Pattern Recognition 64 (2017) 215–225

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Supervised distance metric learning through maximization of the Jeffrey divergence

Bac Nguyen[a,*], Carlos Morell[b], Bernard De Baets[a]

[a] KERMIT, Department of Mathematical Modelling, Statistics and Bioinformatics, Ghent University, Coupure links 653, 9000 Ghent, Belgium
[b] Computer Science Department, Universidad Central "Marta Abreu" de Las Villas, Santa Clara, Cuba

ARTICLE INFO

Keywords:
Distance metric learning
Nearest neighbor
Linear transformation
Jeffrey divergence

ABSTRACT

Over the past decades, distance metric learning has attracted a lot of interest in machine learning and related fields. In this work, we propose an optimization framework for distance metric learning via linear transformations by maximizing the Jeffrey divergence between two multivariate Gaussian distributions derived from local pairwise constraints. In our method, the distance metric is trained on positive and negative difference spaces, which are built from the neighborhood of each training instance, so that the local discriminative information is preserved. We show how to solve this problem with a closed-form solution rather than using tedious optimization procedures. The solution is easy to implement, and tractable for large-scale problems. Experimental results are presented for both a linear and a kernelized version of the proposed method for k-nearest neighbors classification. We obtain classification accuracies superior to the state-of-the-art distance metric learning methods in several cases while being competitive in others.

1. Introduction

Distance metric learning consists in adapting a distance metric using information contained in the training data. The resulting distance metric is used to improve the performance of metric-based methods, such as k-nearest neighbors classification (k-NN) [1], or k-means clustering [2]. Depending on the problem of interest, an appropriate distance metric can yield substantial improvements over the commonly used Euclidean distance metric [3,4]. Learning a good distance metric for a specific problem may be the key to the successful application of metric-based methods. For this reason, distance metric learning plays a crucial role in metric-related pattern recognition tasks (see recent surveys [5,6]), such as classification [3,7], regression [8], clustering [4,9,10], feature selection [11,12], and ranking [13]. Depending on the availability of training instances, distance metric learning methods can be divided into three categories: supervised, semi-supervised, and unsupervised distance metric learning. Supervised methods for classification use the heuristic that instances belonging to the same class should be close to each other, while those from different classes should be farther apart [3,14]. Semi-supervised methods for information retrieval and clustering use the information in the form of pairwise similarity or dissimilarity constraints [9,15]. Unsupervised methods learn a distance metric that preserves the geometric relationships (i.e., distance) between most of the training data for the purpose of unsupervised dimensionality reduction [16,17].

In a supervised setting, we focus on the Mahalanobis distance metric due to its wide use in many application domains and because it provides a flexible way of learning an appropriate distance metric for complex problems [5,6]. The Mahalanobis distance metric is parametrized by a symmetric positive semidefinite matrix $\mathbf{M} \in \mathbb{R}^{D\times D}$, where the distance between two points $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^D$ is computed as

$$d_{\mathbf{M}}(\mathbf{u}, \mathbf{v}) = \sqrt{(\mathbf{u}-\mathbf{v})^{\mathrm{T}}\mathbf{M}(\mathbf{u}-\mathbf{v})}.$$

Since the matrix $\mathbf{M}$ is symmetric positive semidefinite, it can be factorized as $\mathbf{M} = \mathbf{A}\mathbf{A}^{\mathrm{T}}$, where $\mathbf{A} \in \mathbb{R}^{D\times m}$ and $m \le D$. Thus, the Mahalanobis distance between $\mathbf{u}$ and $\mathbf{v}$ is equal to the Euclidean distance between $\mathbf{A}^{\mathrm{T}}\mathbf{u}$ and $\mathbf{A}^{\mathrm{T}}\mathbf{v}$. In other words, learning a Mahalanobis distance metric is equivalent to learning a linear transformation.

We begin by introducing a simple two-class classification problem that motivates the key ideas in the proposed method. For this purpose, we construct a two-dimensional data set, containing 100 positive instances and 100 negative instances (see Fig. 1(a)). Both positive instances and negative instances follow a Gaussian distribution with means $\mu_1 = (-1.250; 0.205)$ and $\mu_2 = (0.60; 0.07)$, respectively, and the same covariance matrix $\Sigma = (1.96, -0.55; -0.55, 0.16)$. The training accuracy of 5-NN using the Euclidean distance metric on this data set is very poor, only 64.0%. However, this performance can be dramatically improved by applying a linear transformation to the

* Corresponding author.
E-mail addresses: bac.nguyencong@ugent.be (B. Nguyen), cmorellp@uclv.edu.cu (C. Morell), bernard.debaets@ugent.be (B. De Baets).

http://dx.doi.org/10.1016/j.patcog.2016.11.010
Received 20 May 2016; Received in revised form 24 August 2016; Accepted 13 November 2016
Available online 16 November 2016
0031-3203/

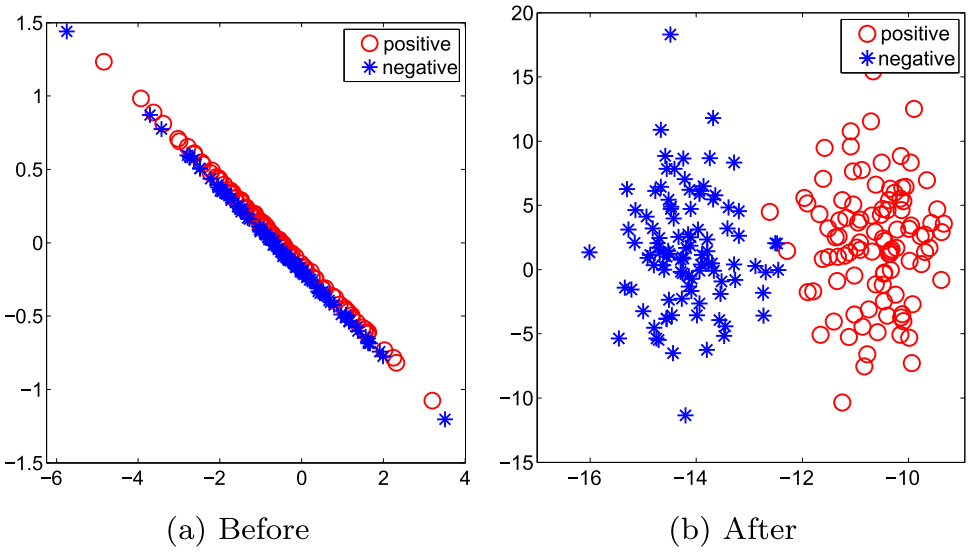


Fig. 1. A synthetic data set illustrating the poor performance of the k-NN classifier using the Euclidean distance metric. The data set consists of 200 instances drawn from two aligned strips, each defining a different class. The red circles denote positive instances, whereas the blue asterisks denote negative instances. (a) Data set before applying the linear transformation and (b) data set after applying the linear transformation. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

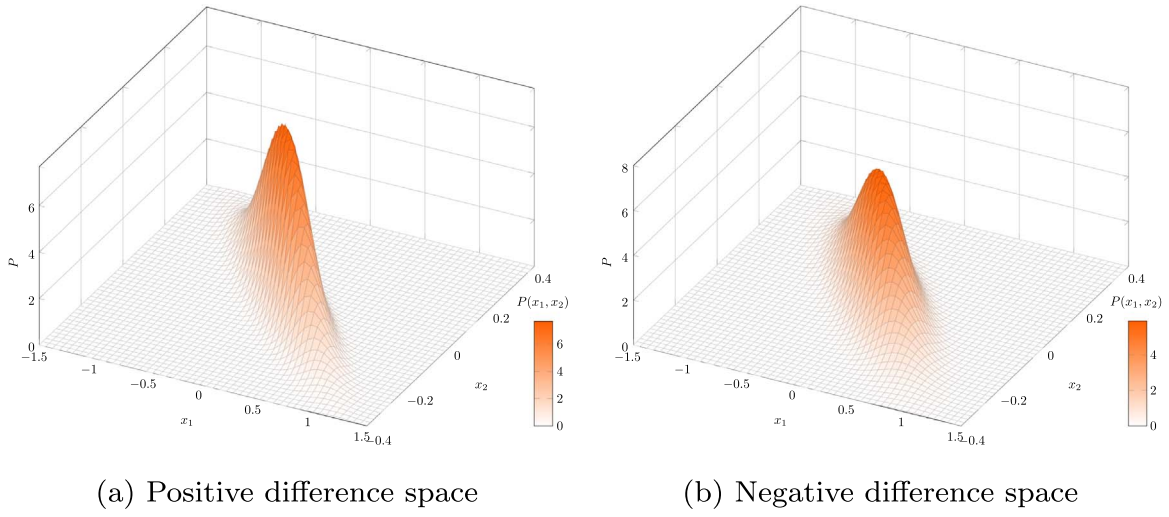


Fig. 2. Visualization of the probability density functions of the difference spaces before applying the linear transformation.

original data. In particular, using our method (as we will describe later) we obtain the linear transformation

$$\mathbf{A} = \begin{pmatrix} 20.11 & -3.02 \\ 70.22 & 0.63 \end{pmatrix},$$

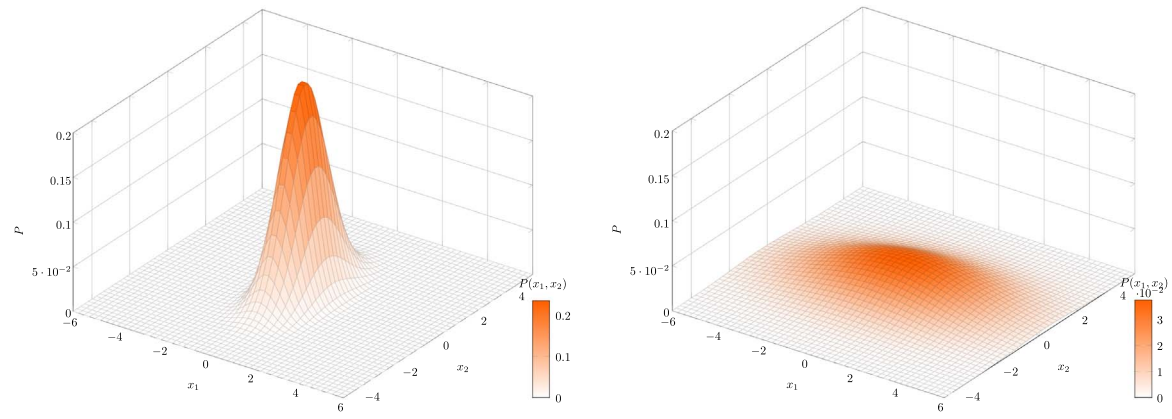
and consequently, the training accuracy is increased to 97.5% (see the resulting transformed data in Fig. 1(b)).

The key question is how to find such a linear transformation $\mathbf{A}$ (or, equivalently, the corresponding Mahalanobis matrix $\mathbf{M}$). Some insights can be obtained when carefully observing how the differences are distributed. Let us informally define the positive (resp. negative) *difference space* as the set of all differences $(\mathbf{x}_i - \mathbf{x}_j)$ between an instance $\mathbf{x}_i$ and its nearest neighbors $\mathbf{x}_j$ from the same (resp. different) class (see Section 2 for the formal definitions). Here, we use five nearest neighbors with the same class label and five nearest neighbors with different class labels for each training instance. Fig. 2 shows the probability density function[1] of data belonging to the positive (Fig. 2(a)) and negative (Fig. 2(b)) difference spaces. It allows us to see how the differences are distributed before applying the linear transformation.

There is a slight difference between these two distributions. However, this difference clearly reveals itself after applying the linear transformation specified by $\mathbf{A}$ (see Fig. 3). Note that our illustration here is based on k=5, but the same phenomenon occurs for other values of k. This particular example suggests a way to find such linear transformation, namely the one that maximizes the difference between these two distributions. The intuition is based on a two-class classification problem, however, it can be also used for multi-class classification problems since the difference spaces are built independently for any number of classes. In the rest of this paper, we develop this idea. In short, our main contributions are the following.

(i) We propose a novel distance metric learning method aimed at finding a linear transformation that maximizes the Jeffrey divergence between two multivariate Gaussian distributions derived from local pairwise constraints. We formulate this task as an unconstrained optimization problem and show that it can be

[1] For illustrative purposes, we use maximum likelihood to estimate the probability density function assuming that the data are normally distributed.



(a) Positive difference space (b) Negative difference space

Fig. 3. Visualization of the probability density functions of the difference spaces after applying the linear transformation.

solved analytically (Section 3.1).

(ii) While the proposed method is limited to learn a global linear transformation, we extend it into a kernelized version to tackle non-linear problems. We show that the kernelized version of the proposed method is more efficient and highly flexible by using the "kernel trick" (Section 3.2).

(iii) The resulting distance metric, when used in conjunction with k-NN, leads to significant improvements in the classification accuracy. We provide an extensive experimental validation to support this claim (Section 5). Several state-of-the-art distance metric learning methods (Section 4) have been used for a fair comparison.

2. Notations

The following notations are used throughout the paper. Vectors are denoted by boldface lowercase letters, such as $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$. Matrices are denoted by boldface capital letters, such as $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. The inner product between two vectors $\mathbf{u}$ and $\mathbf{v}$ is denoted as $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^\top \mathbf{v}$. All scalars are denoted by lowercase or uppercase letters, such as k, n, or D. Sets are denoted by calligraphic uppercase letters, such as $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{V}$. The trace of a matrix $\mathbf{A}$ is denoted as $\mathrm{tr}(\mathbf{A})$. The multivariate Gaussian density function with mean μ and covariance matrix Σ is denoted as $\mathcal{N}(\mu, \Sigma)$.

Let $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^D$ be the training set that contains n labeled instances with corresponding class labels $\mathcal{Y} = \{y_i\}_{i=1}^n$. We first introduce some definitions in order to develop our proposal.

Definition 1 (*k-Positive neighborhood*). Let $k \in \mathbb{N}$ such that $k \geq 1$. The k-positive neighborhood of $\mathbf{x}_i \in \mathcal{X}$ is the set $\mathcal{V}_k^+(\mathbf{x}_i)$ consisting of the k nearest neighbors of $\mathbf{x}_i$ in the input space $\mathcal{X} \setminus \{\mathbf{x}_i\}$, whose class label is equal to y_i.

Definition 2 (*k-Negative neighborhood*). Let $k \in \mathbb{N}$ such that $k \geq 1$. The k-negative neighborhood of $\mathbf{x}_i \in \mathcal{X}$ is the set $\mathcal{V}_k^-(\mathbf{x}_i)$ consisting of the k nearest neighbors of $\mathbf{x}_i$ in the input space $\mathcal{X}$, whose class label is not equal to y_i.

The set of all possible differences for any instance $\mathbf{x}_i$ and its k-positive neighborhood is called the k-positive difference space. The set of all possible differences for any instance $\mathbf{x}_i$ and its k-negative neighborhood is called the k-negative difference space. They are formally defined hereafter.

Definition 3 (*k-Positive difference space*). The k-positive difference space is the following set:

$$\mathcal{S} = \{\mathbf{x}_i - \mathbf{x}_j | \mathbf{x}_i \in \mathcal{X} \text{ and } \mathbf{x}_j \in \mathcal{V}_k^+(\mathbf{x}_i)\}.$$

Definition 4 (*k-Negative difference space*). The k-negative difference space is the following set:

$$\mathcal{D} = \{\mathbf{x}_i - \mathbf{x}_j | \mathbf{x}_i \in \mathcal{X} \text{ and } \mathbf{x}_j \in \mathcal{V}_k^-(\mathbf{x}_i)\}.$$

3. Proposed method

Motivated by the toy example above, the proposed method is based on learning a linear transformation that maximizes the difference between the probability distribution on the positive difference space and that on the negative difference space. Such difference is often measured by the well-known Kullback–Leibler divergence [18], which is widely used in many machine learning applications, such as information retrieval [19], text categorization [20], particularly in the classification of multimedia data with support vector machines [21]. In the distance metric learning context, the Kullback–Leibler divergence was introduced by Davis et al. [7] in information-theoretic metric learning (ITML) and later it was motivated in several other distance metric learning methods [22–25]. Since the Kullback–Leibler divergence can yield substantially different values by changing the order of its arguments, in this work we use the symmetric Kullback–Leibler divergence (also called Jeffrey divergence).

3.1. Problem formulation

Let P denote the distribution of the differences in the positive difference space and let Q denote the distribution of the differences in the negative difference space. We assume that P and Q are multivariate Gaussian distributions with zero mean[2] and covariance matrices $\Sigma_\mathcal{S}$ and $\Sigma_\mathcal{D}$, respectively. As described by Duda et al. [26], linear combinations of jointly normally distributed random variables are normally distributed, even if the variables are not independent. Suppose we perform a linear transformation $\mathbf{x}' = \mathbf{A}^\top \mathbf{x}$, then the transformed distributions $P_\mathbf{A}$ and $Q_\mathbf{A}$ have zero mean and covariance matrices $\mathbf{A}^\top \Sigma_\mathcal{S} \mathbf{A}$ and $\mathbf{A}^\top \Sigma_\mathcal{D} \mathbf{A}$, respectively. Our goal is to find the linear transformation that maximizes the Jeffrey divergence between $P_\mathbf{A}$ and $Q_\mathbf{A}$:

$$\arg\max_{\mathbf{A} \in \mathbb{R}^{D \times m}} f(\mathbf{A}) = \mathrm{KL}(P_\mathbf{A}, Q_\mathbf{A}) + \mathrm{KL}(Q_\mathbf{A}, P_\mathbf{A}). \tag{1}$$

As shown in Appendix B, the Jeffrey divergence between $P_\mathbf{A}$ and $Q_\mathbf{A}$ can

[2] In practice, if an instance $\mathbf{x}_i$ belongs to the neighborhood of an instance $\mathbf{x}_j$, then $\mathbf{x}_j$ usually belongs to the neighborhood of $\mathbf{x}_i$ as well. As a consequence, in practice, the distributions will be symmetric with zero mean.

be calculated as:

$$f(\mathbf{A}) = \frac{1}{2}\text{tr}((\mathbf{A}^\top\Sigma_\mathcal{S}\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_\mathcal{D}\mathbf{A}) + (\mathbf{A}^\top\Sigma_\mathcal{D}\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_\mathcal{S}\mathbf{A})) - m.$$

Since the parameter m in $f(\mathbf{A})$ is constant, we can simplify problem (1) to:

$$\arg\max_{\mathbf{A}\in\mathbb{R}^{D\times m}} J(\mathbf{A}) = \text{tr}((\mathbf{A}^\top\Sigma_\mathcal{S}\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_\mathcal{D}\mathbf{A}) + (\mathbf{A}^\top\Sigma_\mathcal{D}\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_\mathcal{S}\mathbf{A})). \tag{2}$$

Taking the derivative of $J(\mathbf{A})$ with respect to $\mathbf{A}$, by using [27, Eq. (2.4.4)], we obtain

$$\begin{aligned}\frac{\partial}{\partial\mathbf{A}}J(\mathbf{A}) &= -2\Sigma_\mathcal{S}\mathbf{A}(\mathbf{A}^\top\Sigma_\mathcal{S}\mathbf{A})^{-1}\mathbf{A}^\top\Sigma_\mathcal{D}\mathbf{A}(\mathbf{A}^\top\Sigma_\mathcal{S}\mathbf{A})^{-1} + 2\Sigma_\mathcal{D}\mathbf{A}(\mathbf{A}^\top\Sigma_\mathcal{S}\mathbf{A})^{-1} \\ &\quad - 2\Sigma_\mathcal{D}\mathbf{A}(\mathbf{A}^\top\Sigma_\mathcal{D}\mathbf{A})^{-1}\mathbf{A}^\top\Sigma_\mathcal{S}\mathbf{A}(\mathbf{A}^\top\Sigma_\mathcal{D}\mathbf{A})^{-1} + 2\Sigma_\mathcal{S}\mathbf{A}(\mathbf{A}^\top\Sigma_\mathcal{D}\mathbf{A})^{-1} \\ &= \underbrace{(2\Sigma_\mathcal{D}\mathbf{A}\Sigma_{2\mathcal{S}}^{-1} - 2\Sigma_\mathcal{S}\mathbf{A}\Sigma_{2\mathcal{S}}^{-1}\Sigma_{2\mathcal{D}}\Sigma_{2\mathcal{S}}^{-1})}_{\text{first term}} + \underbrace{(2\Sigma_\mathcal{S}\mathbf{A}\Sigma_{2\mathcal{D}}^{-1} - 2\Sigma_\mathcal{D}\mathbf{A}\Sigma_{2\mathcal{D}}^{-1}\Sigma_{2\mathcal{S}}\Sigma_{2\mathcal{D}}^{-1})}_{\text{second term}},\end{aligned}$$

where $\Sigma_{2\mathcal{S}} = \mathbf{A}^\top\Sigma_\mathcal{S}\mathbf{A}$ and $\Sigma_{2\mathcal{D}} = \mathbf{A}^\top\Sigma_\mathcal{D}\mathbf{A}$. The optimal matrix $\mathbf{A}$ should satisfy $\partial J(\mathbf{A})/\partial\mathbf{A} = \mathbf{0}$. Although it seems very complex to solve $\partial J(\mathbf{A})/\partial\mathbf{A} = \mathbf{0}$ for $\mathbf{A}$, the first and second terms can be made zero separately as follows. For the first term, it should hold

$$\Sigma_\mathcal{S}^{-1}\Sigma_\mathcal{D}\mathbf{A} = \mathbf{A}\Sigma_{2\mathcal{S}}^{-1}\Sigma_{2\mathcal{D}}. \tag{3}$$

For the second term, it should hold

$$\Sigma_\mathcal{D}^{-1}\Sigma_\mathcal{S}\mathbf{A} = \mathbf{A}\Sigma_{2\mathcal{D}}^{-1}\Sigma_{2\mathcal{S}}. \tag{4}$$

Before solving these problems, we would like to introduce a theorem, which will help us to find the solution to these problems.

Theorem 1. *Let $\mathbf{A}\in\mathbb{R}^{D\times m}$ be a matrix of m linearly independent eigenvectors of $\Sigma_1^{-1}\Sigma_2$. Then it holds that*

$$\Sigma_1^{-1}\Sigma_2\mathbf{A} = \mathbf{A}(\mathbf{A}^\top\Sigma_1\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_2\mathbf{A}).$$

Proof. Let $\mathbf{D}\in\mathbb{R}^{m\times m}$ be the diagonal matrix containing the corresponding m eigenvalues of $\Sigma_1^{-1}\Sigma_2$, then it holds by definition

$$\Sigma_1^{-1}\Sigma_2\mathbf{A} = \mathbf{A}\mathbf{D}. \tag{5}$$

Multiplying both sides of Eq. (5) by the matrix $\mathbf{A}^\top\Sigma_1$ yields

$$\mathbf{A}^\top\Sigma_2\mathbf{A} = \mathbf{A}^\top\Sigma_1\mathbf{A}\mathbf{D},$$

or, equivalently,

$$(\mathbf{A}^\top\Sigma_1\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_2\mathbf{A}) = \mathbf{D}. \tag{6}$$

Multiplying both sides of Eq. (6) by $\mathbf{A}$ yields

$$\mathbf{A}(\mathbf{A}^\top\Sigma_1\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_2\mathbf{A}) = \mathbf{A}\mathbf{D}. \tag{7}$$

Substituting (5) into (7) leads to

$$\Sigma_1^{-1}\Sigma_2\mathbf{A} = \mathbf{A}(\mathbf{A}^\top\Sigma_1\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_2\mathbf{A}).$$

This concludes the proof. □

Consequently, we have the following corollary.

Corollary 1. *Let $\mathbf{A}\in\mathbb{R}^{D\times m}$ be a matrix of m linearly independent eigenvectors of $\Sigma_1^{-1}\Sigma_2$ and $\mathbf{D}\in\mathbb{R}^{m\times m}$ be the diagonal matrix containing the corresponding m eigenvalues. Then it holds that*

$$\text{tr}((\mathbf{A}^\top\Sigma_1\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_2\mathbf{A})) = \text{tr}(\mathbf{D}).$$

Proof. By definition, it holds that

$$\Sigma_1^{-1}\Sigma_2\mathbf{A} = \mathbf{A}\mathbf{D}. \tag{8}$$

Substituting $\Sigma_1^{-1}\Sigma_2\mathbf{A} = \mathbf{A}(\mathbf{A}^\top\Sigma_1\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_2\mathbf{A})$ into Eq. (8) leads to

$$\mathbf{A}(\mathbf{A}^\top\Sigma_1\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_2\mathbf{A}) = \mathbf{A}\mathbf{D}. \tag{9}$$

Multiplying both sides of Eq. (9) by $\mathbf{A}^\top$ yields

$$(\mathbf{A}^\top\Sigma_1\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_2\mathbf{A}) = \mathbf{D},$$

and hence

$$\text{tr}((\mathbf{A}^\top\Sigma_1\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_2\mathbf{A})) = \text{tr}(\mathbf{D}).$$

We conclude the proof. □

In order to satisfy Eqs. (3) and (4) simultaneously, according to Theorem 1, $\mathbf{A}$ can be a matrix of eigenvectors of both $\Sigma_\mathcal{S}^{-1}\Sigma_\mathcal{D}$ and $\Sigma_\mathcal{D}^{-1}\Sigma_\mathcal{S}$, because the latter two share the same eigenvectors and their diagonal matrices of corresponding eigenvalues are Λ and Λ^{-1} as they are related by $(\Sigma_\mathcal{S}^{-1}\Sigma_\mathcal{D})^{-1} = \Sigma_\mathcal{D}^{-1}\Sigma_\mathcal{S}$. Therefore, by selecting m linearly independent eigenvectors of $\Sigma_\mathcal{S}^{-1}\Sigma_\mathcal{D}$, we satisfy both equations simultaneously and consequently it holds that $\partial J(\mathbf{A})/\partial\mathbf{A} = \mathbf{0}$. According to Corollary 1, the value of $J(\mathbf{A})$ is:

$$\begin{aligned}J(\mathbf{A}) &= \text{tr}((\mathbf{A}^\top\Sigma_\mathcal{S}\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_\mathcal{D}\mathbf{A}) + (\mathbf{A}^\top\Sigma_\mathcal{D}\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_\mathcal{S}\mathbf{A})) \\ &= \text{tr}((\mathbf{A}^\top\Sigma_\mathcal{S}\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_\mathcal{D}\mathbf{A})) + \text{tr}((\mathbf{A}^\top\Sigma_\mathcal{D}\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_\mathcal{S}\mathbf{A})) \\ &= \text{tr}(\Lambda) + \text{tr}(\Lambda^{-1}) = \sum_{i=1}^{m}\left(\lambda_i + \frac{1}{\lambda_i}\right).\end{aligned}$$

In order to maximize $J(\mathbf{A})$, we must select the m linearly independent eigenvectors of $\Sigma_\mathcal{S}^{-1}\Sigma_\mathcal{D}$ corresponding to the m largest values of $(\lambda_i + 1/\lambda_i)$. The eigenvalue λ_i of $\Sigma_\mathcal{S}^{-1}\Sigma_\mathcal{D}$ can be found by solving the equation

$$\Sigma_\mathcal{D}\mathbf{w}_i = \lambda_i\Sigma_\mathcal{S}\mathbf{w}_i,$$

where $\mathbf{w}_i$ is the corresponding eigenvector. It is also known as a generalized eigenvalue problem. Note that we might select the target dimensionality m using a validation set (subset of the training set) to find the dimensionality that realizes the best performance.

As shown in [28], each eigenvalue λ_i of $\Sigma_\mathcal{S}^{-1}\Sigma_\mathcal{D}$ represents the ratio between the variances of Q and P along its corresponding eigenvector $\mathbf{w}_i$. Let $\omega_i^\mathcal{S}$ be the variance of P and let $\omega_i^\mathcal{D}$ be the variance of Q along $\mathbf{w}_i$. When these variances are the same, λ_i becomes 1 and $(\lambda_i + 1/\lambda_i)$ becomes 2, which is the minimum value. Otherwise, when $\omega_i^\mathcal{S}$ is larger or smaller than $\omega_i^\mathcal{D}$, then λ_i becomes smaller or larger than 1. Consequently, $(\lambda_i + 1/\lambda_i)$ becomes larger than 2 in any case. This intuition yields a straightforward explanation of why our formulation is effective in extracting features that realize significant differences between two distributions P and Q.

Using maximum likelihood estimation, the covariance matrices $\Sigma_\mathcal{S}$ and $\Sigma_\mathcal{D}$ are computed as follows:

$$\Sigma_\mathcal{S} = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{n}\left[\sum_{\mathbf{x}_j\in\mathcal{V}_k^+(\mathbf{x}_i)}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top\right], \tag{10}$$

$$\Sigma_\mathcal{D} = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{n}\left[\sum_{\mathbf{x}_j\in\mathcal{V}_k^-(\mathbf{x}_i)}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top\right]. \tag{11}$$

We refer to the proposed method as *Distance Metric Learning through Maximization of the Jeffrey divergence* (DMLMJ). A simplified pseudo-code implementation of DMLMJ is given in Algorithm 1.

Algorithm 1. DMLMJ.

Input: Training set $\mathcal{X}$, $\mathcal{Y}$; number of neighbors k; desired di-

mensionality m
Output: The transformation matrix $\mathbf{A}$
1. Build the difference spaces

$$\mathcal{S} \leftarrow \{\mathbf{x}_i - \mathbf{x}_j | \mathbf{x}_i \in \mathcal{X} \text{ and } \mathbf{x}_j \in \mathcal{V}_k^+(\mathbf{x}_i)\},$$

$$\mathcal{D} \leftarrow \{\mathbf{x}_i - \mathbf{x}_j | \mathbf{x}_i \in \mathcal{X} \text{ and } \mathbf{x}_j \in \mathcal{V}_k^-(\mathbf{x}_i)\}.$$

2. Estimate the covariance matrices

$$\Sigma_{\mathcal{S}} \leftarrow \frac{1}{|\mathcal{S}|}\sum_{\mathbf{u}_i \in \mathcal{S}} \mathbf{u}_i\mathbf{u}_i^\top,\ \Sigma_{\mathcal{D}} \leftarrow \frac{1}{|\mathcal{D}|}\sum_{\mathbf{u}_i \in \mathcal{D}} \mathbf{u}_i\mathbf{u}_i^\top.$$

3. Find the eigenvector matrix $\mathbf{W}$ and the eigenvalue vector λ of $\Sigma_{\mathcal{S}}^{-1}\Sigma_{\mathcal{D}}$ using the generalized eigenvalue decomposition.
4. Construct the matrix $\mathbf{A}$, whose columns are the m column vectors $\mathbf{w}_i \in \mathbf{W}$ corresponding to the m largest values of $(\lambda_i + 1/\lambda_i)$.

3.2. Non-linear distance metric learning

Many real-world data sets contain non-linearities that linear transformations are unable to capture [29,30]. In this section, we will derive the kernelized version of DMLMJ to tackle non-linear problems. The idea of kernelization is to learn a linear transformation in the non-linear feature space induced by a kernel function. This idea has been successfully applied in many other contexts, including non-linear kernel principal component analysis [31], kernel Fisher discriminant analysis [32], and particularly in support vector machines [33].

Let ϕ be a non-linear function that maps the input space from $\mathbb{R}^D$ into the feature space $\mathcal{F}$ of dimensionality N,

$$\phi\colon \mathbb{R}^D \to \mathcal{F}$$
$$\mathbf{x} \mapsto \phi(\mathbf{x}).$$

As a result, each training instance is mapped into a potentially non-linear feature space, in which we can perform linear transformations. Let $\Phi = (\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n))$ be the matrix whose columns are the images of the n training instances under ϕ. Given two points $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^D$, the function that returns the inner product between their images in $\mathcal{F}$ is known as the kernel function, $\ker(\mathbf{u}, \mathbf{v}) = \langle\phi(\mathbf{u}), \phi(\mathbf{v})\rangle$. To find the linear transformation $\mathbf{A}$ in the feature space $\mathcal{F}$, we aim to solve the following problem:

$$\arg\max_{\mathbf{A}\in\mathbb{R}^{N\times m}} J(\mathbf{A}) = \mathrm{tr}((\mathbf{A}^\top\Sigma_{\mathcal{S}}^{\Phi}\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_{\mathcal{D}}^{\Phi}\mathbf{A}) + (\mathbf{A}^\top\Sigma_{\mathcal{D}}^{\Phi}\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_{\mathcal{S}}^{\Phi}\mathbf{A})), \tag{12}$$

where the covariance matrices $\Sigma_{\mathcal{S}}^{\Phi}$ and $\Sigma_{\mathcal{D}}^{\Phi}$ are defined as:

$$\Sigma_{\mathcal{S}}^{\Phi} = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{n}\left[\sum_{\phi(\mathbf{x}_j)\in\mathcal{V}_k^+(\phi(\mathbf{x}_i))}(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^\top\right],$$

$$\Sigma_{\mathcal{D}}^{\Phi} = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{n}\left[\sum_{\phi(\mathbf{x}_j)\in\mathcal{V}_k^-(\phi(\mathbf{x}_i))}(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^\top\right].$$

Note that the dimensionality of $\mathcal{F}$ can be very high or even infinite. In such case, it becomes hard to estimate $\mathbf{A}$, $\Sigma_{\mathcal{S}}^{\Phi}$ and $\Sigma_{\mathcal{D}}^{\Phi}$ directly in the feature space due to the increased computational complexity. Moreover, the mapping ϕ is usually unknown. To overcome these problems, we use the same trick as in [31] for kernel principal component analysis. Instead of explicitly expressing the linear transformation, we find a solution that lies in the span of all training instances. That is, each column vector $\mathbf{w}_i$ of $\mathbf{A}$ is represented as a linear combination of training instances in $\mathcal{F}$:

$$\mathbf{w}_i = \sum_{j=1}^{n} B_{ji}\phi(\mathbf{x}_j),$$

where the matrix $\mathbf{B} \in \mathbb{R}^{n\times m}$ contains the expansion coefficients. Now, we need to find the matrix $\mathbf{B}$. Let $\mathbf{U} = \Phi^\top\Sigma_{\mathcal{S}}^{\Phi}\Phi$ and $\mathbf{V} = \Phi^\top\Sigma_{\mathcal{D}}^{\Phi}\Phi$, then the objective function in (12) becomes:

$$\begin{aligned} J(\mathbf{A}) &= \mathrm{tr}((\mathbf{A}^\top\Sigma_{\mathcal{S}}^{\Phi}\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_{\mathcal{D}}^{\Phi}\mathbf{A}) + (\mathbf{A}^\top\Sigma_{\mathcal{D}}^{\Phi}\mathbf{A})^{-1}(\mathbf{A}^\top\Sigma_{\mathcal{S}}^{\Phi}\mathbf{A})) \\ &= \mathrm{tr}((\mathbf{B}^\top\Phi^\top\Sigma_{\mathcal{S}}^{\Phi}\Phi\mathbf{B})^{-1}(\mathbf{B}^\top\Phi^\top\Sigma_{\mathcal{D}}^{\Phi}\Phi\mathbf{B}) + (\mathbf{B}^\top\Phi^\top\Sigma_{\mathcal{D}}^{\Phi}\Phi\mathbf{B})^{-1}(\mathbf{B}^\top\Phi^\top\Sigma_{\mathcal{S}}^{\Phi}\Phi\mathbf{B})) \\ &= \mathrm{tr}((\mathbf{B}^\top\mathbf{U}\mathbf{B})^{-1}(\mathbf{B}^\top\mathbf{V}\mathbf{B}) + (\mathbf{B}^\top\mathbf{V}\mathbf{B})^{-1}(\mathbf{B}^\top\mathbf{U}\mathbf{B})). \end{aligned}$$

Hence, problem (12) can be rewritten as:

$$\arg\max_{\mathbf{B}\in\mathbb{R}^{n\times m}} J(\mathbf{B}) = \mathrm{tr}((\mathbf{B}^\top\mathbf{U}\mathbf{B})^{-1}(\mathbf{B}^\top\mathbf{V}\mathbf{B}) + (\mathbf{B}^\top\mathbf{V}\mathbf{B})^{-1}(\mathbf{B}^\top\mathbf{U}\mathbf{B})).$$

Analogously to (12), this problem can be solved by finding the eigenvectors of $\mathbf{U}^{-1}\mathbf{V}$. The maximizing solution is a matrix containing the m eigenvectors of $\mathbf{U}^{-1}\mathbf{V}$ corresponding to the m largest values of $(\lambda_i + 1/\lambda_i)$, where λ_i are eigenvalues of $\mathbf{U}^{-1}\mathbf{V}$. We refer to this method as *Kernel Distance Metric Learning through Maximization of the Jeffrey divergence* (KDMLMJ). A simplified pseudo-code implementation of KDMLMJ is given in Algorithm 2.

Algorithm 2. KDMLMJ.

Input: Training set $\mathcal{X}$, $\mathcal{Y}$; number of neighbors k; desired dimensionality m; kernel function ker
Output: The matrix of expansion coefficients $\mathbf{B}$
1. Compute the matrix $\mathbf{U}$ as in (13) and the matrix $\mathbf{V}$ as in (14) using the kernel function ker.
3. Find the eigenvector matrix $\mathbf{W}$ and the eigenvalue vector λ of $\mathbf{U}^{-1}\mathbf{V}$ using the generalized eigenvalue decomposition.
4. Construct the matrix $\mathbf{B}$, whose columns are the m column vectors $\mathbf{w}_i \in \mathbf{W}$ corresponding to the m largest values of $(\lambda_i + 1/\lambda_i)$.

Moreover, the matrices $\mathbf{U}$ and $\mathbf{V}$ can be expressed as:

$$\begin{aligned} \mathbf{U} &= \Phi^\top\Sigma_{\mathcal{S}}^{\Phi}\Phi \\ &= \frac{1}{|\mathcal{S}|}\sum_{i=1}^{n}\left[\sum_{\phi(\mathbf{x}_j)\in\mathcal{V}_k^+(\phi(\mathbf{x}_i))}(\Phi^\top\phi(\mathbf{x}_i) - \Phi^\top\phi(\mathbf{x}_j))(\Phi^\top\phi(\mathbf{x}_i) - \Phi^\top\phi(\mathbf{x}_j))^\top\right] \\ &= \frac{1}{|\mathcal{S}|}\sum_{i=1}^{n}\left[\sum_{\phi(\mathbf{x}_j)\in\mathcal{V}_k^+(\phi(\mathbf{x}_i))}(K(\mathbf{x}_i) - K(\mathbf{x}_j))(K(\mathbf{x}_i) - K(\mathbf{x}_j))^\top\right], \end{aligned} \tag{13}$$

and

$$\begin{aligned} \mathbf{V} &= \Phi^\top\Sigma_{\mathcal{D}}^{\Phi}\Phi \\ &= \frac{1}{|\mathcal{D}|}\sum_{i=1}^{n}\left[\sum_{\phi(\mathbf{x}_j)\in\mathcal{V}_k^-(\phi(\mathbf{x}_i))}(\Phi^\top\phi(\mathbf{x}_i) - \Phi^\top\phi(\mathbf{x}_j))(\Phi^\top\phi(\mathbf{x}_i) - \Phi^\top\phi(\mathbf{x}_j))^\top\right] \\ &= \frac{1}{|\mathcal{D}|}\sum_{i=1}^{n}\left[\sum_{\phi(\mathbf{x}_j)\in\mathcal{V}_k^-(\phi(\mathbf{x}_i))}(K(\mathbf{x}_i) - K(\mathbf{x}_j))(K(\mathbf{x}_i) - K(\mathbf{x}_j))^\top\right], \end{aligned} \tag{14}$$

where

$$\begin{aligned} K(\mathbf{u}) &= \Phi^\top\phi(\mathbf{u}) = (\langle\phi(\mathbf{x}_1), \phi(\mathbf{u})\rangle, \ldots, \langle(\mathbf{x}_n), \phi(\mathbf{u})\rangle)^\top \\ &= (\ker(\mathbf{x}_1, \mathbf{u}), \ldots, \ker(\mathbf{x}_n, \mathbf{u}))^\top. \end{aligned}$$

Since $\mathbf{U}$ and $\mathbf{V}$ are expressed in terms of inner products, we can use a kernel function for mapping all instances and apply a kernel trick as in support vector machines [33], or kernel principal component analysis [31]. Possible kernel functions are Gaussian radial basis function kernels (RBF), $\ker(\mathbf{u}, \mathbf{v}) = \exp(-\|\mathbf{u} - \mathbf{v}\|^2/\sigma)$, or polynomial kernels, $\ker(\mathbf{u}, \mathbf{v}) = \langle\mathbf{u}, \mathbf{v}\rangle^b$, for some positive constants $\sigma \in \mathbb{R}$ and $b \in \mathbb{N}$, respectively (see [33] and the references therein for other kernel functions).

Finally, the Mahalanobis distance in the feature space $\mathcal{F}$ is computed as:

$$
\begin{aligned}
d^2_{\mathbf{M}}(\phi(\mathbf{u}), \phi(\mathbf{v})) &= \|\mathbf{A}^\top(\phi(\mathbf{u}) - \phi(\mathbf{v}))\|^2 \\
&= (\mathbf{A}^\top(\phi(\mathbf{u}) - \phi(\mathbf{v})))^\top(\mathbf{A}^\top(\phi(\mathbf{u}) - \phi(\mathbf{v}))) \\
&= (\mathbf{B}^\top(\Phi^\top\phi(\mathbf{u}) - \Phi^\top\phi(\mathbf{v})))^\top(\mathbf{B}^\top(\Phi^\top\phi(\mathbf{u}) - \Phi^\top\phi(\mathbf{v}))) \\
&= (K(\mathbf{u}) - K(\mathbf{v}))^\top\mathbf{B}\mathbf{B}^\top(K(\mathbf{u}) - K(\mathbf{v})).
\end{aligned}
$$

3.3. Regularization

To get a stable solution $(\lambda_i + 1/\lambda_i)$, where λ_i are eigenvalues of $\Sigma_{\mathcal{S}}^{-1}\Sigma_{\mathcal{D}}$, the covariance matrices $\Sigma_{\mathcal{S}}$ and $\Sigma_{\mathcal{D}}$ are required to be non-singular, which is clearly not always the case. Similarly as Mika et al. [32], we propose a regularization technique to avoid this problem by adding some constant value $\alpha \in]0, 1[$ to the diagonal of the covariance matrix. That is, instead of using the covariance matrix directly, we use

$$\widehat{\Sigma} = (1 - \alpha)\Sigma + \alpha\mathbf{I}.$$

Essentially, it renders the covariance matrix positive definite. Therefore, we make sure that all eigenvalues are sufficiently far from zero, and as a consequence, avoid numerical instability in computing the inverse.

3.4. Computational complexity

We analyze the computational complexity of DMLMJ. The difference spaces $\mathcal{S}$ and $\mathcal{D}$ can be built with a time complexity of $O(kn^2D)$. Next, we compute the covariance matrices $\Sigma_{\mathcal{S}}$ and $\Sigma_{\mathcal{D}}$ in Eqs. (10) and (11) with a time complexity of $O(knD^2)$. The generalized eigenvalue decomposition for $\Sigma_{\mathcal{S}}^{-1}\Sigma_{\mathcal{D}}$ can be performed in $O(D^3)$. Summarizing, the overall time complexity for DMLMJ is $O(kn^2D + knD^2 + D^3)$.

Analogously, we analyze the computational complexity of KDMLMJ. We first compute the kernel matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$, where $K_{ij} = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$, with a time complexity of $O(n^2D)$. Then the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ in the feature space $\mathcal{F}$ can be computed in $O(1)$ as:

$$
\begin{aligned}
d(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)) &= \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\| \\
&= \sqrt{\langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_i)\rangle - 2\langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle + \langle\phi(\mathbf{x}_j), \phi(\mathbf{x}_j)\rangle} \\
&= \sqrt{K_{ii} - 2K_{ij} + K_{jj}}.
\end{aligned}
$$

Therefore, we can find the positive and negative neighborhoods for each training instance with a time complexity of $O(kn)$. The computation of the matrices $\mathbf{U}$ and $\mathbf{V}$ in Eqs. (13) and (14) can be performed in $O(kn^3)$. The time complexity of the generalized eigenvalue decomposition for $\mathbf{U}^{-1}\mathbf{V}$ is $O(n^3)$. Summarizing, the overall time complexity for KDMLMJ is $O(n^2D + kn^3)$.

4. Related work

In order to take into account the positive semidefiniteness constraint, distance metric learning methods are mostly formulated as convex semidefinite programs. However, standard semidefinite programming solvers [34] do not scale well when the number of instances or the dimensionality is high, due to the expensive computational cost in each iteration. A number of methods have been proposed to reduce this heavy computational burden. Weinberger and Saul [3] suggested an efficient solver based on the projected subgradient descent method, but it requires an eigen-decomposition to preserve the positive semidefiniteness of the solution after each iteration. To avoid this eigen-decomposition, Ying and Li [35] used the Frank–Wolfe method, which only requires the largest eigenvalue and corresponding eigenvector, to learn the distance metric. A similar solution requiring only the computation of the largest eigenvalue and corresponding eigenvector, based on a boosting-like method, was presented by Shen et al. [14] to learn a linear positive combination of rank-one matrices. An alternative method proposed by Davis et al. [7] was based on the iterative Bregman projection, where no eigen-decomposition is required. Shi et al. [36] formulated distance metric learning as learning a sparse combination of positive semidefinite rank-one matrices.

Another paradigm aims to learn a distance metric through learning a linear transformation. Since the positive semidefiniteness constraint in this case is not required, the optimization problem can be efficiently solved by a first-order method, such as gradient descent. However, the problem may be no longer convex, thus suffering from spurious local minima. Consequently, the solution will depend on the initialization point. Some popular methods include neighborhood component analysis [37], large margin component analysis [30], and multi-task low-rank metric learning [38].

Unfortunately, the above methods typically require a very large number of iterations for large-scale problems. In practice, it is impossible to satisfy all constraints through online learning or stochastic optimization techniques. That is why learning a Mahalanobis distance metric for large data sets becomes a tremendous challenge as learning is limited by computational resources.

On the other hand, eigenvalue methods have been widely used to learn linear transformations since they only need to compute an eigen-decomposition. The most popular methods include principal component analysis [39] and Fisher's linear discriminant analysis [40]. These methods can also be used in a non-linear input space by applying the kernel trick [31,32]. Bar-Hillel et al. [41] proposed a simple and efficient method, called relevant component analysis (RCA), for semi-supervised applications. RCA computes the Mahalanobis distance metric as a whitening transformation of the within chunklet covariance matrix, which is built from the pairwise similarity constraints. Yeung and Chang [42] extended RCA by incorporating both similarity and dissimilarity constraints. Hoi et al. [43] proposed discriminative component analysis (DCA), where the objective function is based on the ratio of determinants between the within and the discriminative chunklet covariance matrices. A method similar to DCA was introduced in [4] by using the ratio of traces between the covariance matrices as objective function, requiring an iterative method to find the linear transformation. Despite our method is also an eigenvalue method, it differs significantly from previous methods in the way of finding the covariance matrices as well as in the objective function. By considering the local constraints derived from the neighborhood of each training instance, our method can significantly improve the performance of k-NN, as will be shown next.

5. Experiments

In this section, we describe some experiments to evaluate the effectiveness of distance metric learning methods. We compare the proposed methods to the baseline Euclidean distance metric and four state-of-the-art distance metric learning methods, including ITML [7], LMNN [3], DML-eig [35] and SCM [36] (all acronyms are explained in Table 1). First, we use 27 data sets of different sizes to evaluate the linear distance metric learning methods. Second, we conduct an experiment to evaluate the capability of our method to perform

Table 1
Brief description of distance metric learning methods used in the experiments.

Name	Description
Euclidean	The Euclidean distance metric
ITML	Information-theoretic metric learning [7]
LMNN	Large margin nearest neighbor classification [3]
DML-eig	Distance metric learning with eigenvalue optimization [35]
SCML	Sparse compositional metric learning [36]
DMLMJ	Distance metric learning through maximization of the Jeffrey divergence
KDMLMJ	The kernelized version of DMLMJ

dimensionality reduction. Finally, we use two synthetic highly non-linear data sets to evaluate the kernelized version of DMLMJ.

5.1. Experimental settings

In order to make fair comparisons, we use the following configurations throughout this section. All experiments are empirically tested in the context of 5-NN and they are carried out on a PC with 4 Intel Core i5-3570 CPUs (3.40 GHz) and 8 GB RAM. We use the source codes implemented in Matlab of ITML,[3] LMNN,[4] DML-eig[5] and SCML[6] supplied by the authors, and tune their parameters to get the best results. The source codes of DMLMJ and KDMLMJ are available online.[7] For DMLMJ and KDMLMJ, the k-positive neighborhood and k-negative neighborhood are all based on k=5 (see Appendix A for a more detailed analysis of the influence of the number of neighbors).

The first general trend is that the classification accuracy of k-NN using the Euclidean distance metric is significantly improved when using the Mahalanobis distance metric learned by DMLMJ. In general, DMLMJ performs competitively compared with other state-of-the-art methods (Section 5.2).

The second general trend is that DMLMJ can perform distance metric learning and dimensionality reduction simultaneously. It outperforms other distance metric learning methods using principal component analysis (PCA) [39] to reduce the dimensionality. Moreover, it is an order of magnitude faster than the competing methods (Section 5.3).

The third general trend is that KDMLMJ can perform well on highly non-linear data sets, whereas a simple linear transformation cannot improve the performance of k-NN (Section 5.4).

5.2. Linear distance metric learning

We compare the linear distance metric learning methods on 27 data sets from the Knowledge Extraction based on Evolutionary Learning (KEEL) [44] machine learning repository.[8] The information of these data sets is summarized in Table 2. The classification accuracies are obtained by averaging over five runs of 10-fold cross-validation. All divisions of the data sets are randomly split by the KEEL evaluation package. The features of these data sets are normalized into the interval [0, 1].

Table 3 shows the average classification accuracies obtained by the competing methods. On each data set, we rank the methods based on their classification accuracy. We assign rank 1 to the method obtaining the highest accuracy, rank 2 to the method obtaining the second higher accuracy, and so on. The average ranks of the competing methods are listed in the last row of Table 3. To detect whether there are significant differences among the results, we follow the recommendations by Demšar [45] for statistical comparisons of classifiers over multiple data sets.

Firstly, we employ the Friedman test [46] at a confidence level of $\alpha = 0.05$ to test the null hypothesis that all the distance metric learning methods obtain the same results on average. The p-value for the Friedman test is 0.01274. Since the p-value is less than the confidence level α, we reject the null hypothesis. Therefore, we apply the Bonferroni–Dunn test [47] to detect which distance metric learning method performs equivalently or significantly different from the best-ranked method (i.e., DMLMJ, which obtained the lowest rank). The Bonferroni–Dunn test can identify significant differences between the control method (in our case, the best-ranked method) and other methods by computing a critical difference. Two distance metric learning methods are significantly different in performance if their corresponding average ranks differ by at least the critical difference:

$$\mathrm{CD} = q_\alpha \times \sqrt{\frac{n_c(n_c+1)}{6n_t}} = 2.576 \times \sqrt{\frac{6\times(6+1)}{6\times 27}} = 1.3116,$$

where n_c and n_t are the number of competing methods and the number

Table 2
The KEEL data sets used in our experiments.

#	Data sets	# of features	# of instances	# of classes
1.	*appendicitis*	7	106	2
2.	*balance*	4	625	3
3.	*banana*	2	5300	2
4.	*bupa*	6	345	2
5.	*ionosphere*	33	351	2
6.	*iris*	4	150	3
7.	*led7digit*	7	500	10
8.	*letter*	16	20 000	26
9.	*magic*	10	19 020	2
10.	*monk-2*	6	432	2
11.	*movement_libras*	90	360	15
12.	*optdigits*	64	5620	10
13.	*page-blocks*	10	5472	5
14.	*phoneme*	5	5404	2
15.	*pima*	8	768	2
16.	*ring*	20	7400	2
17.	*satimage*	36	6435	7
18.	*segment*	19	2310	7
19.	*sonar*	60	208	2
20.	*spambase*	57	4597	2
21.	*texture*	40	5500	11
22.	*twonorm*	20	7400	2
23.	*vehicle*	18	846	4
24.	*vowel*	13	990	11
25.	*wdbc*	30	569	2
26.	*wine*	13	178	3
27.	*wisconsin*	9	683	2

Table 3
Classification accuracies on the KEEL data sets. The best result is highlighted in boldface.

#	Euclidean	ITML	LMNN	DML-eig	SCML	DMLMJ
1.	85.00	86.00	**88.82**	87.00	86.91	87.91
2.	86.24	91.84	84.64	87.52	**94.25**	92.63
3.	89.28	89.34	89.34	89.17	**89.36**	89.26
4.	64.28	62.05	61.90	62.47	65.05	**65.69**
5.	85.17	87.17	**89.75**	84.90	86.33	**89.75**
6.	95.33	94.67	96.00	96.67	**97.33**	95.33
7.	**70.40**	69.80	69.80	69.40	65.00	67.80
8.	95.55	95.37	96.72	84.42	96.54	**97.50**
9.	83.60	83.73	83.74	83.15	**84.79**	84.30
10.	94.75	89.43	97.04	**100.00**	99.55	99.55
11.	75.28	74.72	**82.50**	67.22	63.33	81.94
12.	98.75	98.70	**99.04**	97.44	97.21	99.00
13.	95.78	96.03	96.24	95.29	**96.56**	95.78
14.	87.75	87.75	87.43	**87.84**	87.49	87.79
15.	73.32	72.93	73.19	73.06	72.92	**73.84**
16.	69.12	81.54	69.22	84.31	80.12	**87.28**
17.	90.78	90.71	91.28	89.54	89.08	**91.79**
18.	95.41	96.36	96.23	**96.84**	95.97	95.84
19.	84.52	81.69	84.05	**85.05**	80.19	**85.05**
20.	87.77	87.91	**90.08**	89.82	88.04	89.39
21.	98.49	99.29	**99.89**	98.98	99.58	99.51
22.	96.99	97.08	96.97	**97.54**	97.09	97.28
23.	71.75	73.77	77.89	72.81	75.89	**80.97**
24.	94.85	91.82	95.35	94.65	94.04	**95.45**
25.	97.01	96.83	96.30	**97.36**	96.48	95.95
26.	95.49	96.67	97.78	96.63	**98.86**	98.33
27.	97.09	96.80	**97.10**	96.96	96.67	96.51
Rank	4.167	4.056	2.944	3.648	3.574	**2.611**

[3] http://www.cs.utexas.edu/pjain/itml/download/itml-1.2.tar.gz
[4] http://www.cse.wustl.edu/kilian/code/files/mLMNN2.3.zip
[5] http://secamlocal.ex.ac.uk/people/staff/yy267/dml-eig-copy.zip
[6] http://mloss.org/media/code_archive/SCMLv1.11.zip
[7] http://users.ugent.be/bacnguy~/DMLMJ.zip
[8] http://sci2s.ugr.es/keel/datasets.php

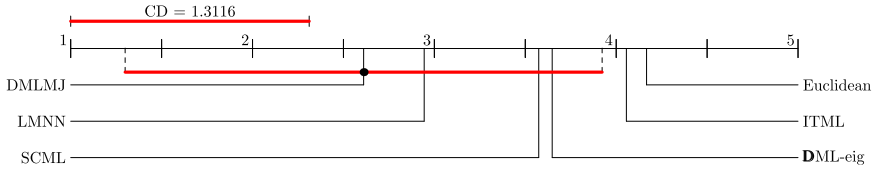


Fig. 4. Comparison of the control method against the others with the Bonferroni–Dumn test. All methods with ranks outside the marked interval are significantly different from the control.

Table 4
Holm post-hoc test for the competing methods with $\alpha = 0.05$.

Control method: DMLMJ					
i	**Method**	***z*-Value**	***p*-Value**	**Holm's adjusted α**	**Hypothesis**
5	**Euclidean**	3.0551	0.0023	0.0100	**Rejected**
4	**ITML**	2.8368	0.0046	0.0125	**Rejected**
3	**DML-eig**	2.0367	0.0417	0.0167	**Accepted**
2	**SCML**	1.8912	0.0586	0.0250	**Accepted**
1	**LMNN**	0.6547	0.5127	0.0500	**Accepted**

of data sets, respectively, and q_α is the critical value [48]. Fig. 4 graphically represents the significant differences among the performances of the different distance metric learning methods. Any distance metric learning method with rank outside this marked area is significantly different from the control method (i.e., DMLMJ).

Additionally, we also apply Holm's step-down procedure [49] to compare the best-ranked method with the remaining methods. Table 4 presents the z-value, p-value, and adjusted α for the Holm test at a confidence level of $\alpha = 0.5$. According to Table 4, the Holm test rejects hypotheses 4 and 5 since the corresponding p-value is less than the adjusted α. But hypotheses 1–3 cannot be rejected.

The statistical results allow us to draw the following conclusions. First, DMLMJ significantly outperforms ITML, but it only shows a slightly better behavior compared to LMNN, DML-eig and SCML in the context of k-NN. Second, the Mahalanobis distance metric learned by DMLMJ consistently outperforms the Euclidean distance metric.

5.3. Dimensionality reduction

We compare the performance of k-NN using DMLMJ against other distance metric learning methods using PCA as a preprocessing step to reduce the dimensionality. Our experiment is based on the Isolet (Isolated Letter Speech Recognition) data set [50], which consists of 6238 training instances, 1559 test instances with 617 features and 26 classes corresponding to 26 spoken letters. The Isolet data set has been used in various distance metric learning studies such as [13,51]. More details about the features can be found in [50]. Training and test sets were predefined.[9] All features are continuous, real values, and scaled into the range $[-1, 1]$.

Fig. 5(a) illustrates the classification accuracy of k-NN based on different distance metric learning methods with a varying number of features. We observe that DMLMJ performs better than other methods on this data set. When the dimensionality is small, all methods perform poorly as a consequence of the loss of information from the original feature space, however, DMLMJ is still much more effective. PCA discards the valuable class label information contained in the training set, and the projection made by PCA may intertwine the useful features and noisy features, thus leading to the poor performance of methods based on PCA.

Fig. 5(b) illustrates the training time (in seconds) of these five methods. Clearly, our method is an order of magnitude faster than the other methods.

5.4. Non-linear distance metric learning

To illustrate the potential of KDMLMJ, we conduct experiments on two synthetic two-dimensional data sets shown in Figs. 6(a) and 7(a). The first one consists of 200 instances drawn from two concentric circles. The second one consists of 500 instances drawn from two banana-shaped distributions. All instances belonging to the same class are shown in the same style and color. Similar experiments on these data sets were discussed by Weinberger and Saul [3] and by Baghshah and Shouraki [52]. According to the non-linear structure in these data sets, a linear transformation may not suffice to improve the classification accuracy of k-NN.

In this experiment, the RBF kernel, $\ker(\mathbf{u}, \mathbf{v}) = \exp(-\| \mathbf{u} - \mathbf{v} \|^2/\sigma)$, where $\sigma > 0$ is the kernel width, is adopted for the KDMLMJ method. The parameter σ is tuned by cross-validation on the training set considering as set of values $\{2^{-15}, \ldots, 2^3\}$. For a visual representation, the data sets are plotted in the transformed space using the non-linear transformation learned by KDMLMJ (see Figs. 6(b) and 7(b)) and the linear transformation learned by DMLMJ (see Figs. 6(c) and 7(c)). Our illustration here is based on DMLMJ, but the same phenomenon occurs for the other linear distance metric learning methods. According to Figs. 6 and 7, KDMLMJ outperforms on both data sets since it is able to produce a highly non-linear decision boundary through the use of the kernel function.

6. Conclusion

In this paper, we have developed a novel linear transformation method for distance metric learning. We have shown that learning a linear transformation can be formulated as maximizing the Jeffrey divergence between two distributions derived from local pairwise constraints. Then we have demonstrated that this problem is equivalent to solving a generalized eigenvalue decomposition problem with a closed-form solution. We have also developed the kernelized version of the proposed method to handle non-linear data sets. The experimental results on the synthetic and real-world data sets demonstrate that the proposed method performs competitively compared with other state-of-the-art distance metric learning methods, while being an order of magnitude faster in training.

Conflict of interest

None declared.

[9] Available at https://archive.ics.uci.edu/ml/datasets/ISOLET.

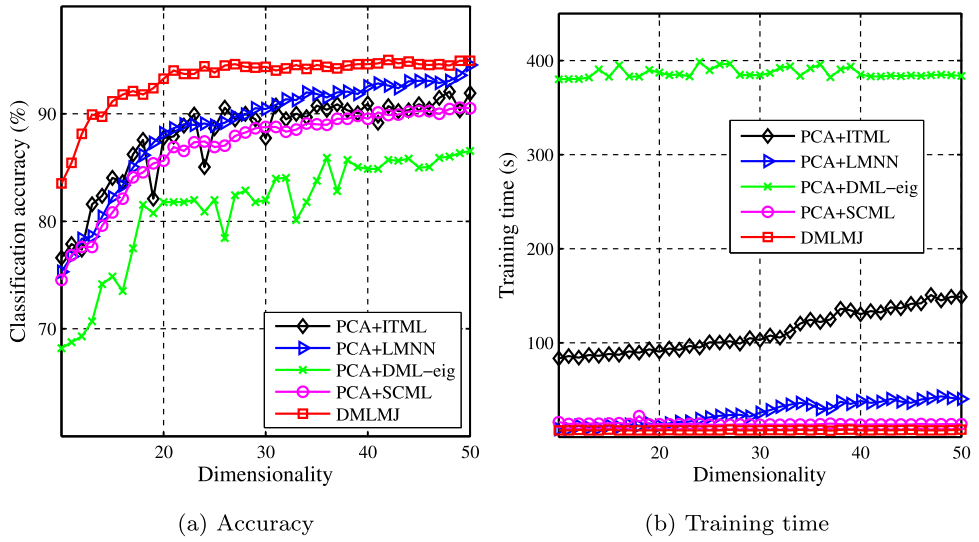


Fig. 5. Experimental results on the Isolet data set. (a) Classification accuracy vs. dimensionality and (b) training time vs. dimensionality.

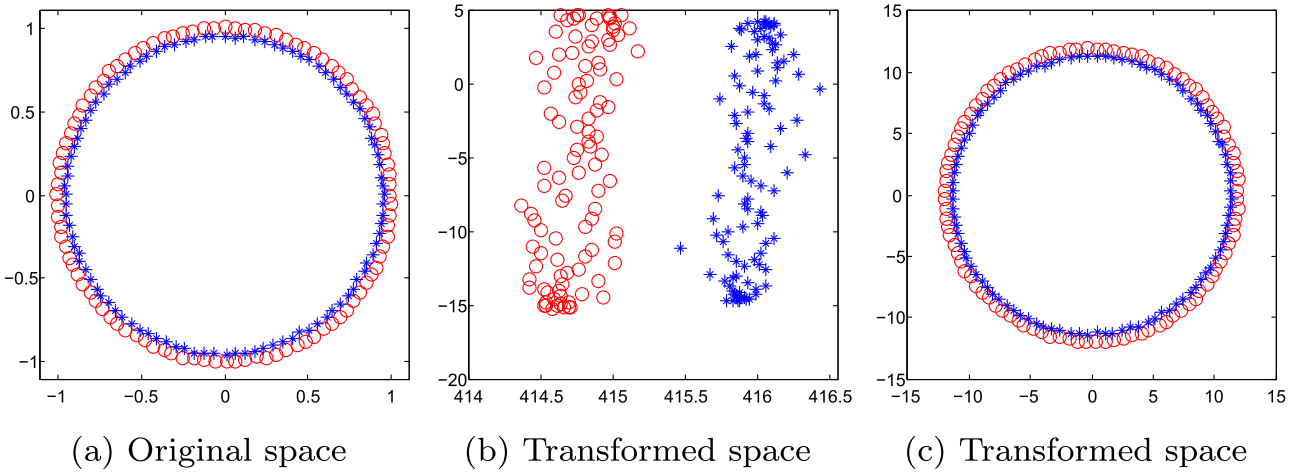


Fig. 6. Illustration of a synthetic data set drawn from two concentric circles: (a) original space, (b) transformed space learned by KDMLMJ using an RBF kernel, and (c) transformed space learned by DMLMJ. Instances belonging to the same class are denoted in the same color and style. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

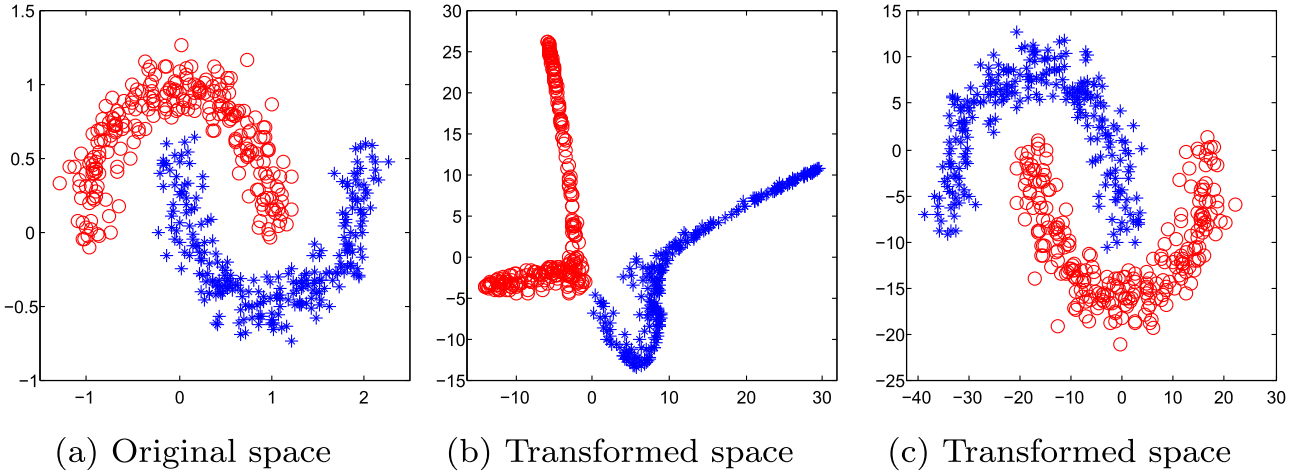


Fig. 7. Illustration of a synthetic data set drawn from two banana-shaped distributions: (a) original space, (b) transformed space learned by KDMLMJ using an RBF kernel, and (c) transformed space learned by DMLMJ. Instances belonging to the same class are denoted in the same color and style. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Appendix A. Influence of the choice of the difference spaces

In this appendix, we study the influence of the choice of the difference spaces on the performance of DMLMJ. Since the difference spaces are built from the nearest neighbors of each training instance, it is interesting to compare the performance in experiments using different neighborhood sizes. Let k_1, k_2 denote the number of neighbors for constructing the positive and negative difference spaces, respectively. Fig. A1 shows the accuracy of 5-NN classification on the *balance* data set with different numbers of neighbors k_1 and k_2. From the figure, we can see that when $k_1 >> k_2$ the classification accuracy is very low. This can be explained by the fact that the positive neighborhoods are more likely to undergo divergence than the negative neighborhoods, which implies that DMLMJ will extract the features that maximize the variance between instances of the same class and minimize the variance between instances of different classes. Consequently, the performance of k-NN cannot be improved. On the other hand, if more instances of different classes are considered to build the negative difference space, the classification accuracy is significantly increased. When k_1 and k_2 approach 100, the difference spaces tend to use the information from the whole data set instead of using only information contained in the neighborhoods. In this case, DMLMJ performs similarly to other global distance metric learning methods, such as distance metric learning for clustering [9] and ITML [7]. The performance is relatively stable when k_1 and k_2 are small enough to find the local discriminative information from the neighborhoods.

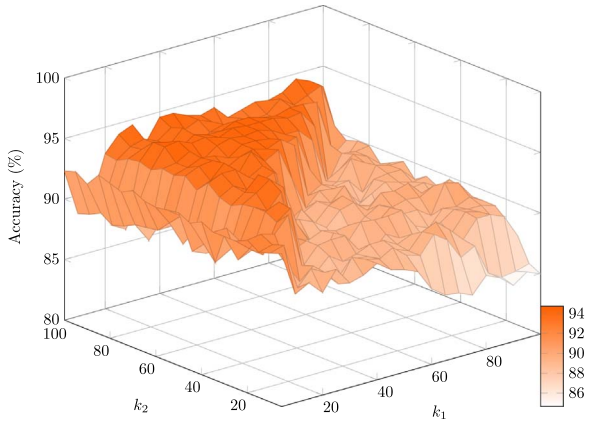


Fig. A1. Experimental results on the balance data set. Classification accuracy of the 5-NN classifier vs. the number of neighbors used for constructing the difference spaces, where k_1 denotes the number of neighbors used in the positive difference space and k_2 denotes the number of neighbors used in the negative difference space.

Appendix B. Jeffrey divergence

Let P_1 and P_2 be two D-dimensional multivariate Gaussian distributions with means μ_1 and μ_2, covariance matrices Σ_1 and Σ_2, and corresponding probability density functions p_1 and p_2, respectively. The Kullback–Leibler divergence between P_1 and P_2 is defined as:

$$\begin{aligned}\mathrm{KL}(P_1, P_2) &= \int \ln\left(\frac{p_1(\mathbf{x})}{p_2(\mathbf{x})}\right) p_1(\mathbf{x})\mathrm{d}\mathbf{x} \\ &= \frac{1}{2}\left[\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) - D + \mathrm{tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2-\mu_1)^\top\Sigma_2^{-1}(\mu_2-\mu_1)\right].\end{aligned}$$

The proof of the latter expression can be found in [53]. We now consider the symmetric Kullback–Leibler divergence or Jeffrey divergence between P_1 and P_2:

$$\begin{aligned}\mathrm{JF}(P_1, P_2) &= \mathrm{KL}(P_1, P_2) + \mathrm{KL}(P_2, P_1) \\ &= \frac{1}{2}\left[\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) - D + \mathrm{tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2-\mu_1)^\top\Sigma_2^{-1}(\mu_2-\mu_1)\right] + \frac{1}{2}\left[\log\left(\frac{|\Sigma_1|}{|\Sigma_2|}\right) - D + \mathrm{tr}(\Sigma_1^{-1}\Sigma_2) + (\mu_1-\mu_2)^\top\Sigma_1^{-1}(\mu_1-\mu_2)\right] \\ &= \frac{1}{2}\mathrm{tr}(\Sigma_1^{-1}\Sigma_2 + \Sigma_2^{-1}\Sigma_1) + \frac{1}{2}\mathrm{tr}((\mu_2-\mu_1)^\top\Sigma_2^{-1}(\mu_2-\mu_1)) + \frac{1}{2}\mathrm{tr}((\mu_1-\mu_2)^\top\Sigma_1^{-1}(\mu_1-\mu_2)) - D \\ &= \frac{1}{2}\mathrm{tr}(\Sigma_1^{-1}\Sigma_2 + \Sigma_2^{-1}\Sigma_1) - D + \frac{1}{2}\mathrm{tr}((\Sigma_1^{-1}+\Sigma_2^{-1})(\mu_2-\mu_1)(\mu_2-\mu_1)^\top).\end{aligned}$$

If $\mu_1 = \mu_2 = \mathbf{0}$, then the Jeffrey divergence between P_1 and P_2 reduces to:

$$\mathrm{JF}(P_1, P_2) = \frac{1}{2}\mathrm{tr}(\Sigma_1^{-1}\Sigma_2 + \Sigma_2^{-1}\Sigma_1) - D.$$

References

[1] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27.
[2] S.P. Lloyd, Least squares quantization in PCM, IEEE Trans. Inf. Theory 28 (2) (1982) 129–137.
[3] K.Q. Weinberger, L.K. Saul, Distance metric learning for large margin nearest neighbor classification, J. Mach. Learn. Res. 10 (2009) 207–244.
[4] S. Xiang, F. Nie, C. Zhang, Learning a Mahalanobis distance metric for data clustering and classification, Pattern Recognit. 41 (12) (2008) 3600–3612.
[5] B. Kulis, Metric learning: a survey, Found. Trends Mach. Learn. 5 (4) (2012) 287–364.
[6] A. Bellet, A. Habrard, M. Sebban, Metric learning, Synth. Lect. Artif. Intell. Mach.

Learn. 9 (1) (2015) 1–151.
[7] J.V. Davis, B. Kulis, P. Jain, S. Sra, I.S. Dhillon, Information-theoretic metric learning, in: Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 209–216.
[8] B. Nguyen, C. Morell, B. De Baets, Large-scale distance metric learning for -nearest neighbors regression, Neurocomputing **214** (2016) 805–814.
[9] E.P. Xing, M.I. Jordan, S. Russell, A. Ng, Distance metric learning with application to clustering with side-information, in: Advances in Neural Information Processing Systems, vol. 14, 2002, pp. 505–512.
[10] Y.G. Zhang, C.S. Zhang, D. Zhang, Distance metric learning by knowledge embedding, Pattern Recognit. 37 (1) (2004) 161–163.
[11] B. Liu, B. Fang, X. Liu, J. Chen, Z. Huang, X. He, Large margin subspace learning for feature selection, Pattern Recognit. 46 (10) (2013) 2798–2806.
[12] C.-C. Chang, Generalized iterative RELIEF for supervised distance metric learning, Pattern Recognit. 43 (8) (2010) 2971–2981.
[13] B. McFee, G. Lanckriet, Metric learning to rank, in: Proceedings of the 27th International Conference on Machine Learning, 2010, pp. 775–782.
[14] C. Shen, J. Kim, L. Wang, A. Van Den Hengel, Positive semidefinite metric learning using boosting-like algorithms, J. Mach. Learn. Res. 13 (1) (2012) 1007–1036.
[15] Q. Wang, P.C. Yuen, G. Feng, Semi-supervised metric learning via topology preserving multiple semi-supervised assumptions, Pattern Recognit. 46 (9) (2013) 2576–2587.
[16] J.B. Tenenbaum, D.S. Vin, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.
[17] L.K. Saul, S.T. Roweis, Think globally, fit locally: unsupervised learning of low dimensional manifolds, J. Mach. Learn. Res. 4 (2003) 119–155.
[18] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1) (1951) 79–86.
[19] B. Bigi, R.D. Mori, M. El-Bèze, T. Spriet, A fuzzy decision strategy for topic identification and dynamic selection of language models, Signal Process. 80 (6) (2000) 1085–1097.
[20] B. Bigi, Using Kullback–Leibler distance for text categorization, in: Proceedings of the 25th European Conference on IR Research, 2003, pp. 305–319.
[21] P.J. Moreno, P.P. Ho, N. Vasconcelos, A Kullback–Leibler divergence based kernel for SVM classification in multimedia applications, in: Advances in Neural Information Processing Systems, vol. 16, 2004, pp. 1385–1392.
[22] G.-J. Qi, J. Tang, Z.-J. Zha, T.-S. Chua, H.-J. Zhang, An efficient sparse metric learning in high-dimensional space via ℓ_1-penalized log-determinant regularization, in: Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 841–848.
[23] P. Jain, B. Kulis, I.S. Dhillon, K. Grauman, Online metric learning and fast similarity search, in: Advances in Neural Information Processing Systems, vol. 21, 2009, pp. 761–768.
[24] J. Mei, M. Liu, H. Karimi, H. Gao, Logdet divergence-based metric learning with triplet constraints and its applications, IEEE Trans. Image Process. 23 (11) (2014) 4920–4931.
[25] A. Globerson, S.T. Roweis, Metric learning by collapsing classes, in: Advances in Neural Information Processing Systems, vol. 18, 2006, pp. 451–458.
[26] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, Wiley-Interscience, New York, United States, 2000.
[27] K.B. Petersen, M.S. Pedersen, The Matrix Cookbook, Technical Report, Technical University of Denmark, November 2012.
[28] K. Fukunaga, Introduction to Statistical Pattern Recognition, 2nd ed., Academic Press, San Diego, California, United States, 1990.
[29] J. He, M. Li, H.-J. Zhang, H. Tong, C. Zhang, Manifold-ranking based image retrieval, in: Proceedings of the 12th Annual ACM International Conference on Multimedia, 2004, pp. 9–16.
[30] L. Torresani, K.-C. Lee, Large margin component analysis, in: Advances in Neural Information Processing Systems, vol. 19, 2007.
[31] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319.
[32] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, K.-R. Müller, Fisher discriminant analysis with kernels, in: Proceedings of the IEEE International Workshop Neural Networks for Signal Processing, vol. IX, 1999, pp. 41–48.
[33] B. Schölkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, Massachusetts, United States, 2001.
[34] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, New York, United States, 2004.
[35] Y. Ying, P. Li, Distance metric learning with eigenvalue optimization, J. Mach. Learn. Res. 13 (1) (2012) 1–26.
[36] Y. Shi, A. Bellet, F. Sha, Sparse compositional metric learning, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014, pp. 2078–2084.
[37] J. Goldberger, S. Roweis, G. Hinton, R. Salakhutdinov, Neighbourhood components analysis, in: Advances in Neural Information Processing Systems, vol. 17, 2005, pp. 513–520.
[38] P. Yang, K. Huang, C.-L. Liu, Multi-task low-rank metric learning based on common subspace, in: Proceedings of the 18th International Conference on Neural Information Processing, Springer, Berlin, Heidelberg, 2011, pp. 151–159.
[39] I. Jolliffe, Principal Component Analysis, Wiley Online Library, New York, United States, 2005.
[40] R.A. Fisher, The use of multiple measurements in taxonomic problems, Ann. Eugen. 7 (2) (1936) 179–188.
[41] A. Bar-Hillel, T. Hertz, N. Shental, D. Weinshall, Learning a Mahalanobis metric from equivalence constraints, J. Mach. Learn. Res. 6 (6) (2005) 937–965.
[42] D.-Y. Yeung, H. Chang, Extending the relevant component analysis algorithm for metric learning using both positive and negative equivalence constraints, Pattern Recognit. 39 (5) (2006) 1007–1010.
[43] S.C. Hoi, W. Liu, M.R. Lyu, W.Y. Ma, Learning distance metrics with contextual constraints for image retrieval, in: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, vol. 2, 2006, pp. 2072–2078.
[44] J. Alcalá-Fdez, L. Sánchez, S. García, M. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, J. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Comput. 13 (3) (2009) 307–318.
[45] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[46] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Ann. Math. Stat. 11 (1) (1940) 86–92.
[47] O.J. Dunn, Multiple comparisons among means, J. Am. Stat. Assoc. 56 (293) (1961) 52–64.
[48] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, 4th ed., Chapman & Hall/CRC, Boca Raton, Florida, United States, 2007.
[49] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. 6 (2) (1979) 65–70.
[50] R. Cole, M. Fanty, Spoken letter recognition, in: Proceedings of the Third DARPA Speech and Natural Language Workshop, 1990, pp. 385–390.
[51] S. Parameswaran, K.Q. Weinberger, Large margin multi-task metric learning, in: Advances in Neural Information Processing Systems, vol. 23, 2010, pp. 1867–1875.
[52] M.S. Baghshah, S.B. Shouraki, Non-linear metric learning using pairwise similarity and dissimilarity constraints and the geometrical structure of data, Pattern Recognit. 43 (8) (2010) 2982–2992.
[53] J.V. Davis, I.S. Dhillon, Differential entropic clustering of multivariate Gaussians, in: Advances in Neural Information Processing Systems, vol. 19, MIT Press, Cambridge, Massachusetts, United States, 2007, pp. 337–344.

Bac Nguyen received his B.Sc. and M.Sc. (summa cum laude) degrees in Computer Science from the Universidad Central de Las Villas, Cuba, in 2014 and 2015, respectively. He is currently a Ph.D. candidate in the Department of Mathematical Modelling, Statistics and Bioinformatics, Ghent University, Belgium. His research interests are in the areas of data mining, machine learning, and their applications.

Carlos Morell received his B.Sc. degree in Computer Science and his Ph.D. in Artificial Intelligence from the Universidad Central de Las Villas, in 1995 and 2005, respectively. Currently, he is a Professor in the Department of Computer Science at the same university. In addition, he leads the Artificial Intelligence Research Laboratory. His teaching and research interests include machine learning, soft computing and programming languages.

Bernard De Baets holds an M.Sc. in Maths (1988), a postgraduate degree in Knowledge Technology (1991) and a Ph.D. in Maths (1995), all summa cum laude from Ghent University (Belgium). He is a Full Professor in Applied Maths (1999) at Ghent University, where he is leading KERMIT, the Research Unit Knowledge-Based Systems. He is a Government of Canada Award holder (1988), an Honorary Professor of Budapest Tech (2006) and an IFSA Fellow (2011). His publications comprise more than 400 papers in international journals and about 60 book chapters. He serves on the editorial boards of various international journals, in particular as Co-Editor-in-Chief of Fuzzy Sets and Systems. B. De Baets is a Member of the Board of Directors of EUSFLAT and of the Administrative Board of the Belgian OR Society.