $x, x' \in \mathbb{R}^m$ of the form

$$d(x, x') = \|x - x'\|_A = \sqrt{(x - x')^\top A (x - x')}, \tag{1}$$

where $^\top$ is the transpose operator, and $A \in \mathbb{R}^{m\times m}$ is a symmetric positive semi-definite matrix to be learned[2]. The probability of labeling $(x, x') \in \mathbb{R}^m \times \mathbb{R}^m$ with $y = \pm 1$ is denoted by $p^A(y \mid x, x')$ that is explicitly parameterized by the matrix A. When the pair (x, x') comes from $\mathcal{S} \cup \mathcal{D} \cup \mathcal{U}$, $p^A(y \mid x_i, x_j)$ is abbreviated into $p^A_{i,j}(y)$.

2.2 Basic model

To begin with, we derive a probabilistic model to investigate the conditional probability of $y = \pm 1$ given $(x, x') \in \mathbb{R}^m \times \mathbb{R}^m$. We resort to a parametric form of $p^A(y \mid x, x')$ and we will focus on this form as it is optimal in the following sense.

The *maximum entropy principle* (Jaynes, 1957; Berger et al., 1996) suggests us to choose the probability distribution with the maximum entropy out of all probability distributions that match the data moments. Let[3]

$$H(p^A_{i,j}) = -\sum\nolimits_y p^A_{i,j}(y) \ln p^A_{i,j}(y)$$

be the entropy of the conditional probability $p^A_{i,j}(y)$, and

$$f(x, x', y; A) : \mathbb{R}^m \times \mathbb{R}^m \times \{+1, -1\} \mapsto \mathbb{R}$$

[2]In the rest of this paper, the matrix A is always assumed to be symmetric and positive semi-definite if it is an optimization variable, and the constraints $A = A^\top$ and $A \succeq 0$ will not be explicitly written for convenience.

[3]Throughout this paper, we adopt that $0 \ln 0 = \lim_{x\to 0^+} x \ln x = 0$.

be a feature function that is convex with respect to A, then the constrained optimization problem is

$$\begin{aligned} \max_{A,p^A_{i,j},\xi} \quad & \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} H(p^A_{i,j}) - \frac{1}{2\gamma}\xi^2 \\ \text{s.t.} \quad & \left|\sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \mathbb{E}_{p^A_{i,j}}[f(x_i, x_j, y; A)] - \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} f(x_i, x_j, y_{i,j}; A)\right| \leq \xi, \end{aligned} \tag{2}$$

where ξ is a slack variable and $\gamma > 0$ is a regularization parameter. After the introduction of ξ, distributions are allowed to match two data moments in a way that is not strictly exact. The penalty $\xi^2/(2\gamma)$ in the objective function presumes the Gaussian prior of the expected data moment

$$\sum_{\mathcal{S}\cup\mathcal{D}} \mathbb{E}_{p^A_{i,j}}[f(x_i, x_j, y; A)]$$

from the empirical data moment

$$\sum_{\mathcal{S}\cup\mathcal{D}} f(x_i, x_j, y_{i,j}; A),$$

which is consistent with the *generalized maximum entropy principle* (Dudík and Schapire, 2006). Please see Section 4.2 for the alternative explanation of optimization (2) in the sense of the generalized maximum entropy principle, particularly the necessity of introducing the slack variable ξ from a theoretical point of view.

Theorem 1. *The primal solution $p^{*A}(y \mid x, x')$ to optimization (2) can be given in terms of the dual solution (A^*, κ^*) by*

$$p^{*A}(y \mid x, x') = \frac{\exp(\kappa^* f(x, x', y; A^*))}{Z(x, x'; A^*, \kappa^*)}, \tag{3}$$

where

$$Z(x, x'; A, \kappa) = \sum\nolimits_{y'} \exp(\kappa f(x, x', y'; A))$$

is the partition function, and (A^*, κ^*) can be obtained by solving the dual problem

$$\min_{A,\kappa} \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \ln Z(x_i, x_j; A, \kappa) - \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \kappa f(x_i, x_j, y_{i,j}; A) + \frac{\gamma}{2}\kappa^2. \tag{4}$$

Let $p^A(y \mid x, x')$ take the form of $p^{*A}(y \mid x, x')$ in (3). Define the regularized log-likelihood function on labeled data (i.e., on observed weak labels) as

$$\mathcal{L}_1(A, \kappa) = \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \ln p^A_{i,j}(y_{i,j}) - \frac{\gamma}{2}\kappa^2.$$

Then for supervised metric learning, the regularized maximum log-likelihood estimation

$$\max_{A,\kappa} \mathcal{L}_1(A, \kappa)$$

and the generalized maximum entropy estimation (2) are equivalent.[4]

[4]The proofs of all theorems are in Appendix A.

When considering $f(x, x', y; A)$ that should take moments about the metric information into account, we propose[5]

$$f(x, x', y; A, \eta) = -\frac{y}{2}(\|x - x'\|_A^2 - \eta), \tag{5}$$

where $\eta > 0$ is a hyperparameter that serves as the threshold to separate the similar and dissimilar data pairs in $\mathcal{S}$ and $\mathcal{D}$ under the new metric $d(x, x')$. Now the probabilistic model (3) becomes

$$p^A(y \mid x, x') = \frac{1}{1 + \exp(\kappa y(\|x - x'\|_A^2 - \eta))}.$$

For the optimal solution (p^{*A}, A^*, κ^*) and reasonable η, we hope for two properties:

(i) The feature function can indicate the correctness of the observed weak labels, i.e.,

$$f(x_i, x_j, y_{i,j}; A^*, \eta) = -y_{i,j}(\|x_i - x_j\|_{A^*}^2 - \eta)/2 > 0;$$

(ii) The probabilistic model can correctly classify the observed weak labels, i.e.,

$$p^{*A}(y_{i,j} \mid x_i, x_j) = 1/\left(1 + \exp(\kappa^* y_{i,j}(\|x_i - x_j'\|_{A^*}^2 - \eta))\right) > 1/2.$$

Therefore, there must be $\kappa^* > 0$.

Note that the generalized maximum entropy estimation for supervised metric learning is a general framework, and it is not limited to supervised metric learning based on weak labels. Although we use (5) as our feature function, other feature functions emphasizing different perspectives of the metric information are possible. For instance, a local distance metric feature function

$$f(x, x', y; A) = -\frac{y}{2}(\|x - x'\|_A^2 - \|x - x'\|_2^2)$$

replaces the global threshold η with a local one $\|x - x'\|_2^2$ and focuses on the changes of pairwise distances. In fact, optimization (2) can even be applied to other problem settings such as multi-label metric learning with a global distance metric feature function

$$f(x, x', y, y'; A, \eta) = \left(\frac{1}{2} - \frac{\langle y, y' \rangle}{\|y\|_2 \|y'\|_2}\right)(\|x - x'\|_A^2 - \eta),$$

where the labels y and y' are binary-valued vectors.

[5]Note that in Niu et al. (2012) the feature function is $f(x, x', y; A, \eta) = y(\|x - x'\|_A^2 - \eta)/2$ that has the opposite sign with the feature function in Eq. (5). However, they are equivalent feature functions, since the signs of κ^* are also opposite here and there.

2.3 Regularization

In this subsection, we extend $\mathcal{L}_1(A, \kappa)$ defined above to semi-supervised metric learning via entropy regularization, and further regularize it by trace-norm regularization.

Our unsupervised part extracts both the similarity and dissimilarity information of unlabeled data according to the new Mahalanobis distance metric $d(x, x')$. In order to do so, it follows the *minimum entropy principle* (Grandvalet and Bengio, 2005), and hence $p^A_{i,j}(y)$ should have low entropy (which in turn means low uncertainty) for unlabeled data $(x_i, x_j) \in \mathcal{U}$. Generally speaking, the resultant discriminative probabilistic models prefer peaked distributions on unlabeled data such that unlabeled data can be classified with high confidence, which can carry out a probabilistic *low-density separation*. Subsequently, according to Grandvalet and Bengio (2005), our optimization becomes

$$\max_{A,\kappa} \mathcal{L}_2(A, \kappa) = \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \ln p^A_{i,j}(y_{i,j}) + \mu \sum\nolimits_{\mathcal{U}} \sum\nolimits_y p^A_{i,j}(y) \ln p^A_{i,j}(y) - \frac{\gamma}{2}\kappa^2,$$

where $\mu \geq 0$ is a regularization parameter.

In addition, we hope for the dimensionality reduction ability by encouraging low-rank projection matrices associated with A. It would be helpful in dealing with corrupted data or data distributed intrinsically in a low-dimensional subspace. It is known that the trace is a convex relaxation of the rank for positive semi-definite matrices, so we revise our optimization problem into

$$\max_{A,\kappa} \mathcal{L}(A, \kappa) = \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \ln p^A_{i,j}(y_{i,j}) + \mu \sum\nolimits_{\mathcal{U}} \sum\nolimits_y p^A_{i,j}(y) \ln p^A_{i,j}(y) - \frac{\gamma}{2}\kappa^2 - \lambda \operatorname{tr}(A), \quad (6)$$

where $\operatorname{tr}(A)$ is the trace of A, and $\lambda \geq 0$ is a regularization parameter.

Optimization problem (6) is the final model of SERAPH. We say that it is equipped with the hyper-sparsity when both μ and λ are positive and hence both regularization terms are active. The hyper-sparsity, as well as the posterior and projection sparsity, will be discussed in Section 4.1. Moreover, SERAPH possesses standard kernel and manifold extensions, and we will explain them in Sections 7.1 and 7.2 respectively.

3 SERAPH, the Algorithm

In this section, we reduce optimization (6) to a form that is easy to handle, and develop two practical algorithms for solving the reduced optimization.

3.1 Reduction

While optimization (6) involves a dual variable κ, we would like to focus on the variable A just as many previous metric learning methods. The theorem below guarantees that we can eliminate κ from (6) to get an equivalent but simpler optimization, thanks to the fact that we use a single feature function (5) in optimization (2).

Theorem 2. *Define the reduced optimization problem as*[6]

$$\max_{A} \ \hat{\mathcal{L}}(A) = \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \ln \hat{p}^{A}_{i,j}(y_{i,j}) + \mu \sum\nolimits_{\mathcal{U}} \sum\nolimits_{y} \hat{p}^{A}_{i,j}(y) \ln \hat{p}^{A}_{i,j}(y) - \hat{\lambda}\,\mathrm{tr}(A), \tag{7}$$

where the reduced probabilistic model is

$$\hat{p}^{A}(y \mid x, x') = \frac{1}{1+\exp(y(\|x-x'\|_A^2 - \hat{\eta}))}. \tag{8}$$

Let (A^*, κ^*) be a locally optimal solution to (6)*. Then, there exist well-defined $\hat{\eta}$ and $\hat{\lambda}$, such that $\hat{A} = \kappa^* A^*$ is also a locally optimal solution to* (7) *and it satisfies*

(i) $d(x, x')$ parameterized by $\hat{A}$ is equivalent to $d(x, x')$ parameterized by A^*, i.e.,

$$\forall x, x' \in \mathbb{R}^m, \ \frac{d(x, x'; \hat{A})}{d(x, x'; A^*)} = \text{Const.}; \tag{9}$$

(ii) $\hat{p}^{A}(y \mid x, x')$ parameterized by $\hat{A}$ and $\hat{\eta}$ is identical to the original $p^{A}(y \mid x, x')$ parameterized by A^*, κ^* and η, i.e.,

$$\forall x, x' \in \mathbb{R}^m, y \in \{-1, +1\}, \ \hat{p}^{A}(y \mid x, x'; \hat{A}, \hat{\eta}) = p^{A}(y \mid x, x'; A^*, \kappa^*, \eta). \tag{10}$$

Remark 1. After the reduction of Theorem 2, γ has been dropped, η and λ have been modified, but the regularization parameter μ remains the same, which means that the tradeoff between the supervised and unsupervised parts has not been affected.

3.2 Two algorithms

There are several approaches for solving optimization (7). For example, gradient projection and expectation maximization (cf. Grandvalet and Bengio, 2006, pp. 155–158). By no means an approach can always be better than another for non-convex optimizations. Hence, we explore both of them and find they can solve (7) efficiently and stably.

Our first solver for (7) is a direct gradient projection algorithm (Polyak, 1967). The gradient matrix $\nabla\mathcal{L}(A)$ is simply

$$\begin{aligned} \nabla\mathcal{L}(A) = &- \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} y_{i,j}(1 - p^{A}_{i,j}(y_{i,j}))(x_i - x_j)(x_i - x_j)^\top \\ &- \mu \sum\nolimits_{\mathcal{U}} \sum\nolimits_{y} y(1 + \ln p^{A}_{i,j}(y)) p^{A}_{i,j}(y)(1 - p^{A}_{i,j}(y))(x_i - x_j)(x_i - x_j)^\top - \lambda I_m. \end{aligned} \tag{11}$$

The projection of the symmetric matrix resulted from a gradient update back to the cone of symmetric positive semi-definite matrices includes eigen-decomposing that symmetric matrix and recovering it from its positive eigenvalues and eigenvectors associated with

[6] The new functions and parameters are denoted by $\hat{\cdot}$ within this theorem for the sake of clarity.

those eigenvalues. Although this algorithm must converge, many heuristic tricks are necessary in order to find a reasonable locally optimal solution to (7) since the unsupervised part is highly non-convex.

Our second solver for (7) is an indirect EM-like iterative algorithm. It runs as follows. In the beginning, we initialize a probability $q(y \mid x_i, x_j)$ for each pair $(x_i, x_j) \in \mathcal{U}$. The initial solution in our current implementation is $q(y = -1 \mid x_i, x_j) = 1$, which means that at the beginning we treat all unlabeled pairs as dissimilar pairs. Then, the M-step and the E-step get executed repeatedly until certain stopping conditions are satisfied. At the t-th M-step, we find new metric $A^{(t)}$ through a surrogate optimization:

$$\max_A \ \mathcal{F}(A) = \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \ln p^A_{i,j}(y_{i,j}) + \mu \sum\nolimits_{\mathcal{U}} \sum\nolimits_y q(y \mid x_i, x_j) \ln p^A_{i,j}(y) - \lambda \operatorname{tr}(A), \tag{12}$$

where $q(y \mid x_i, x_j)$ is generated in the last E-step. Since the feature function $f(x, x', y; A)$ is convex with respect to A, the objective function $\mathcal{F}(A)$ is concave with respect to A and optimization (12) is convex according to Boyd and Vandenberghe (2004, p. 74). Thus, we could solve optimization (12) using the gradient projection method without worry about local maxima, where the gradient matrix $\nabla\mathcal{F}(A)$ is

$$\begin{aligned} \nabla\mathcal{F}(A) = &- \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} y_{i,j}(1 - p^A_{i,j}(y_{i,j}))(x_i - x_j)(x_i - x_j)^\top \\ &- \mu \sum\nolimits_{\mathcal{U}} \sum\nolimits_y y q(y \mid x_i, x_j)(1 - p^A_{i,j}(y))(x_i - x_j)(x_i - x_j)^\top - \lambda I_m. \end{aligned} \tag{13}$$

At the t-th E-step, we update $q(y \mid x_i, x_j)$ for each pair $(x_i, x_j) \in \mathcal{U}$ as

$$q(y \mid x_i, x_j) = \frac{\left(p^A_{i,j}(y)\right)^{1+\mu/t}}{\sum_{y'} \left(p^A_{i,j}(y')\right)^{1+\mu/t}}, \tag{14}$$

where $p^A_{i,j}(y)$ is parameterized by $A^{(t)}$ found in the last M-step. Although this algorithm may not converge, it works fairly well in practice. No matter how we design the M-step, it is insensitive to the step size of the gradient update and it gives a deterministic solution after fixing the initial solution and the stopping conditions. In other words, the EM-like iterative algorithm can easily be de-randomized by the initial solution and the stopping conditions, which is a nice algorithmic property for non-convex optimizations.

For the details of our implementation, please see Appendix B.

3.3 Theoretical analyses

The gradient projection and EM-like algorithms developed above are able to solve optimization (7) efficiently and stably. Let us figure out their asymptotic time complexities. Generally speaking, the asymptotic time complexity of both algorithms is $O(n^2m + m^3)$, where n is the number of data and m is the number of features (recall that the training set $\mathcal{X}$ contains n points each with m features). Specifically, each iteration of the gradient projection algorithm consumes $O(n^2m + nm^2)$ for the gradient update and $O(m^3)$ for the

projection, which has an asymptotic time complexity $O(n^2m + m^3)$, since $O(nm^2)$ could never dominate $O(n^2m)$ and $O(m^3)$ simultaneously. Additionally, it is common to set in advance a maximum number of iterations T_{GP} for such a non-convex optimization solver, and the overall asymptotic time complexity of the gradient projection algorithm is

$$O\left((n^2m + m^3)T_{\mathrm{GP}}\right).$$

For the EM-like iterative algorithm, each iteration of the M-step is same as the gradient projection algorithm, and each E-step costs $O(n^2)$ which is negligible compared with the computational complexity of the whole M-step. As a consequence, the overall asymptotic time complexity of the EM-like algorithm is

$$O\left((n^2m + m^3)T'_{\mathrm{GP}}T_{\mathrm{EM}}\right),$$

where T'_{GP} is the maximum number of iterations of the M-step and T_{EM} is the maximum number of iterations of the EM-like algorithm.

It is obvious that which algorithm is empirically faster depends primarily on which of T_{GP} or $T'_{\mathrm{GP}}T_{\mathrm{EM}}$ is smaller. In fact, the gradient projection method for (12) is much easier than for (7) since (12) is a convex optimization, which means the M-step of the EM-like algorithm itself is much easier than the gradient projection algorithm. Furthermore, it is unnecessary to solve the M-step exactly in such an EM-like algorithm. As a result, T'_{GP} is supposed to be significantly smaller than T_{GP}. On the other hand, the temporary $A^{(t)}$ of EM-like iterations make up a deterministic sequence $(A^{(1)}, \ldots, A^{(t)}, \ldots)$ for fixed initial $q(y \mid x_i, x_j)$, and a small T_{EM} is usually enough for finding a reasonable solution. To sum up, we can set $T'_{\mathrm{GP}}T_{\mathrm{EM}}$ to be smaller than T_{GP} in practice, and then expect the EM-like algorithm to be faster than the gradient projection algorithm with comparable qualities of the learned distance metrics.

The gradient projection and EM-like algorithms are not only computationally efficient but also computationally stable. The following theorem shows that the gradient matrices of $\mathcal{L}(A)$ and $\mathcal{F}(A)$ given in Eqs. (11) and (13) are uniformly bounded, regardless of the scale of A, i.e., the magnitude of $\mathrm{tr}(A)$. It also implies that compared with maximizing $\mathcal{L}(A)$, maximizing $\mathcal{F}(A)$ should be more stable even without considering that $\mathcal{F}(A)$ is a concave function.

Theorem 3. *The objective functions $\mathcal{L}(A)$ and $\mathcal{F}(A)$ of optimizations* (7) *and* (12) *are Lipschitz continuous, and the best Lipschitz constants with respect to the Frobenius norm $\|\cdot\|_{\mathrm{Fro}}$ satisfy*

$$\mathrm{Lip}_{\|\cdot\|_{\mathrm{Fro}}}(\mathcal{L}) \leq (\#\mathcal{S} + \#\mathcal{D} + (1 + \ln 2)\mu\#\mathcal{U})(\mathrm{diam}(\mathcal{X}))^2 + \lambda m, \tag{15}$$

$$\mathrm{Lip}_{\|\cdot\|_{\mathrm{Fro}}}(\mathcal{F}) \leq (\#\mathcal{S} + \#\mathcal{D} + \mu\#\mathcal{U})(\mathrm{diam}(\mathcal{X}))^2 + \lambda m, \tag{16}$$

where $\mathrm{diam}(\mathcal{X}) = \max_{x_i, x_j \in \mathcal{X}} \|x_i - x_j\|_2$ *is the diameter of $\mathcal{X}$, and # measures the cardinality of a set.*

Last but not least, we would like to comment on Eq. (14), i.e., the E-step of the EM-like algorithm. It has the same idea as the deterministic annealing EM-like algorithm in Grandvalet and Bengio (2006), and it is the analytical solution to

$$\min_q \ \mathrm{KL}(q \,||\, p^A_{i,j}) - (\mu/t) \sum\nolimits_y q(y \mid x_i, x_j) \ln p^A_{i,j}(y)$$

similarly to Graça et al. (2009) and Gillenwater et al. (2011), where KL is the Kullback-Leibler divergence. It is easy to see that our E-step is different from the standard E-step if $\mu \neq 0$, while for any μ it approaches the standard one as $t \to \infty$. In other words, the EM-like algorithm does not solve optimization (7) exactly, but (7) is indeed the limit of a sequence of optimizations which the algorithm solves at different EM-like iterations. If $\mu \neq 0$ and $t = 0$, $q(y \mid x_i, x_j)$ becomes the hard assignments

$$q(y \mid x_i, x_j) = \begin{cases} 1 & \text{if } p^A_{i,j}(y) > 0.5, \\ 0.5 & \text{if } p^A_{i,j}(y) = 0.5, \\ 0 & \text{if } p^A_{i,j}(y) < 0.5. \end{cases}$$

This is the reason for initializing $q(y \mid x_i, x_j) \in \{0, 1\}$ in our current implementation.

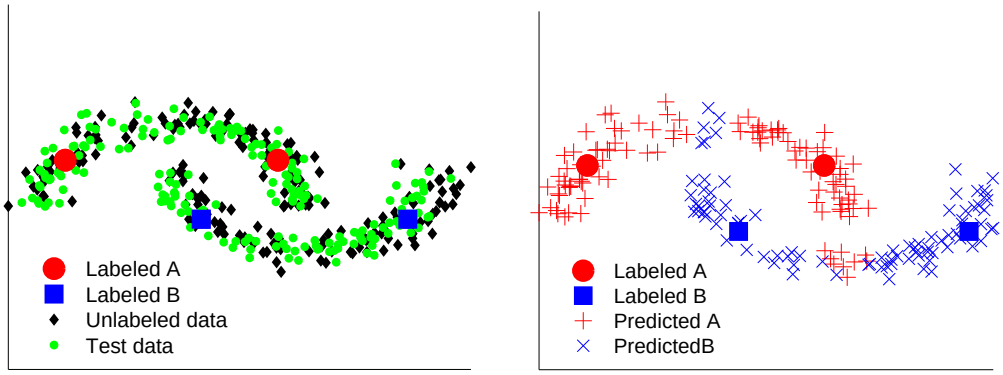
4 Discussions

We have left out a few theoretical arguments when we proposed the model of Seraph in order to keep the presentation as concise and comprehensible as possible. In this section, we discuss the sparsity issue in the sense of metric learning and present two additional justifications for our model.

4.1 Posterior sparsity and projection sparsity

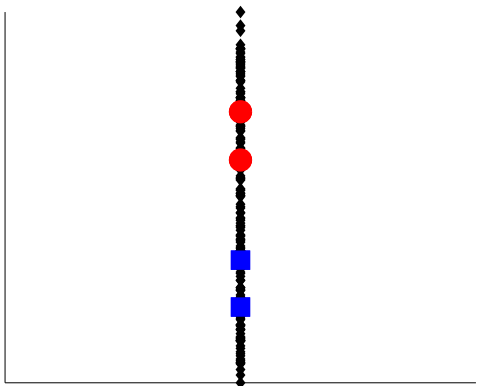
Sparse metric learning might have different meanings, since we learn a metric with low-rank linear projections by optimizing a conditional probability, where the optimization variable is actually a square matrix. First of all, we would like to explain the meaning of our sparsity and claim that we can obtain the *posterior sparsity* (Graça et al., 2009) by entropy regularization and the *projection sparsity* (Ying et al., 2009) by trace-norm regularization. The arguments are as follows.

By a "sparse" posterior distribution, we mean that the uncertainty (e.g., the entropy or variance) of $p^A_{i,j}(y)$ for $(x_i, x_j) \in \mathcal{U}$ is low, such that (x_i, x_j) can be classified to be a similar or dissimilar pair with high confidence. Figure 2 is an illustrative example. Recall that supervised metric learning aims at finding a new distance metric under which data in the same class are close and data from different classes are far apart. It would result in the metric which ignores the horizontal feature and only focuses on the vertical feature. Nevertheless, the horizontal feature is useful, and taking care of the posterior sparsity would lead to a better metric as shown in subfigures (e) and (f). As a consequence, we

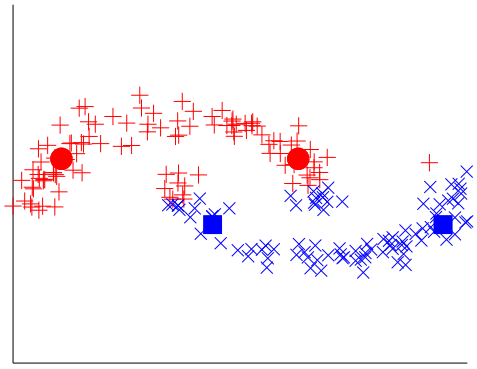


(a) Data set

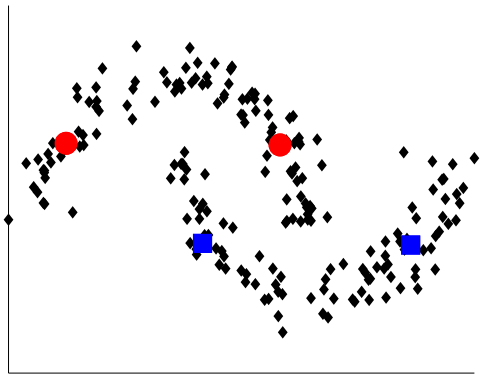
(b) Euclidean



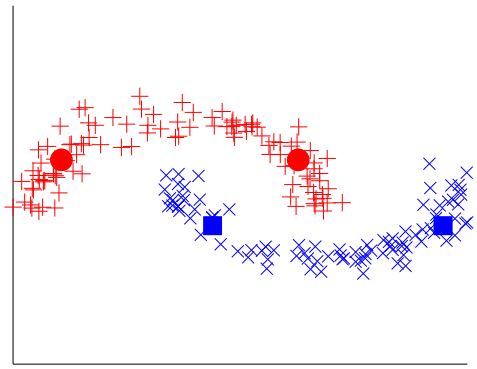
(c) Non-sparse



(d) Non-sparse



(e) Sparse



(f) Sparse

Figure 2: Sparse vs. non-sparse posterior distributions. Six weak labels were constructed according to the four class labels. The left three panels show the original data and the projected data by metrics learned with/without the posterior sparsity. The right three panels exhibit one-nearest-neighbor classification results based on the Euclidean distance and the two learned metrics.

prefer taking the posterior sparsity into account in addition to the aforementioned goal of supervised metric learning, and then the risk of overfitting weakly labeled data can be significantly reduced.

When considering the posterior sparsity, our optimization via entropy regularization is equivalent to soft posterior regularization (Graça et al., 2009; Gillenwater et al., 2011), that is, we can rewrite $\mathcal{L}_2(A, \kappa)$ as an objective function of a soft posterior regularization. More specifically, let the auxiliary feature function be

$$g(x, x', y) = -\ln p^A(y \mid x, x'),$$

then maximizing $\mathcal{L}_2(A, \kappa)$ is equivalent to

$$\max_{A,\kappa} \ \mathcal{L}_1(A, \kappa) - \mu \sum\nolimits_{\mathcal{U}} \mathbb{E}_{p^A_{i,j}}[g(x_i, x_j, y)]. \tag{17}$$

On the other hand, according to optimization (7) of Graça et al. (2009), the soft posterior regularization objective should take a form as

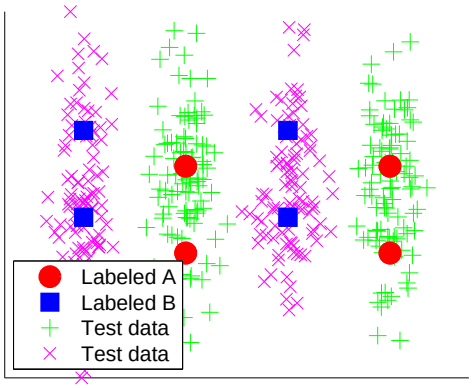
$$\begin{aligned} \max_{A,\kappa} \ & \mathcal{L}_1(A, \kappa) - \min_{q} \left(\mathrm{KL}(q \mid\mid p^A) + \mu \sum\nolimits_{\mathcal{U}} \xi_{i,j} \right) \\ \text{s.t.} \ & \mathbb{E}_q[g(x_i, x_j, y)] \leq \xi_{i,j}, \forall (x_i, x_j) \in \mathcal{U}, \end{aligned} \tag{18}$$

where $\xi_{i,j}$ are slack variables. Since q is unconstrained, we can optimize q with respect to fixed A and κ. It is easy to see that q should be p^A restricted on $\mathcal{U}$, so the KL divergence term is zero and the expectation term is the entropy, which implies the equivalence of optimizations (17) and (18).

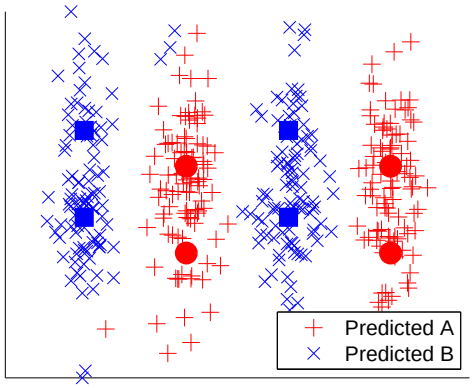
Besides the posterior sparsity, we also hope for the projection sparsity that may guide the new distance metric to a better generalization performance. Figure 3 illustrates its effect, where the horizontal feature is dominant and the vertical feature is uninformative. The underlying technique is known as mixed-norm regularization (Argyriou et al., 2007) or group lasso (Yuan and Lin, 2006). Denote the $\ell_{(2,1)}$-norm of a symmetric matrix M as

$$\|M\|_{(2,1)} = \sum_{k=1}^{m} \left(\sum_{k'=1}^{m} M_{k,k'}^2 \right)^{1/2}.$$

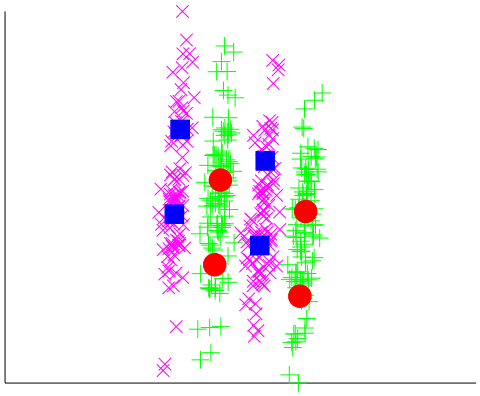
Similarly to Ying et al. (2009), let $P \in \mathbb{R}^{m \times m}$ be a linear projection, $W = P^\top P$ be the symmetric positive semi-definite matrix of the metric induced from P, and P_i and W_i be the i-th columns of P and W. If P_i is identically zero, the i-th component of x has no contribution to $z = Px$. Since the column-wise sparsity of W and P are equivalent, we can reach the column-wise sparsity of P by penalizing $\|W\|_{(2,1)}$. Nevertheless, this is the ability of feature selection rather than dimensionality reduction. Note that the goal is to select a few most representative directions of input data which are not restricted to the coordinate axes. The solution is to pick an extra transformation $V \in \mathcal{O}^m$ to rotate x before projecting x where $\mathcal{O}^m$ is the set of orthonormal matrices of size m. Consequently,



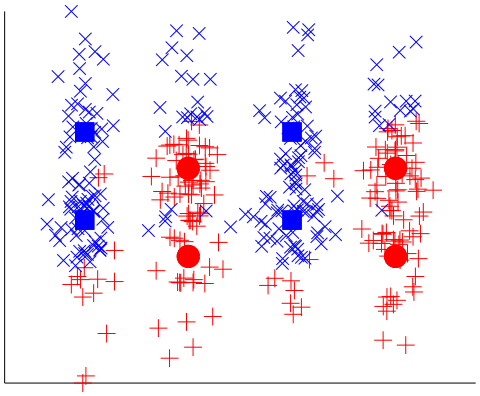
(a) Data set



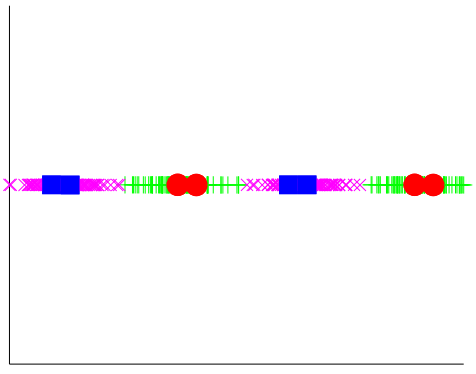
(b) Euclidean



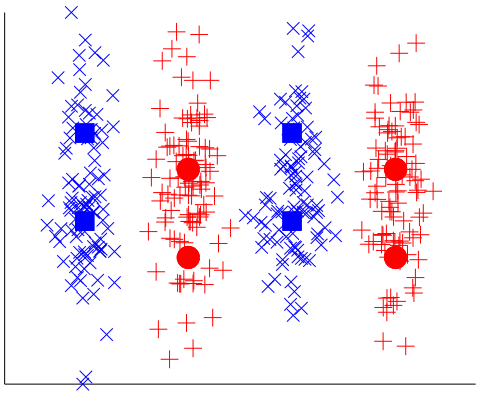
(c) Non-sparse



(d) Non-sparse



(e) Sparse



(f) Sparse

Figure 3: Sparse vs. non-sparse projections. Twenty-eight weak labels were constructed according to the eight class labels. The left three panels show the original data and the projected data by metrics learned with/without the projection sparsity. The right three panels exhibit one-nearest-neighbor classification results based on the Euclidean distance and the two learned metrics.

we penalize $\|W\|_{(2,1)}$, project x to $z = PVx$, and since $A = (PV)^\top(PV) = V^\top WV$, we arrive at

$$\begin{aligned} \max_{A,\kappa,W,V} \quad & \mathcal{L}_2(A,\kappa) - \lambda\|W\|_{(2,1)} \\ \text{s.t.} \quad & A = V^\top WV, W = W^\top, W \succeq 0, V \in \mathcal{O}^m. \end{aligned} \tag{19}$$

Remember that the final model of SERAPH was given by optimization (6) as

$$\max_{A,\kappa} \ \mathcal{L}_2(A,\kappa) - \lambda \operatorname{tr}(A).$$

The equivalence of optimizations (6) and (19) is guaranteed by Lemma 1 of Ying et al. (2009). By unifying the posterior sparsity and the projection sparsity mentioned above, we obtain a property that we call the *hyper-sparsity*.

4.2 Generalized maximum entropy principle

The basic model defined in optimization (2) contains an inequality constraint instead of some equality constraint, since the regularization term $-\gamma\kappa^2/2$ in $\mathcal{L}_1(A,\kappa)$ is indispensable. Otherwise we would have $\kappa^* = +\infty$ for the optimal solution (A^*, κ^*), which means that the optimization would be degenerated, and the learned metric might easily overfit weakly labeled data. This phenomenon is owing to the single-point prior of the expected data moment from the empirical data moment. The regularization term $-\gamma\kappa^2/2$ reflects the Gaussian prior in the generalized maximum entropy principle (Dudík and Schapire, 2006), while the ordinary maximum entropy principle (Jaynes, 1957; Berger et al., 1996) assumes the single-point prior and applies no regularization on the dual variable.

The potential function underlies the generalized maximum entropy distribution estimation. By the potential function and the slack variable, we could obtain the same dual problem. Let the potential function $U_f(\cdot)$ and its target value u_f be

$$\begin{aligned} U_f(x) &= \frac{1}{2\gamma}(x - u_f)^2, \\ u_f &= \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} f(x_i, x_j, y_{i,j}). \end{aligned}$$

Redefine optimization (2) as an equivalent form

$$\max_{A, p^A_{i,j}} \ \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} H(p^A_{i,j}) - U_f\left(\sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \mathbb{E}_{p^A_{i,j}}[f(x_i, x_j, y)]\right),$$

where the equivalence is due to Fenchel's Duality Theorem of Dudík and Schapire (2006) plus the fact that the conjugate of $U_f(x)$ is $U_f^*(\kappa) = \gamma\kappa^2/2$. Subsequently,

$$\max_{A,\kappa} \ \mathcal{L}_2(A,\kappa) = \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \ln p^A_{i,j}(y_{i,j}) - U_f^*(-\kappa) - \mu U_g\left(\sum\nolimits_{\mathcal{U}} \mathbb{E}_{p^A_{i,j}}[g(x_i, x_j, y)]\right)$$

is an optimization problem with two potential functions $U_f(\cdot)$ and $U_g(\cdot)$ under the posterior regularization framework (Graça et al., 2008, 2009; Bellare et al., 2009; Gillenwater et al., 2011), and hence SERAPH can be viewed as a semi-supervised maximum entropy estimation equipped with the additional projection sparsity.

4.3 Information maximization principle

The final model defined in optimization (6) can also be viewed as an information maximization approach to semi-supervised metric learning based on weak labels. The regularized information maximization framework (Gomes et al., 2010) advocates the preference for maximizing the mutual information between data and labels as well as the necessity for regularizing the model parameters.

Let $p(y)$ be the prior distribution

$$p(y) = \iint_{\mathbb{R}^m \times \mathbb{R}^m} p^A(y \mid x, x')p(x)p(x')\mathrm{d}x\mathrm{d}x',$$

and $\hat{p}(y)$ be its estimate

$$\hat{p}(y) = \frac{1}{\#\mathcal{U}} \sum\nolimits_{\mathcal{U}} p^A_{i,j}(y).$$

Let $I(y; x, x')$ be the mutual information between the data pair and the weak label

$$I(y; x, x') = \iint_{\mathbb{R}^m \times \mathbb{R}^m} \sum\nolimits_y p^A(y \mid x, x')p(x)p(x') \ln\left(\frac{p^A(y \mid x, x')}{p(y)}\right) \mathrm{d}x\mathrm{d}x',$$

and $I(y;\mathcal{U})$ be its estimate, that is, the mutual information between unlabeled data and unobserved weak labels

$$I(y;\mathcal{U}) = \frac{1}{\#\mathcal{U}} \sum\nolimits_{\mathcal{U}} \sum\nolimits_y p^A_{i,j}(y) \ln\left(\frac{p^A_{i,j}(y)}{\hat{p}(y)}\right).$$

Given the supervised part of SERAPH, regularized information maximization would suggest

$$\max_{A,\kappa} \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \ln p^A_{i,j}(y_{i,j}) + \mu' I(y;\mathcal{U}) - \frac{\gamma}{2}\kappa^2 - \lambda \operatorname{tr}(A),$$

where we assume the regularization parameter μ' satisfies $\mu' = \#\mathcal{U}\mu$. Then by decomposing $I(y;\mathcal{U})$, it could be rewritten as

$$\max_{A,\kappa} \mathcal{L}(A, \kappa) + \mu' H(\hat{p}(y)).$$

The entropy term encourages a balanced prior distribution of y under the metric $d(x, x')$. However, the number of similar and dissimilar data pairs (i.e., $y = +1$ and $y = -1$) are inherently imbalanced in all metric learning problem settings. Therefore, we simply drop the regularization term $\mu' H(\hat{p}(y))$ and attain optimization (6).

Notice that this explanation elicits a nice heuristic value of the regularization parameter

$$\mu = \frac{\#(\mathcal{S} \cup \mathcal{D})}{\#\mathcal{U}}.$$

In fact, let $H(y \mid x, x')$ be the conditional entropy of the weak label on the data pair

$$H(y \mid x, x') = \iint_{\mathbb{R}^m \times \mathbb{R}^m} H(p^A(y \mid x, x'))\mathrm{d}x\mathrm{d}x',$$

then $H(y \mid x, x')$ can be estimated by

$$H(y \mid \mathcal{S} \cup \mathcal{D}) = \frac{1}{\#(\mathcal{S} \cup \mathcal{D})} \sum\nolimits_{\mathcal{S} \cup \mathcal{D}} H(p_{i,j}^A).$$

As a result, the conditional entropy $H(y \mid \mathcal{S} \cup \mathcal{D})$ as the supervised part and the mutual information $I(y; \mathcal{U})$ as the unsupervised part become equally important in $\mathcal{L}_2(A, \kappa)$ and $\mathcal{L}(A, \kappa)$ if setting $\mu = \#(\mathcal{S} \cup \mathcal{D})/\#\mathcal{U}$.

5 Related Works

Xing et al. (2003) initiated the research of metric learning based on pairwise similarity and dissimilarity constraints by global distance metric learning (GDM). Inspired by miscellaneous motivations, several excellent metric learning methods have been developed in the last decade, such as neighborhood component analysis (NCA) (Goldberger et al., 2005), large margin nearest neighbor classification (LMNN) (Weinberger et al., 2006), information-theoretic metric learning (ITML) (Davis et al., 2007), and so on.

Both ITML and SERAPH are information-theoretic, but the ideas and models are quite different. ITML defines a generative Gaussian model

$$p^A(x) = \frac{1}{Z} \exp\left(-\frac{1}{2}\|x - \mu\|_A^2\right),$$

where μ is the unknown mean value, Z is a normalizing constant, and both of them can be canceled out in the constrained optimization. Compared with GDM, ITML regularizes the Kullback-Leibler divergence between $p^{A_0}(x)$ and $p^A(x)$ where A_0 is the prior metric, and then transforms this term to a log-det regularization. By specifying $A_0 = \frac{1}{n} I_m$, it becomes the maximum entropy estimation of $p^A(x)$. Thus, it prefers the distance metric close to the Euclidean distance. On the other hand, the supervised part of SERAPH also follows the maximum entropy principle, but the probabilistic model $p^A(y \mid x, x')$ is discriminative.

A probabilistic GDM was designed intuitively as a baseline method in the experimental part of Yang et al. (2006). It can be viewed as a special case of our supervised part, but the final model of SERAPH is much more general. Please refer to Sections 2.2, 7.1 and 7.2 for details.

Due to the limitation of supervised metric learning when few labeled data are available, semi-supervised models and algorithms that incorporate off-the-shelf unsupervised techniques to existing supervised approaches have been proposed in recent years. Local distance metric learning LDM (Yang et al., 2006) is the pioneer. Unlike later manifold-based methods, it embeds the unsupervised information by assuming that the eigenvectors of the optimal A are the principal components of all training data. Hoi et al. (2008) borrows the idea of Laplacian eigenmaps (Belkin and Niyogi, 2002) and combines manifold regularization to the min-max principle of GDM. Baghshah and Shouraki (2009) then shows that Fisher discriminant analysis can be regularized by locally linear embedding (Roweis and Saul, 2000), and the resulting manifold Fisher discriminant analysis (MFDA)

is extremely computationally efficient. Liu et al. (2010) brings the element-wise matrix sparsity of A to Hoi et al. (2008). In general, any unsupervised embedding method that preserves the local neighborhood information can be modified into a semi-supervised extension. Check *DistLearnKit*[7] for a partial list of such methods.

The manifold extension which will be described in Section 7.2 is so general that it can be attached to all metric learning methods, whereas our information-theoretic extension can only be applied to probabilistic metric learning methods. Nevertheless, any probabilistic method with an explicit expression of the posterior distribution such as NCA, LDM and SERAPH can have two semi-supervised extensions, while deterministic methods like GDM, LMNN and MFDA cannot benefit from our semi-supervised extension. ITML utilizes a generative Gaussian model whose parameters are not estimated by the algorithm, so it is non-trivial to apply our extension to it.

Here we leave out sparse metric learning and robust metric learning, and instead, we recommend Huang et al. (2009, pp. 8–9) and Huang et al. (2010, p. 2) for the reviews of sparse and robust metric learning. Moreover, a comprehensive literature survey on metric learning, Bellet et al. (2013), becomes available online now and can be a good reference.

6 Experiments

In this section, we numerically evaluate the performance of metric learning methods.

6.1 Setup

In our experiments, we compared the proposed SERAPH with six representative metric learning methods (plus the Euclidean distance):

- Global distance metric learning (GDM; Xing et al., 2003)[8];
- Neighborhood component analysis (NCA; Goldberger et al., 2005)[9];
- Large margin nearest neighbor classification (LMNN; Weinberger et al., 2006)[10];
- Information-theoretic metric learning (ITML; Davis et al., 2007)[11];
- Local distance metric learning (LDM; Yang et al., 2006)[12];
- Manifold Fisher discriminant analysis (MFDA; Baghshah and Shouraki, 2009)[13].

GDM, NCA, LMNN and ITML are supervised methods, while LDM and MFDA are semi-supervised methods. SERAPH as well as GDM, ITML and LDM utilize the global metric information; and NCA, LMNN and MFDA benefit from the local metric information.

[7] A Matlab toolkit for distance metric learning: http://www.cs.cmu.edu/~liuy/distlearn.htm.
[8] The code was from http://www.cs.cmu.edu/~epxing/papers/Old_papers/code_Metric_online.tar.gz.
[9] The code was from http://www.cs.berkeley.edu/~fowlkes/software/nca/nca_demo.tar.gz.
[10] The code was from http://www.cse.wustl.edu/~kilian/code/files/mLMNN.zip.
[11] The code was from http://www.cs.utexas.edu/~pjain/itml/download/itml-1.2.tar.gz.
[12] The code was from http://www.cs.cmu.edu/~liuy/ldm_scripts_2.zip.
[13] We wrote its code based on locally linear embedding (http://www.cs.nyu.edu/~roweis/lle/code.html).

Table 1: Specification of benchmark data sets. For each data set, c means the number of classes, m means the number of features, $n_{\text{train}}/n_{\text{test}}$ means the number of training/test data points, and n_{label} means the number of class labels to construct $\mathcal{S}$ and $\mathcal{D}$.

	c	m	n_{train}	n_{test}	n_{label}	$\mathbb{E}\#\mathcal{S}$	$\mathbb{E}\#\mathcal{D}$	$\#\mathcal{U}$
iris	3	4	100	38	10	15.10	29.90	4905
wine	3	13	100	78	10	13.98	31.02	4905
ionosphere	2	34	100	251	20	97.50	92.50	4760
balance	3	4	100	465	10	20.38	24.62	4905
breast cancer	2	30	100	469	10	23.54	21.46	4905
diabetes	2	8	100	668	10	23.02	21.98	4905
	c	m	n_{train}	n_{test}	n_{label}	$\#\mathcal{S}$	$\#\mathcal{D}$	$\#\mathcal{U}$
$\text{USPS}_{1-5,20}$	5	64	100	2500	10	5	40	4905
$\text{USPS}_{1-5,40}$	5	64	200	2500	20	30	160	19710
$\text{USPS}_{1-10,20}$	10	64	200	2500	20	10	180	19710
$\text{USPS}_{1-10,40}$	10	64	400	2500	40	60	720	79020
$\text{MNIST}_{1,7}$	2	196	100	1000	4	2	4	4944
$\text{MNIST}_{3,5,8}$	3	196	150	1500	9	9	27	11139

Table 1 describes the specification of benchmark data sets in our experiments. The top six data sets (i.e., iris, wine, ionosphere, balance, breast cancer, and diabetes) come from the *UCI machine learning repository*[14], and *USPS* and *MNIST* are available at the homepage of the late Sam Roweis[15]. The gray-scale images of handwritten digits in USPS were downsampled to 8×8 pixel resolution resulting in 64-dimensional vectors. Similarly, the gray-scale images in MNIST were downsampled to 14×14 pixel resolution resulting in 196-dimensional vectors. The symbol $\text{USPS}_{1-5,20}$ means 20 training data from each of the first 5 classes, $\text{USPS}_{1-10,40}$ means 40 training data from each of all 10 classes, $\text{MNIST}_{1,7}$ means digits 1 versus 7, and so forth. Note that in the last two tasks, the dimensionality of data is greater than the number of all training data: The number of parameters to be learned in A is $m(m+1)/2 = 19306$, whereas the number of training data points n_{train} is 100 or 150 and then the number of training data pairs $n_{\text{train}}(n_{\text{train}}-1)/2$ is only 4950 or 11175.

All metric learning methods were repeatedly run on 50 random samplings of a given task. For each random sampling, we constructed $\mathcal{S}$ and $\mathcal{D}$, which include the similar and dissimilar data pairs for training, according to the class labels of the first few data points for training: Let y_i and y_j be the class labels of x_i and x_j, then

- $(x_i, x_j) \in \mathcal{S}$ and $y_{i,j} = +1$ if $y_i = y_j$;
- $(x_i, x_j) \in \mathcal{D}$ and $y_{i,j} = -1$ if $y_i \neq y_j$.

The sizes of $\mathcal{S}$ and $\mathcal{D}$ were dependent on the specific random sampling of each UCI task, but fixed for all samplings of each USPS and MNIST task. We measured the performance

[14]The data sets are available at http://archive.ics.uci.edu/ml/.

[15]The data sets are available at http://cs.nyu.edu/~roweis/data.html.

of the one-nearest-neighbor classifiers based on the learned metrics and the computation time for learning the metrics, where the "training data" for our classifiers included only the few data points having class labels.

For SERAPH, we fixed $\eta = 1$ for simplicity. Then, four hyperparameter settings were considered:

- $\text{SERAPH}_{\text{none}}$ stands for $\mu = 0$ and $\lambda = 0$;
- $\text{SERAPH}_{\text{post}}$ stands for $\mu = \frac{\#(\mathcal{S} \cup \mathcal{D})}{\#\mathcal{U}}$ and $\lambda = 0$;
- $\text{SERAPH}_{\text{proj}}$ stands for $\mu = 0$ and $\lambda = 1$;
- $\text{SERAPH}_{\text{hyper}}$ stands for $\mu = \frac{\#(\mathcal{S} \cup \mathcal{D})}{\#\mathcal{U}}$ and $\lambda = 1$.

There was no cross-validation for each random sampling, because we would like the metrics learned by different methods to be independent of those nearest-neighbor classifiers whose performance had a large deviation given limited supervised information.[16] The hyperparameters of other methods, e.g., the number of reduced dimensions, the number of nearest neighbors, as well as the percentage of principal components, were selected as the best candidate value based on another 10 random samplings, if no default or heuristic value was provided by the original authors of the codes.

6.2 Results

Artificial data sets Figures 2 and 3 had already displayed the visually comprehensive results of the posterior and projection sparsity regularization on two artificial data sets respectively. More specifically,

- In both figures, subfigures (c) and (d) were generated with $\mu = \lambda = 0$;
- In Figure 2, (e) and (f) were generated with $\mu = 100 \cdot \frac{\#(\mathcal{S} \cup \mathcal{D})}{\#\mathcal{U}}$ and $\lambda = 0$;
- In Figure 3, (e) and (f) were generated with $\mu = 0$ and $\lambda = 100$,

where the gradient projection algorithm was used. We can see from Figures 2 and 3 that the sparsity regularization can dramatically improve the generalized maximum entropy estimation.

Gradient projection algorithm vs. EM-like algorithm Before comparing the proposed SERAPH with other metric learning methods, we evaluated the gradient projection algorithm (GP) and the EM-like iterative algorithm (EM). Table 2 shows their performance where the hyperparameter setting $\text{SERAPH}_{\text{hyper}}$ was used. By the paired t-test at the significance level 5%, GP and EM both won 2 times and tied 8 times, and therefore GP and EM are basically comparable as two different solvers to the same optimization problem. However, EM was computationally more efficient than GP in our experiments consistently and the difference of their average computation time was remarkable, which suggests EM as an indirect solver could be a good alternative to the direct solver GP.

[16] Even if we allow the learned metrics to be dependent on the following classifiers, it is nontrivial to crossly validate the hyperparameters given limited supervised information.

Table 2: Gradient projection algorithm (GP) vs. EM-like algorithm (EM). Means with standard errors of the nearest-neighbor misclassification rate (in %) are shown, together with results of the paired t-test at the significance level 5%. The computation time ratio means the average computation time of GP over that of EM.

	GP	EM	paired t-test	computation time ratio
iris	5.58 ± 0.57	6.11 ± 0.66	Tie	3.00
wine	7.87 ± 0.62	7.46 ± 0.51	Tie	2.94
ionosphere	19.65 ± 0.50	19.53 ± 0.44	Tie	2.04
balance	21.55 ± 0.75	20.59 ± 0.64	Tie	1.94
breast cancer	9.51 ± 0.49	9.97 ± 0.49	Tie	1.43
diabetes	29.93 ± 0.65	30.07 ± 0.61	Tie	1.67
$\text{USPS}_{1-5,20}$	31.46 ± 0.79	32.82 ± 0.77	GP win	2.58
$\text{USPS}_{1-5,40}$	25.23 ± 0.58	25.30 ± 0.56	Tie	2.77
$\text{USPS}_{1-10,20}$	45.45 ± 0.60	44.93 ± 0.58	EM win	2.81
$\text{USPS}_{1-10,40}$	34.14 ± 0.48	33.45 ± 0.47	EM win	2.43
$\text{MNIST}_{1,7}$	4.33 ± 0.25	8.15 ± 0.59	GP win	1.23
$\text{MNIST}_{3,5,8}$	35.46 ± 0.84	35.77 ± 0.83	Tie	1.99

Benchmark data sets The experimental results in terms of the nearest-neighbor misclassification rate are reported in Table 3, where the EM-like algorithm was used. GDM was very slow for high-dimensional data and excluded from the comparison. SERAPH was fairly promising, especially the hyper-sparsity setting (i.e., $\mu = \frac{\#(\mathcal{S} \cup \mathcal{D})}{\#\mathcal{U}}$ and $\lambda = 1$). It was best or tie over all 12 tasks. It often statistically significantly outperformed other methods except ITML on six UCI data sets, and it was superior to all other competitors including $\text{SERAPH}_{\text{post}}$ and $\text{SERAPH}_{\text{proj}}$ in 4 USPS tasks. Furthermore, it successfully improved the accuracy even in two ill-posed MNIST tasks. To sum up, SERAPH can reduce the risk of overfitting weakly labeled data with the help of unlabeled data, and hence our sparsity regularization would be reasonable and practical.

In vivid contrast with SERAPH that exhibited the nice generalization capability, supervised methods might learn a metric even worse than the Euclidean distance due to overfitting problems, especially NCA that optimized the expected leave-one-out classification error on a limited amount of labeled data. The powerful LMNN did not behave satisfyingly, since it was hardly fulfilled to find a lot of neighbors belonging to the same class within labeled data. ITML worked very well despite that it can only access weakly labeled data, but it became less useful for high-dimensional data. On the other hand, we observed that LDM might fail when the principal components of all training data were not close to the eigenvectors of the optimal matrix being learned, and MFDA might fail if the amount of training data cannot afford to recover the data manifold.

An observation is that the methods using the global metric information usually outperformed the methods using the local metric information in our experiments since the supervised information was insufficient, which is opposite to the phenomena observed in supervised metric learning problem settings. It indicates that the methods using the local

Table 3: Means with standard errors of the nearest-neighbor misclassification rate (in %) on UCI, USPS and MNIST benchmarks. For each data set, the best method and comparable ones based on the unpaired t-test at the significance level 5% are highlighted in boldface.

	iris	wine	ionosphere	balance	breast cancer	diabetes
EUCLIDEAN	9.58 ± 0.73	12.93 ± 0.83	23.60 ± 0.89	27.15 ± 0.75	14.11 ± 1.07	32.94 ± 0.65
GDM	8.95 ± 0.71	11.52 ± 0.77	$\mathbf{20.82 \pm 0.82}$	22.89 ± 1.08	11.86 ± 0.83	$\mathbf{30.73 \pm 0.59}$
NCA	10.32 ± 0.83	15.03 ± 1.12	26.68 ± 0.82	32.97 ± 1.31	14.63 ± 1.09	32.95 ± 0.65
LMNN	9.81 ± 0.79	14.83 ± 0.97	22.25 ± 0.75	24.00 ± 1.34	13.86 ± 0.84	32.02 ± 0.60
ITML	$\mathbf{5.57 \pm 0.53}$	$\mathbf{8.22 \pm 0.66}$	$\mathbf{20.35 \pm 0.64}$	$\mathbf{22.04 \pm 0.80}$	$\mathbf{9.60 \pm 0.49}$	$\mathbf{31.21 \pm 0.73}$
LDM	7.27 ± 0.72	17.21 ± 1.41	24.54 ± 0.92	$\mathbf{21.22 \pm 0.93}$	14.85 ± 0.92	34.33 ± 0.60
MFDA	6.58 ± 0.54	11.55 ± 1.03	23.66 ± 0.91	23.61 ± 1.00	11.21 ± 0.80	31.64 ± 0.62
SERAPH$_{\text{none}}$	$\mathbf{5.42 \pm 0.52}$	9.31 ± 0.63	$\mathbf{19.45 \pm 0.46}$	$\mathbf{21.04 \pm 0.71}$	11.61 ± 0.48	$\mathbf{29.76 \pm 0.61}$
SERAPH$_{\text{post}}$	$\mathbf{5.21 \pm 0.43}$	$\mathbf{7.97 \pm 0.60}$	$\mathbf{19.82 \pm 0.46}$	$\mathbf{20.80 \pm 0.75}$	11.41 ± 0.48	$\mathbf{29.85 \pm 0.57}$
SERAPH$_{\text{proj}}$	$\mathbf{5.63 \pm 0.53}$	$\mathbf{8.28 \pm 0.63}$	$\mathbf{19.42 \pm 0.49}$	$\mathbf{20.78 \pm 0.63}$	$\mathbf{9.54 \pm 0.45}$	$\mathbf{30.51 \pm 0.72}$
SERAPH$_{\text{hyper}}$	$\mathbf{6.11 \pm 0.66}$	$\mathbf{7.46 \pm 0.51}$	$\mathbf{19.53 \pm 0.44}$	$\mathbf{20.59 \pm 0.64}$	$\mathbf{9.97 \pm 0.49}$	$\mathbf{30.07 \pm 0.61}$
	USPS$_{1-5,20}$	USPS$_{1-5,40}$	USPS$_{1-10,20}$	USPS$_{1-10,40}$	MNIST$_{1,7}$	MNIST$_{3,5,8}$
EUCLIDEAN	36.63 ± 0.80	28.43 ± 0.60	49.17 ± 0.50	39.30 ± 0.39	10.42 ± 0.67	$\mathbf{37.30 \pm 0.81}$
GDM	37.62 ± 0.77	-	-	-	-	-
NCA	37.55 ± 0.84	28.39 ± 0.60	57.01 ± 0.82	49.21 ± 0.66	10.42 ± 0.67	$\mathbf{37.75 \pm 0.92}$
LMNN	36.43 ± 0.78	28.93 ± 0.61	48.12 ± 0.57	43.68 ± 0.58	9.99 ± 0.71	$\mathbf{36.49 \pm 0.82}$
ITML	35.86 ± 0.74	27.40 ± 0.65	47.40 ± 0.60	39.44 ± 0.57	9.94 ± 0.69	40.83 ± 0.93
LDM	47.19 ± 1.51	32.52 ± 0.85	59.13 ± 0.73	43.18 ± 0.53	14.54 ± 1.41	45.53 ± 1.16
MFDA	42.52 ± 0.82	28.82 ± 0.62	52.13 ± 0.59	37.78 ± 0.50	$\mathbf{9.35 \pm 0.72}$	42.39 ± 0.92
SERAPH$_{\text{none}}$	36.23 ± 0.76	28.15 ± 0.63	47.55 ± 0.58	38.41 ± 0.55	9.99 ± 0.71	$\mathbf{36.54 \pm 0.84}$
SERAPH$_{\text{post}}$	35.83 ± 0.76	27.41 ± 0.60	47.17 ± 0.58	37.31 ± 0.53	10.99 ± 0.79	$\mathbf{36.48 \pm 0.84}$
SERAPH$_{\text{proj}}$	36.62 ± 0.77	27.10 ± 0.63	48.16 ± 0.63	36.49 ± 0.55	$\mathbf{9.54 \pm 0.66}$	$\mathbf{36.21 \pm 0.83}$
SERAPH$_{\text{hyper}}$	$\mathbf{32.82 \pm 0.77}$	$\mathbf{25.30 \pm 0.56}$	$\mathbf{44.93 \pm 0.58}$	$\mathbf{33.45 \pm 0.47}$	$\mathbf{8.15 \pm 0.59}$	$\mathbf{35.77 \pm 0.84}$

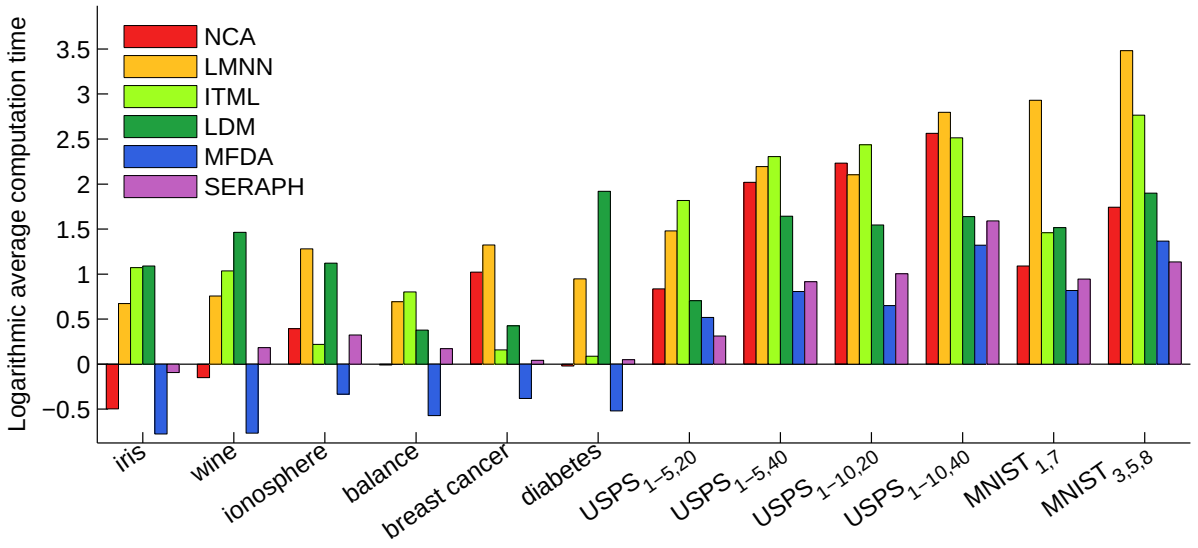


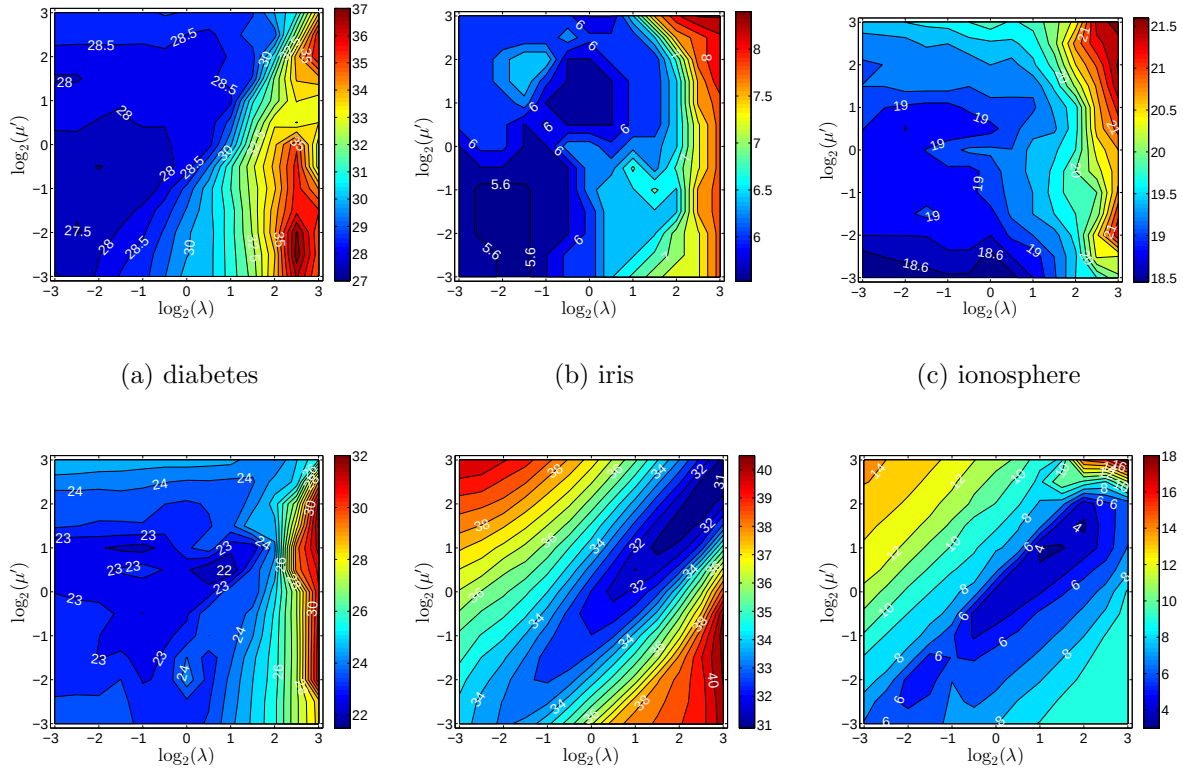
Figure 4: Average computation time of different metric learning methods on UCI, USPS and MNIST benchmarks. The computation time was measured in seconds and drawn in a logarithmic scale with 10 as the base.

metric information tends to fit the given information too much and suffers from overfitting problems, since the local metric information always focuses on a small amount of data in a local neighborhood and thus has a relatively large deviation.

Computational efficiency Figure 4 summarizes the corresponding experimental results in terms of the average computation time (GDM was excluded from the comparison due to its low speed). The computation time was measured in seconds and drawn in a logarithmic scale with 10 as the base. The shortest average computation time was 0.1677 second of MFDA for iris, and the longest time was 3023 seconds of LMNN for $\text{MNIST}_{3,5,8}$. Generally speaking, SERAPH (when the EM-like algorithm was used) was the second most computationally-efficient method, and the most computationally-efficient method MFDA just consists of two steps: Solve a linear system of locally linear embedding (Roweis and Saul, 2000) and then solve a generalized eigenvalue problem as Fisher discriminant analysis (Fisher, 1936). Improvements may be expected if we program in Matlab with C/C++ as NCA and LMNN.

Sensitivity to regularization parameters Recall that there was no cross-validation within each random sampling, so it would be helpful to test the sensitivity of SERAPH to the regularization parameters μ and λ. Six benchmark data sets were included:

- diabetes, iris, and ionosphere, on which $\text{SERAPH}_{\text{none}}$, $\text{SERAPH}_{\text{post}}$, and $\text{SERAPH}_{\text{proj}}$ had lowest means of the nearest-neighbor misclassification rate in Table 3, respectively;



(a) diabetes (b) iris (c) ionosphere

(d) balance (e) $\text{USPS}_{1-5,20}$ (f) $\text{MNIST}_{1,7}$

Figure 5: Contours of the mean misclassification rates (in %) of the one-nearest-neighbor classifiers based on the metrics learned by SERAPH given different regularization parameters μ' and λ. The actual regularization parameter μ being used was $\mu = \mu' \cdot \frac{\#(\mathcal{S} \cup \mathcal{D})}{\#\mathcal{U}}$.

- balance, $\text{USPS}_{1-5,20}$, and $\text{MNIST}_{1,7}$, on which $\text{SERAPH}_{\text{hyper}}$ had lowest means of the nearest-neighbor misclassification rate in Table 3.

We considered geometrically progressed candidates for both μ' and λ ranging from 2^{-3} to 2^{+3} with $2^{0.5}$ as the factor, namely,

$$\mu', \lambda \in \left\{ \frac{1}{8}, \frac{\sqrt{2}}{8}, \frac{1}{4}, \frac{\sqrt{2}}{4}, \frac{1}{2}, \frac{\sqrt{2}}{2}, 1, \sqrt{2}, 2, 2\sqrt{2}, 4, 4\sqrt{2}, 8 \right\},$$

and the actual regularization parameter μ being used was $\mu = \mu' \cdot \frac{\#(\mathcal{S} \cup \mathcal{D})}{\#\mathcal{U}}$. The gradient projection algorithm was repeatedly run on 10 random samplings, which were the first 10 random samplings of those 50 random samplings, given all combinations of μ' and λ, and the resulting contours are displayed in Figure 5.

We can see that SERAPH worked well in large areas of the contour plots in Figure 5, and we can clearly observe two phenomena.

Firstly, SERAPH was more sensitive to λ for low-dimensional tasks diabetes, iris, ionosphere and balance, and the learned metrics became worse when λ became large. Even for $\text{USPS}_{1-5,20}$, the learned metrics also became worse when λ became large for small μ'. Note that large λ implies the strong regularization on the trace norm of A, which upper bounds the rank of A, and the rank of A ultimately controls the number of parameters to be learned in A. This explains why the contours of $\text{MNIST}_{1,7}$ were different from others as there were so many parameters in A that large λ did not make SERAPH significantly over-regularized.

Secondly, SERAPH was also sensitive to μ' for high-dimensional tasks $\text{USPS}_{1-5,20}$ and $\text{MNIST}_{1,7}$, while this time when μ' became large, the learned metrics became worse for small λ but better for large λ. In other words, SERAPH easily got over-regularized by emphasizing unlabeled data too much if the number of parameters to be learned in A was improperly large, whereas it hardly got over-regularized if the number of parameters was properly small. Additionally, for $\text{USPS}_{1-5,20}$ and $\text{MNIST}_{1,7}$ neither the posterior sparsity nor the projection sparsity worked alone, but they became very powerful after integrated into the hyper-sparsity. A final caveat is that the hyper-sparsity could never be a panacea for such high-dimensional tasks and we should not employ it too much: The contours of $\text{MNIST}_{1,7}$ have exhibited a typical effect that the learned metrics became worse suddenly and rapidly along the line $\log_2(\mu') = \log_2(\lambda)$ when $\mu' = \lambda$ became very large.

7 Extensions

In this section, we explain the kernel and manifold extensions of SERAPH. The technique for kernelizing a metric learning method was originally proposed in Jain et al. (2010), and the technique for manifold regularizing a metric learning method was originally proposed in Hoi et al. (2008).

7.1 Kernel extension

Suppose that we have a *kernel function* $k : \mathbb{R}^m \times \mathbb{R}^m \mapsto \mathbb{R}$ with the *feature map* $\phi : \mathbb{R}^m \mapsto \mathbb{R}^{\widetilde{m}}$ such that $k(x, x') = \phi(x)^\top \phi(x')$. Consider learning a Mahalanobis distance metric for $x, x' \in \mathbb{R}^m$ of the form

$$d(x, x') = \|\phi(x) - \phi(x')\|_W = \sqrt{(\phi(x) - \phi(x'))^\top W (\phi(x) - \phi(x'))}, \tag{20}$$

where $W \in \mathbb{R}^{\widetilde{m} \times \widetilde{m}}$ is a symmetric positive semi-definite matrix to be learned. However, it is impractical or impossible to learn W directly, since $\widetilde{m}$ is often very large and possibly infinite. In order to learn W indirectly, we rewrite optimization (7) with respect to W:

$$\begin{aligned} \min_W \quad & \lambda \operatorname{tr}(W) + \sum\nolimits_{\mathcal{S} \cup \mathcal{D}} \xi_{i,j} + \mu \sum\nolimits_{\mathcal{U}} \xi_{i,j} \\ \text{s.t.} \quad & \ln p^W_{i,j}(y_{i,j}) \geq -\xi_{i,j}, \forall (x_i, x_j) \in \mathcal{S} \cup \mathcal{D} \\ & \sum\nolimits_y p^W_{i,j}(y) \ln p^W_{i,j}(y) \geq -\xi_{i,j}, \forall (x_i, x_j) \in \mathcal{U}, \end{aligned}$$

where $p^W(y \mid x, x')$ is similar to Eq. (8):

$$p^W(y \mid x, x') = \frac{1}{1 + \exp(y(\|\phi(x) - \phi(x')\|_W^2 - \eta))}.$$

Subsequently, according to Jain et al. (2010), any optimal solution W^* will be in the form of $W^* = \Phi^\top A^* \Phi$, where $\Phi = (\phi(x_1), \ldots, \phi(x_n))^\top \in \mathbb{R}^{n \times \widetilde{m}}$ is the design matrix obtained by applying ϕ to the training set $\mathcal{X}$, and $A^* \in \mathbb{R}^{n \times n}$ is an optimal solution to

$$\begin{aligned} \min_A \ & \lambda \operatorname{tr}(A) + \sum\nolimits_{\mathcal{S} \cup \mathcal{D}} \xi_{i,j} + \mu \sum\nolimits_{\mathcal{U}} \xi_{i,j} \\ \text{s.t.} \ & \ln p_{i,j}^A(y_{i,j}) \geq -\xi_{i,j}, \forall (x_i, x_j) \in \mathcal{S} \cup \mathcal{D} \\ & \sum\nolimits_y p_{i,j}^A(y) \ln p_{i,j}^A(y) \geq -\xi_{i,j}, \forall (x_i, x_j) \in \mathcal{U}, \end{aligned}$$

which is actually optimization (7) with respect to A but $p^A(y \mid x, x')$ in Eq. (8) is replaced with

$$p^A(y \mid x, x') = \frac{1}{1 + \exp(y((\phi(x) - \phi(x'))^\top \Phi^\top A \Phi(\phi(x) - \phi(x')) - \eta))}. \tag{21}$$

Next let us simplify our notations to remove the feature map ϕ from our equations. We introduce the *empirical kernel map* (Schölkopf and Smola, 2001, p. 42) defined by

$$\psi(x) = \Phi\phi(x) = (k(x_1, x), \ldots, k(x_n, x))^\top,$$

and then Eqs. (20) and (21) can be expressed by

$$\begin{aligned} d(x, x') &= \|\phi(x) - \phi(x')\|_W = \|\psi(x) - \psi(x')\|_A \\ &= \sqrt{(\psi(x) - \psi(x'))^\top A (\psi(x) - \psi(x'))}, \\ p^A(y \mid x, x') &= \frac{1}{1 + \exp(y(\|\psi(x) - \psi(x')\|_A^2 - \eta))}. \end{aligned}$$

Moreover, let $K \in \mathbb{R}^{n \times n}$ be the *kernel matrix* and $k_1, \ldots, k_n$ be the columns of K, then for any $x_i, x_j \in \mathcal{X}$,

$$\begin{aligned} d(x_i, x_j) &= \|\psi(x_i) - \psi(x_j)\|_A = \|k_i - k_j\|_A, \\ p_{i,j}^A(y) &= \frac{1}{1 + \exp(y(\|k_i - k_j\|_A^2 - \eta))}. \end{aligned}$$

All components of SERAPH remain the same after replacing x_i with the corresponding k_i. The resultant Mahalanobis distance metric $d(x, x')$ will be highly non-linear with respect to the original input data domain.

The experimental results based on the kernel extension are reported in Table 4, where the EM-like algorithm was used, and for convenience the best hyperparameter setting in Table 3 is also listed in Table 4. The four hyperparameter settings were same as before, but here they were kernelized and with a symbol "ker" in front of them. More specifically,

three kernels were involved: The linear kernel was used for iris; the Gaussian kernel was used for other UCI data sets; the sparse variant of the cosine kernel was used for USPS and MNIST. The linear kernel is

$$k(x, x') = x^\top x'.$$

The Gaussian kernel is

$$k(x, x') = \exp\left(-\frac{\|x - x'\|_2^2}{2\sigma^2}\right)$$

with a hyperparameter σ, and σ was set to the median pairwise distance, i.e., the median value of the Euclidean distances between all training data pairs. Note that we just need compute the empirical kernel map ψ in which the first argument of the kernel k must be from $\mathcal{X}$, and hence the sparse variant of the cosine kernel is

$$k(x_i, x) = \begin{cases} \dfrac{x_i^\top x}{\|x_i\|_2 \|x\|_2} & \text{if } x_i \sim_{\widetilde{k}} x, \\ 0 & \text{otherwise,} \end{cases}$$

with a hyperparameter $\widetilde{k}$, where $x_i \sim_{\widetilde{k}} x$ means that x_i is one of the $\widetilde{k}$ nearest neighbors of x in $\mathcal{X}$, and $\widetilde{k}$ was set to 11 so that 10 nearest neighbors were found for $x_i \in \mathcal{X}$ except itself. We can see from Table 4 that SERAPH still performed well and even better after applying the kernel extension. Among all 12 tasks on UCI, USPS and MNIST data sets, the records were improved by the kernel extension in 7 tasks, and the improvement was all significant under the paired t-test at the significance level 5%. We may roughly compare the experimental results in Table 4 with similar results in Jain et al. (2010)[17] and Wang et al. (2011), while we should be aware that the training data for SERAPH as well as for the following nearest-neighbor classifiers were much less than theirs, and a single kernel was also much weaker than multiple kernels.

7.2 Manifold extension

Without loss of generality, we adopt the kernel matrix K used in the kernel extension as our adjacency matrix of the underlying similarity graph for manifold regularization. Let $d_i = \sum_{j=1}^n K_{i,j}$ be the degree of x_i, and $D = \mathrm{diag}(d_1, \ldots, d_n)$ be the degree matrix, then the *unnormalized graph Laplacian* is given by $L = D - K$.

Let $P \in \mathbb{R}^{\widetilde{m} \times m}$ be a projection matrix associated with A such that $A = P^\top P$, $X = (x_1, \ldots, x_n)^\top \in \mathbb{R}^{n \times m}$ be the design matrix of $\mathcal{X}$, and $Z = (z_1, \ldots, z_n)^\top \in \mathbb{R}^{n \times \widetilde{m}}$ be the design matrix of the projected data such that $z_i = Px_i$ and $Z = XP^\top$. The manifold assumption suggests that $\forall x, x' \in \mathbb{R}^m$, if x and x' are close, z and z' should also be close, and we should minimize a manifold regularization term defined by

$$\mathcal{M}(A) = \mathrm{tr}(Z^\top L Z) = \mathrm{tr}(X^\top L X A).$$

[17] Though called semi-supervised, the proposed method in Jain et al. (2010) does not involve $\mathcal{U}$.

Table 4: Means with standard errors of the nearest-neighbor misclassification rate (in %) based on the kernel extension. For each data set, the best hyperparameter setting and comparable ones based on the paired t-test at the significance level 5% are in boldface.

	Table 3 best	ker none	ker post	ker proj	ker hyper
iris	$\mathbf{5.21 \pm 0.43}$	8.74 ± 0.88	8.47 ± 0.90	8.89 ± 0.86	8.63 ± 0.89
wine	7.46 ± 0.51	7.54 ± 0.58	$\mathbf{6.41 \pm 0.57}$	$\mathbf{6.82 \pm 0.69}$	$\mathbf{6.15 \pm 0.54}$
ionosphere	19.42 ± 0.49	14.03 ± 0.97	$\mathbf{13.04 \pm 0.72}$	14.48 ± 1.04	$\mathbf{13.20 \pm 0.71}$
balance	$\mathbf{20.59 \pm 0.64}$	25.09 ± 1.08	24.22 ± 1.07	23.02 ± 0.96	23.54 ± 0.99
breast cancer	$\mathbf{9.54 \pm 0.45}$	11.22 ± 0.74	10.98 ± 0.71	11.22 ± 0.87	10.93 ± 0.67
diabetes	$\mathbf{29.76 \pm 0.61}$	32.43 ± 0.67	32.31 ± 0.67	32.02 ± 0.70	32.20 ± 0.64
$\text{USPS}_{1-5,20}$	32.82 ± 0.77	29.73 ± 0.88	30.04 ± 0.89	29.77 ± 0.93	$\mathbf{28.80 \pm 0.89}$
$\text{USPS}_{1-5,40}$	25.30 ± 0.56	$\mathbf{22.29 \pm 0.54}$	22.47 ± 0.55	$\mathbf{22.16 \pm 0.55}$	$\mathbf{22.14 \pm 0.53}$
$\text{USPS}_{1-10,20}$	44.93 ± 0.58	$\mathbf{44.07 \pm 0.47}$	44.26 ± 0.47	44.39 ± 0.44	$\mathbf{43.83 \pm 0.47}$
$\text{USPS}_{1-10,40}$	33.45 ± 0.47	33.10 ± 0.40	33.30 ± 0.40	$\mathbf{32.81 \pm 0.41}$	$\mathbf{32.66 \pm 0.39}$
$\text{MNIST}_{1,7}$	$\mathbf{8.15 \pm 0.59}$	17.51 ± 1.47	17.54 ± 1.47	17.47 ± 1.47	12.44 ± 1.24
$\text{MNIST}_{3,5,8}$	35.77 ± 0.84	29.51 ± 0.84	29.52 ± 0.84	29.61 ± 0.85	$\mathbf{27.48 \pm 0.84}$

More specifically, the similarity between x_i and x_j is measured by $k(x_i, x_j)$, whereas the dissimilarity of z_i and z_j is measured by the Euclidean distance $\|z_i - z_j\|_2$. The manifold assumption translates into that $\|z_i - z_j\|_2^2$ should be penalized more for larger $k(x_i, x_j)$ than smaller $k(x_i, x_j)$. Consequently, we have

$$\begin{aligned}
\mathcal{M}(A) &= \frac{1}{2} \sum_{i,j=1}^{n} K_{i,j} \|z_i - z_j\|_2^2 \\
&= \sum_{i,j=1}^{n} K_{i,j} (z_i^\top z_i - z_i^\top z_j) \\
&= \sum_{i=1}^{n} d_i z_i^\top z_i - \sum_{i,j=1}^{n} K_{i,j} z_i^\top z_j \\
&= \operatorname{tr}(Z^\top D Z) - \operatorname{tr}(Z^\top K Z) \\
&= \operatorname{tr}(Z^\top L Z) \\
&= \operatorname{tr}(X^\top L X A).
\end{aligned}$$

It would affect neither the concavity nor the Lipschitz continuity if $\mathcal{M}(A)$ is attached to concave or Lipschitz continuous functions, since $\mathcal{M}(A)$ is again linear with respect to A. Therefore, our optimization becomes

$$\max_A \ \widetilde{\mathcal{L}}(A) = \mathcal{L}(A) - \omega \mathcal{M}(A),$$

where $\omega \geq 0$ is a regularization parameter, and at each M-step of the EM-like algorithm, we still solve a convex optimization

$$\max_A \ \widetilde{\mathcal{F}}(A) = \mathcal{F}(A) - \omega \mathcal{M}(A).$$

Table 5: Means with standard errors of the nearest-neighbor misclassification rate (in %) based on the manifold extension. For each data set, the best hyperparameter setting and comparable ones based on the paired t-test at the significance level 5% are in boldface.

	Table 3 best	mani	post+mani	proj+mani	hyper+mani
iris	$\mathbf{5.21 \pm 0.43}$	8.21 ± 0.78	9.58 ± 1.04	8.37 ± 1.35	9.84 ± 0.98
wine	$\mathbf{7.46 \pm 0.51}$	23.54 ± 1.29	29.95 ± 1.42	21.79 ± 1.30	28.77 ± 1.36
ionosphere	$\mathbf{19.42 \pm 0.49}$	20.60 ± 0.68	20.57 ± 0.66	20.61 ± 0.68	20.55 ± 0.70
balance	$\mathbf{20.59 \pm 0.64}$	28.00 ± 1.31	24.28 ± 0.98	29.43 ± 1.50	25.12 ± 1.16
breast cancer	$\mathbf{9.54 \pm 0.45}$	21.79 ± 0.79	25.74 ± 0.78	17.90 ± 0.92	21.64 ± 0.84
diabetes	$\mathbf{29.76 \pm 0.61}$	32.32 ± 0.91	31.28 ± 0.72	33.26 ± 1.02	31.73 ± 0.89
$\text{USPS}_{1-5,20}$	32.82 ± 0.77	42.21 ± 0.77	36.46 ± 0.77	36.68 ± 0.73	$\mathbf{31.46 \pm 0.76}$
$\text{USPS}_{1-5,40}$	25.30 ± 0.56	30.42 ± 0.56	26.09 ± 0.56	26.96 ± 0.58	$\mathbf{24.41 \pm 0.54}$
$\text{USPS}_{1-10,20}$	44.93 ± 0.58	53.42 ± 0.51	46.58 ± 0.55	47.77 ± 0.64	$\mathbf{43.80 \pm 0.57}$
$\text{USPS}_{1-10,40}$	33.45 ± 0.47	38.22 ± 0.52	33.71 ± 0.48	35.58 ± 0.52	$\mathbf{32.60 \pm 0.48}$
$\text{MNIST}_{1,7}$	$\mathbf{8.15 \pm 0.59}$	15.32 ± 1.23	15.30 ± 1.08	9.68 ± 0.70	$\mathbf{7.88 \pm 0.60}$
$\text{MNIST}_{3,5,8}$	$\mathbf{35.77 \pm 0.84}$	43.81 ± 0.80	39.76 ± 0.87	37.42 ± 0.86	$\mathbf{36.28 \pm 0.89}$

The gradient matrices of $\widetilde{\mathcal{L}}(A)$ and $\widetilde{\mathcal{F}}(A)$ are simply

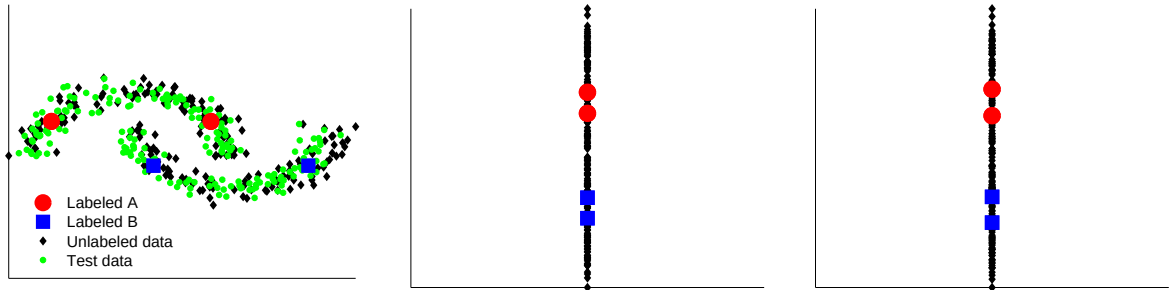
$$\nabla\widetilde{\mathcal{L}}(A) = \nabla\mathcal{L}(A) - \omega\nabla\mathcal{M}(A) = \nabla\mathcal{L}(A) - \omega X^\top LX,$$
$$\nabla\widetilde{\mathcal{F}}(A) = \nabla\mathcal{F}(A) - \omega\nabla\mathcal{M}(A) = \nabla\mathcal{F}(A) - \omega X^\top LX,$$

where $\nabla\mathcal{L}(A)$ and $\nabla\mathcal{F}(A)$ were given by Eqs. (11) and (13) respectively.

Four additional hyperparameter settings were considered in our experiments:

- $\text{SERAPH}_{\text{mani}}$ stands for $\mu = 0$, $\lambda = 0$, and $\omega = \frac{\#(\mathcal{S}\cup\mathcal{D})}{n(n-1)}$;
- $\text{SERAPH}_{\text{post+mani}}$ stands for $\mu = \frac{\#(\mathcal{S}\cup\mathcal{D})}{\#\mathcal{U}}$, $\lambda = 0$, and $\omega = \frac{\#(\mathcal{S}\cup\mathcal{D})}{n(n-1)}$;
- $\text{SERAPH}_{\text{proj+mani}}$ stands for $\mu = 0$, $\lambda = 1$, and $\omega = \frac{\#(\mathcal{S}\cup\mathcal{D})}{n(n-1)}$;
- $\text{SERAPH}_{\text{hyper+mani}}$ stands for $\mu = \frac{\#(\mathcal{S}\cup\mathcal{D})}{\#\mathcal{U}}$, $\lambda = 1$, and $\omega = \frac{\#(\mathcal{S}\cup\mathcal{D})}{n(n-1)}$.

The experimental results based on the manifold extension are reported in Table 5, where the EM-like algorithm was used, and for convenience the best hyperparameter setting in Table 3 is also listed in Table 5. We can see that $\text{SERAPH}_{\text{mani}}$ did not work at all, while $\text{SERAPH}_{\text{post+mani}}$ and $\text{SERAPH}_{\text{proj+mani}}$ were not too bad. Even $\text{SERAPH}_{\text{hyper+mani}}$ failed to improve the records on UCI data sets, but it did successfully improve 5 records on USPS and MNIST, and the improvement on USPS was all significant under the paired t-test at the significance level 5%. This is because the manifold extension has the same drawback with MFDA if the amount of training data cannot afford to recover the data manifold, and the manifold structures are vague for UCI data sets but clear for USPS and MNIST. Nevertheless, the manifold extension can hardly deal with the artificial data set shown in Figure 2 that has an extremely clear manifold structure. As illustrated in Figure 6, the manifold extension failed to stop the supervised part merging and compressing the two



(a) Data set (b) $\omega' = 1/10$ (c) $\omega' = 1/\sqrt{10}$

(d) $\omega' = 1$ (e) $\omega' = \sqrt{10}$ (f) $\omega' = 10$

Figure 6: The projected data by metrics learned exclusively with the manifold extension. The actual regularization parameter ω being used was $\omega = \omega' \cdot \frac{\#(\mathcal{S} \cup \mathcal{D})}{n(n-1)}$.

disconnected data clouds without the help of the posterior sparsity, even though it had detected and was able to recover the data manifold.

8 Conclusions

We proposed a general information-theoretic approach to semi-supervised metric learning called SERAPH (SEmi-supervised metRic leArning Paradigm with Hyper-sparsity) which follows entropy regularization rather than manifold regularization. The generalized maximum entropy distribution estimation for supervised metric learning based on weak labels was our foundation. Then, a semi-supervised extension which can achieve the posterior sparsity was obtained via entropy regularization, and low-dimensional projections which can achieve the projection sparsity were encouraged by trace-norm regularization. The non-convex optimization problem could be solved efficiently and stably by the proposed gradient projection algorithm or EM-like iterative algorithm. Last but not least, SERAPH could be easily kernelized or manifold regularized in standard manners.

Experiments on benchmark data sets demonstrated that given limited supervised in-

formation, SERAPH usually outperformed state-of-the-art supervised and semi-supervised metric learning methods, especially that the learned Mahalanobis distance possessed high discriminability even under a noisy environment. A final note is that in our experiments the posterior sparsity and the projection sparsity were shown to be very helpful for high-dimensional data only when they were combined with each other, i.e., integrated into the hyper-sparsity.

Acknowledgments

The authors would like to thank anonymous reviewers for the suggestive comments. GN was supported by the MEXT scholarship and the FIRST program, MY was supported by the JST PRESTO program, and MS was supported by the MEXT KAKENHI 23120004.

A Proofs

In this appendix, we prove the theorems appeared in Sections 2 and 3.

A.1 Proof of Theorem 1

To simplify our notations and make the proof compact, let us denote

$$p^+_{i,j} \triangleq p^A_{i,j}(+1), \qquad p^-_{i,j} \triangleq p^A_{i,j}(-1),$$
$$f^+_{i,j} \triangleq f(x_i, x_j, +1), \qquad f^-_{i,j} \triangleq f(x_i, x_j, -1), \qquad \tilde{f}_{i,j} \triangleq f(x_i, x_j, y_{i,j}),$$

respectively. Foremost, expand optimization (2) into its complete form:

$$\begin{aligned} \max_{A, p^A_{i,j}, \xi} \; & -\sum\nolimits_{\mathcal{S}\cup\mathcal{D}} (p^+_{i,j} \ln p^+_{i,j} + p^-_{i,j} \ln p^-_{i,j}) - \frac{1}{2\gamma}\xi^2 \\ \text{s.t.} \; & \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} (p^+_{i,j} f^+_{i,j} + p^-_{i,j} f^-_{i,j}) - \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \tilde{f}_{i,j} - \xi \le 0, \\ & \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \tilde{f}_{i,j} - \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} (p^+_{i,j} f^+_{i,j} + p^-_{i,j} f^-_{i,j}) - \xi \le 0, \\ & p^+_{i,j} + p^-_{i,j} = 1, \forall (x_i, x_j) \in \mathcal{S}\cup\mathcal{D}. \end{aligned}$$

The terms $\ln p^+_{i,j}$ and $\ln p^-_{i,j}$ in the objective function plus $p^+_{i,j} + p^-_{i,j} = 1$ in the constraints have already implied that

$$0 \le p^+_{i,j}, p^-_{i,j} \le 1.$$

By introducing dual variables $\kappa_1 \geq 0, \kappa_2 \geq 0$ for the first and second constraints, and $\delta_{i,j} \in \mathbb{R}$ for the third group of constraints, the Lagrangian is expressed as

$$\begin{aligned} L(A, p^A_{i,j}, \xi, \kappa_1, \kappa_2, \delta_{i,j}) = & -\sum\nolimits_{\mathcal{S}\cup\mathcal{D}} (p^+_{i,j} \ln p^+_{i,j} + p^-_{i,j} \ln p^-_{i,j}) - \frac{1}{2\gamma}\xi^2 \\ & - \kappa_1 \left(\sum\nolimits_{\mathcal{S}\cup\mathcal{D}} (p^+_{i,j} f^+_{i,j} + p^-_{i,j} f^-_{i,j}) - \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \tilde{f}_{i,j} - \xi \right) \\ & - \kappa_2 \left(\sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \tilde{f}_{i,j} - \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} (p^+_{i,j} f^+_{i,j} + p^-_{i,j} f^-_{i,j}) - \xi \right) \\ & + \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \delta_{i,j} (p^+_{i,j} + p^-_{i,j} - 1). \end{aligned}$$

Differentiating the function $L(A, p^A_{i,j}, \xi, \kappa_1, \kappa_2, \delta_{i,j})$ with respect to $p^+_{i,j}$ and $p^-_{i,j}$, and equating the derivatives to zero will give us

$$\begin{aligned} \ln p^+_{i,j} &= \kappa f^+_{i,j} + \delta_{i,j} - 1, \\ \ln p^-_{i,j} &= \kappa f^-_{i,j} + \delta_{i,j} - 1, \end{aligned} \tag{22}$$

where $\kappa = \kappa_2 - \kappa_1 \in \mathbb{R}$. Note that Eq. (22) says that

$$\ln p^+_{i,j} - \ln p^-_{i,j} = \kappa f^+_{i,j} - \kappa f^-_{i,j},$$

i.e.,

$$\frac{p^+_{i,j}}{p^-_{i,j}} = \frac{\exp(\kappa f^+_{i,j})}{\exp(\kappa f^-_{i,j})}. \tag{23}$$

Hence Eq. (3) follows with

$$\delta_{i,j} = 1 - \ln Z^A_{i,j}. \tag{24}$$

Next, differentiating $L(A, p^A_{i,j}, \xi, \kappa_1, \kappa_2, \delta_{i,j})$ with respect to ξ and equating the derivative to zero will give us

$$\xi = \gamma(\kappa_1 + \kappa_2). \tag{25}$$

According to the Karush-Kuhn-Tucker condition about the dual complementary slackness

$$\begin{aligned} \kappa_1 \left(\sum\nolimits_{\mathcal{S}\cup\mathcal{D}} (p^+_{i,j} f^+_{i,j} + p^-_{i,j} f^-_{i,j}) - \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \tilde{f}_{i,j} - \xi \right) &= 0, \\ \kappa_2 \left(\sum\nolimits_{\mathcal{S}\cup\mathcal{D}} \tilde{f}_{i,j} - \sum\nolimits_{\mathcal{S}\cup\mathcal{D}} (p^+_{i,j} f^+_{i,j} + p^-_{i,j} f^-_{i,j}) - \xi \right) &= 0, \end{aligned}$$

we know that $\kappa_1\kappa_2 = 0$, which means

$$(\kappa_1 + \kappa_2)^2 = (\kappa_1 - \kappa_2)^2 = \kappa^2. \tag{26}$$

Substituting Eq. (22)–Eq. (26) into $L(A, p^A_{i,j}, \xi, \kappa_1, \kappa_2, \delta_{i,j})$ accomplishes dual problem (4).

Finally, the optimization of the regularized maximum log-likelihood estimation is

$$\max_{A,\kappa} \ \mathcal{L}_1(A, \kappa).$$

By plugging the probabilistic model (3) into it we get optimization (4) exactly, which is the dual problem of the generalized maximum entropy estimation (2). □

A.2 Proof of Theorem 2

As mentioned before, there must be $\kappa^* > 0$. It is clear that $\kappa^* < +\infty$ and $\mathrm{tr}(A^*) < +\infty$, since they are penalized in optimization (6). Let $\hat{\eta} = \kappa^*\eta$ and $\hat{\lambda} = \lambda/\kappa^*$. Then $\hat{\eta}$ and $\hat{\lambda}$ are well-defined hyperparameters as finite positive real numbers, and $\hat{A}$ is a feasible solution to (7) as a finite-trace symmetric positive semi-definite matrix. Notice that the two properties described in Eqs. (9) and (10) hold, and then the theorem follows if $\hat{A}$ is a local maximum of $\hat{\mathcal{L}}(A)$.

Differentiate p^A and $\hat{p}^A$ with respect to A,

$$\partial p^A/\partial A = \kappa y p^A(1-p^A)(x-x')(x-x')^\top, \tag{27}$$

$$\partial \hat{p}^A/\partial A = -y\hat{p}^A(1-\hat{p}^A)(x-x')(x-x')^\top. \tag{28}$$

From (10) we have

$$\partial\hat{\mathcal{L}}/\partial\hat{p}^A\big|_{A=\hat{A}} = \partial\mathcal{L}/\partial p^A\big|_{A=A^*,\kappa=\kappa^*}.$$

Thus from

$$\partial\hat{p}^A/\partial A\big|_{A=\hat{A}} = -(1/\kappa^*)\partial p^A/\partial A\big|_{A=A^*,\kappa=\kappa^*},$$

$\partial\,\mathrm{tr}(A)/\partial A = I_m$ where I_m is the identity matrix, and the KKT stationarity condition of optimization (6)

$$\partial\mathcal{L}/\partial A\big|_{A=A^*,\kappa=\kappa^*} = 0_{m\times m}$$

where $0_{m\times m}$ is the zero matrix in $\mathbb{R}^{m\times m}$, we get

$$\partial\hat{\mathcal{L}}/\partial A\big|_{A=\hat{A}} = -(1/\kappa^*)\partial\mathcal{L}/\partial A\big|_{A=A^*,\kappa=\kappa^*} = 0_{m\times m}.$$

This implies that $\hat{A}$ is a stationary point of $\hat{\mathcal{L}}(A)$.

Similarly, it is not difficult to verify that

$$\partial^2\hat{\mathcal{L}}/\partial(\hat{p}^A)^2\big|_{A=\hat{A}} = \partial^2\mathcal{L}/\partial(p^A)^2\big|_{A=A^*,\kappa=\kappa^*},$$
$$\partial^2\hat{p}^A/\partial A^2\big|_{A=\hat{A}} = (1/\kappa^*)^2\partial^2 p^A/\partial A^2\big|_{A=A^*,\kappa=\kappa^*},$$

and

$$\partial^2\mathcal{L}/\partial A^2 = (\partial^2\mathcal{L}/\partial(p^A)^2)\cdot(\partial p^A/\partial A)^2 + (\partial\mathcal{L}/\partial p^A)\cdot(\partial^2 p^A/\partial A^2).$$

As a result,

$$\partial^2\hat{\mathcal{L}}/\partial A^2\big|_{A=\hat{A}} = (1/\kappa^*)^2\partial^2\mathcal{L}/\partial A^2\big|_{A=A^*,\kappa=\kappa^*}.$$

Hence, $\partial_A^2\hat{\mathcal{L}}(\hat{A})$ is negative definite if $\partial_A^2\mathcal{L}(A^*,\kappa^*)$ is negative definite, and $\hat{A}$ is actually a local maximum of $\hat{\mathcal{L}}(A)$. □

A.3 Proof of Theorem 3

Let us first consider $\mathcal{F}(A)$. Obviously, it is differentiable as long as we allow unbounded derivatives. Since the conjugate norm of the Frobenius norm is still the Frobenius norm, the best Lipschitz constant of $\mathcal{F}$ with respect to $\|\cdot\|_{\text{Fro}}$ is expressed as

$$\text{Lip}_{\|\cdot\|_{\text{Fro}}}(\mathcal{F}) = \sup_{A\succeq 0} \|\nabla\mathcal{F}(A)\|_{\text{Fro}}.$$

So we are going to prove that $\|\nabla\mathcal{F}(A)\|_{\text{Fro}}$ is uniformly bounded for fixed training set $\mathcal{X}$. It is sufficient to bound $\|(\partial\mathcal{F}/\partial p^A_{i,j})(\partial p^A_{i,j}/\partial A)\|_{\text{Fro}}$ for all $(x_i, x_j) \in \mathcal{S}\cup\mathcal{D}\cup\mathcal{U}$ from above uniformly, which is more convenient than to bound $\|\nabla\mathcal{F}(A)\|_{\text{Fro}}$ directly.

Recall that the partial derivative of the simplified $p^A(y \mid x, x')$ with respect to A was given by Eq. (28) as

$$\partial p^A/\partial A = -yp^A(1-p^A)(x-x')(x-x')^\top.$$

On the other hand,

$$\partial\mathcal{F}/\partial p^A_{i,j} = \begin{cases} 1/p^A_{i,j}(y_{i,j}) & \text{if } (x_i,x_j)\in\mathcal{S}\cup\mathcal{D} \\ \mu q(y \mid x_i, x_j)/p^A_{i,j}(y) & \text{if } (x_i,x_j)\in\mathcal{U}, y\in\{1,-1\}. \end{cases}$$

Hence when $(x_i, x_j) \in \mathcal{S}\cup\mathcal{D}$,

$$\begin{aligned} \|(\partial\mathcal{F}/\partial p^A_{i,j})(\partial p^A_{i,j}/\partial A)\|_{\text{Fro}} &= \|-y_{i,j}(1-p^A_{i,j}(y_{i,j}))(x_i-x_j)(x_i-x_j)^\top\|_{\text{Fro}} \\ &\le \|(x_i-x_j)(x_i-x_j)^\top\|_{\text{Fro}} \\ &= \|x_i-x_j\|_2^2 \\ &\le (\text{diam}(\mathcal{X}))^2, \end{aligned}$$

where we use the fact that

$$\|zz^\top\|^2_{\text{Fro}} = \sum_{i,j=1}^m (z_iz_j)^2 = \left(\sum_{i=1}^m z_i^2\right)\left(\sum_{j=1}^m z_j^2\right) = \|z\|_2^4.$$

Similarly, we have that when $(x_i, x_j) \in \mathcal{U}$ for fixed y,

$$\|(\partial\mathcal{F}/\partial p^A_{i,j})(\partial p^A_{i,j}/\partial A)\|_{\text{Fro}} \le \mu q(y \mid x_i, x_j)(\text{diam}(\mathcal{X}))^2,$$

and thus

$$\sum\nolimits_y \|(\partial\mathcal{F}/\partial p^A_{i,j})(\partial p^A_{i,j}/\partial A)\|_{\text{Fro}} \le \mu(\text{diam}(\mathcal{X}))^2.$$

As a consequence, there exists a finite $\text{Lip}_{\|\cdot\|_{\text{Fro}}}(\mathcal{F})$, and (16) can be obtained by applying the triangle inequality of the Frobenius norm.

Now let us consider $\mathcal{L}(A)$. The only difference is that for $(x_i, x_j) \in \mathcal{U}$ and fixed y,

$$\partial\mathcal{L}/\partial p^A_{i,j} = \mu(1+\ln p^A_{i,j}(y)),$$

and thus

$$\begin{aligned}\sum\nolimits_y \|(\partial\mathcal{L}/\partial p_{i,j}^A)(\partial p_{i,j}^A/\partial A)\|_{\mathrm{Fro}} &\le \mu(\mathrm{diam}(\mathcal{X}))^2 \sum\nolimits_y |p_{i,j}^A(y) + p_{i,j}^A(y)\ln p_{i,j}^A(y)| \\ &\le \mu(\mathrm{diam}(\mathcal{X}))^2 \left(\sum\nolimits_y p_{i,j}^A(y) - \sum\nolimits_y p_{i,j}^A(y)\ln p_{i,j}^A(y)\right) \\ &\le (1+\ln 2)\mu(\mathrm{diam}(\mathcal{X}))^2,\end{aligned}$$

where the second step is because $p_{i,j}^A(y)\ln p_{i,j}^A(y)$ is negative, and the last step is because

$$-\sum\nolimits_y p_{i,j}^A(y)\ln p_{i,j}^A(y) = H(p_{i,j}^A) \le \ln 2.$$

Then, (15) can be obtained similarly to (16). □

B Implementation Details

In this appendix, we explain the details of our implementation, in particular two simple tricks for the speedup. The pseudo codes are in Matlab, while the tricks apply to similar high-level programming languages.

To begin with, we must compute the Mahalanobis distances

$$\|x_i - x_j\|_A^2 = (x_i - x_j)^\top A(x_i - x_j)$$

for all $(x_i, x_j) \in \mathcal{S}\cup\mathcal{D}\cup\mathcal{U}$. If implemented naively, each pair will cost $O(m^2)$ time and there are $O(n^2)$ pairs, so it will consume $O(n^2m^2)$ time. The trick here is to compute the distances for all pairs simultaneously:

$$\begin{aligned}\bar{x} &\leftarrow \mathrm{diag}(XAX^\top), \\ M &\leftarrow \mathrm{repmat}(\bar{x}, 1, n) + \mathrm{repmat}(\bar{x}^\top, n, 1) - 2XAX^\top,\end{aligned}$$

where $\mathrm{repmat}(\bar{x}, 1, n)$ means to replicate the column vector $\bar{x}$ and form a 1-by-n tiling of copies of $\bar{x}$ and $\mathrm{repmat}(\bar{x}^\top, n, 1)$ means to replicate the row vector $\bar{x}^\top$ and form an n-by-1 tiling of copies of $\bar{x}^\top$. This would result in $M \in \mathbb{R}^{n\times n}$ such that $M_{i,j} = \|x_i - x_j\|_A^2$. The computation of $XAX^\top$ costs $O(nm^2)$, the computation of M costs $O(n^2)$, and hence this trick reduces the computational complexity from $O(n^2m^2)$ to $O(nm^2 + n^2)$. After we get M, we can compute $p_{i,j}^A(y=+1)$ for all $(x_i, x_j) \in \mathcal{S}\cup\mathcal{D}\cup\mathcal{U}$ simply by

$$P \leftarrow 1./(1 + \exp(M - \eta 1_{n\times n})),$$

where "./" and "exp" are the element-wise division and exponential on matrices. We can similarly compute $\mathcal{L}(A)$, $\mathcal{F}(A)$, and $q(y \mid x_i, x_j)$ for all $(x_i, x_j) \in \mathcal{U}$ with the computational complexity $O(n^2)$.

However, the computation of $\nabla\mathcal{L}(A)$ or $\nabla\mathcal{F}(A)$, which requires $O(n^2m)$ time, is the main bottleneck. In practice, we should avoid using the computationally-inefficient structure "double for loops", and fortunately there is such a trick. Without loss of generality,

Algorithm 1 Efficient Computation of $\nabla\mathcal{F}(A)$

Input: $A \in \mathbb{R}^{m\times m}$ that is the current solution,
$X \in \mathbb{R}^{n\times m}$ that is the design matrix of $\mathcal{X}$,
$S \in \mathbb{R}^{n\times n}$ such that $S_{i,j} = S_{j,i} = 1$ if $(x_i, x_j) \in \mathcal{S}$ and $S_{i,j} = 0$ otherwise,
$D \in \mathbb{R}^{n\times n}$ such that $D_{i,j} = D_{j,i} = 1$ if $(x_i, x_j) \in \mathcal{D}$ and $D_{i,j} = 0$ otherwise,
$Q \in \mathbb{R}^{n\times n}$ such that $Q_{i,j} = q(y = +1 \mid x_i, x_j)$ for $(x_i, x_j) \in \mathcal{U}$
Output: $\nabla\mathcal{F}(A)$

1: $\bar{x} \leftarrow \mathrm{diag}(XAX^\top)$
$M \leftarrow \mathrm{repmat}(\bar{x}, 1, n) + \mathrm{repmat}(\bar{x}^\top, n, 1) - 2XAX^\top$
2: $P \leftarrow 1./(1 + \exp(M - \eta 1_{n\times n}))$
3: $C \leftarrow 0_{n\times n}$
4: $O \leftarrow 1_{n\times n}$
$C_S \leftarrow P_S - O_S$
$C_D \leftarrow P_D$
5: $U \leftarrow O - S - D - I_n$
$C_U \leftarrow \mu(P_U - Q_U)$
6: $\nabla\mathcal{F}(A) \leftarrow X^\top(\mathrm{repmat}(C1_n, 1, m). * X) - X^\top CX - \lambda I_m$

we describe the efficient computation of $\nabla\mathcal{F}(A)$ in Algorithm 1. We have observed that in our experiments Algorithm 1 was at least twenty times faster than the computation of $\nabla\mathcal{F}(A)$ using double for loops. In this algorithm, we use a matrix $C \in \mathbb{R}^{n\times n}$ to store all coefficients of $(x_i - x_j)(x_i - x_j)^\top$, and the entries of C are computed separately for S, D and U:

$$\begin{aligned} C_S &\leftarrow P_S - O_S, \\ C_D &\leftarrow P_D, \\ C_U &\leftarrow \mu(P_U - Q_U) \end{aligned}$$

where the subscript S, D or U means that the involved element-wise operations are done only for the entries corresponding to $S_{i,j} = 1$, $D_{i,j} = 1$ or $U_{i,j} = 1$. Since C was initialized as the zero matrix, we would have

$$C_{i,j} = C_{j,i} = \begin{cases} 0 & \text{if } i = j, \\ -y_{i,j}(1 - p^A_{i,j}(y_{i,j})) & \text{if } (x_i, x_j) \in \mathcal{S} \cup \mathcal{D}, \\ -\mu \sum_y yq(y \mid x_i, x_j)(1 - p^A_{i,j}(y)) & \text{if } (x_i, x_j) \in \mathcal{U}. \end{cases}$$

At last, $\nabla\mathcal{F}(A)$ can be obtained by

$$\nabla\mathcal{F}(A) \leftarrow X^\top(\mathrm{repmat}(C1_n, 1, m). * X) - X^\top CX - \lambda I_m.$$

Note that $\nabla\mathcal{F}(A) \neq 2X^\top(\mathrm{repmat}(C1_n, 1, m). * X) - 2X^\top CX - \lambda I_m$, since we have considered each pair (x_i, x_j) twice in the algorithm by $C_{i,j}$ and $C_{j,i}$ but it appears only once

in Eq. (13). Moreover, it suffices to replace $C_U \leftarrow \mu(P_U - Q_U)$ in Algorithm 1 with

$$C_U \leftarrow \mu(\ln P_U - \ln(O_U - P_U)). * P_U . * (P_U - O_U)$$

to efficiently compute $\nabla\mathcal{L}(A)$, where ".∗" is the element-wise multiplication on matrices.

References

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems 19 (NIPS)*, 2007.

M. Baghshah and S. Shouraki. Semi-supervised metric learning using pairwise constraints. In *Proceedings of 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14 (NIPS)*, 2002.

K. Bellare, G. Druck, and A. McCallum. Alternating projections for learning with expectation constraints. In *Proceedings of 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2009.

A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. http://arxiv.org/abs/1306.6709, 2013.

A. Berger, S. Pietra, and V. Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22:39–71, 1996.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

F. Chiaromonte and R. Cook. Sufficient dimension reduction and graphics in regression. *Annals of the Institute of Statistical Mathematics*, 54:768–795, 2002.

J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. Information-theoretic metric learning. In *Proceedings of 24th International Conference on Machine Learning (ICML)*, 2007.

M. Dudík and R. E. Schapire. Maximum entropy distribution estimation with generalized regularization. In *Proceedings of 19th Annual Conference on Learning Theory (COLT)*, 2006.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

K. Fukumizu, F. Bach, and M. I. Jordan. Kernel dimension reduction in regression. *Annals of Statistics*, 37(4):1871–1905, 2009.

J. Gillenwater, K. Ganchev, J. Graça, F. Pereira, and B. Taskar. Posterior sparsity in usupervised dependency parsing. *Journal of Machine Learning Research*, 12:455–490, 2011.

A. Globerson and S. Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems 18 (NIPS)*, 2006.

J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 17 (NIPS)*, 2005.

R. Gomes, A. Krause, and P. Perona. Discriminative clustering by regularized information maximization. In *Advances in Neural Information Processing Systems 23 (NIPS)*, 2010.

J. Graça, K. Ganchev, and B. Taskar. Expectation maximization and posterior constraints. In *Advances in Neural Information Processing Systems 20 (NIPS)*, 2008.

J. Graça, K. Ganchev, B. Taskar, and F. Pereira. Posterior vs. parameter sparsity in latent variable models. In *Advances in Neural Information Processing Systems 22 (NIPS)*, 2009.

Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems 17 (NIPS)*, 2005.

Y. Grandvalet and Y. Bengio. Entropy regularization. In O. Chapelle, B. Schölkopf, and A. Zien, editors, *Semi-Supervised Learning*, pages 151–168. MIT Press, 2006.

S. Hoi, W. Liu, and S.-F. Chang. Semi-supervised distance metric learning for collaborative image retrieval. In *Proceedings of 21st IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

K. Huang, Y. Ying, and C. Campbell. GSML: A unified framework for sparse metric learning. In *Proceedings of 9th IEEE International Conference on Data Mining (ICDM)*, 2009.

K. Huang, R. Jin, Z. Xu, and C. Liu. Robust metric learning by smooth optimization. In *Proceedings of 26th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010.

P. Jain, B. Kulis, and I. Dhillon. Inductive regularized learning of kernel functions. In *Advances in Neural Information Processing Systems 23 (NIPS)*, 2010.

E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4): 620–630, 1957.

W. Liu, S. Ma, D. Tao, J. Liu, and P. Liu. Semi-supervised sparse metric learning using alternating linearization optimization. In *Proceedings of 16th ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2010.

G. Niu, B. Dai, M. Yamada, and M. Sugiyama. Information-theoretic semi-supervised metric learning via entropy regularization. In *Proceedings of 29th International Conference on Machine Learning (ICML)*, 2012.

B. T. Polyak. A general method for solving extremal problems (in Russian). *Soviet Mathematics Doklady*, 174(1):33–36, 1967.

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

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2001.

M. Sugiyama. Dimensionality reduction of multimodal labeled data by local Fisher discriminant analysis. *Journal of Machine Learning Research*, 8:1027–1061, 2007.

M. Sugiyama, T. Idé, S. Nakajima, and J. Sese. Semi-supervised local Fisher discriminant analysis for dimensionality reduction. *Machine Learning*, 78(1-2):35–61, 2010.

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

L. Torresani and K. Lee. Large margin component analysis. In *Advances in Neural Information Processing Systems 19 (NIPS)*, 2007.

J. Wang, H. Do, A. Woznica, and A. Kalousis. Metric learning with multiple kernels. In *Advances in Neural Information Processing Systems 24 (NIPS)*, 2011.

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

E. Xing, A. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15 (NIPS)*, 2003.

L. Yang, R. Jin, R. Sukthankar, and Y. Liu. An efficient algorithm for local distance metric learning. In *Proceedings of 21st National Conference on Artificial Intelligence (AAAI)*, 2006.

Y. Ying, K. Huang, and C. Campbell. Sparse metric learning via smooth optimization. In *Advances in Neural Information Processing Systems 22 (NIPS)*, 2009.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society*, Series B, 68(1):49–67, 2006.

Z. Zha, T. Mei, M. Wang, Z. Wang, and X. Hua. Robust distance metric learning with auxiliary knowledge. In *Proceedings of 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.