Data-Adaptive Metric Learning with Scale Alignment

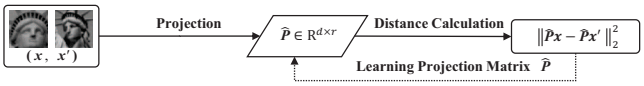
Shuo Chen[†], Chen Gong[†], Jian Yang[†*], Ying Tai[‡], Lei Hui[†], Jun Li[†]
[†]PCA Lab, Key Lab of Intelligent Perception and System for High-Dimensional Information of Ministry of Education
[†]Jiangsu Key Lab of Image and Video Undersatanding for Social Security
[†]School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
[‡]Youtu Lab, Tencent
{shuochen, chen.gong, csjyang, le.hui}@njust.edu.cn, yingtai@tencent.com, junl.mldl@gmail.com

Abstract

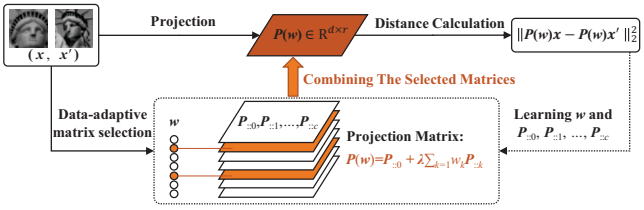
The central problem for most existing metric learning methods is to find a suitable projection matrix on the differences of all pairs of data points. However, a single unified projection matrix can hardly characterize all data similarities accurately as the practical data are usually very complicated, and simply adopting one global projection matrix might ignore important local patterns hidden in the dataset. To address this issue, this paper proposes a novel method dubbed "Data-Adaptive Metric Learning" (DAML), which constructs a data-adaptive projection matrix for each data pair by selectively combining a set of learned candidate matrices. As a result, every data pair can obtain a specific projection matrix, enabling the proposed DAML to flexibly fit the training data and produce discriminative projection results. The model of DAML is formulated as an optimization problem which jointly learns candidate projection matrices and their sparse combination for every data pair. Nevertheless, the over-fitting problem may occur due to the large amount of parameters to be learned. To tackle this issue, we adopt the Total Variation (TV) regularizer to align the scales of data embedding produced by all candidate projection matrices, and thus the generated metrics of these learned candidates are generally comparable. Furthermore, we extend the basic linear DAML model to the kernerlized version (denoted "KDAML") to handle the non-linear cases, and the Iterative Shrinkage-Thresholding Algorithm (ISTA) is employed to solve the optimization model. Intensive experimental results on various applications including retrieval, classification, and verification clearly demonstrate the superiority of our algorithm to other state-of-the-art metric learning methodologies.

Introduction

Metric learning aims to learn a distance function for data pairs to faithfully measure their similarities. It has played an important role in many pattern recognition applications, such as face verification (Liu et al. 2018), person re-identification (Si et al. 2018), and image retrieval (Zhan et al. 2009; Liu, Tsang, and Müller 2017). The well-studied metric learning models are usually global, which means that they directly learn a single semi-positive definite (SPD) matrix $\widehat{\boldsymbol{M}} = \widehat{\boldsymbol{P}}^\top \widehat{\boldsymbol{P}}$ to decide a Mahalanobis distance function $D_{\widehat{\boldsymbol{P}}}(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} - \boldsymbol{x}')^\top \widehat{\boldsymbol{M}}(\boldsymbol{x} - \boldsymbol{x}')$[1] for all data pairs $(\boldsymbol{x}, \boldsymbol{x}')$ (see Fig. 1(a)), where $\widehat{\boldsymbol{P}}$ can be understood as a projection matrix. The detailed implementations can be linear projection (Harandi, Salzmann, and Hartley 2017) or non-linear deep neural networks (DNN) (Oh Song et al. 2016). The primitive linear works utilized the supervised information (*e.g.* must-link and cannot-link) to control the learned distance during their training phases, such as Distance Metric Learning for Clustering (Xing et al. 2003), Large Margin Nearest Neighbor (LMNN) (Weinberger, Blitzer, and Saul 2006), and Information-Theoretic Metric Learning (ITML) (Davis et al. 2007). To enhance the fitting performance and effectively discover the structure for more complicated data, some recent non-linear methods including Projection Metric Learning on Grassmann Manifold (Huang et al. 2015) and Geometric Mean Metric Learning (GMML) (Zadeh, Hosseini, and Sra 2016) are proposed to learn the matrices $\widehat{\boldsymbol{M}}$ and $\widehat{\boldsymbol{P}}$ on a manifold instead of the original linear space. Moreover, by adaptively enriching the training data, Adversarial Metric Learning (Chen et al. 2018; Duan et al. 2018) showed further improvement on the linear metric by dis-



(a). Framework of Traditional Metric Learning.



(b). Framework of Our Proposed DAML.

Figure 1: The comparison of traditional metric learning and our proposed model. (a) The traditional method learns a single global projection matrix $\widehat{\boldsymbol{P}}$ to distinguish the similarity of $(\boldsymbol{x}, \boldsymbol{x}')$. (b) Our proposed DAML jointly learns multiple projection matrices $\boldsymbol{P}_{::0}$, $\boldsymbol{P}_{::1}$, $\cdots$, $\boldsymbol{P}_{::c}$ and their weight vector $\boldsymbol{w} \in \mathbb{R}^c$ for each data pair $(\boldsymbol{x}, \boldsymbol{x}')$.

[*]Corresponding author.

[1]For simplicity, the notation of "square" on $D_{\widehat{\boldsymbol{P}}}(\boldsymbol{x}, \boldsymbol{x}')$ has been omitted and it will not influence the final output.

criminating the confusing yet critical data pairs produced by the generator. There are some other methods that replace the linear projection $\widehat{\boldsymbol{P}}\boldsymbol{x}$ by the non-linear form $\widehat{\boldsymbol{P}}\boldsymbol{\mathcal{W}}(\boldsymbol{x})$ so that the model representation ability can be boosted, in which the mapping $\boldsymbol{\mathcal{W}}(\cdot)$ usually indicates a deep neural network. For example, the Convolutional Neural Network (CNN) is adopted by Siamese-Net (Zagoruyko and Komodakis 2015) while Multi-Layer Perceptron (MLP) is employed by Discriminative Deep Metric Learning (DDML) (Hu, Lu, and Tan 2014). However, the above global metric learning methods are not flexible for handling complex or heterogeneous data, because they all use the same projection operator for all data pairs, which might be inappropriate to characterize the local data properties. As a result, some important patterns carried by the training data are ignored and the learned metric can be inaccurate.

To improve the flexibility of metric learning for fitting complex data pairs, various local models were proposed from different viewpoints. For example, LMNN was extended to a local version by learning a specific metric for each class based on certain classification criterion (Weinberger and Saul 2008). Afterwards, the Instance Specific Distance was proposed to further enhance the metric flexibility on each of the training examples (Zhan et al. 2009). Recently, in Parametric Local Metric Learning (Wang, Kalousis, and Woznica 2012), the authors proposed the weighted Mahalanobis distance in which multiple metric values are linearly combined. Based on the similar way, the traditional methods LMNN and GMML have also been extended to the local forms by introducing the weighted distances (Bohné et al. 2014; Su, King, and Lyu 2017). In contrast to the combination of multiple metrics of PLML, a Gaussian mixture based model (Luo and Huang 2018) partitioned the metric $\widehat{\boldsymbol{M}}$ into multiple blocks and proposed a localized norm to improve the model flexibility. However, these improvements on metric $\widehat{\boldsymbol{M}}$ are not guaranteed to consistently render reasonable projections for discriminating the similar pairs from dissimilar ones. Therefore, there are also some works aiming at directly refining the projection operator $\widehat{\boldsymbol{P}}$. For instance, Gated Siamese-Net (Varior, Haloi, and Wang 2016) employed a gating function to selectively emphasize the local pattern for the projected data. Similarly, an attention mechanism was utilized in the image matching model (Si et al. 2018), by which the feature-pair alignment can be performed on the projection results.

Although the existing metric learning models have achieved promising results to some extent, most of them cannot adaptively find the suitable projection strategy for different data pairs, as they are not data-adaptive and thus the learning flexibility is rather limited. To this end, we propose a novel metric learning model that generalizes the single projection matrix to multiple ones, and establish a selective mechanism to adaptively utilize them according to the local property of data points (see Fig. 1(b)). In other words, our method jointly learns multiple candidate matrices and their sparse combinations for different training pairs. On one hand, every pair of examples is associated with a definite projection matrix which is constructed by wisely selecting and combining a small fraction of candidate matrices. On the other hand, the candidate matrices are also automatically learned to minimize the training loss on each training pair. Considering that such a data-adaptive projection may bring about over-fitting, we further introduce the concept of "metric scale" and employ the Total Variation (TV) regularizer to enforce the embedded data produced by different candidate projection matrices are generally aligned in the same scale. Consequently, the solution space for learning the candidate projection matrices shrinks and the over-fitting problem caused by scale variations can be effectively alleviated. Thanks to the sparse selections of projection matrices and the operation of scale alignment, every data pair is able to acquire suitable projection to reach discriminative representation for further distance calculations. Therefore, our proposed method is termed as "Data-Adaptive Metric Learning" (DAML). The main contributions of this paper are summarized below:

- We propose a novel metric learning framework dubbed DAML, which is able to learn adaptive projections for different data pairs to enhance the discriminability and flexibility of the learned metric.
- A kenerlized version is devised to enable the DAML model to successfully handle the non-linear cases, and an efficient optimization algorithm is designed to solve the proposed model which is guaranteed to converge.
- DAML is empirically validated on various typical datasets and the results suggest that DAML outperforms other state-of-the-art metric learning methodologies.

The Proposed DAML Model

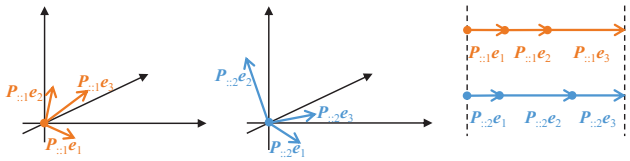
In this section, we first introduce some necessary notations. After that, we establish the basic DAML model in linear space, and then extend it to a kernerlized form to handle the non-linear cases. Finally, we derive the Iterative Shrinkage-Thresholding Algorithm (ISTA) to solve the proposed optimization problem.

Notations

Throughout this paper, we write matrices as bold uppercase characters and vectors as bold lowercase characters. Tensors are written as Euclid uppercase characters. Let $\boldsymbol{y} = (y_1, y_2, \cdots, y_n)^\top$ be the label vector of training data pairs $\mathcal{X} = \{(\boldsymbol{x}_1, \boldsymbol{x}'_1), (\boldsymbol{x}_2, \boldsymbol{x}'_2), \cdots, (\boldsymbol{x}_n, \boldsymbol{x}'_n)\}$ with $\boldsymbol{x}_i, \boldsymbol{x}'_i \in \mathbb{R}^d$, where $y_i = 1$ if $\boldsymbol{x}_i$ and $\boldsymbol{x}'_i$ are similar, and $y_i = 0$ otherwise. Here d is the data dimensionality and n is the total number of data pairs. Given $\boldsymbol{\mathcal{P}}$ as a three-order tensor, and then the k-th slice of tensor $\boldsymbol{\mathcal{P}}$ is written as $\boldsymbol{P}_{::k}$. The notations $||\cdot||_F$, $||\cdot||_1$, and $||\cdot||_2$ denote the Frobenius-norm, l_1-norm, and l_2-norm, respectively. The Total Variation (TV) norm $||\boldsymbol{a}||_{\text{tv}}$ for a vector $\boldsymbol{a} \in \mathbb{R}^d$ is defined as $\sum_{i=1}^{d-1} |a_i - a_{i+1}|$. The signum function $\text{sign}(z) = 1$ if $z \geq 0$, and $\text{sign}(z) = -1$ otherwise.

Linear DAML Model

Given the matrix $\widehat{\boldsymbol{M}} \in \mathbb{R}^{d\times d}$ in the traditional Mahalanobis distance, we know that $\widehat{\boldsymbol{M}}$ can be decomposed as



(a). $\boldsymbol{e}_i$'s projection on $\boldsymbol{P}_{::1}$. (b). $\boldsymbol{e}_i$'s projection on $\boldsymbol{P}_{::2}$. (c). Scale Alignment.

Figure 2: Illustration of scale alignment. The lengths of projected $\boldsymbol{e}_i$ ($i = 1, 2, \cdots, d$) by different projection matrices are summed up to the same value. Here d is simply set to 3 for visualization.

$\widehat{\boldsymbol{M}} = \widehat{\boldsymbol{P}}^\top \widehat{\boldsymbol{P}}$, and thus the squared Mahalanobis distance $(\boldsymbol{x}_i - \boldsymbol{x}'_i)^\top \widehat{\boldsymbol{M}}(\boldsymbol{x}_i - \boldsymbol{x}'_i)$ between $\boldsymbol{x}_i$ and $\boldsymbol{x}'_i$ is equivalent to the Euclidean distance after their projections by $\widehat{\boldsymbol{P}}$, *i.e.,*

$$D_{\widehat{\boldsymbol{P}}}(\boldsymbol{x}_i, \boldsymbol{x}'_i) = ||\widehat{\boldsymbol{P}}\boldsymbol{x}_i - \widehat{\boldsymbol{P}}\boldsymbol{x}'_i||_2^2, \tag{1}$$

where $\widehat{\boldsymbol{P}}$ is the projection matrix of the size $d \times r$, and r is the dimensionality of projection results. Since only one global projection matrix $\widehat{\boldsymbol{P}}$ is adopted, the traditional methods are not sufficiently flexible for learning the similarities of all data pairs. To address this limitation, we build a data-adaptive metric learning scheme which automatically generates the suitable projection matrix for each of the n data pairs $(\boldsymbol{x}_i, \boldsymbol{x}'_i)$ $(i = 1, 2, \cdots, n)$. As such, the local property of dataset can be exploited and an improved metric can be finally learned. To be specific, we propose the following distance regarding tensor $\boldsymbol{\mathcal{P}}$, namely

$$D_{\boldsymbol{\mathcal{P}}}(\boldsymbol{x}_i, \boldsymbol{x}'_i) = ||\boldsymbol{P}(\boldsymbol{w}_i)\boldsymbol{x}_i - \boldsymbol{P}(\boldsymbol{w}_i)\boldsymbol{x}'_i||_2^2, \tag{2}$$

where $\boldsymbol{\mathcal{P}}$ is a three-order tensor stacked by a primitive projection matrix $\boldsymbol{P}_{::0}$ and c candidate matrices $\boldsymbol{P}_{::1}, \boldsymbol{P}_{::2} \cdots, \boldsymbol{P}_{::c}$ in depth. For the i-th data pair, its specific projection matrix has the form

$$\boldsymbol{P}(\boldsymbol{w}_i) = \boldsymbol{P}_{::0} + \lambda \sum\nolimits_{k=1}^{c} w_{ik}\boldsymbol{P}_{::k}, \tag{3}$$

where $\boldsymbol{w}_i = (w_{i1}, w_{i2}, \cdots, w_{ic})^\top$ and w_{ik} is the weight of candidate projection matrix $\boldsymbol{P}_{::k}$ for constructing $\boldsymbol{P}(\boldsymbol{w}_i)$. The parameter λ is tuned by the user, and $\lambda = 0$ degenerates our model to the traditional metric learning. From Eq. (3), we see that the data-adaptive projection matrix $\boldsymbol{P}(\boldsymbol{w}_i)$ is decided by the sum of a primitive projection matrix $\boldsymbol{P}_{::0}$ and the linear combination of the candidate projection matrices. Therefore, we have to jointly learn the tensor $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{r\times d\times(c+1)}$ for all data pairs as well as the weight vector $\boldsymbol{w}_i$ for the i-th training data pair $(\boldsymbol{x}_i, \boldsymbol{x}'_i)$. By taking all n training pairs into consideration and putting all $\boldsymbol{w}_i$ into a matrix $\boldsymbol{W} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_n)$, the basic empirical loss for our DAML model is formed as

$$\mathcal{L}(\boldsymbol{\mathcal{P}}, \boldsymbol{W}) = \frac{1}{n}\sum\nolimits_{i=1}^{n} l(D_{\boldsymbol{\mathcal{P}}}(\boldsymbol{x}_i, \boldsymbol{x}'_i), y_i), \tag{4}$$

in which the function $l(D_{\boldsymbol{\mathcal{P}}}(\boldsymbol{x}_i, \boldsymbol{x}'_i), y_i)$ evaluates the inconsistency between the label y_i and the model prediction $D_{\boldsymbol{\mathcal{P}}}(\boldsymbol{x}_i, \boldsymbol{x}'_i)$ for the data pair $(\boldsymbol{x}_i, \boldsymbol{x}'_i)$. Note that practically not all candidate projection matrices are needed to generate the data-adaptive projection matrix $\boldsymbol{P}(\boldsymbol{w}_i)$ for the i-th data pair, so we use the regularizer $||\boldsymbol{W}||_1$ to encourage the algorithm to sparsely select a small subset of candidate matrices $\boldsymbol{P}_{::0}, \boldsymbol{P}_{::1}, \cdots, \boldsymbol{P}_{::c}$ to suitably reconstruct $\boldsymbol{P}(\boldsymbol{w}_i)$.

The good news by introducing the combination of candidate matrices is that the fitting ability of our model can be enhanced. Nevertheless, the bad news is that if we merely minimize the loss function Eq. (4) equipped with the sparse regularizer $||\boldsymbol{W}||_1$ to learn our metric, the over-fitting problem may occur due to the large amount of entries in $\{\boldsymbol{P}_{::0}, \boldsymbol{P}_{::1}, \cdots, \boldsymbol{P}_{::c}\}$ to be learned. Therefore, we should find a way to constrain the final solution to a suitable hypothesis space. Ideally, the linear combination of $c+1$ candidate matrices in $\{\boldsymbol{P}_{::0}, \boldsymbol{P}_{::1}, \cdots, \boldsymbol{P}_{::c}\}$ can produce exact label for every specific data pair when we minimize $l(D_{\boldsymbol{\mathcal{P}}}(\boldsymbol{x}_i, \boldsymbol{x}'_i), y_i)$, which is undesirable and cannot acquire the reasonable general metric. This is because the candidate matrices in $\{\boldsymbol{P}_{::0}, \boldsymbol{P}_{::1}, \cdots, \boldsymbol{P}_{::c}\}$ can produce the results with arbitrary scales.

To address the over-fitting problem caused by scale variations, we introduce the notation of "metric scale" $s(\boldsymbol{P}_{::k})$ for $\boldsymbol{P}_{::k}$ $(k = 0, 1, \cdots, c)$ and require all scales yielded by $\{\boldsymbol{P}_{::0}, \boldsymbol{P}_{::1}, \cdots, \boldsymbol{P}_{::c}\}$ to be as close as possible. To this end, we devise the TV regularizer on the scale vector $\boldsymbol{s}(\boldsymbol{\mathcal{P}}) = (s(\boldsymbol{P}_{::0}), s(\boldsymbol{P}_{::1}), \cdots, s(\boldsymbol{P}_{::c}))^\top \in \mathbb{R}^{c+1}$, such that the difference between any two of $\{s(\boldsymbol{P}_{::0}), s(\boldsymbol{P}_{::1}), \cdots, s(\boldsymbol{P}_{::c})\}$ can be minimized. Since all projection matrices adopt the comparable scales to measure the distance between pairs of data points, the overfitting caused by scale variations of projection matrices can be effectively alleviated. Specifically, $s(\boldsymbol{P}_{::k})$ is defined as the sum of squared Mahalanobis distances between the projection on orthonormal bases (*i.e.,* $\boldsymbol{P}_{::k}\boldsymbol{e}_i$) and the origin, *i.e.,*

$$s(\boldsymbol{P}_{::k}) = \sum\nolimits_{i=1}^{d} D_{\boldsymbol{P}_{::k}}(\boldsymbol{e}_i, \boldsymbol{0}) = \sum\nolimits_{i=1}^{d} ||\boldsymbol{P}_{::k}\boldsymbol{e}_i - \boldsymbol{0}||_2^2, \tag{5}$$

where $\boldsymbol{e}_i$ is the i-th orthonormal base in $\mathbb{R}^d$. As shown in Fig. 2, the lengths of projected $\boldsymbol{e}_i$ $(i = 1, 2, \cdots, d)$ by different projection matrices are summed up to the same value so that the generated scales are perfectly aligned. Since Eq. (5) can be simplified to $s(\boldsymbol{P}_{::k}) = ||\boldsymbol{P}_{::k}||_F^2$, the TV regularizer $||\boldsymbol{s}(\boldsymbol{\mathcal{P}})||_{\text{tv}}$ can be easily tackled in the following optimization.

By combining the above empirical loss Eq. (4), sparse regularizer $||\boldsymbol{W}||_1$ and TV regularizer $s(\boldsymbol{P}_{::k})$, our DAML model is formally formulated as

$$\min_{\boldsymbol{\mathcal{P}}, \boldsymbol{W}} \mathcal{L}(\boldsymbol{\mathcal{P}}, \boldsymbol{W}) + \alpha||\boldsymbol{s}(\boldsymbol{\mathcal{P}})||_{\text{tv}} + \beta||\boldsymbol{W}||_1, \tag{6}$$

in which the TV regularizer $||\boldsymbol{s}(\boldsymbol{\mathcal{P}})||_{\text{tv}}$ facilitates the scale alignment and the l_1-norm regularizer $||\boldsymbol{W}||_1$ performs the sparse selection of candidate projection matrices for data-adaptive projections.

In test stage, given a new test data pair $(\boldsymbol{z}, \boldsymbol{z}')$ from the d-dimensional example space, we need to decide the weights for selecting candidate projection matrices, as the optimal weight matrix $\boldsymbol{W}$ (denoted $\boldsymbol{W}^*$) is merely learned for training data. Inspired by the regression mechanism in Low-Rank Representation (LRR) (Liu et al. 2013), here we employ the linear regression to predict the weight vector $\boldsymbol{w}_{\boldsymbol{z}} \in \mathbb{R}^c$ for a new data pair $(\boldsymbol{z}, \boldsymbol{z}')$, *i.e.,*

$$\boldsymbol{w}_{\boldsymbol{z}} = \boldsymbol{Q}^*(\boldsymbol{z} + \boldsymbol{z}'), \tag{7}$$

where $\boldsymbol{Q}^* \in \mathbb{R}^{c\times d}$ is learned from the linear regression by minimizing $||\boldsymbol{Q}\overline{\boldsymbol{X}} - \boldsymbol{W}^*||_F^2$, and $\overline{\boldsymbol{X}} = (\boldsymbol{x}_1 + \boldsymbol{x}'_1, \boldsymbol{x}_2 + \boldsymbol{x}'_2, \cdots, \boldsymbol{x}_n + \boldsymbol{x}'_n)$. Based on the predicted weight vector

$\boldsymbol{w_z}$, we know that the distance between $\boldsymbol{z}$ and $\boldsymbol{z}'$ equals to

$$D_{\boldsymbol{\mathcal{P}}^*}(\boldsymbol{z},\, \boldsymbol{z}') = ||\boldsymbol{P}^*(\boldsymbol{w_z})\boldsymbol{z} - \boldsymbol{P}^*(\boldsymbol{w_z})\boldsymbol{z}'||_2^2, \quad (8)$$

in which $\boldsymbol{\mathcal{P}}^*$ is learned by solving Eq. (6).

Kernelized DAML Model

In this part, we show that the linear DAML model proposed above can be easily extended to a kernelized form (denoted "KDAML") to handle the non-linear cases. A symmetric similarity function κ is a kernel (Bishop 2006) if there exists a (possibly implicit) mapping function $\boldsymbol{\phi}(\cdot) : \mathbb{X} \to \mathbb{H}$ from the instance space $\mathbb{X}$ to a Hilbert space $\mathbb{H}$ such that κ can be written as an inner product in $\mathbb{H}$, *i.e.,*

$$\kappa(\boldsymbol{x},\, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}'), \quad (9)$$

where $\boldsymbol{x}$ and $\boldsymbol{x}'$ are examples from the instance space $\mathbb{X}$.

To perform the kernel extension, we replace the examples $\boldsymbol{x}_i$ and $\boldsymbol{x}_i'$ with their feature mapping results $\boldsymbol{\phi}(\boldsymbol{x}_i)$ and $\boldsymbol{\phi}(\boldsymbol{x}_i')$, and thus the kernelized distance $D^\kappa_{\boldsymbol{\mathcal{P}}}(\boldsymbol{x}_i,\, \boldsymbol{x}_i')$ which follows Eq. (2) is written as

$$D^\kappa_{\boldsymbol{\mathcal{P}}}(\boldsymbol{x}_i,\, \boldsymbol{x}_i') = ||\boldsymbol{P}(\boldsymbol{w}_i)\boldsymbol{\phi}(\boldsymbol{x}_i) - \boldsymbol{P}(\boldsymbol{w}_i)\boldsymbol{\phi}(\boldsymbol{x}_i')||_2^2, \quad (10)$$

in which the mapping results $\boldsymbol{\phi}(\boldsymbol{x}_i)$, $\boldsymbol{\phi}(\boldsymbol{x}_i') \in \mathbb{R}^h$ and h is the dimensionality of the Hilbert space $\mathbb{H}$. Notice that in the above kernelized distance, the size of candidate matrices $\boldsymbol{P}_{::k}$ ($k = 0,\, 1,\, \cdots,\, c$) are increased to $r \times h$ rather than its original size of $r \times d$. Therefore, it is unrealistic to directly learn the parameters in $\boldsymbol{P}_{::k}$ within $\mathbb{R}^{r\times h}$, because the dimensionality of Hilbert space h is usually assumed to be very high or even infinite (Bishop 2006; Liu and Tsang 2017), which means that the large-scale matrix $\boldsymbol{P}_{::k}$ cannot be computed within the limited time. Therefore, according to (Weinberger and Tesauro 2007), we express the candidate matrix $\boldsymbol{P}_{::k}$ as the following form regarding the mapping results $\boldsymbol{\phi}(\boldsymbol{x}_i)$ ($i = 1, 2, \cdots, n$), and obtain

$$\boldsymbol{P}_{::k} = \boldsymbol{R}_{::k}\boldsymbol{\varphi}^\top, \quad (11)$$

where $\boldsymbol{R}_{::k} \in \mathbb{R}^{r\times n}$ and $\boldsymbol{\varphi} = (\boldsymbol{\phi}(\boldsymbol{x}_1), \boldsymbol{\phi}(\boldsymbol{x}_2), \cdots, \boldsymbol{\phi}(\boldsymbol{x}_n)) \in \mathbb{R}^{h\times n}$. After that, the problem has been transformed to learn $\boldsymbol{R}_{::0}, \boldsymbol{R}_{::1}, \cdots, \boldsymbol{R}_{::c}$, of which the sizes are independent with h, and thus the mathematical operations in the original high-dimensional Hilbert space is avoided. By further denoting the n-dimensional vectors as

$$\begin{cases} \boldsymbol{k}_i = (\kappa(\boldsymbol{x}_i, \boldsymbol{x}_1), \kappa(\boldsymbol{x}_i, \boldsymbol{x}_2), \cdots, \kappa(\boldsymbol{x}_i, \boldsymbol{x}_n))^\top, \\ \boldsymbol{k}_i' = (\kappa(\boldsymbol{x}_i', \boldsymbol{x}_1), \kappa(\boldsymbol{x}_i', \boldsymbol{x}_2), \cdots, \kappa(\boldsymbol{x}_i', \boldsymbol{x}_n))^\top, \end{cases} \quad (12)$$

we have

$$\begin{aligned} &D^\kappa_{\boldsymbol{\mathcal{R}}}(\boldsymbol{x}_i,\, \boldsymbol{x}_i') \\ &= (\boldsymbol{\phi}(\boldsymbol{x}_i) - \boldsymbol{\phi}(\boldsymbol{x}_i'))^\top (\boldsymbol{R}_{::0}\boldsymbol{\varphi}^\top + \textstyle\sum_{k=1}^c \boldsymbol{R}_{::k}\boldsymbol{\varphi}^\top)^\top \\ &\times (\boldsymbol{R}_{::0}\boldsymbol{\varphi}^\top + \textstyle\sum_{k=1}^c \boldsymbol{R}_{::k}\boldsymbol{\varphi}^\top)(\boldsymbol{\phi}(\boldsymbol{x}_i) - \boldsymbol{\phi}(\boldsymbol{x}_i')) \\ &= (\boldsymbol{\phi}(\boldsymbol{x}_i) - \boldsymbol{\phi}(\boldsymbol{x}_i'))^\top \boldsymbol{\varphi}\boldsymbol{R}(\boldsymbol{w}_i)^\top \boldsymbol{R}(\boldsymbol{w}_i)\boldsymbol{\varphi}^\top (\boldsymbol{\phi}(\boldsymbol{x}_i) - \boldsymbol{\phi}(\boldsymbol{x}_i')) \\ &= (\boldsymbol{k}_i - \boldsymbol{k}_i')^\top \boldsymbol{R}(\boldsymbol{w}_i)^\top \boldsymbol{R}(\boldsymbol{w}_i)(\boldsymbol{k}_i - \boldsymbol{k}_i'), \end{aligned} \quad (13)$$

in which the tensor $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{r\times n\times (c+1)}$ is stacked by the candidate matrices $\boldsymbol{R}_{::0}$, $\boldsymbol{R}_{::1}$, $\cdots$, $\boldsymbol{R}_{::c}$, and the matrix $\boldsymbol{R}(\boldsymbol{w}_i) \in \mathbb{R}^{r\times n}$ corresponds to the data-adaptive projection matrix $\boldsymbol{P}(\boldsymbol{w}_i)$ in Eq. (2). Compared with the linear DAML model, and additional step required by the kernelized DAML is that the vectors $\boldsymbol{k}_i$ and $\boldsymbol{k}_i'$ in Eqs. (12) and (13) should be pre-computed for the i-th pair. As long as the kernel $\kappa(\cdot)$ is specified, we may finally obtain the kernelized distance by Eq. (13). In this paper, we adopt the Gaussian kernel function as $\kappa(\cdot)$ due to its popularity and computational easiness. Since the distance formulation of KDAML (*i.e.* Eq. (13)) shares the equivalent mathematical expression with that of linear DAML (*i.e.* Eq. (2)), it can be directly solved via the same optimization algorithm as the linear DAML. The optimization process is detailed in the next section.

Optimization

For algorithm implementation, the loss function $l(D_i,\, y_i)$ in Eq. (4) can be squared loss, squared hinge loss or other popular formulations, which are continuous and have the derivative $l'_{D_i}(D_i,\, y_i)$ regarding ${D_i}$[2]. Then we employ the Iterative Shrinkage-Thresholding Algorithm (ISTA) (Rolfs et al. 2012) to solve our problem in Eq. (6). The general ISTA solves a continuous optimization problem with the form

$$\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) + g(\boldsymbol{\theta}), \quad (14)$$

where $\boldsymbol{\theta}$ is the optimization variable, $f(\boldsymbol{\theta})$ is derivable, and $g(\boldsymbol{\theta})$ is usually non-smooth. The solution to Eq. (14) can be found by iteratively optimizing the following function, namely

$$\begin{aligned} &\Gamma_L(\boldsymbol{\theta},\, \boldsymbol{\theta}^{(t)}) \\ &= f(\boldsymbol{\theta}^{(t)}) + (\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)})^\top \nabla f(\boldsymbol{\theta}^{(t)}) + \frac{L}{2}||\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)}||_2^2 + g(\boldsymbol{\theta}), \end{aligned} \quad (15)$$

where L is Lipschitz constant, and $\nabla f(\boldsymbol{\theta}^{(t)})$ computes the gradient of f on $\boldsymbol{\theta}^{(t)}$. Eq. (15) admits a unique minimizer, which is

$$\boldsymbol{\Pi}_L(\boldsymbol{\theta}^{(t)}) = \arg\min_{\boldsymbol{\theta}} g(\boldsymbol{\theta}) + \frac{L}{2}||\boldsymbol{\theta} - (\boldsymbol{\theta}^{(t)} - \frac{1}{L}\nabla f(\boldsymbol{\theta}^{(t)}))||_2^2, \quad (16)$$

where L is usually manually tuned to satisfy

$$f(\boldsymbol{\Pi}_L(\boldsymbol{\theta}^{(t)})) + g(\boldsymbol{\Pi}_L(\boldsymbol{\theta}^{(t)})) \le \Gamma_L(\boldsymbol{\Pi}_L(\boldsymbol{\theta}^{(t)}),\, \boldsymbol{\theta}^{(t)}). \quad (17)$$

To solve our model, we let $\boldsymbol{\theta} = (\boldsymbol{\mathcal{P}},\, \boldsymbol{W})$, so we have $f(\boldsymbol{\theta})$ and $g(\boldsymbol{\theta})$ in our model as

$$f(\boldsymbol{\mathcal{P}},\, \boldsymbol{W}) = \mathcal{L}(\boldsymbol{\mathcal{P}},\, \boldsymbol{W}) + \alpha||\boldsymbol{s}(\boldsymbol{\mathcal{P}})||_{\text{tv}}, \quad (18)$$

and

$$g(\boldsymbol{W}) = \beta||\boldsymbol{W}||_1. \quad (19)$$

Then in each iteration, we have to minimize the following function

$$\begin{aligned} &J(\boldsymbol{\mathcal{P}},\, \boldsymbol{W}) \\ &= \beta||\boldsymbol{W}||_1 + \frac{L}{2}||\boldsymbol{W} - (\boldsymbol{W}^{(t)} - \frac{1}{L}\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{\mathcal{P}}^{(t)},\, \boldsymbol{W}^{(t)}))||_F^2 \\ &+ \frac{L}{2}\left\|\boldsymbol{\mathcal{P}} - \left[\boldsymbol{\mathcal{P}}^{(t)} - (\frac{1}{L}(\nabla_{\boldsymbol{\mathcal{P}}}\mathcal{L}(\boldsymbol{\mathcal{P}}^{(t)}, \boldsymbol{W}^{(t)}) + \alpha\nabla_{\boldsymbol{\mathcal{P}}}||\boldsymbol{s}(\boldsymbol{\mathcal{P}})||_{\text{tv}})\right]\right\|_F^2, \end{aligned} \quad (20)$$

[2]Here the notation $D_{\boldsymbol{\mathcal{P}}}(\boldsymbol{x}_i,\, \boldsymbol{x}_i')$ is simplified as D_i for convenience.

in which[3]

$$\begin{cases}\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{\mathcal{P}}^{(t)},\boldsymbol{W}^{(t)})=\frac{1}{n}\sum_{i=1}^{n}l'_{D_i}(D_i,y_i)\nabla_{\boldsymbol{W}}D_i,\\ \nabla_{\boldsymbol{\mathcal{P}}}\mathcal{L}(\boldsymbol{\mathcal{P}}^{(t)},\boldsymbol{W}^{(t)})=\frac{1}{n}\sum_{i=1}^{n}l'_{D_i}(D_i,y_i)\nabla_{\boldsymbol{\mathcal{P}}}D_i.\end{cases} \tag{21}$$

To minimize above Eq. (20), here we provide the computation results of $\nabla_{\boldsymbol{W}}D_i$ and $\nabla_{\boldsymbol{\mathcal{P}}}D_i$ respectively. By using the chain rule of derivate, we can easily obtain that[4]

$$\begin{cases}\nabla_{\boldsymbol{w}_i}D_i=2(\boldsymbol{A}_{1:i}^{\top},\boldsymbol{A}_{2:i}^{\top},\cdots,\boldsymbol{A}_{c:i}^{\top})^{\top}\overline{\boldsymbol{w}}_i,\\ \nabla_{\boldsymbol{\mathcal{P}}_{::k}}D_i=2\sum_{j=0}^{c}\overline{w}_{ik}\overline{w}_{ij}\overline{\boldsymbol{x}}_i\overline{\boldsymbol{x}}_i^{\top}\boldsymbol{\mathcal{P}}_{::j}^{\top}.\end{cases} \tag{22}$$

Based on above results, the minimizer of Eq. (20) (*i.e.* the iteration rule) can be summarized as

$$\begin{cases}\boldsymbol{W}^{(t+1)}=\mathcal{T}_{\frac{\beta}{L}}(\boldsymbol{W}^{(t)}-\frac{1}{L}\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{\mathcal{P}}^{(t)},\boldsymbol{W}^{(t)})),\\ \boldsymbol{\mathcal{P}}^{(t+1)}=\boldsymbol{\mathcal{P}}^{(t)}-\frac{1}{L}(\nabla_{\boldsymbol{\mathcal{P}}}\mathcal{L}(\boldsymbol{\mathcal{P}}^{(t)},\boldsymbol{W}^{(t)})+\alpha\nabla_{\boldsymbol{\mathcal{P}}}||\boldsymbol{s}(\boldsymbol{\mathcal{P}})||_{\text{tv}},\end{cases} \tag{23}$$

in which the Soft Threshold Operator $\mathcal{T}_{\mu}(v)$ (Cai, Candès, and Shen 2010) is defined as

$$\mathcal{T}_{\mu}(v) \quad =\begin{cases}v-\mu, & \text{if } v>\mu,\\ v+\mu, & \text{if } v<-\mu,\\ 0, & \text{otherwise.}\end{cases} \tag{24}$$

We summarize the training phase of DAML in Algorithm 1, where Eq. (23) is denoted as

$$(\boldsymbol{\mathcal{P}}^{(t+1)},\boldsymbol{W}^{(t+1)})=\boldsymbol{\Pi}_L(\boldsymbol{\mathcal{P}}^{(t)},\boldsymbol{W}^{(t)}). \tag{25}$$

Based on the output of Algorithm 1, the testing steps for a new data pair are described in Algorithm 2. Since KDAML has the equivalent mathematical expression with linear DAML, the training procedure (Algorithm 1) and testing steps (Algorithm 2) are directly applicable to KDAML.

Finally, we want to explain the convergence of Algorithm 1. Although the traditional ISTA is designed for convex optimization, some extended convergence proofs for non-convex problems have been provided in the prior works such as (Cui 2018). Therefore, although the objective function of our model is non-convex, the adopted optimization process is still theoretically guaranteed to converge to a stationary point.

Experiments

In this section, intensive empirical investigations are conducted to validate the effectiveness of our proposed method. In detail, we compare the performance of the DAML and KDAML models with: 1) the classical linear metric learning methods ITML (Davis et al. 2007) and LMNN (Weinberger, Blitzer, and Saul 2006); 2) the DNN based metric learning method DDML (Hu, Lu, and Tan 2014); and 3) state-of-the-art metric learning methods GMDRML (Luo and Huang 2018), LGMML (Su, King, and Lyu 2017), and AML (Chen et al. 2018). All methods are evaluated on retrieval, classification and verification tasks. For the compared methods, we follow the authors' suggestions to choose the optimal parameters. For the tuning parameters, c is fixed to 10 while α, β and λ are all tuned by searching the grid $\{0, 0.2, 0.4, \cdots, 2\}$ to get the best performances. We follow ITML and use the squared hinge loss as $l(D_{\boldsymbol{\mathcal{P}}}(\boldsymbol{x}_i,\boldsymbol{x}'_i),y_i)$ in Eq. (6) for our objective function. The Gaussian kernel function (Bishop 2006) is employed for implementing KDAML.

[3] $\nabla_{\boldsymbol{\mathcal{P}}_{::k}}||\boldsymbol{s}(\boldsymbol{\mathcal{P}})||_{\text{tv}}=2\text{sign}(||\boldsymbol{\mathcal{P}}_{::k}||_F^2-||\boldsymbol{\mathcal{P}}_{::k+1}||_F^2)\boldsymbol{\mathcal{P}}_{::k}-2\text{sign}(||\boldsymbol{\mathcal{P}}_{::k}||_F^2-||\boldsymbol{\mathcal{P}}_{::k-1}||_F^2)\boldsymbol{\mathcal{P}}_{::k}$.

[4] Here $\overline{\boldsymbol{w}}_i=(1,\lambda w_1,\lambda w_2,\cdots,\lambda w_c)^{\top}$ and $\overline{\boldsymbol{x}}_i=\boldsymbol{x}_i-\boldsymbol{x}'_i$, respectively. The element A_{kji} in the tensor $\boldsymbol{\mathcal{A}}\in\mathbb{R}^{(c+1)\times(c+1)\times n}$ equals to $\boldsymbol{x}_i^{\top}\boldsymbol{\mathcal{P}}_{::k}\boldsymbol{\mathcal{P}}_{::j}\boldsymbol{x}_i+\boldsymbol{x}_i'^{\top}\boldsymbol{\mathcal{P}}_{::k}\boldsymbol{\mathcal{P}}_{::j}\boldsymbol{x}'_i-2\boldsymbol{x}_i^{\top}\boldsymbol{\mathcal{P}}_{::k}\boldsymbol{\mathcal{P}}_{::j}\boldsymbol{x}'_i$.

Algorithm 1 Solving Eq. (6) via ISTA.

Input: Training data pairs $\mathcal{X}=\{(\boldsymbol{x}_i,\boldsymbol{x}'_i)|1\le i\le n\}$; labels $\boldsymbol{y}\in\{0,1\}^n$; parameters α, β, λ.
Initialize: $t=1$; $L_0=1$; $\eta>1$; $\boldsymbol{W}^{(0)}=\boldsymbol{0}$; $\boldsymbol{\mathcal{P}}^{(0)}=\boldsymbol{0}$.
Repeat:

1). Find the smallest nonnegative integers i_t such that with $\widehat{L}=\eta^{i_t}L^{(t-1)}$

$$f(\boldsymbol{\Pi}_{\widehat{L}}(\boldsymbol{\mathcal{P}}^{(t)},\boldsymbol{W}^{(t)}))+g(\boldsymbol{\Pi}_{\widehat{L}}(\boldsymbol{\mathcal{P}}^{(t)},\boldsymbol{W}^{(t)}))\le\Gamma_{\widehat{L}}(\boldsymbol{\Pi}_{\widehat{L}}(\boldsymbol{\mathcal{P}}^{(t)},\boldsymbol{W}^{(t)}),(\boldsymbol{\mathcal{P}}^{(t)},\boldsymbol{W}^{(t)})).$$

2). Set $L^{(t)}=\widehat{L}$ and use Eq. (23) to update

$$(\boldsymbol{\mathcal{P}}^{(t+1)},\boldsymbol{W}^{(t+1)})=\boldsymbol{\Pi}_{L^{(t)}}(\boldsymbol{\mathcal{P}}^{(t)},\boldsymbol{W}^{(t)}).$$

3). Update $t=t+1$.

Until Convergence.
Output: The converged $\boldsymbol{\mathcal{P}}$ and $\boldsymbol{W}$.

Algorithm 2 Distance Computation for New Test Data Pair.

Input: Test data pair $(\boldsymbol{z},\boldsymbol{z}')$; the learnd projection tensor $\boldsymbol{\mathcal{P}}^*$ and selection weights $\boldsymbol{W}^*$.
Initialize: Regression matrix $\boldsymbol{Q}^*=(\overline{\boldsymbol{X}}\overline{\boldsymbol{X}}^{\top})^{-1}\boldsymbol{W}^*(\overline{\boldsymbol{X}})^{\top}$.
Procedure:

1). Predict the weights

$$\boldsymbol{w}_z=\boldsymbol{Q}^*(\boldsymbol{z}+\boldsymbol{z}').$$

2). Compute the distance

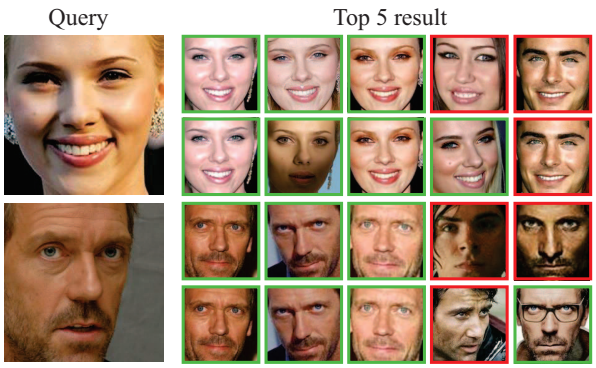
$$D_{\boldsymbol{\mathcal{P}}^*}(\boldsymbol{z},\boldsymbol{z}')=||\boldsymbol{P}^*(\boldsymbol{w}_{\boldsymbol{z}})\boldsymbol{z}-\boldsymbol{P}^*(\boldsymbol{w}_{\boldsymbol{z}})\boldsymbol{z}'||_2^2.$$

End.
Output: The predicted distance $D_{\boldsymbol{\mathcal{P}}^*}(\boldsymbol{z},\boldsymbol{z}')$.

Experiments on Retrieval

Retrieval is one of the most typical applications of metric learning, which aims to search the most similar instances for a query instance (Zhan et al. 2016). In our experiments, we use the *PubFig* face image dataset (Nair and Hinton 2010) and the *Outdoor Scene Recognition* (*OSR*) dataset (Parikh and Grauman 2011) to evaluate the capabilities of all compared methods.

In the first experiment, we use the cropped *PubFig* face image dataset, which includes 771 images from 8 individuals. We follow the experimental setting in (Luo and Huang 2018) and use a 512-dimensional Dense Scale Invariant Feature Transform (DSIFT) features (Cheung and Hamarneh 2009) to represent each image. This experiment is run 5 times, where 30 images per person are randomly selected each time as the training data. Fig. 3(a) shows the retrieval



(a) Results of 5 nearest neighbors based on queried images on Pubfig dataset. For each queried image, the first row shows the results of LGMML, and the second row presents the results of our method.

(b) Results of 5 nearest neighbors based on queried images on OSR dataset. For each queried image, the first row shows the results of LGMML, and the second row presents the results of our method.

Figure 3: Results of 5 nearest neighbors on *PubFig* and *OSR* datasets for image retrieval. The green box means that the retrieval result is correct, and the red box denotes that the retrieval result is incorrect.

Table 1: The classification error rates (%) of the top 5 retrieval results predicted by all methods on the *PubFig* and *OSR* datasets. The best two results in each dataset are highlighted in red and blue, respectively. Notation "•" indicates that DAML and KDAML are significantly better than the best baseline method.

Datasets	ITML (Davis et al. 2007)	LMNN (Wein. et al. 2006)	GMDRML (Luo et al. 2018)	DDML (Hu et al. 2014)	LGMML (Su et al. 2017)	AML (Ch. et al. 2018)	DAML (Ours)	KDAML (Ours)	t-test
PubFig	13.89 ± 0.12	13.65 ± 0.05	13.22 ± 0.34	13.23 ± 0.07	13.15 ± 0.14	13.89 ± 0.11	12.95 ± 0.09	12.89 ± 0.11	•
OSR	24.59 ± 0.11	24.68 ± 0.13	23.51 ± 0.24	23.61 ± 0.17	24.39 ± 0.14	24.32 ± 0.21	23.41 ± 0.12	23.18 ± 0.22	•

results of two queries, and the average classification error rates of top 5 retrieval results are presented in Tab. 1. It can be seen that our proposed DAML and KDAML achieve the lowest error rates, *i.e.,* 12.95% and 12.89%, respectively. Moreover, the DNN based method DDML and local method LGMML also yield good performances, but they are still slightly worse than our DAML and KDAML models are revealed in Tab. 1.

The second experiment is performed on the *OSR* dataset. It includes 2688 images from 8 scene categories, which are described by the high-level attribute features (Huo, Nie, and Huang 2016). We also use 30 images for each category as training data, and the other images are used as test data. We repeat this procedure 5 times and use the average error rate to evaluate all methods in Tab. 1. It can be found that our methods DAML and KDAML still obtain the best results 23.41% and 23.18% among all comparators. Fig. 3(b) shows the retrieval results of two queries, and it is clear that our method learns a better distance metric than the traditional Mahalanobis distance. Notably, we find that it is quite difficult to distinguish "mountain" and "tall-building" when their images contain blue sky background. Therefore, LGMML fails to consistently return the correct results (see the first row), while our method can still render the satisfactory results. We also perform the t-test (significance level 0.05) to validates the superiority of our method to the best baseline method.

Experiments on Classification

To evaluate the performances of various compared methods on classification task, we follow the existing work (Ye et al. 2017) and adopt the k-NN classifier ($k = 5$) based on the learned metrics to investigate the classification error rates of various methods. The datasets are from the well-known UCI repository (Asuncion and Newman 2007), which include *MNIST, Autompg*, *Sonar*, *Australia*, *Balance*, *Isolet*, and *Letters*. We compare all methods over 20 random trials. In each trial, 80% of examples are randomly selected as the training examples, and the rest are used for testing. By following the recommendation in (Zadeh, Hosseini, and Sra 2016), the training pairs are generated by randomly picking up $1000c(c-1)$ pairs among the training examples. The average classification error rates of compared methods are showed in Tab. 2, and we find that DAML and KDAML obtain the best results. It can be noted that KDAML is generally better than DAML, as the kernel mapping improves the non-linear ability of our model.

Among the compared methods, it can be noticed that the main difference between ITML and our DAML is the form of their projection matrices, in which ITML employs single fixed projection matrix while our DAML utilizes multiple candidate matrices to generate the data-adaptive projection matrices. From the experimental results in Tab. 2), we can clearly observe that DAML is able to obtain significantly better performances than ITML, and therefore the proposed data-adaptive projection matrix is indeed useful to enhance the accuracy of the learned metric.

Experiments on Verification

We use two face datasets and one image matching dataset to evaluate the capabilities of all compared methods on image verification. For the *PubFig* face dataset (as described before), the first 80% data pairs are selected for training and the rest are used for testing. Similar experiments are performed

Table 2: The classification error rates (%) of all methods on the *MNIST, Autompg*, *Sonar*, *German-Credit*, *Balance*, *Isolet* and *Letters* datasets. The best two results in each dataset are highlighted in red and blue, respectively. Notation "•" indicates that DAML and KDAML are significantly better than the best baseline method.

Datasets	ITML (Davis et al. 2007)	LMNN (Wein. et al. 2006)	GMDRML (Luo et al. 2018)	DDML (Hu et al. 2014)	LGMML (Su et al. 2017)	AML (Ch. et al. 2018)	DAML (Ours)	KDAML (Ours)	t-test
MNIST	14.31 ± 2.32	17.46 ± 5.32	12.21 ± 0.82	11.56 ± 0.07	12.19 ± 0.14	11.52 ± 0.27	11.34 ± 0.12	11.12 ± 0.21	•
Autompg	26.62 ± 3.21	25.92 ± 3.32	24.32 ± 2.71	23.95 ± 1.52	23.56 ± 1.41	25.31 ± 3.22	23.95 ± 5.21	21.45 ± 1.21	•
Sonar	17.02 ± 3.52	16.04 ± 5.31	17.12 ± 5.64	15.31 ± 2.56	21.35 ± 4.01	16.53 ± 3.51	15.64 ± 1.64	15.55 ± 3.65	
Australia	17.52 ± 2.13	15.51 ± 2.53	14.12 ± 2.04	15.12 ± 5.23	18.35 ± 1.84	12.95 ± 2.27	13.17 ± 1.97	13.11 ± 2.01	•
Balance	9.31 ± 2.21	9.93 ± 1.62	6.56 ± 2.14	8.12 ± 1.97	7.36 ± 2.14	7.98 ± 2.11	6.21 ± 0.12	6.15 ± 0.17	•
IsoLet	9.23 ± 1.12	3.23 ± 1.23	3.12 ± 0.34	2.68 ± 0.71	3.91 ± 1.14	2.91 ± 2.17	2.98 ± 1.15	2.79 ± 1.32	
Letters	6.24 ± 0.23	4.21 ± 2.05	3.54 ± 1.22	5.14 ± 1.04	4.56 ± 2.45	3.22 ± 1.23	3.11 ± 1.05	3.01 ± 1.23	•

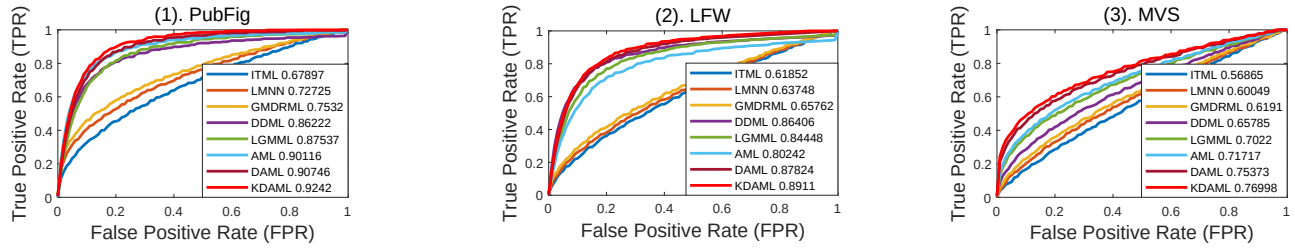


Figure 4: ROC curves of different methods on (a) *PubFig*, (b) *LFW* and (c) *MVS* datasets. AUC values are presented in the legends.

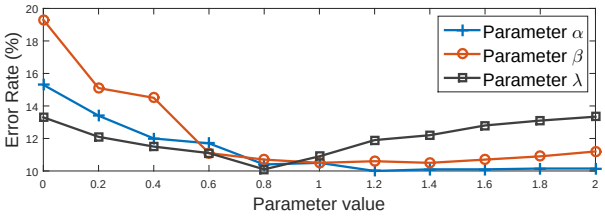


Figure 5: Parametric Sensitivities on *MNIST* dataset. The test error rate is reported by changing one of α, β and γ, and keeping the remaining two parameteres as constants.

on the *LFW* face dataset (Huo, Nie, and Huang 2016) which includes 13233 unconstrained face images of 5749 individuals. The image matching dataset *MVS* (Brown, Hua, and Winder 2011) consists of over 3×10^4 gray-scale images sampled from 3D reconstructions of the Statue of Liberty (LY), Notre Dame (ND) and Half Dome in Yosemite (YO). By following the settings in (Zagoruyko and Komodakis 2015), LY and ND are put together to form a training set with over 10^5 image patch pairs, and 10^4 patch pairs in YO are used for testing. The adopted features are extracted by DSIFT (Cheung and Hamarneh 2009) and Siamese-CNN (Zagoruyko and Komodakis 2015) for face datasets (*i.e. PubFig* and *LFW*) and image patch dataset (*i.e. MVC*), respectively. We plot the Receiver Operator Characteristic (ROC) curve by changing the thresholds of different distance metrics. Then the values of Area Under Curve (AUC) are calculated to quantitatively evaluate the performances of all comparators. From the ROC curves and AUC values in Fig. 4, it is clear to see that DAML and KDAML consistently outperform other methods.

Parametric Sensitivity

In our proposed DAML, there are three parameters which might influence the model performance. The parameters λ in Eq. (3) determines the importance of the linearly combined projection matrix. Parameters α and β in Eq. (6) respectively control the amount and the differences of selected matrices.

As shown in Fig. 5, the parameters α and β render the lowest test errors within the range $(0.8, 1.8)$. It means that the sparse regularizer and TV regularizer in Eq. (6) are indeed necessary for model performances. We may also find that the performances are relatively stable in such a range, and thus the two parameters can be easily tuned for practical use. When the parameter λ increases within $(0, 0.8)$, the prediction result graduatelly becomes accurate, because the selection and combination of candidates begin to take effects. Moreover, we notice that it is not wise to use a very large value for λ, because such a setting intrinsically discards the shared primitive projection matrix $\boldsymbol{P}_{::0}$, which is also useful to extract the shared features in the data.

Conclusion

In this paper, we propose a metric learning framework named Data-Adaptive Metric Learning (DAML), which generalizes the single global projection matrix of traditional Mahalanobis distance to a local data-adaptive form. The proposed projection matrix is combined by a series of weighted candidate matrices for a specific data pair, in which the l_1-norm is employed to sparsely select a small subset of candidates to form the suitable combination. Meanwhile, a TV regularizer is utilized to align the produced scales of candidates so that the over-fitting caused by the arbitrary scale variations can be avoided. Furthermore, we show that such a linear data-adaptive metric can be easily kernelized to handle the non-linear cases. The experimental results on various tasks show that the proposed DAML is able to flexibly discover the local data property and acquire more reliable and precise metric than the state-of-the-art metric learning methods. Since the proposed DAML framework is general in nature, it is promising to apply DAML to more manifold and DNN based metric learning models for the future works.

Acknowledgments

The authors would like to thank the anonymous reviewers for their critical and constructive comments and suggestions. This work was supported by the National Science Fund (NSF) of China under Grant Nos. U1713208, 61472187 and 61602246, Program for Changjiang Scholars, NSF of Jiangsu Province (No: BK20171430), the Fundamental Research Funds for the Central Universities (No: 30918011319), the "Summit of the Six Top Talents" Program (No: DZXX-027), the "CAST Lift Program for Young Talents", and the "Outstanding PhD of NJUST" Program (No: AE88902).

References

Asuncion, A., and Newman, D. 2007. Uci machine learning repository.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning*. springer.

Bohné, J.; Ying, Y.; Gentric, S.; and Pontil, M. 2014. Large margin local metric learning. In *ECCV*, 679–694. Springer.

Brown, M.; Hua, G.; and Winder, S. 2011. Discriminative learning of local image descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33(1):43–57.

Cai, J.-F.; Candès, E. J.; and Shen, Z. 2010. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization* 20(4):1956–1982.

Chen, S.; Gong, C.; Yang, J.; Li, X.; Wei, Y.; and Li, J. 2018. Adversarial metric learning. In *IJCAI*, 123–131.

Cheung, W., and Hamarneh, G. 2009. n-sift: n-dimensional scale invariant feature transform. *IEEE Transactions on Image Processing* 18(9):2012–2021.

Cui, A. 2018. Iterative thresholding algorithm based on nonconvex method for modified lp-norm regularization minimization. *arXiv preprint arXiv:1804.09385*.

Davis, J. V.; Kulis, B.; Jain, P.; Sra, S.; and Dhillon, I. S. 2007. Information-theoretic metric learning. In *ICML*.

Duan, Y.; Zheng, W.; Lin, X.; Lu, J.; and Zhou, J. 2018. Deep adversarial metric learning. In *CVPR*, 2780–2789.

Harandi, M.; Salzmann, M.; and Hartley, R. 2017. Joint dimensionality reduction and metric learning: A geometric take. In *ICML*, 1943–1950.

Hu, J.; Lu, J.; and Tan, Y.-P. 2014. Discriminative deep metric learning for face verification in the wild. In *CVPR*.

Huang, Z.; Wang, R.; Shan, S.; and Chen, X. 2015. Projection metric learning on grassmann manifold with application to video based face recognition. In *CVPR*, 140–149.

Huo, Z.; Nie, F.; and Huang, H. 2016. Robust and effective metric learning using capped trace norm. In *KDD*.

Liu, W., and Tsang, I. W. 2017. Making decision trees feasible in ultrahigh feature and label dimensions. *The Journal of Machine Learning Research*.

Liu, G.; Lin, Z.; Yan, S.; Sun, J.; Yu, Y.; and Ma, Y. 2013. Robust recovery of subspace structures by low-rank representation. *IEEE transactions on pattern analysis and machine intelligence*.

Liu, W.; Xu, D.; Tsang, I.; and Zhang, W. 2018. Metric learning for multi-output tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Liu, W.; Tsang, I. W.; and Müller, K.-R. 2017. An easy-to-hard learning paradigm for multiple classes and multiple labels. *The Journal of Machine Learning Research*.

Luo, L., and Huang, H. 2018. Matrix variate gaussian mixture distribution steered robust metric learning. In *AAAI*.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*, 807–814.

Oh Song, H.; Xiang, Y.; Jegelka, S.; and Savarese, S. 2016. Deep metric learning via lifted structured feature embedding. In *CVPR*, 4004–4012.

Parikh, D., and Grauman, K. 2011. Relative attributes. In *ICCV*.

Rolfs, B.; Rajaratnam, B.; Guillot, D.; Wong, I.; and Maleki, A. 2012. Iterative thresholding algorithm for sparse inverse covariance estimation. In *NIPS*, 1574–1582.

Si, J.; Zhang, H.; Li, C.-G.; Kuen, J.; Kong, X.; Kot, A. C.; and Wang, G. 2018. Dual attention matching network for context-aware feature sequence based person reidentification. In *CVPR*, 132–141.

Su, Y.; King, I.; and Lyu, M. 2017. Learning to rank using localized geometric mean metrics. In *SIGIR*, 233–241.

Varior, R. R.; Haloi, M.; and Wang, G. 2016. Gated siamese convolutional neural network architecture for human re-identification. In *ECCV*, 791–808. Springer.

Wang, J.; Kalousis, A.; and Woznica, A. 2012. Parametric local metric learning for nearest neighbor classification. In *NIPS*, 1601–1609.

Weinberger, K. Q., and Saul, L. K. 2008. Fast solvers and efficient implementations for distance metric learning. In *ICML*, 1160–1167. ACM.

Weinberger, K. Q., and Tesauro, G. 2007. Metric learning for kernel regression. In *Artificial Intelligence and Statistics*.

Weinberger, K. Q.; Blitzer, J.; and Saul, L. K. 2006. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, 1473–1480.

Xing, E. P.; Jordan, M. I.; Russell, S. J.; and Ng, A. Y. 2003. Distance metric learning with application to clustering with side-information. In *NIPS*, 521–528.

Ye, H.-J.; Zhan, D.-C.; Si, X.-M.; and Jiang, Y. 2017. Learning mahalanobis distance metric: Considering instance disturbance helps. In *IJCAI*, 866–872.

Zadeh, P.; Hosseini, R.; and Sra, S. 2016. Geometric mean metric learning. In *ICML*, 2464–2471.

Zagoruyko, S., and Komodakis, N. 2015. Learning to compare image patches via convolutional neural networks. In *ICCV*, 4353–4361.

Zhan, D.-C.; Li, M.; Li, Y.; and Zhou, Z. 2009. Learning instance specific distances using metric propagation. In *ICML*.

Zhan, D.-C.; Hu, P.; Chu, Z.; and Zhou, Z.-H. 2016. Learning expected hitting time distance. In *AAAI*, 2309–2314.