Neural Comput & Applic
DOI 10.1007/s00521-017-3102-9

ORIGINAL ARTICLE

When sparse coding meets ranking: a joint framework for learning sparse codes and ranking scores

Jim Jing-Yan Wang[1] · Xuefeng Cui[1] · Ge Yu[2] · Lili Guo[2] · Xin Gao[1]

Received: 31 October 2016 / Accepted: 15 June 2017

Abstract Sparse coding, which represents a data point as a sparse reconstruction code with regard to a dictionary, has been a popular data representation method. Meanwhile, in database retrieval problems, learning the ranking scores from data points plays an important role. Up to now, these two problems have always been considered separately, assuming that data coding and ranking are two independent and irrelevant problems. However, is there any internal relationship between sparse coding and ranking score learning? If yes, how to explore and make use of this internal relationship? In this paper, we try to answer these questions by developing the first joint sparse coding and ranking score learning algorithm. To explore the local distribution in the sparse code space, and also to bridge coding and ranking problems, we assume that in the neighborhood of each data point, the ranking scores can be approximated from the corresponding sparse codes by a local linear function. By considering the local approximation error of ranking scores, the reconstruction error and sparsity of sparse coding, and the query information provided by the user, we construct a unified objective function for learning of sparse codes, the dictionary and ranking scores. We further develop an iterative algorithm to solve this optimization problem.

Keywords Database retrieval · Data representation · Sparse coding · Learning to rank · Nearest neighbors

1 Introduction

Sparse coding is a popular data representation method [11]. It tries to reconstruct a given data point as a linear combination of some basic elements in a dictionary, which are referred to as codewords. The linear combination coefficients are imposed to be sparse, e.g., most of the combination coefficients are zeros. The linear combination coefficient vector of a data point can be used as its new representation, and we call it the sparse code due to its sparsity. Because of its ability to explore the latent part-based nature of the data, it has been widely used to represent data in pattern classification, image understanding, and database retrieval problems. Many sparse coding algorithms were proposed to learn the dictionary and sparse codes [2, 8, 11, 21–23].

Meanwhile, in nearest neighbor-based classification and content-based database retrieval problems, the data points are usually ranked according to their similarity measures to the queries. The similarity measures are referred to as the ranking scores. Recently, methods to learn the ranking scores from the data points were proposed and showed their power in retrieval problems [28]. By considering both the query information provided by the users and the distribution of the data points, efficient algorithms were

✉ Xin Gao
xin.gao@kaust.edu.sa

Jim Jing-Yan Wang
jimjywang@gmail.com

Xuefeng Cui
xuefeng.cui@kaust.edu.sa

[1] Computer, Electrical and Mathematical Sciences and Engineering (CEMSE) Division, Computational Bioscience Research Center (CBRC), King Abdullah University of Science and Technology (KAUST), Thuwal 23955, Saudi Arabia

[2] Key Laboratory of Space Utilization, Technology and Engineering Center for Space Utilization, Chinese Academy of Sciences, Beijing 100094, China

Published online: 28 June 2017

developed to learn the ranking scores [4, 19, 20, 25, 27, 28].

It is possible to use both sparse coding and ranking score learning techniques to boost the performance of nearest neighbor searching. One may firstly map the data points to the sparse codes using a sparse coding algorithm, and then learn the ranking scores in the sparse code space. However, this strategy uses sparse coding and ranking methods independently and assumes that they are two irrelevant problems. In this paper, we ask the following two questions about sparse coding and ranking score learning:

1. Is there any internal relationship between sparse coding and ranking score learning?
2. If yes, how can we explore it to boost both the data representation and ranking simultaneously?

To answer these two questions, we propose to learn the sparse codes and ranking scores jointly to explore their internal relationship. Actually, in [14], Mairal et al. proposed to learn sparse codes, a dictionary and a classifier jointly to explore the internal relationship between sparse coding and classification. However, up to now, there is no existing work considering both sparse coding and ranking problems simultaneously.

To this end, we propose to perform sparse coding to all the data points and use the query information provided by the user to regularize the learning of the ranking scores. More importantly, to bridge the learning of sparse codes and ranking scores, and also to utilize the local distribution of the data points, we assume that in a local neighborhood of each data point, the ranking scores can be approximated from the sparse codes using a local linear function. By considering the reconstruction error and sparsity of the sparse coding problem, the local approximation error and the complexity of local ranking score approximation, and the query information regularization problems simultaneously, we construct a unified objective function for learning of the sparse codes, the dictionary and ranking scores. By optimizing this objective function, sparse codes and ranking scores can regularize the learning of each other, and thus the internal relationship can be explored. An iterative algorithm is developed to optimize the objective function with regard to the sparse codes, the dictionary and ranking scores, using the alternate optimization strategy.

The rest parts of this paper are organized as follows: in Sect. 2, we briefly introduce related works on sparse coding and ranking score learning. In Sects. 3 and 4, we introduce the proposed joint sparse coding and ranking score learning method. In Sect. 5, we show the performance of the proposed algorithm on nearest neighbor retrieval problems using six benchmark data sets. In Sect. 6, the paper is concluded with some future work.

2 Related work

Since our method is based on sparse coding, ranking score learning and local learning, we give some brief introduction to the relevant works. The most widely used sparse coding method was proposed by Lee et al. [11], which is based on iteratively solving a ℓ_2-constrained least square problem and a ℓ_1-regularized least square problem. The solutions can achieve a significant speedup for sparse coding. This method ignores the local manifold structure of the distribution of the data points. To solve this issue, Gao et al. [8] proposed Laplacian sparse coding (LapSc) to explore the local manifold structure of the data set, which is presented by a nearest neighbor graph, and used it to regularize the learning of sparse codes. A nearest neighbor graph is constructed from the data points to present the local manifold structure, and the learned sparse codes of neighboring data points are imposed to be close.

In the learning to rank problems, Zhou et al. [28] also used a nearest neighbor graph to regularize the learning of ranking scores. A disadvantage of using nearest neighbor graphs is that the ranking performance is usually sensitive to the graph parameters. To solve this problem, Yang et al. [27] proposed a local regression and global alignment (LRGA) algorithm to use a local linear function to predict the ranking scores from the original data points in the neighborhood of each data point to explore the local manifold information.

Although local manifold information has been utilized to improve the learning of both sparse codes and ranking scores [8, 27, 28], it is still not clear if there is any connection between the sparse codes and the ranking scores in a local manifold context. In this paper, we will try to predict ranking scores from the sparse codes in the local neighborhood of each data point to explore such a connection.

3 Joint learning of sparse coding and ranking

In this section, we will introduce the proposed unified sparse coding and ranking score learning method.

3.1 Problem formulation

Assume we have a data set of n data points, denoted as $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional feature vector of the ith data point. In this data set, one point is provided by the user, which is named as the query, while the remaining data points are from a given database. To indicate the query data point, we define a query indicator vector $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_n] \in \{1, 0\}^n$, where $\lambda_i = 1$ if $\mathbf{x}_i$ is a query, and 0 otherwise. The problem of data retrieval is to return some data points from the database which are the most similar to the

query. To this end, ranking scores are learned for the data points as similarities to the query so that the data points can be ranked according to the ranking scores, and the top ranked data points are returned as the retrieval results. The ranking scores of the data points in $\mathcal{X}$ are organized in a ranking score vector $\mathbf{f} = [f_1, \ldots, f_n] \in \mathbb{R}^n$, where f_i is the ranking score for the ith data point. To learn the ranking score, we represent the data points as sparse codes of a dictionary first, and meanwhile learn the ranking scores from the sparse codes and query information. The following problems are considered to construct a unified objective function to learn both the sparse codes and the ranking scores.

– *Sparse coding* The sparse coding problem aims to learn a dictionary with m codewords $\{\mathbf{d}_l\}_{l=1}^m$, and reconstruct a data point $\mathbf{x}_i$ as a sparse linear combination of the codewords,

$$\mathbf{x}_i \approx \sum_{l=1}^{m} \mathbf{d}_l s_{il} = D\mathbf{s}_i, \tag{1}$$

where $D = [\mathbf{d}_1, \ldots, \mathbf{d}_m] \in \mathbb{R}^{d\times m}$ is the dictionary matrix, $\mathbf{d}_l \in \mathbb{R}^d$ is the lth codeword, and $\mathbf{s}_i = [s_{i1}, \ldots, s_{im}]^\top \in \mathbb{R}^m$ is the sparse code of $\mathbf{x}_i$. To learn the dictionary and the sparse codes of the data points, the following minimization problem is considered,

$$\begin{aligned} &\min_{D,\{\mathbf{s}_i\}_{i=1}^n} \sum_{i=1}^{n} \left(\|\mathbf{x}_i - D\mathbf{s}_i\|_2^2 + \alpha \|\mathbf{s}_i\|_1 \right), \\ &\text{s.t. } \|\mathbf{d}_l\|_2^2 \le C, l = 1, \ldots, m, \end{aligned} \tag{2}$$

where $\|\mathbf{x}_i - D\mathbf{s}_i\|_2^2$ is the reconstruction error of the ith data point measured by the squared ℓ_2-norm distance, $\|\mathbf{s}_i\|_1$ is a ℓ_1-norm based sparsity measure of the sparse code $\mathbf{s}_i$, and α is a tradeoff parameter. In (2), we use the same loss function as regression problem, but the target and variables are different. In the regression problem, the target is to proximate the response of each data point, while in our problem, the target is to reconstruct the input feature vector of each data point. Meanwhile, the variables of our formula are the sparse codes and the dictionary, while in the regression problem, the variable is the parameter of the regression function. The motive of (2) is to learn reconstructive representations which are discriminative and effective for ranking problem. We can also use the regression model solution to solve the optimization problem in (2).

– *Local ranking score learning* To unitize the local structure of the sparse code space, we propose to learn a local linear function for the neighborhood of each data point to approximate the ranking scores. The set of the k-nearest neighboring data points of $\mathbf{x}_i$ is denoted as $\mathcal{N}_i$. The impact of the k-nearest neighboring to the performance is not significant. In our experiments we have varied the number of k from 5 to 30, and we did not observed much change over the performance. In the experiment, we set $k = 10$ for data sets of Yale B, USPS, and COIL100, and $k = 5$ for Glass, Climate, and Ionosphere. We use the ℓ_2 norm distance to calculate the k-nearest neighboring set for each point. We propose to learn a linear function $h_i(\mathbf{s}_j)$ to approximate the ranking scores $f_j|_{j:\mathbf{x}_j\in\mathcal{N}_i}$ of data points in this neighborhood from their sparse codes $\mathbf{s}_j|_{j:\mathbf{x}_j\in\mathcal{N}_i}$,

$$f_j \approx h_i(\mathbf{s}_j) = \mathbf{w}_i^\top \mathbf{s}_j, \tag{3}$$

where $\mathbf{w}_i \in \mathbb{R}^m$ is the parameter vector of the linear function of the $\mathcal{N}_i$. The ranking score is resented by sparse code $\mathbf{s}$ and $\mathbf{w}$, while $\mathbf{s}$ is resented by D and $\mathbf{x}$; thus, ranking score is indirectly resented by $\mathbf{s}$, $\mathbf{w}$, and D. To learn $\mathbf{w}_i$, we propose the following minimization problem for each $\mathcal{N}_i$,

$$\min_{\{\mathbf{s}_j f_j\}_{j:\mathbf{x}_j\in\mathcal{N}_i}, \mathbf{w}_i} \left(\sum_{j:\mathbf{x}_j\in\mathcal{N}_i} \left\| f_j - \mathbf{w}_i^\top \mathbf{s}_j \right\|_2^2 + \beta \|\mathbf{w}_i\|_2^2 \right), \tag{4}$$

where $\left\| f_j - \mathbf{w}_i^\top \mathbf{s}_j \right\|_2^2$ is the approximation error of ranking scores measured by squared ℓ_2-norm, $\|\mathbf{w}_i\|_2^2$ is a square ℓ_2-norm based regularization term used to control the complexity of the local linear function, and β is a tradeoff parameter. An overall problem is obtained by summing up the local minimization problems over all the data points,

$$\min_{\{\mathbf{s}_i f_i, \mathbf{w}_i\}_{i=1}^n} \sum_{i=1}^{n} \left(\sum_{j:\mathbf{x}_j\in\mathcal{N}_i} \left\| f_j - \mathbf{w}_i^\top \mathbf{s}_j \right\|_2^2 + \beta \|\mathbf{w}_i\|_2^2 \right). \tag{5}$$

Note that not only the local function parameters $\mathbf{w}_i|_{i=1}^n$ are to be solved, but also the sparse codes and ranking scores.

– *Query regularization* To unitize the query information provided by the users, we also regularize the learning of the ranking scores with the query indicator. If a data point is a query, its ranking score should be large since it is similar to itself. Thus we define a large value constant y and force the ranking scores of the queries to be close to it. The following minimization problem is obtained,

$$\min_{\{f_i\}_{i=1}^n} \sum_{i=1}^{n} \|f_i - y\|_2^2 \lambda_i. \tag{6}$$

In this problem, when a data point $\mathbf{x}_i$ is a query ($\lambda_i = 1$), we minimize the squared ℓ_2-norm distance between its ranking score f_i and the large constant value y. We set y as a constant value, 100.

The final optimization problem is obtained by combining the problems in (2), (5), and (6),

$$\begin{aligned}\min_{D,\{\mathbf{s}_i,f_i,\mathbf{w}_i\}|_{i=1}^n} & \left\{\sum_{i=1}^n\left(\|\mathbf{x}_i - D\mathbf{s}_i\|_2^2 + \alpha\|\mathbf{s}_i\|_1\right)\right. \\ & + \gamma\sum_{i=1}^n\left(\sum_{j:\mathbf{x}_j\in\mathcal{N}_i}\left\|f_j - \mathbf{w}_i^\top\mathbf{s}_j\right\|_2^2 + \beta\|\mathbf{w}_i\|_2^2\right) \\ & \left. + \delta\sum_{i=1}^n\|f_i - y\|_2^2\lambda_i\right\}, \\ \text{s.t. } & \|\mathbf{d}_l\|_2^2 \le C, l = 1,\ldots,m,\end{aligned} \tag{7}$$

where γ and δ are tradeoff parameters. In this problem, we need to solve a dictionary D, the corresponding sparse codes $\{\mathbf{s}_i\}_{i=1}^n$, the ranking scores $\{f_i\}_{i=1}^n$, and the local linear ranking score predictor parameters $\{\mathbf{w}_i\}_{i=1}^n$ of the data points. The learning of sparse codes and ranking scores are unified in a single optimization problem, and thus the learning of them are regularized by each other. This is the critical difference between the proposed method and the traditional independent sparse coding and ranking score learning algorithms which ignore the inherent connection between them.

3.2 Optimization

Directly solving this problem is difficult; thus, we adapt the alternate optimization strategy to solve it. The ranking scores, sparse codes and the dictionary are updated in an iterative algorithm. In each iteration, one of them is solved while the others are fixed, then their roles are switched. The iterations are repeated until a maximum iteration number is reached.

3.2.1 Solving ranking scores

When the ranking scores $\{f_i\}_{i=1}^n$ are being solved, we fix D and $\{\mathbf{s}_i\}_{i=1}^n$, remove the objective terms irrelevant to ranking scores from (7), and obtain the following problem,

$$\begin{aligned}\min_{\{f_i,\mathbf{w}_i\}|_{i=1}^n} & \left\{\gamma\sum_{i=1}^n\left(\sum_{j:\mathbf{x}_j\in\mathcal{N}_i}\left\|f_j - \mathbf{w}_i^\top\mathbf{s}_j\right\|_2^2 + \beta\|\mathbf{w}_i\|_2^2\right)\right. \\ & + \delta\sum_{i=1}^n\|f_i - y_i\|_2^2\lambda_i \\ & \left. = \gamma\sum_{i=1}^n g(S_i,\mathbf{f}_i,\mathbf{w}_i) + \delta\sum_{i=1}^n\|f_i - y_i\|_2^2\lambda_i\right\},\end{aligned} \tag{8}$$

where $g(S_i,\mathbf{f}_i,\mathbf{w}_i) = \sum_{j:\mathbf{x}_j\in\mathcal{N}_i}\left\|f_j - \mathbf{w}_i^\top\mathbf{s}_j\right\|_2^2 + \beta\|\mathbf{w}_i\|_2^2$ is defined as the local objective for each local ranking score learning problem of $\mathbf{x}_i$. To rewrite it in the matrix form, we define a local ranking score vector for each $\mathcal{N}_i$ as $\mathbf{f}_i = [f_{i1}\ldots,f_{ik}] \in \mathbb{R}^k$, where f_{ij} is the ranking score of the jth nearest neighbor point of $\mathbf{x}_i$. Similarly, we define a local sparse code matrix for each $\mathcal{N}_i$ as $S_i = [\mathbf{s}_{i1},\ldots,\mathbf{s}_{ik}] \in \mathbb{R}^{m\times k}$, where $\mathbf{s}_{ij}$ is the sparse code of the jth nearest neighbor point of $\mathbf{x}_i$. In this way, we rewrite $g(S_i,\mathbf{f}_i,\mathbf{w}_i)$ as

$$\begin{aligned}g(S_i,\mathbf{f}_i,\mathbf{w}_i) &= \sum_{j:\mathbf{x}_j\in\mathcal{N}_i}\left\|f_j - \mathbf{w}_i^\top\mathbf{s}_j\right\|_2^2 + \beta\|\mathbf{w}_i\|_2^2 \\ &= \left\|\mathbf{f}_i - \mathbf{w}_i^\top S_i\right\|_2^2 + \beta\|\mathbf{w}_i\|_2^2.\end{aligned} \tag{9}$$

The objective function of (8) is composed of the local objective functions of all data points; thus, this local objective function is to be minimized. To minimize this local objective function, we set its partial derivative with regard to $\mathbf{w}_i$ to zero,

$$\begin{aligned}&\frac{\partial g}{\partial\mathbf{w}_i} = -2S_i\mathbf{f}_i^\top + 2S_iS_i^\top\mathbf{w}_i + 2\beta\mathbf{w}_i = 0, \\ &\Rightarrow \mathbf{w}_i = \left(S_iS_i^\top + \beta I\right)^{-1}S_i\mathbf{f}_i^\top = \Phi_i\mathbf{f}_i^\top,\end{aligned} \tag{10}$$

where

$$\Phi_i = \left(S_iS_i^\top + \beta I\right)^{-1}S_i \in \mathbb{R}^{m\times k}. \tag{11}$$

By substituting it to (9), we can eliminate $\mathbf{w}_i$ from (9) and rewrite it as

$$\begin{aligned}g(S_i,\mathbf{f}_i) &= \left\|\mathbf{f}_i - \left(\Phi_i\mathbf{f}_i^\top\right)^\top S_i\right\|_2^2 + \beta\left\|\Phi_i\mathbf{f}_i^\top\right\|_2^2 \\ &= \left\|\mathbf{f}_i\left(I - \Phi_i^\top S_i\right)\right\|_2^2 + \beta\left\|\mathbf{f}_i\Phi_i^\top\right\|_2^2 \\ &= \mathbf{f}_i\left[\left(I - \Phi_i^\top S_i\right)\left(I - \Phi_i^\top S_i\right)^\top + \beta\Phi_i^\top\Phi_i\right]\mathbf{f}_i^\top \\ &= \mathbf{f}_iL_i\mathbf{f}_i^\top,\end{aligned} \tag{12}$$

where

$$L_i = \left[\left(I - \Phi_i^\top S_i\right)\left(I - \Phi_i^\top S_i\right)^\top + \beta\Phi_i^\top\Phi_i\right] \in R^{k\times k} \tag{13}$$

is a local regularization matrix for learning $\mathbf{f}_i$.

Moreover, to consider the summation of the local objective functions of all the data points in (8), we can rewrite $\mathbf{f}_i$ as the product of $\mathbf{f}$ and a nearest neighbor indicator matrix $H_i = \{1,0\}^{n\times k}$ for each $\mathcal{N}_i$ to indicate which data points are in $\mathcal{N}_i$. The (j,j')th element of H_i is defined as

$$H_{ijj'} = \begin{cases}1, & \mathbf{x}_j \text{ is the } j'\text{th nearest neighor of } \mathbf{x}_i, \\ 0, & \text{otherwise.}\end{cases} \tag{14}$$

Then $\mathbf{f}_i$ can be rewritten as,

$$\mathbf{f}_i = \mathbf{f}H_i. \tag{15}$$

Substituting both (12) and (15) to (8), the first term of (8) can be rewritten as

$$\begin{aligned}\sum_{i=1}^n g(S_i,\mathbf{f}_i) &= \sum_{i=1}^n\mathbf{f}_iL_i\mathbf{f}_i^\top \\ &= \sum_{i=1}^n\mathbf{f}H_iL_iH_i^\top\mathbf{f}^\top = \mathbf{f}\left(\sum_{i=1}^n H_iL_iH_i^\top\right)\mathbf{f}^\top.\end{aligned} \tag{16}$$

The second term of (8) can also be rewritten in a matrix form as,

$$\sum_{i=1}^{n} \|f_i - y\|_2^2 \lambda_i = (\mathbf{f} - \mathbf{y})\mathrm{diag}(\boldsymbol{\lambda})(\mathbf{f} - \mathbf{y})^\top, \tag{17}$$

where $\mathrm{diag}(\boldsymbol{\lambda}) \in \mathbb{R}^{n\times n}$ is a diagonal matrix with its diagonal vector as $\boldsymbol{\lambda}$, and $\mathbf{y} = [y, \ldots, y] \in \mathbb{R}^n$ is an n-dimensional vector with all its elements as y.

Finally, we substitute (16) and (17) to (8) and obtain the optimization problem with regard to the ranking score vector $\mathbf{f}$,

$$\min_{\mathbf{f}} \left\{ h(\mathbf{f}) = \gamma \mathbf{f} \left(\sum_{i=1}^{n} H_i L_i H_i^\top \right) \mathbf{f}^\top + \delta(\mathbf{f} - \mathbf{y})\mathrm{diag}(\boldsymbol{\lambda})(\mathbf{f} - \mathbf{y})^\top \right\}, \tag{18}$$

where $h(\mathbf{f})$ is the objective function for the problem of learning $\mathbf{f}$. This problem can be easily solved by setting the partial derivative of $h(\mathbf{f})$ with regard to $\mathbf{f}$ to zero,

$$\begin{aligned} &\frac{\partial h(\mathbf{f})}{\partial \mathbf{f}} = 2\gamma \mathbf{f} \left(\sum_{i=1}^{n} H_i L_i H_i^\top \right) + 2\delta(\mathbf{f} - \mathbf{y})\mathrm{diag}(\boldsymbol{\lambda}) = 0 \\ &\Rightarrow \mathbf{f} = \delta \mathbf{y} \mathrm{diag}(\boldsymbol{\lambda}) \left[\gamma \left(\sum_{i=1}^{n} H_i L_i H_i^\top \right) + \delta \mathrm{diag}(\boldsymbol{\lambda}) \right]^{-1}. \end{aligned} \tag{19}$$

3.2.2 Solving sparse codes

When the ranking scores $\{f_i\}_{i=1}^n$ and the dictionary D are fixed, and the terms irrelevant to sparse codes are removed, the optimization problem in (7) is reduced to,

$$\begin{aligned} \min_{\{\mathbf{s}_i, \mathbf{w}_i\}_{i=1}^n} &\sum_{i=1}^{n} \left(\|\mathbf{x}_i - D\mathbf{s}_i\|_2^2 + \alpha \|\mathbf{s}_i\|_1 \right) \\ &+ \gamma \sum_{i=1}^{n} \left(\sum_{j:\mathbf{x}_j \in \mathcal{N}_i} \left\| f_j - \mathbf{w}_i^\top \mathbf{s}_j \right\|_2^2 + \beta \|\mathbf{w}_i\|_2^2 \right). \end{aligned} \tag{20}$$

As indicated in (10), the optimal solution of $\mathbf{w}_i$ is also a function of the sparse codes of data points in $\mathcal{N}_i$. Directly solving this problem is complicated, and we choose to use an EM-like algorithm to solve it. In each iteration, $\mathbf{w}_i$ is firstly estimated using the sparse codes solved in the previous iteration, and then it is fixed when the sparse codes $\{\mathbf{s}_i\}_{i=1}^n$ are updated. Moreover, we also choose to update the sparse codes one by one. When the sparse code $\mathbf{s}_i$ is considered, the others $\{\mathbf{s}_j\}_{j;j\neq i}$ are fixed. This reduces the problem in (20) to

$$\min_{\mathbf{s}_i} \|\mathbf{x}_i - D\mathbf{s}_i\|_2^2 + \alpha \|\mathbf{s}_i\|_1 + \gamma \sum_{j:\mathbf{x}_i \in \mathcal{N}_j} \left\| f_i - \mathbf{w}_j^\top \mathbf{s}_i \right\|_2^2. \tag{21}$$

This problem can be easily solved by the feature-sign search algorithm [11].

3.2.3 Solving the dictionary

When ranking scores and sparse codes are fixed, only the dictionary D is considered, and the irrelevant terms are moved, the problem in (7) is turned to

$$\begin{aligned} &\min_{D} \sum_{i=1}^{n} \|\mathbf{x}_i - D\mathbf{s}_i\|_2^2, \\ &\text{s.t. } \|\mathbf{d}_k\|_2^2 \leq C, l = 1, \ldots, m. \end{aligned} \tag{22}$$

This is a typical dictionary learning problem of sparse coding, and it can be solved by the Lagrange dual method [11].

3.3 Algorithm

Based on the optimization results, we develop an iterative algorithm, which is shown in Algorithm 1. The iterations are repeated until it meets a maximum iteration time T.

Algorithm 1 Iterative joint sparse coding and ranking score learning algorithm.

Input: A training set of n data points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$, and a query indicator vector $\boldsymbol{\lambda}$;
Input: Tradeoff parameters α, β, γ and δ;
Initialize $\{\mathbf{s}_i^0\}_{i=1}^n$, D^0, and $t = 1$;
Find the k nearest neighbors $\mathcal{N}_i|_{i=1}^n$ and construct the nearest neighbor indicator matrix $H_i|_{i=1}^n$ for the data points $\mathbf{x}_i|_{i=1}^n$.
repeat
1. Update $\Phi_i^t|_{i=1}^n$ and $L_i^t|_{i=1}^n$ as in (11) and (13) by fixing the sparse codes as $\{\mathbf{s}_i^{t-1}\}_{i=1}^n$;
2. Update the ranking score vector $\mathbf{f}^t$ as in (19);
3. Update the local ranking score predictor parameters $\mathbf{w}_i^t|_{i=1}^n$ as in (10) by fixing $\Phi_i^t|_{i=1}^n$ and $\mathbf{f}_i^t|_{i=1}^n$;
4. Update the sparse codes $\mathbf{s}_i^t|_{i=1}^n$ one by one by solving (21) by fixing D^{t-1}, $\mathbf{w}_i^t|_{i=1}^n$ and $f_i^t|_{i=1}^n$;
5. Update the dictionary D^t by solving (22) by fixing $\mathbf{s}_i^t|_{i=1}^n$;
6. $t = t + 1$;

until $t \geq T$
Output: Ranking scores $f_i^{t-1}|_{i=1}^n$, sparse codes $\mathbf{s}_i^{t-1}|_{i=1}^n$ and a dictionary D^{t-1}.

4 Off-line and on-line extensions

A shortage of Algorithm 1 is its high computational complexity. To calculate a ranking vector for one single query, the dictionary and the sparse codes of all the data points are updated in each iteration. This is unacceptable for an on-line retrieval system especially when the database size is large. To solve this problem, we propose a two-step strategy including an off-line learning procedure to learn the dictionary and sparse codes of the data points of a database, and an on-line ranking procedure to learn the sparse code of a query and its ranking score vector.

4.1 Off-line learning of dictionary and sparse codes

In the off-line learning procedure, we only have the database of n data points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$, while not knowing the query. To regularize the learning of the dictionary and the sparse codes of the data points by ranking, we randomly select some presentative data points from the data set and treat them as queries. The selected query set is denoted as $\mathcal{Q} \subset \mathcal{X}$. For each query $\mathbf{x}_q \in \mathcal{Q}$, we want to learn a ranking vector $\mathbf{f}^q = [f_1^q, \ldots, f_n^q] \in \mathbb{R}^n$, where f_i^q is the ranking score of query $\mathbf{x}_q$ against the ith data point. To learn the sparse codes, the dictionary and the ranking score vectors of the queries, we extend (7)–(23) to consider multiple queries in $\mathcal{Q}$,

$$\begin{aligned}
&\min \Bigg\{ \sum_{i=1}^{n} \left(\|\mathbf{x}_i - D\mathbf{s}_i\|_2^2 + \alpha \|\mathbf{s}_i\|_1 \right) \\
&\quad + \sum_{q:\mathbf{x}_q \in \mathcal{Q}} \left[\gamma \sum_{i=1}^{n} \left(\sum_{j:\mathbf{x}_j \in \mathcal{N}_i} \left\| f_j^q - \mathbf{w}_i^{q\top} \mathbf{s}_j \right\|_2^2 + \beta \|\mathbf{w}_i^q\|_2^2 \right) \right. \\
&\quad \left. + \delta \sum_{i=1}^{n} \|f_i^q - y\|_2^2 \lambda_i^q \right] \Bigg\} \\
&w.r.t. D, \mathbf{s}_i|_{i=1}^n, \{f_i^q, \mathbf{w}_i^q\}|_{i,q:\mathbf{x}_i \in \mathcal{X}, \mathbf{x}_q \in \mathcal{Q}}, \\
&\text{s.t. } \|\mathbf{d}_l\|_2^2 \le C, l = 1, \ldots, m,
\end{aligned} \tag{23}$$

where $\mathbf{w}_i^q \in \mathbb{R}^m$ is the parameter vector of the linear function of the $\mathcal{N}_i$ to predict ranking scores of query $\mathbf{x}_q$ from the sparse codes, and $\lambda_i^q = 1$ if $\mathbf{x}_i$ is query $\mathbf{x}_q$, and 0 otherwise. To solve this problem, we adapt a similar alternate optimization strategy as the method used to solve (7). The sparse codes and the dictionary are solved in the same way as in Sects. 3.2.2 and 3.2.3, respectively. The ranking score vector for each query is solved independently as in Sect. 3.2.1.

4.2 On-line ranking

In the on-line ranking procedure, given the database $\mathcal{X}$ with n data points and a new query $\mathbf{x}_{n+1} \in \mathbb{R}^d$, we need to calculate an $n+1$ ranking score vector $\mathbf{f} = [f_1, \ldots, f_{n+1}] \in \mathbb{R}^{n+1}$ for the query. We already have the sparse codes $\mathbf{s}_1, \ldots, \mathbf{s}_n$ for data points in $\mathcal{X}$ and the dictionary D learned in the off-line procedure. Thus we only need to calculate the sparse code $\mathbf{s}_{n+1}$ of the new query data point $\mathbf{x}_{n+1}$, while fixing the sparse codes of the remaining data points. We extend (7)–(24) to consider the additional query $\mathbf{x}_{n+1}$ in the on-line retrieval procedure to learn its sparse code $\mathbf{s}_{n+1}$ and its ranking scores $f_i|_{i=1}^{n+1}$,

$$\begin{aligned}
\min_{\mathbf{s}_{n+1}, f_i|_{i=1}^{n+1}} \Bigg\{ & \left(\|\mathbf{x}_{n+1} - D\mathbf{s}_{n+1}\|_2^2 + \alpha \|\mathbf{s}_{n+1}\|_1 \right) \\
& + \gamma \sum_{i=1}^{n+1} \left(\sum_{j:\mathbf{x}_j \in \mathcal{N}_i} \left\| f_j - \mathbf{w}_i^\top \mathbf{s}_j \right\|_2^2 + \beta \|\mathbf{w}_i\|_2^2 \right) \\
& + \delta \sum_{i=1}^{n+1} \|f_i - y\|_2^2 \lambda_i \Bigg\},
\end{aligned} \tag{24}$$

where $\lambda_i = 1$ if $i = n+1$, and 0 otherwise. This problem can also be solved with an alternate optimization strategy. In an iterative algorithm, $\mathbf{s}_{n+1}$ and $f_i|_{i=1}^{n+1}$ are updated alternately. Moreover, we also assume the k-nearest neighbors in $\mathcal{N}_i$ of each $\mathbf{x}_i \in \mathcal{X}$ lie within $\mathcal{X}$ while not considering $\mathbf{x}_{n+1}$. In this way, in the on-line retrieval procedure, we only need to search the nearest neighbors $\mathcal{N}_{n+1}$ of $\mathbf{x}_{n+1}$, while leaving $\mathcal{N}_i|_{i=1}^n$ fixed. Actually, when we try to solve the ranking scores as in (19), the local regularization matrices $L_i|_{i=1}^n$ for the first n data points are the same as the ones calculated in the off-line learning procedure and can be fixed, and we only need to update L_{n+1}. When $\mathbf{s}_{n+1}$ is solved, the first n local learning regularization terms can be ignored because $\mathbf{x}_{n+1}$ is not in any $\mathcal{N}_i|_{i=1}^n$, and only the regularization in $\mathcal{N}_{n+1}$ needs to be considered. Thus both the computations of $\mathbf{s}_{n+1}$ and $f_i|_{i=1}^{n+1}$ are low cost.

5 Experiments

To evaluate the proposed algorithm, we conducted experiments on six benchmark data sets and compared it to individual sparse coding and ranking score learning algorithms, as well as their simple combinations.

5.1 Data sets and setup

We used the Yale face database B [9], the USPS handwritten digit database [10], the COIL100 object image database [15], the glass identification data set [3], the climate model simulation crashes data set [13], and the ionosphere data set [16]. The statistical information of these data sets is given in Table 1. To conduct the retrieval experiments, we employed the fourfold cross-validation. A

Table 1 Statistical information of data sets

Data set	# Data points	# Classes	# Features
Yale B	2414	38	1024
USPS	9298	10	256
COIL100	7200	100	1024
Glass	214	6	10
Climate	540	2	18
Ionosphere	351	2	34

data set was split to fourfolds randomly, and each fold was used as a query set, while the remaining threefolds were combined and used as the database set. We first performed the off-line learning procedure on the database set and then performed the on-line ranking procedure to each query in the query set. The retrieval performance of the ranking is measured by the receiver operating characteristic (ROC) curve and the recall-precision curve. The area under ROC curve (AUC) was also used as a single performance measure [12].

5.2 Results

5.2.1 Comparison against independent sparse coding and ranking methods

We compared our joint sparse coding and ranking score learning algorithm with a state-of-the-art sparse coding method, LapSc [8], and a state-of-the-art ranking score learning algorithm, LRGA [27], and their simple combination, i.e., using LapSc to learn sparse codes and then using the sparse codes with LRGA to learn the ranking. Both of these two individual sparse coding and ranking algorithms are based on manifold learning. Note that we did not consider supervised sparse coding algorithms for fair comparison since the proposed algorithm is an

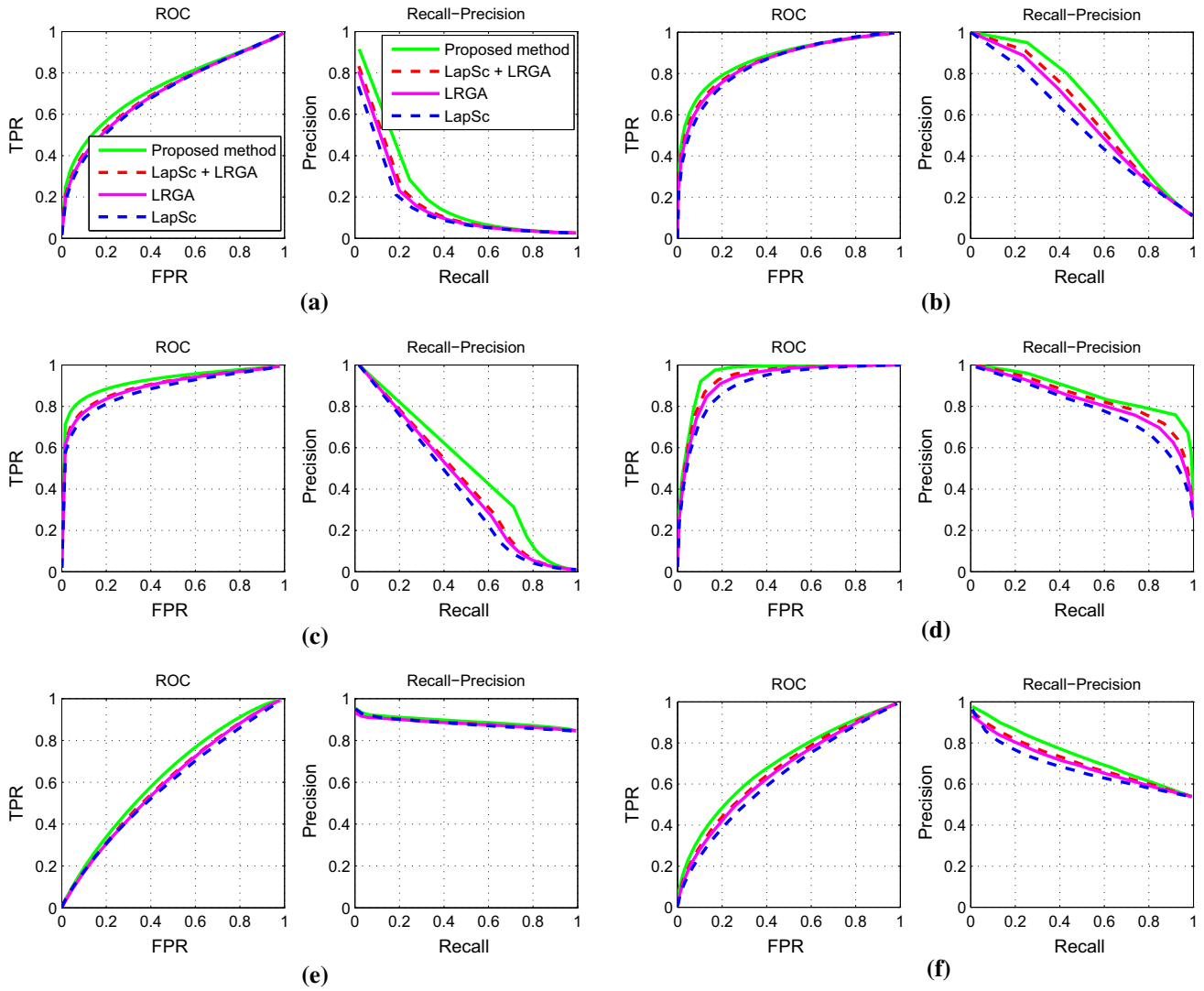

Fig. 1 ROC and recall-precision curves of different methods on six data sets. **a** Yale B. **b** USPS. **c** COIL100. **d** Glass. **e** Climate. **f** Ionosphere

Table 2 AUC values of different methods

Data set	Proposed method	LapSc + LRGA	LRGA	LapSc
Yale B	0.7333	0.7130	0.7091	0.7032
USPS	0.8524	0.8401	0.8365	0.8293
COIL100	0.9070	0.8834	0.8793	0.8637
Glass	0.9666	0.9492	0.9403	0.9216
Climate	0.6097	0.5902	0.5862	0.5821
Ionosphere	0.6946	0.6692	0.6589	0.6362

unsupervised learning algorithm. The ROC curves of the compared methods are given in Fig. 1. From these figures, we can see that the proposed method clearly outperforms the independent sparse coding algorithm, the ranking score learning algorithm, and their simple combination on the six different data sets. In all the plots, the ROC curves of the proposed method are closer to the top-left corner of the figures than any other method, while the recall-precision curves of the proposed method are closer to the top-right corner of the figures than other methods. This indicates an overall better retrieval performance. These are strong evidences of the advantage of the joint sparse coding and ranking method over the independent sparse coding and ranking methods. This claim can be further supported by the AUC values of ROC curves in Table 2. Over the six data sets, the proposed method achieves the highest AUC values. For example, for data set COIL100, only the proposed method achieves an AUC value higher than 0.90. Moreover, it is interesting to see that LRGA outperforms LapSc in most cases, while incorporating LapSc to LRGA in a simple way does not achieve significant improvement over LRGA. For example, in Fig. 1b, the recall-precision curve of LRGA is significantly closer to the top-right corner than that of LapSc, and the recall-precision curves of LRGA and the simple combination LRGA + LapSc are close. Although both LapSc and LRGA explore the manifold structure of the data set, LapSc applies manifold regularization in the sparse code space, while LRGA directly regularizes the ranking scores by the manifold. This means manifold learning in the representation space does not guarantee an effective ranking result from this space, and it is necessary to perform local learning to the ranking score space like LRGA. Moreover, performing LRGA in the sparse code space provided by LapSc can also improve the retrieval results, but the improvement is marginal. Only when sparse coding and ranking is performed jointly by the proposed method, significant improvements are achieved. This means sparse coding has the potential to improve the performance of ranking, but it is necessary to explore the inner relation between them.

5.2.2 *Sensitivity to parameters*

There are three tradeoff parameters α, β and γ in the objective function (7). We are also interested in the sensitivity of the proposed method to these parameters, and we plot the AUC values against different values of the parameters in Fig. 2. The parameter sensitivity analysis is performed over the Yale face database B. In Fig. 2a, we can see that AUC tends to increase when α is increased, indicating that a sparser representation can achieve better performance. However, it seems the performance is stable when a large value of α is given. From Fig. 2b, it can be seen that the proposed algorithm is stable to the parameter β, while from Fig. 2c, it seems that a large value of γ reduces the weight of local learning and obtains a lower AUC. This indicates the importance of the local learning.

5.2.3 *Analysis of algorithm convergence*

The proposed learning algorithm is an iterative algorithm, and we are also interested in the convergence of the algorithm. This we plot the loss function values against the numbers of iterations. The convergence curves over data sets of Yale B and Glass are shown in Fig. 3. According to the results in Fig. 3, the proposed algorithm converges after 30 iterations.

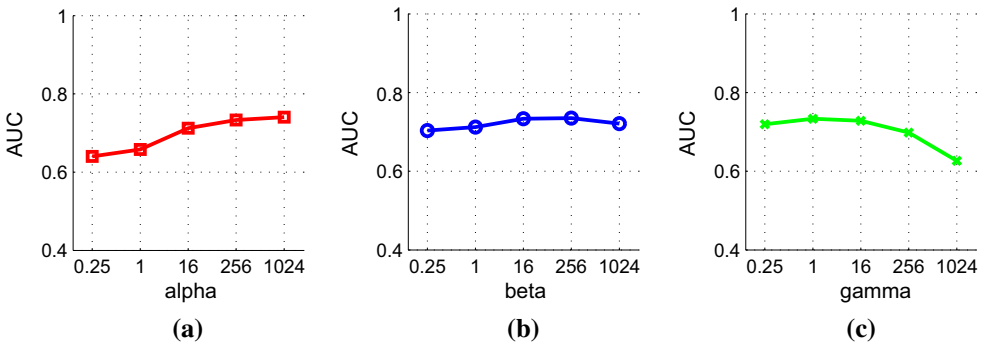

Fig. 2 Parameter sensitivity curves. **a** α. **b** β. **c** γ

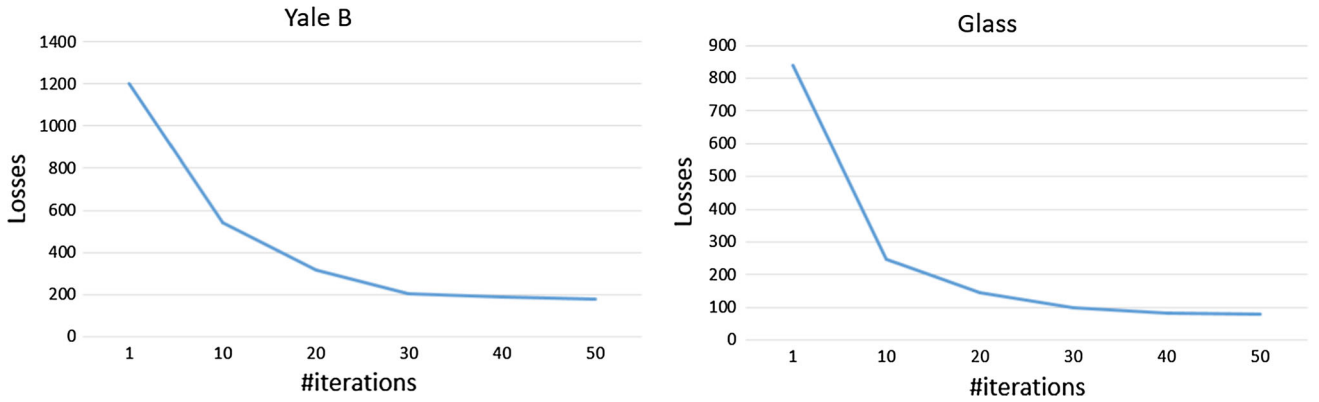

Fig. 3 Convergence curves

Table 3 Running time (s)

Data sets	Off-line training	On-line querying
Yale B	830.15	15.13
Glass	103.33	3.13

5.2.4 Running time analysis

We also provide an experimental analysis for the proposed algorithm. The running time of the off-line training and on-line querying processes over data sets of Yale B and Glass are shown in Table 3. According to the results reported in Table 3, the off-line training process consumes most of the running time. Meanwhile the running time of experiments over Yale B data set is much longer than that of data set Glass. This is also not surprising since our algorithm running time is sensitive to the size of the data set.

6 Conclusion and future work

Is there any internal relationship between a popular data representation method, sparse coding, and an important procedure of the nearest neighbor search problem, ranking score learning? To answer this question, in this paper, we assume such a relationship exists and propose to explore it by using a local linear function to approximate the ranking scores from the sparse codes in the local neighborhood of each data point. A unified objective function is constructed based on the local learning of ranking scores from sparse codes, and also based on the sparse coding and query information regularization problems. By iteratively optimizing it with regard to the sparse codes, the dictionary, and ranking scores, we develop the first joint sparse coding and ranking score learning algorithm. If the assumption holds, it is expected that the joint method which takes the advantage of this internal relationship should outperform the independent sparse coding and ranking algorithms which ignore this relationship. The proposed algorithm demonstrates superior performance over the existing sparse coding algorithm, the ranking score learning algorithm, and their simple combination. This verifies our assumption and reveals the existence of the internal relationship between the sparse coding and ranking score learning problems.

In the future, we will extend the proposed method to big data ranking, by using distributed computing models [1, 17, 18]. Moreover, we will investigate more representation methods for ranking purpose besides sparse coding, such as using Bayesian networks for data representation and ranking score learning [5–7]. In the proposed model, we use a simple squared ℓ_2-norm distance to measure the loss of ranking score learning. However, in the test process, we use the AUC as the performance measure. In the future, we will also study how to minimize a loss function that directly corresponds to AUC instead of the squared ℓ_2-norm distance to obtain the optimal performance measure directly [24, 26].

Acknowledgements The research reported in this publication was supported by funding from King Abdullah University of Science and Technology (KAUST) and the National Natural Science Foundation of China under the Grant No. 61502463.

Compliance with ethical standards

Conflict of interest The authors declare no conflict of interests.

References

1. Al Marri WJ, Malluhi Q, Ouzzani M, Tang M, Aref WG (2016) The similarity-aware relational database set operators. Inf Syst 59:79–93
2. Al-Shedivat M, Wang JJY, Alzahrani M, Huang J, Gao X (2014) Supervised transfer sparse coding. In: AAAI, vol 3, pp 1665–1672
3. Evett IW, Spiehler EJ (1987) Rule induction in forensic science. Tech. rep, Central Research Establishment, Home Office Forensic Science Service

4. Fan J, Liang RZ (2016) Stochastic learning of multi-instance dictionary for earth mover's distance-based histogram comparison. Neural Comput Appl. doi:10.1007/s00521-016-2603-2
5. Fan X, Malone B, Yuan C (2014) Finding optimal Bayesian network structures with constraints learned from data. In: UAI, pp 200–209
6. Fan X, Yuan C (2015) An improved lower bound for Bayesian network structure learning. In: AAAI, pp 3526–3532
7. Fan X, Yuan C, Malone B (2014) Tightening bounds for Bayesian network structure learning. AAAI 4:2439–2445
8. Gao S, Tsang IW, Chia LT, Zhao P (2010) Local features are not lonely—Laplacian sparse coding for image classification. In: CVPR, pp 3555–3561
9. Georghiades A, Belhumeur P, Kriegman D (2001) From few to many: illumination cone models for face recognition under variable lighting and pose. TPAMI 23(6):643–660
10. Kaynak C (1995) Methods of combining multiple classifiers and their applications to handwritten digit recognition. Master's thesis, Institute of Graduate Studies in Science and Engineering, Bogazici University
11. Lee H, Battle A, Raina R, Ng AY (2006) Efficient sparse coding algorithms. In: NIPS, pp 801–808
12. Liang RZ, Shi L, Wang H, Meng J, Wang JJY, Sun Q, Gu Y (2016) Optimizing top precision performance measure of content-based image retrieval by learning similarity function. In: 2016 23rd International conference on pattern recognition (ICPR), pp 2954–2958. IEEE
13. Lucas DD, Klein R, Tannahill J, Ivanova D, Brandon S, Domyancic D, Zhang Y (2013) Failure analysis of parameter-induced simulation crashes in climate models. Geosci Model Dev Discuss 6(1):585–623
14. Mairal J, Ponce J, Sapiro G, Zisserman A, Bach FR (2009) Supervised dictionary learning. In: NIPS, pp 1033–1040
15. Nene SA, Nayar SK, Murase H, et al (1996) Columbia object image library (coil-20). Tech. rep., Technical Report CUCS-005-96
16. Sigillito VG, Wing SP, Hutton LV, Baker KB (1989) Classification of radar returns from the ionosphere using neural networks. Johns Hopkins APL Tech. Dig, vol 10, pp 262–266
17. Tang M, Tahboub RY, Aref WG, Atallah MJ, Malluhi QM, Ouzzani M, Silva YN (2016) Similarity group-by operators for multi-dimensional relational data. IEEE Trans Knowl Data Eng 28(2):510–523
18. Tang M, Yu Y, Malluhi QM, Ouzzani M, Aref WG (2016) Locationspark: a distributed in-memory data management system for big spatial data. Proc VLDB Endow 9(13):1565–1568
19. Wang J, Gao X, Wang Q, Li Y (2012) Prodis-contshc: learning protein dissimilarity measures and hierarchical context coherently for protein–protein comparison in protein database retrieval. BMC Bioinform 13(SUPPL.7), S2
20. Wang JJY, Bensmail H, Gao X (2012) Multiple graph regularized protein domain ranking. BMC Bioinform 13(1):307
21. Wang JJY, Bensmail H, Gao X (2014) Feature selection and multi-kernel learning for sparse representation on a manifold. Neural Netw 51:9–16
22. Wang JJY, Bensmail H, Yao N, Gao X (2013) Discriminative sparse coding on multi-manifolds. Knowl Based Syst 54:199–206
23. Wang JJY, Gao X (2014) Semi-supervised sparse coding. In: IJCNN, pp 1630–1637
24. Wang JJY, Gao X (2015) Partially labeled data tuple can optimize multivariate performance measures. In: CIKM, pp 1915–1918
25. Wang JJY, Sun Y, Gao X (2014) Sparse structure regularized ranking. Multimed Tools Appl 74(2):635–654
26. Wang JJY, Tsang IWH, Gao X (2016) Optimizing multivariate performance measures from multi-view data. In: AAAI
27. Yang Y, Xu D, Nie F, Luo J, Zhuang Y (2009) Ranking with local regression and global alignment for cross media retrieval. In: ACM MM, pp 175–184
28. Zhou D, Weston J, Gretton A, Bousquet O, Schölkopf B (2004) Ranking on data manifolds. In: NIPS, pp 169–176