Tsinghua Science and Technology

Volume 24 | Issue 4 Article 8

2019

LKLR: A Local Tangent Space-Alignment Kernel Least-Squares Regression Algorithm

Chao Tan
the School of Computer Science and Engineering, Southeast University, Nanjing 210096 and the School of Computer Science and Technology, Nanjing Normal University, Nanjing 210023, China.

Genlin Ji
the School of Computer Science and Technology, Nanjing Normal University, Nanjing 210023, China.

TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 03/11 pp389–399
DOI: 10.26599/TST.2018.9010120
Volume 24, Number 4, August 2019

LKLR: A Local Tangent Space-Alignment Kernel Least-Squares Regression Algorithm

Chao Tan* and Genlin Ji

Abstract: In the fields of machine learning and data mining, label learning is a nascent area of research, and within this paradigm, there is much room for improving multi-label manifold learning algorithms for high-dimensional data. Thus far, researchers have experimented with mapping relationships from the feature space to the traditional logical label space (using neighbors in the label space, for example, to predict logical label vectors from the feature space's manifold structure). Here we combine the feature manifold's and label space's local topological structures to reconstruct the label manifold. To achieve this, we use a nonlinear manifold learning algorithm to transform the local topological structure from the feature space to the label space. Our algorithm adopts a regularized least-squares kernel method to realize the reconstruction process, employing an optimization function to find the best solution. Extensive experiments show that our algorithm significantly improves multi-label manifold learning in terms of learning accuracy and time complexity.

Key words: multi-label manifold learning; local topological structure; regularized least squares kernel method

1 Introduction

Multi-Label Learning (MLL) is a flourishing topic in machine learning and data mining research; essentially, this learning process works by mapping an instance and then assigning a label[1]. In the existing learning paradigm, there are two primary patterns of label assignment: (1) a single label is assigned to one instance; (2) multiple labels are assigned to one instance. Single-Label Learning (SLL) assumes that all the examples in a training set are labeled following the first pattern. MLL[2] allows training instances to be labeled following the second pattern. A benefit of MLL is that it can handle ambiguous situations when an instance belongs to multiple classes (labels).

• Chao Tan is with the School of Computer Science and Engineering, Southeast University, Nanjing 210096 and the School of Computer Science and Technology, Nanjing Normal University, Nanjing 210023, China. E-mail: 101101451@seu.edu.cn.
• Genlin Ji is with the School of Computer Science and Technology, Nanjing Normal University, Nanjing 210023, China. E-mail: glji@njnu.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2018-07-09; accepted: 2018-09-01

Although many problems with label uncertainty can be processed using MLL, it is often unsuitable for practical applications where the degree of a label's importance is globally distributed. To address this issue, some machine learning researchers recently proposed a new learning paradigm named Label Distribution Learning (LDL)[3]. Label distribution covers a certain number of labels, representing the extent to which each label can describe an instance. LDL is a more general learning framework that includes SLL and MLL for special conditions. In LDL, each instance is not associated with one or a group of labels but is associated with a label distribution. A label distribution covers all possible labels and gives an explicit degree of the instance described by each label. Under this definition, traditional SLL and MLL can be seen as special cases of LDL. There are many datasets with label distribution information in the real world, and whenever label distribution information is incomplete, a complete label distribution can be generated using prior knowledge or machine learning methods. Thus, researchers explore LDL as a new machine learning paradigm that is more generalized than the traditional learning paradigm, with the understanding that it could

have broader applications.

The traditional label space spanned by label vector y_i is logical, and y_i's elements can be seen as logical labels. To study the label manifold, the label space should be extended to a Euclidean space. Each dimension in the space corresponds to one label in Y, while the value is extended from a logical value to a real value. Such labels are called numerical labels, which carry more semantic information and describe the instances more completely than logical labels. However, label manifolds are not explicitly provided in training examples.

The relationship between feature and label manifolds does not embed or reduce dimensionality. Instead, the manifolds share the local topological structure in two different spaces, based on the smoothness assumption. This approach leads to problems regarding how to use label information in the label manifold space to instruct unsupervised manifold learning methods, so that the supervised information in the label manifold space can improve feature extraction accuracy; to make the feature manifold and label manifold learning guide and promote each other; and to accelerate the entire algorithm's convergence and improve the result's accuracy.

To solve these problems, Hou et al.[4] proposed representing the feature manifold with a graph and approximating it with overlapping patches in local linear neighborhoods. The edge weights in each patch are solved by a least-squares planning procedure. Then the label manifold and the transferred local topological structure are reconstructed from the feature manifold and existing logic labels. A quadratic programming method is used to realize the reconstruction process. The label manifold has three advantages. First, it helps to use the correlation between the labels by transferring the topological structure from feature space based on the smoothness assumptions. Second, it extends the traditional logical labels to numerical labels, which describe the instances in more detail and increase the likelihood of improving performance. Third, it helps make more complex decisions based on numerical labels, because the value of a numerical label indicates the label's relative importance.

To study the label manifold, we extend the label space to a Euclidean space. According to the smoothness assumption, the local topological structure can be shared between the feature and label manifolds. In this paper, we propose a new algorithm (using a feature-extraction method to align the tangent space of characters) to guide the label manifold's reconstruction while preserving the neighboring structural information. Then, in the Euclidean label space, we study the manifold structure. On the basis of nonlinear manifold learning methods such as the local tangent space alignment learning algorithm, we transform the local topological structure from the feature space to the label space and obtain the manifold's structure in the label space. This approach has four major advantages: It is easier to learn the spatial structure relationship of high-dimensional samples, find the manifold structure information hidden in high-dimensional streaming big data, guide the label manifold's reconstruction, and preserve the neighborhood structural information.

When extracting features from big data streams, the feature manifold is obtained by overlapping local linear neighborhood blocks via an incremental alignment learning method. The label manifold is obtained from the local topological structure and transformed from the feature manifold and the existing logical labels. Here, we use our algorithm, the Local tangent space-alignment Kernel Least squares Regression (LKLR) method, to achieve reconstruction. Our method's optimization model has a simple kernel transformation form and extracts the manifold structure's characteristics; moreover, it only needs to perform inverse operations on a sparse matrix and has a high computational efficiency.

To solve the above mentioned problems, in this paper, we first use a feature extraction method to guide the reconstruction of label manifold, and then use the LKLR method to process the reconstruction; finally, we achieve the mutual guidance and promotion of feature manifold learning and label manifold learning. The contributions of this paper are as follows:

- We use a feature-extraction method with a tangent space-alignment property to reconstruct the label manifold and preserve the neighborhood structure information in order to better understand the space-structure relationship of high-dimensional samples.
- We propose an LKLR algorithm to realize the reconstruction process. The method has a simple kernel transformation form that extracts the features of the manifold structure. The method only needs to compute the sparse inverse matrix and has a high computational efficiency.
- We employ a kernel matrix by the feature space

spanned in the previous steps. The optimized objective function has a simple kernel-transformation form that extracts manifold structure features combined with the label space in order to predict the multi-label predictive model.

This paper is organized as follows. First, related works are discussed in Section 2. Then, the details of LKLR are provided in Section 3. Next, we report the results of a comparative experiment in Section 4. Finally, Section 5 concludes the paper.

2 Related Work

Existing MLL algorithms can be roughly divided into three categories based on the idea of label relevance order[5]. The simplest one is a first-order algorithm that assumes independence among class labels[6]. Then multi-label classification becomes a series of binary classification problems. In contrast, the second-order approach considers the correlation between class label pairs[7], and the higher-order method considers the correlation between label subsets or all class labels[8]. Hou et al.[4] explored manifolds in label space and treated the labels as numbers. In this case, the label manifold contains more semantic information and is beneficial to the learning process.

Previous works on MLL have transformed the logical label space into a Euclidean label space. In one instance[9], researchers projected the feature and label spaces into a new space and maximized the correlation between the two spaces' projections. This reduced the dimension of the label space, creating a new space mapped from the original label space. However, the Multi-Label Manifold Learning (ML^2) algorithm proposed by Hou et al.[4] is not characterized by dimensionality reduction, and the label manifold is reconstructed automatically from logical multi-label data. The relationship between the feature and label manifolds avoids embedding or reducing dimensionality. The manifolds are located in two different spaces. According to the smoothness assumption, they only share a local topology structure. Our label manifold reconstruction process is similar to the Local Tangent Space-Alignment (LTSA) manifold learning method[10], which transfers the local topology from the feature space to the label space.

Belkin et al.[11] proposed a semi-supervised learning framework to study how to exploit and combine labeled and unlabeled data. In this framework, the regularized least-squares method is used for optimization. The use of the reproducing kernel Hilbert space provided a theoretical basis for the algorithm's proof. The method naturally extends to new samples and effectively uses unlabeled data, making it suitable for unsupervised and semi-supervised situations. The framework exports kernel-based algorithms for classification and regression. Niyogi[12] then discussed how to incorporate these ideas into geometry based on a kernel regularization framework. There is no similar unified study at present.

Chen et al.[13] pointed out that Laplacian support vector machines[14] and Laplacian regularized least squares[14] lay a solid foundation for the manifold regularization learning framework. However, because of the matrix's high computational cost, most of these optimization algorithms are limited to small-scale problems. In the Laplacian embedded regression framework[13], the algorithm has a simple form of kernel transformation that extracts the characteristics of the manifold structure and only needs to implement the inverse operation on a sparse matrix, which is more computationally efficient.

Moreover, Qiao et al.[15] proposed an adaptive linearized method of multipliers for a regularized optimization problem. This work inspired us to use the Lagrange multiplier method to solve the optimization problem.

In our approach, we use feature vectors (which reflect the feature manifold structure into the label space's loss function), mapping them into the subspace spanned by the overlapped patches after alignment in the tangent space, to better reflect the data manifold and solve the regularized least-squares problem via kernel transformation. Then, our approach transforms the local topological structure from the feature space to the label space to find the best solution of the optimization function, optimize the dimensionality reduction error, and improve system learning efficiency.

3 The Proposed Algorithm: LKLR

3.1 Formulation

The training set for MLL can be expressed as $D = \{(x_i, y_i)|1 \leqslant i \leqslant n\}$. Given any instance $x_i \in \Re^d$ where $\Re^d$ is a d-dimensional manifold space, and logic label $y_i \in \{+1, -1\}^q$, $\mu_i \in \Re^q$ is used to represent the numerical label vector[4]. Note that here -1 instead of 0 is used to indicate that the logical label vector is irrelevant to an instance. As in many graph-based learning methods, the topological structure is composed by graph $G = \langle \nu, \varepsilon, \omega \rangle$, where ν is the set of vertices,

and ε is a set of edges; each edge e_i^j represents the relationship between data x_i and x_j, and ω is the weight matrix, where each element w_i^j indicates the weight of edge e_i^j.

According to the smoothness assumption, the feature space topological structure is transferred locally to the local numerical label space. To maintain the locality, we must use the local neighborhood information of each point to construct the topological structure. In big data feature extraction, the feature manifold can be obtained by the overlapped local linear neighborhood patches via an alignment learning method[10]. In this paper, we approximate the feature manifold to induce the minimization of the reconstruction error E to construct the loss function:

$$E_i = T_i \left(I - \frac{1}{k} e e^{\mathrm{T}} \right) (I - U_i^{\mathrm{T}} U_i) \tag{1}$$

where $T_i = [t_{i_1}, \ldots, t_{i_k}]$ are the low-dimensional feature vectors of $X = [x_1, \ldots, x_n]$ in space $\Re^d$, I is the unit vector, k is the nearest neighbor coefficient. e is an N-dimensional unit column vector, matrix $U \in \Re^{m \times d}$ is an orthonormal basis of the affine subspace, $U = [U(x_1), \ldots, U(x_n)]^{\mathrm{T}}$, and U_i is the optimal low-ranking representation of the matrix $(X_n - \bar{X_n})(X_n - \bar{X_n})^{\mathrm{T}}$ (where $\bar{X_n}$ is the mean of X).

Then, we convert Eq. (1) to solve the approximate least-squares programming problem:

$$\min_{T_i} ||E_i||_{\mathrm{F}}^2 = \mathrm{Tr}(T_i R_i R_i^{\mathrm{T}} T_i^{\mathrm{T}}) \tag{2}$$

where Tr is the trace of matrix, $R_i = (I - \frac{1}{k} e e^{\mathrm{T}})(I - U_i^{\mathrm{T}} U_i)$, s.t. $T T^{\mathrm{T}} = I$.

The optimal solution of Eq. (2) is given by the eigenvectors corresponding to the first d smallest eigenvalues of $R_i R_i^{\mathrm{T}}$.

In the work of Hou et al.[4], the authors generalized the one-dimensional Support Vector Regressor (SVR) to solve a multidimensional case similar to Multi-dimensional Support Vector Regressor (MSVR). Additionally, they proposed a regressor based on MSVR by finding regressors w and b to minimize the multidimensional regression estimation problem and set an L_2 loss function.

$$L(w,b) = \frac{1}{2} \sum_{j=1}^{Q} ||w^j||^2 + C_1 \sum_{i=1}^{l} L_1(u_i) + C_2 \sum_{i=1}^{l} \sum_{j=1}^{Q} L_2(t_i^j) \tag{3}$$

where $W = [w^1, \ldots, w^Q]$, $b = [b^1, \ldots, b^Q]^{\mathrm{T}}$, C_1 and C_2 are penalized coefficients, L_1 and L_2 are horms, $u_i = ||e_i|| = \sqrt{e_i^{\mathrm{T}} e_i}$, $e_i^{\mathrm{T}} = y_i^{\mathrm{T}} - \phi(x_i)^{\mathrm{T}} W - b^{\mathrm{T}}$, and $t_i^j = y_i^j (\phi(x_i)^{\mathrm{T}} w^j + b^j)$.

We use the low-rank representation $U(x_i)$ in Eq. (1) to replace the nonlinear transformation $\phi(x_i)$, also known as the feature space in the loss function of Eq. (3).

To yield a single support vector for all dimensions, the L_1 loss function in Eq. (3) is set as the Vapnik ε-insensitive loss function.

$$L_1(u) = \begin{cases} 0, & u < \varepsilon; \\ u^2 - 2u\varepsilon + \varepsilon^2, & u \geqslant \varepsilon \end{cases} \tag{4}$$

To make the signs of the numerical and logical labels as similar as possible, the L_2 loss function in Eq. (3) is set as

$$L_2(t) = -t\sigma(-t) = \begin{cases} 0, & t > 0; \\ -t, & t \leqslant 0 \end{cases} \tag{5}$$

where $\sigma(-t)$ is an activation function with a value of 0 if $-t$ is negative, or 1 if otherwise.

To get an optimal solution in Eq. (3), we obtain w and b by setting the derivatives of L with respect to the least-squares problem $L''(w,b)$ to be 0.

$$\frac{\partial L''(w,b)}{\partial w} = \begin{cases} w^j - C_1 \sum\limits_i U(x_i)\alpha_i (y_i^j - U(x_i)^{\mathrm{T}} w^j - \\ b^j) = 0,\ t_i^j = y_i^j (U(x_i)^{\mathrm{T}} w^j + b^j) > 0; \\ w^j - C_1 \sum\limits_i U(x_i)\alpha_i (y_i^j - U(x_i)^{\mathrm{T}} w^j - \\ b^j) - C_2 \sum\limits_{i=1}^{l} \sum\limits_{j=1}^{Q} (y_i^j U(x_i)^{\mathrm{T}} + y_i^j b^j) = \\ 0, t_i^j \leqslant 0 \end{cases} \tag{6}$$

where

$$\alpha_i = \begin{cases} 0, & u_i < \varepsilon; \\ \frac{2(u_i - \varepsilon)}{u_i}, & u_i \geqslant \varepsilon \end{cases} \tag{7}$$

$$\frac{\partial L''(w,b)}{\partial b} = \begin{cases} -C_1 \sum\limits_i \alpha_i (y_i^j - U(x_i)^{\mathrm{T}} w^j - b^j) = 0, \\ t_i^j = y_i^j (U(x_i)^{\mathrm{T}} w^j + b^j) > 0; \\ -C_1 \sum\limits_i \alpha_i (y_i^j - U(x_i)^{\mathrm{T}} w^j - b^j) - \\ C_2 \sum\limits_{i=1}^{l} \sum\limits_{j=1}^{Q} y_i^j = 0, t_i^j \leqslant 0 \end{cases} \tag{8}$$

According to Refs. [4, 16], under fairly general conditions, the learning problem's linear system of Eqs. (6) and (8) can be expressed as a linear combination of the training examples in the feature space.

$$\begin{bmatrix} C_1U^{\mathrm{T}}D_\alpha U + I & C_1U^{\mathrm{T}}\alpha \\ C_1\alpha^{\mathrm{T}}U & C_1I^{\mathrm{T}}\alpha \end{bmatrix}\begin{bmatrix} w^j \\ b^j \end{bmatrix} = \begin{bmatrix} C_1U^{\mathrm{T}}D_\alpha y^j + C_2U^{\mathrm{T}}D_j y^j \\ C_1\alpha^{\mathrm{T}}y^j + C_2(\sigma^j)^{\mathrm{T}}y^j \end{bmatrix} \quad (9)$$

where, $\alpha = [\alpha_1, \ldots, \alpha_n]^{\mathrm{T}}$, $(D_\alpha)_i^k = \alpha_i\delta_i^k$, $(D_j)_i^k = \sigma(-t_i^j)\delta_i^k$, δ_i^k is Kronecker's delta function.

3.2 Constructing kernel space for optimization

A kernel trick can be applied to accommodate nonlinear predictive models. The kernel method is an approach that uses kernel functions to project the original data as an inner product to a high-dimensional eigenspace. The kernel method uses polynomial kernel or inner-product kernel transformations to convert the linear learning algorithms from a linear space to a kernel space, thereby transforming the unclassifiable points in the original linear space to linearly separable points in the kernel space. The kernel method solves the problem of poor separability of nonlinear data and exploits the kernel functions' generalization capability to improve out-of-sample learning ability (that is, the ability to learn outside the sample).

The kernel feature space (the inner product of the transformed vectors, i.e., $k(x_i, x_j) = \phi(x_i)^{\mathrm{T}}\phi(x_i)$ is typically used to replace the whole nonlinear mapping[16]. In a previous work[16], the predictive model was expressed as a linear combination of the training examples in the feature space.

If we insert this expression into the objective function, Eq. (9) is expressed as follows, where $K_{ij} = k(x_i, x_j)$ is the kernel matrix.

$$\begin{bmatrix} C_1(K + D_\alpha^{-1}) & C_1I \\ C_1\alpha^T K & C_1I^{\mathrm{T}}\alpha \end{bmatrix}\begin{bmatrix} w^j \\ b^j \end{bmatrix} = \begin{bmatrix} C_1U^{\mathrm{T}}D_\alpha y^j + C_2U^{\mathrm{T}}D_j y^j \\ C_1\alpha^{\mathrm{T}}y^j + C_2(\sigma^j)^{\mathrm{T}}y^j \end{bmatrix} \quad (10)$$

According to our previous work[17], the following three definitions reformulate the proposed method from the kernel perspective.

Definition 1. For a binary function $k\colon X \times X \to \mathbf{R}$, if there is an inner product space $(H, k(x, y))$ and mapping $\Phi\colon X \to H$, such that $k(x, y) = \langle\Phi(x), \Phi(y)\rangle$, then k is known as the kernel function, H is known as the feature space, and Φ is known as the feature mapping.

Definition 2. Given a kernel function k and the input $x_1, \ldots, x_n \in X$, $n \times n$-order matrix $K := (k(x_i, x_j))_{ij}$ is known as k's nuclear matrix with regard to $x_1, \ldots, x_n$ (or Gram matrix).

Definition 3. If the function $k\colon X \times X \to \mathbf{R}$ satisfies a symmetrical characteristic, i.e., $k(x, y) = k(y, x)$, and for any $n \in \mathbf{Z}^+, x_1, x_2, \ldots, x_n \in X$, $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbf{R}$, such that $\sum_{i,j=1}^{n} \alpha_i\alpha_j k(x_i, x_j) \geqslant 0$, then the function k is regarded as positive definite, and the corresponding matrix $K := (k(x_i, x_j))_{ij}$ is known as a positive definite matrix.

The kernel function choice is quite important. The kernel function describes the dataset characteristics. Thus, the kernel matrix is a description of the original dataset characteristics, and can effectively recover the dataset intrinsic nonlinear structure. The kernel matrix specifies the mapping from the original dataset to the feature space, and its decomposition avoids the feature decomposition in the high-dimensional feature space, thereby simplifying the computational complexity.

The kernel matrix is semi-positive definite and meets the conditions of kernel matrices. Therefore, our algorithm can be described in the kernel view given in the following theorem[17].

Theorem 1 The matrix K is a kernel matrix if and only if it is positive definite.

According to Ref. [18], the last step of LTSA is to search the low-dimensional embedding coordinates T, which can optimally minimize the matrix E. The cost function is as follows: $\min_{T_i} ||E_i||_{\mathrm{F}}^2 = \mathrm{Tr}(T_i R_i R_i^{\mathrm{T}} T_i^{\mathrm{T}})$. We replace part of this cost function with $M = RR^{\mathrm{T}}$. The original cost function optimization objective is expressed as $\mathrm{Tr}(TMT^{\mathrm{T}})$.

The coordinates T minimizing the cost function are the required value of d dimensional embedding coordinates, i.e., the eigenvectors according to d smallest eigenvalues of matrix M. In Ref. [18], LTSA kernel matrix is defined as $K = (\lambda_{\max}I - M)$, where $\lambda_{\max}$ is the largest eigenvalue of M. Because M is positive definite, it can be proved that K is also positive definite; therefore, the kernel matrix condition is satisfied, and the proof process is found in the following theorem[18].

Theorem 2 When $M = WW^{\mathrm{T}}$ is positive definite, K is also positive definite.

Proof The expression of eigen-decomposition on matrix M is $TM = T\lambda$. Multiply both sides by T^{T}: $TMT^{\mathrm{T}} = TT^{\mathrm{T}}\lambda = \lambda$, and by substituting $M = \lambda_{\max}I - K$, we get

$$\begin{aligned} &T(\lambda_{\max}I - K)T^{\mathrm{T}} = \lambda \Rightarrow \\ &T\lambda_{\max}IT^{\mathrm{T}} - TKT^{\mathrm{T}} = \lambda \Rightarrow \\ &\lambda_{\max}I - \lambda = TKT^{\mathrm{T}} \end{aligned} \quad (11)$$

Matrix M is positive definite, and we know M's eigenvalue λ is positive by equation $YM = Y\lambda$. While $\lambda_{\max}$ is M's largest eigenvalue, it is positive and $\lambda_{\max} I - \lambda > 0$. As a result, the equation is positive on both sides. Thus, K is also positive definite, satisfying the kernel matrix condition. ■

Because the feature manifold's construction process is obtained by aligning local linear neighborhood patches, we can convert the least-squares approach of Eq. (2) by minimizing the reconstruction error (when we approximate the feature manifold), thereby optimizing the nonlinear mapping $U(\cdot)$ with a kernel function. Then we design the kernel matrix $K = \lambda_{\max} I - M$ for the original cost function optimization $\min\limits_{T_i} ||E_i||_{\mathrm{F}}^2 = \mathrm{Tr}(T_i R_i R_i^{\mathrm{T}} T_i^{\mathrm{T}})$ in Eq. (2), where $M = RR^{\mathrm{T}}$. The inner product matrix M satisfies the positive definite constraints. The proof can be found in Ref. [18].

It is important to solve the problem of out-of-sample learning ability in applications such as data classification. Because the kernel function performs well in out-of-sample learning, a linear transformation is introduced in our algorithm. On the other hand, our algorithm works under the assumption that the local tangent space of the input sample point's coordinates can reconstruct the global embedding coordinates' geometric structure. Therefore, the kernel linear transformation introduced in our method can make the feature spaces of original samples guide their label spaces and make both promote each other.

We therefore add a linear transformation to our algorithm's objective function in the kernel framework, using the Lagrange multiplier method to solve the optimization problem and find a linear subspace suitable for classification. This objective function must satisfy the centralization constraint condition.

We bring the linear expression $T = U^{\mathrm{T}}X$ into the optimization function in kernel frame $\min \mathrm{Tr}(TKT^{\mathrm{T}})$, where $K = \lambda_{\max} I - M$, $M = RR^{\mathrm{T}} = \sum\limits_{i=1}^{n} R_i R_i^{\mathrm{T}}$, $R_i = (I - \frac{1}{k} ee^{\mathrm{T}})(I - U_i^{\mathrm{T}} U_i)$. The centralization constraint condition T is $TT^{\mathrm{T}} = I$. The optimization function becomes $\min\limits_{U} \mathrm{Tr}(U^{\mathrm{T}} XK(U^{\mathrm{T}}X)^{\mathrm{T}}) = \min\limits_{U} \mathrm{Tr}(U^{\mathrm{T}}(XKX^{\mathrm{T}})U)$. We rewrite the constraint condition as $TT^{\mathrm{T}} = U^{\mathrm{T}}X(U^{\mathrm{T}}X)^{\mathrm{T}} = I$, i.e., $U^{\mathrm{T}}XX^{\mathrm{T}}U = I$. The optimization function can be represented as a constrained objective function:

$$\begin{cases} \min\limits_{U} \mathrm{Tr}(U^{\mathrm{T}}XK(U^{\mathrm{T}}X)^{\mathrm{T}}) = \min\limits_{U} \mathrm{Tr}(U^{\mathrm{T}}(XKX^{\mathrm{T}})U), \\ U^{\mathrm{T}}XX^{\mathrm{T}}U = I \end{cases} \tag{12}$$

We then use a Lagrange multiplier method to solve Eq. (12). We bring the constraint into the optimization function using a Lagrangian multiplier method, construct Eq. (12) as an equation, and differentiate the variable U and make it equal to 0. We obtain the following expression:

$$\begin{aligned} L(U) &= U^{\mathrm{T}}(XKX^{\mathrm{T}})U + \lambda(I - U^{\mathrm{T}}XX^{\mathrm{T}}U) \Rightarrow \\ \frac{\partial L(U)}{\partial U} &= 2(XKX^{\mathrm{T}})U - 2\lambda XX^{\mathrm{T}}U = 0 \Rightarrow \\ (XKX^{\mathrm{T}})U &= \lambda XX^{\mathrm{T}}U \end{aligned} \tag{13}$$

The problem is now transformed into an eigen-decomposition of matrix XKX^{T} to obtain d largest eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_d$, with corresponding eigenvectors $u_1, u_2, \ldots, u_d$. From the aforementioned eigen equation (Eq. (13)), we can see that when the matrix U is composed of eigenvectors $u_1, u_2, \ldots, u_d$, the objective function (Eq. (12)) will take the minimum value. Therefore, the solutions of the optimization problem (Eq. (12)) are $u_1, u_2, \ldots, u_d$.

Using the coefficient matrix U, we can obtain the low-dimensional output coordinates from the correlation function $T = U^{\mathrm{T}}X$ between the previous local coordinates and low-dimensional global coordinates. After obtaining the explicit mapping function, the relevant low-dimensional coordinate of the new point is directly calculated through this expression. This makes our algorithm extremely useful for processing incremental data.

Then, we use the optimal solution direction of Eq. (10) as the descending direction for the optimization approach $L(w, b)$, and for solving the next iteration of w^{j+1} and b^{j+1}. The label information in the label manifold space is used to guide the unsupervised manifold learning method. At that point, the supervised information in the label manifold space can be used to improve feature extraction accuracy in the manifold space. Finally, the information mutually guides both the feature manifold and label manifold learning, enables them promote each other and accelerate convergence of the algorithm.

The main procedure of the LKLR algorithm is summarized as follows, and the input of the LKLP algorithm is shown in Algorithm 1.

(1) According to Eq. (2), R_i is constructed as $R_i = (I - \frac{1}{k} ee^{\mathrm{T}})(I - U_i^{\mathrm{T}} U_i)$. Then, we perform eigen-decomposition to obtain the eigenvectors corresponding to the first d minimum eigenvalues of $R_i R_i^{\mathrm{T}}$, thereby obtaining an optimal solution for Eq. (2).

Algorithm 1 LKLR algorithm

Input: $X = [x_1, \ldots, x_l, x_{l+1}, \ldots, x_N] \in \Re^{d \times N}$, X represents N data points, with part of them labeled as possibly sampled from a d-dimensional manifold.

Output: Y: the predicted label set for x

(2) Construct the function Eq. (3), where $W = [w^1, \ldots, w^Q]$, $b = [b^1, \ldots, b^Q]^{\mathrm{T}}$, $u_i = ||e_i|| = \sqrt{e_i^{\mathrm{T}} e_i}$, $e_i^{\mathrm{T}} = y_i^{\mathrm{T}} - \phi(x_i)^{\mathrm{T}} W - b^{\mathrm{T}}$. The low-ranking representation $U(x_i)$ in Eq. (1) is used to replace the nonlinear transformation $\phi(x_i)$ in the loss function $L(w, b)$. w and b are obtained by setting $L''(w, b)$ to 0, as shown in Eqs. (6) and (8).

(3) Construct the kernel matrix $K = \lambda_{\max} I - M$, $M = RR^{\mathrm{T}}$, where $\lambda_{\max}$ is the largest eigenvalue of M. According to the linear expression $T = U^{\mathrm{T}} X$, perform eigen-decomposition on the matrix XKX^{T} to obtain the coefficient eigenvectors $u_1, u_2, \ldots, u_d$, corresponding to d largest eigenvalues. The eigenvectors $u_1, u_2, \ldots, u_d$ replace the coefficient matrix U in Eq. (10).

(4) The optimal solution direction of objective function Eq. (10) is used as the descending direction for the optimization approach $L(w, b)$, and the solution for the next iteration of w^{j+1} and b^{j+1} is obtained in the process.

3.3 Algorithm convergence and computational complexity

Note that the direction of the optimal solution of Eq. (10) is the descending direction for the optimization of $L(w, b)$ in Eq. (3); the optimal solution's direction should be an aggregate of descending directions for each component to be estimated. That is, the objective value of Eq. (10) does not increase after each iteration. Thus, algorithm LKLR's objective function is repeated iteratively until a minimization solution is obtained. We therefore conclude that the algorithm is convergent.

The first step of algorithm LKLR is eigen-decomposition of $R_i R_i^{\mathrm{T}}$, and its computational complexity is $O(n^2)$. For the loss function $L(w, b)$, $L''(w, b)$ is a quadratic approximation from a first-order Taylor expansion $L'(w, b)$, which is a convex function. To obtain w and b, we need to solve the least-squares problem $L''(w, b)$. The loss function can be extended to multiple dimensions, but being based on an L_1 norm, it needs to account for each dimension independently, which makes the solution's complexity grow linearly with the number of dimensions. Therefore, the computational complexity of the second step of algorithm LKLR is $O(n)$. In the third step, a kernel matrix K is introduced into the objective function Eq. (10). As mentioned above, the direction of the optimal solution of Eq. (10) is used as the descending direction for the optimization of $L(w, b)$, and also as the direction of the next iteration of w^{j+1} and b^{j+1}. So the third step of algorithm LKLR has an upper bound. Note that the low-ranking matrix U is required to be smaller than $\min(d, n)$. In summary, the time complexity of each iteration of algorithm LKLR is $O(n^2)$.

4 Experiments

4.1 Experimental setup

4.1.1 Datasets

To compare our algorithm with state-of-the-art MLL algorithms, we selected 10 real multi-label datasets for performance evaluation. Table 1 summarizes the real datasets' detailed features selected from the Mulan website[19]. Half of them are regular-sized and half are large-scale; thus, they cover a wide range of multi-

Table 1 Characteristics of multi-label datasets.

	Dataset	S	T	dim(S)	$L(S)$	LCard(S)	LDen(S)	DL(S)	Domain
Regular-sized	Emotions	415	178	72	6	1.869	0.311	27	Music
	Medical	645	333	1449	45	1.245	0.028	94	Text
	Cal500	250	252	68	174	26.044	0.150	502	Music
	Birds	320	325	260	19	1.014	0.053	133	Audio
	Enron	1123	579	1001	53	3.378	0.064	753	Text
Large-scale	Yeast	1200	1217	103	14	4.237	0.303	198	Biology
	Image	1000	1000	294	5	1.236	0.247	20	Image
	Scene	1211	1196	294	6	1.074	0.179	15	Image
	Corel5k	2500	2500	499	374	3.522	0.009	3175	Image
	Bibtex	3700	3695	1836	159	2.402	0.015	2856	Text

label attributes. S is the number of examples, T is the number of testing samples, dim(S) denotes the feature dimensions, $L(S)$ represents the number of class labels, and LCard(S) is the label cardinality. Other multi-label statistics include the label density LDen(S), distinct label sets DL(S), and domain of datasets Domain.

4.1.2 Comparing algorithms

We chose four well-established MLL algorithms to compare with LKLR: ML^2[4], Multi-Label k-Nearest-Neighbor (ML-kNN)[6], Multi-Label Naive Bayes classifier (MLNB)[20], and MLL with Feature-induced labeling information Enrichment (MLFE)[21]. The number of ML^2's neighbors K is set to $q+1$, because K must be larger than q to generate a q-dimensional space using K vectors. Parameters λ, C_1, and C_2 are set to 1, 1, and 10, respectively, according to the work of Hou et al.[4]

4.1.3 Evaluation metrics

We selected five evaluation metrics that are widely used for MLL: Hamming Loss, Ranking Loss, One-Error, Coverage, and Macro-averaging Accuracy (AUC)[5]. For AUC, the larger the value, the better the performance. For the other four metrics, the smaller the values, the better the performance.

4.2 Experiment results

Tables 2 and 3 detail the experiment results of LKLR and the compared algorithms. We randomly selected half the examples on each dataset as a training set, and used the other half to form the test set.

The average performance of each dataset was recorded. For each evaluation metric, ↑ indicates the larger the better, and ↓ means the smaller the better. The tables show the best performance among the five comparison algorithms in boldface.

From the experimental results, we can see that for the regular-sized and large-scale datasets, LKLR ranks first in more than half of the evaluation metrics. This fully validates the effectiveness of LKLR for MLL.

4.3 Time performance comparison

To further illustrate the superiority of LKLR algorithm, the comparison of time performance can be also found in Tables 2 and 3. According to the tables, our algorithm computes effectively and ranks highly when compared to well-established MLL algorithms. Combined with the results of recognition accuracy experiments from the previous section, we ascertain that LKLR performs well

Table 2 Predictive performance of each compared algorithm (mean value) on regular-sized datasets.

Compared algorithm	Hamming Loss↓				
	Emotions	Medical	Cal500	Birds	Enron
LKLR	**0.2378**	0.0174	0.1976	0.1265	0.2924
ML^2	0.2388	0.0114	0.1578	0.0636	0.0546
ML-kNN	0.2706	0.0153	0.1416	**0.0546**	0.0620
MLNB	0.2804	0.0339	**0.1395**	0.0779	0.1145
MLFE	0.2434	**0.0112**	0.1549	0.0615	**0.0543**
Compared algorithm	Ranking Loss↓				
	Emotions	Medical	Cal500	Birds	Enron
LKLR	**0.1818**	0.1415	0.4615	**0.2084**	0.3145
ML^2	0.2228	0.1084	0.4721	0.3288	0.3210
ML-kNN	0.2724	0.0540	0.1928	0.3070	0.1220
MLNB	0.2150	0.0599	**0.1927**	0.2266	0.1768
MLFE	0.2061	**0.0209**	0.2089	0.3210	**0.0958**
Compared algorithm	One-Error↓				
	Emotions	Medical	Cal500	Birds	Enron
LKLR	**0.3333**	0.3947	**0.0759**	0.8421	0.6346
ML^2	0.5000	0.3421	0.0805	0.7895	0.6731
ML-kNN	0.4213	0.2492	0.1190	0.7356	0.3921
MLNB	0.4848	0.4234	0.1190	**0.5517**	0.5233
MLFE	0.3708	**0.1471**	0.1984	0.7471	**0.2608**
Compared algorithm	Coverage↓				
	Emotions	Medical	Cal500	Birds	Enron
LKLR	**0.1483**	0.5973	0.2282	**0.2695**	0.4401
ML^2	0.160	0.5236	0.2302	0.2836	0.4523
ML-kNN	0.2247	0.3441	**0.1319**	0.3606	0.1631
MLNB	0.2871	0.1925	0.1346	0.2695	0.2313
MLFE	0.1887	**0.1475**	0.1354	0.3763	**0.1495**
Compared algorithm	Macro-averaging AUC ↑				
	Emotions	Medical	Cal500	Birds	Enron
LKLR	**0.8573**	**0.9491**	0.7388	0.6407	0.7512
ML^2	0.7764	0.8904	**0.7758**	0.6430	**0.9056**
ML-kNN	0.7142	0.7695	0.5054	0.6173	0.5512
MLNB	0.6807	0.5227	0.5120	0.6955	0.5569
MLFE	0.7901	0.8745	0.5377	**0.7047**	0.6581
Compared algorithm	Time↓				
	Emotions	Medical	Cal500	Birds	Enron
LKLR	**0.0001**	**0.0156**	**0.0001**	**0.0001**	**0.1560**
ML^2	0.0625	0.2496	0.1404	0.0001	0.1716
ML-kNN	0.0312	0.2184	0.2028	0.0468	0.4992
MLNB	0.8899	0.5269	0.2186	0.7372	0.8964
MLFE	0.0312	0.2652	1.0452	0.1560	0.5148

Table 3 Predictive performance of each compared algorithm (mean value) on large-scale datasets.

Compared algorithm	Hamming Loss↓				
	Yeast	Image	Scene	Corel5k	Bibtex
LKLR	0.3058	0.2054	0.1558	0.2251	0.0921
ML^2	0.2073	0.1642	**0.0847**	0.0098	0.0126
ML-kNN	**0.1980**	0.1862	0.0989	**0.0094**	0.0136
MLNB	0.2166	0.2300	0.1299	0.0145	0.0824
MLFE	0.2038	**0.1616**	0.0903	0.0101	**0.0124**
Compared algorithm	Ranking Loss↓				
	Yeast	Image	Scene	Corel5k	Bibtex
LKLR	0.2973	**0.1372**	**0.0592**	0.4392	0.0914
ML^2	0.3022	0.1467	0.0612	0.4177	**0.0897**
ML-kNN	**0.1715**	0.1927	0.0931	0.2663	0.2234
MLNB	0.2323	0.2420	0.1124	**0.1267**	0.1584
MLFE	0.1777	0.1443	0.0713	0.3156	0.0921
Compared algorithm	One-Error↓				
	Yeast	Image	Scene	Corel5k	Bibtex
LKLR	**0.0714**	**0**	**0**	0.9390	**0.3585**
ML^2	0.2857	0.20000	0	0.9360	0.3899
ML-kNN	0.2345	0.3600	0.2425	0.7892	0.6225
MLNB	0.4170	0.4390	0.2851	0.8804	0.5876
MLFE	0.2356	0.2680	0.2157	**0.7832**	0.3710
Compared algorithm	Coverage↓				
	Yeast	Image	Scene	Corel5k	Bibtex
LKLR	0.8690	0.9686	0.9745	**0.1813**	**0.2417**
ML^2	0.8749	0.9510	0.9282	0.1827	0.2472
ML-kNN	0.6414	1.0420	0.5686	0.1978	0.5723
MLNB	**0.2499**	1.2450	0.6564	0.2102	0.3819
MLFE	0.6503	**0.8410**	**0.4582**	0.2238	0.2586
Compared algorithm	Macro-averaging AUC ↑				
	Yeast	Image	Scene	Corel5k	Bibtex
LKLR	**0.8243**	0.8573	0.9287	**0.6358**	0.9044
ML^2	0.8228	0.8555	0.9329	0.5974	**0.9261**
ML-kNN	0.6642	0.8187	0.9108	0.5233	0.6528
MLNB	0.6936	0.7788	0.8993	0.3559	0.8209
MLFE	0.6996	**0.8617**	**0.9385**	0.5549	0.8672
Compared algorithm	Time↓				
	Yeast	Image	Scene	Corel5k	Bibtex
LKLR	**0.1404**	**0.0936**	0.1560	1.3572	**4.4772**
ML^2	0.2808	0.1248	**0.1404**	**1.3260**	4.5864
ML-kNN	0.5148	0.2808	0.5928	4.1028	10.1869
MLNB	0.4108	0.4365	0.6040	7.0152	9.3436
MLFE	0.4680	0.6084	0.4056	12.4957	12.3397

on average, and is significant in label manifold learning.

4.4 Friedman test and critical difference diagram

To systematically analyze the algorithm relative performance, we use the Friedman test[22]. This statistical test compares relative performance among multiple algorithms over a number of datasets. Table 4 summarizes the Friedman statistics F_F and the corresponding critical values on each evaluation metric. As Table 4 shows, the null hypothesis of indistinguishable performance among the algorithms is rejected at a 0.05 significance level on each evaluation metric. Consequently, the Bonferroni-Dunn test[22] is employed as a post hoc test, to show the relative performances of the algorithms. Here, LKLR is treated as the control algorithm. The average rank difference between LKLR and a compared algorithm is calibrated using Critical Difference (CD). Accordingly, the performance between LKLR and a compared algorithm is deemed significantly different if their average ranks differ by at least one CD (CD = 1.766 in this paper: number of comparing algorithms k = 5, number of datasets N = 10).

Figure 1 illustrates the CD diagrams[22] for each evaluation metric and Fig. 2 illustrates the CD diagrams for time complexity, where the average rank of each compared algorithm is marked along the axis (lower ranks to the right). In each subfigure, any compared algorithm whose average rank is within one CD of LKLR interconnects with a thick line. Otherwise, it is considered to have a significantly different performance from LKLR. Based on the experimental results, we make the following observations:

(1) LKLR achieves an optimal average rank in terms of the Macro-averaging AUC evaluation metric (Fig. 1d). Furthermore, LKLR performs second best for all the other evaluation metrics.

(2) LKLR significantly outperforms MLNB for all

Table 4 Friedman statistics F_F, in terms of each evaluation metric and the critical value at 0.05 significance level (number of comparing algorithms k = 5, number of datasets N =10).

Evaluation metric	F_F	Critical value
Hamming Loss	9.07	2.498
Ranking Loss	1.04	
One-Error	1.60	
Coverage	0.30	
Macro-averaging AUC	17.79	
Time	21.41	

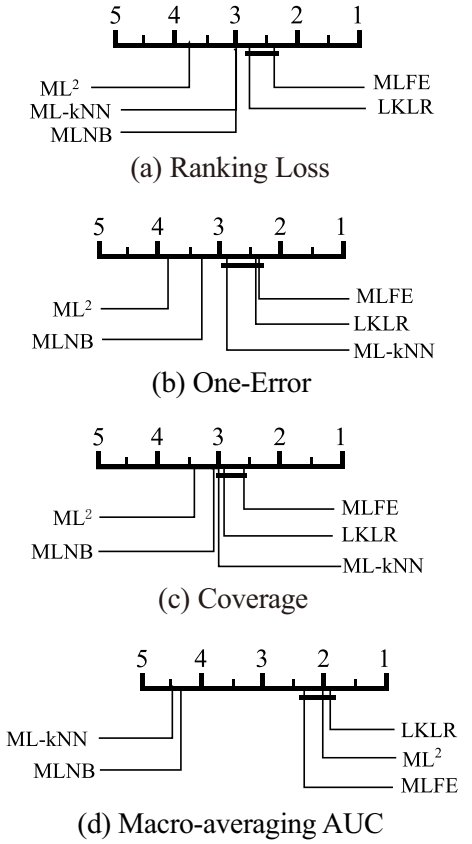

Fig. 1 Comparison of LKLR (the control algorithm) with four algorithms based on several evaluation metrics using the Bonferroni-Dunn test. Algorithms not connected with LKLR in the CD diagram are considered to have significantly different performance from the control algorithm (CD = 1.766 at 0.05 significance level).

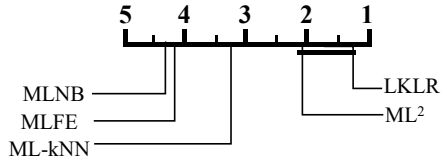

Fig. 2 Comparison of LKLR (the control algorithm) with four algorithms on time complexity using the Bonferroni-Dunn test. Algorithms not connected with LKLR in the CD diagram are considered to have significantly different performance from the control algorithm (CD = 1.766 at 0.05 significance level).

the evaluation metrics.

(3) LKLR is comparable to ML^2 in terms of time complexity, and significantly outperforms MLNB, MLFE, and ML-kNN for the time-complexity evaluation metric.

5 Conclusion

The ability to mine and understand data in an automated fashion continues to be an important feature in machine learning. A promising approach to aid such learning is to map instances and then assign labels. Then, according to the smoothness assumption, feature and label manifolds can share a local topological structure. Here, we propose a new approach to study the manifold structure in the Euclidean label space, and exploit a feature-extraction method based on tangent space alignment to guide the label manifold reconstruction while preserving the neighborhood information. Our algorithm, the LKLR method, implemented the reconstruction process. After establishing a learning model for feature extraction in the label space, we used a feature-extraction method of tangent space alignment to guide the reconstruction of label manifolds. Then we used LKLR to realize the reconstruction process. Extensive testing shows that LKLR allows mutual guidance and promotion, and makes great strides in improving multi-label manifold learning in terms of learning accuracy and time complexity.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 61702270 and 41471371) and the Project funded by China Postdoctoral Science Foundation (No. 2017M621592).

References

[1] Z. Q. Wang, J. C. Xin, H. X. Yang, S. Tian, G. Yu, C. R. Xu, and Y. D. Yao, Distributed and weighted extreme learning machine for imbalanced big data learning, *Tsinghua Science and Technology*, vol. 22, no. 2, pp. 160–173, 2017.

[2] G. Tsoumakas and I. Katakis, Multi-label classification: An overview, *International Journal of Data Warehousing Mining*, vol. 3, no. 3, pp. 1–13, 2007.

[3] X. Geng, Label distribution learning, *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1734–1748, 2016.

[4] P. Hou, X. Geng, and M. L. Zhang, Multi-label manifold learning, in *Proc. 13th AAAI Conf. Artificial Intelligence*, Palo Alto, CA, USA, 2016, pp. 1680–1686.

[5] M. L. Zhang and Z. H. Zhou, A review on multi-label learning algorithms, *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1819–1837, 2014.

[6] M. L. Zhang and Z. H. Zhou, ML-KNN: A lazy learning approach to multi-label learning, *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[7] J. Fürnkranz, E. Hüllermeier, E. L. Mencía, and K. Brinker, Multilabel classification via calibrated label ranking, *Machine Learning*, vol. 73, no. 2, pp. 133–153, 2008.

[8] G. Tsoumakas, I. Katakis, and I. Vlahavas, Random k-labelsets for multilabel classification, *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 7, pp. 1079–1089, 2011.

[9] L. Sun, S. W. Ji, and J. P. Ye, Canonical correlation analysis

for multilabel classification: A least-squares formulation, extensions, and analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 194–200, 2011.

[10] Z. Y. Zhang and H. Y. Zha, Principal manifolds and nonlinear dimensionality reduction via tangent space alignment, *SIAM Journal on Scientific Computing*, vol. 26, no. 1, pp. 313–338, 2004.

[11] M. Belkin, P. Niyogi, and V. Sindhwani, Manifold regularization: A geometric framework for learning from labeled and unlabeled examples, *Journal of Machine Learning Research*, vol. 7, pp. 2399–2434, 2006.

[12] P. Niyogi, Manifold regularization and semi-supervised learning: Some theoretical analyses, *Journal of Machine Learning Research*, vol. 14, pp. 1229–1250, 2013.

[13] L. Chen, I. W. Tsang, and D. Xu, Laplacian embedded regression for scalable manifold regularization, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 6, pp. 902–915, 2012.

[14] M. Belkin, P. Niyogi, and V. Sindhwani, On manifold regularization, https://newtraell.cs.uchicago.edu/files/tr_authentic/TR-2004-05.pdf, 2005.

[15] L. B. Qiao, B. F. Zhang, X. C. Lu, and J. S. Su, Adaptive linearized alternating direction method of multipliers for non-convex compositely regularized optimization problems, *Tsinghua Science and Technology*, vol. 22, no. 3, pp. 328–341, 2017.

[16] D. Tuia, J. Verrelst, L. Alonso, F. Prez-Cruz, and G. Camps-Valls, Multioutput support vector regression for remote sensing biophysical parameter estimation, *IEEE Geoscience and Remote Sensing Letters*, vol. 8, no. 4, pp. 804–808, 2011.

[17] C. Tan, C. Chen, and J. H. Guan, A nonlinear dimension reduction method with both distance and neighborhood preservation, in *Knowledge Science, Engineering and Management*, M. Wang, ed. Springer, 2013, pp. 48–63.

[18] C. Tan, J. H. Guan, and S. G. Zhou, IKLTSA: An incremental kernel LTSA method, in *Machine Learning and Data Mining in Pattern Recognition*, P. Perner, ed. Springer, 2015, pp. 70–83.

[19] Mulan, multi-label datasets for multi-label learning, http://mulan.sourceforge.net/datasets-mlc.html, 2018.

[20] M. L. Zhang, J. M. Peña, and V. Robles, Feature selection for multi-label naive Bayes classification, *Information Sciences*, vol. 179, no. 19, pp. 3218–3229, 2009.

[21] Q. W. Zhang, Y. Zhong, and M. L. Zhang, Feature-induced labeling information enrichment for multi-label learning, in *Proc. 32nd AAAI Conf. Artificial Intelligence*, New Orleans, LA, USA, 2018, pp. 4446–4453.

[22] J. Demšar, Statistical comparisons of classifiers over multiple data sets, *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

Chao Tan received the BEng and MEng degrees from Southeast University in 2005 and 2009, respectively, and received the PhD degree in computer science and technology from Tongji University in 2015. She joined Nanjing Normal University as a lecturer in 2015 and is an associate professor at present. She is now also a postdoctoral researcher in Southeast University. Her research interests generally focus on machine learning, multi-label manifold learning, and data mining.

Genlin Ji received the BEng and MEng degrees from Nanjing University of Aeronautics and Astronautics in 1986 and 1989, respectively, and received the PhD degree from Southeast University in 2004. He is now a professor in Nanjing Normal University. His research interests generally focus on data mining and its application.