Neural Computing and Applications manuscript No.
(will be inserted by the editor)

Exploiting label consistency in structured sparse representation for classification

Yan Huang · Yuhui Quan* · Tao Liu · Yong Xu

Abstract Sparse representation with adaptive dictionaries has emerged as a promising tool in computer vision and pattern analysis. While standard sparsity promoted by ℓ_0 or ℓ_1 regularization has been widely used, recent approaches seek for kinds of structured sparsity to improve the discriminability of sparse codes. For classification, label consistency is one useful concept regarding structured sparsity, which relates class labels to dictionary atoms for generating discriminative sparsity patterns. Motivated by the limitations of existing label-consistent regularization methods, in this paper, we investigate the exploitation of label consistency and propose an effective sparse coding approach. The proposed approach enforces the sparse approximation of a label consistency matrix by sparse code during dictionary learning, which encourages the supports of sparse codes to be consistent for intra-class signals and distinct for inter-class signals. Thus, the learned dictionary can induce discriminative sparsity patterns when used in sparse coding. Moreover, the proposed method is computationally efficient, as the label consistency regularization developed in our method brings very little additional computational cost in solving the related sparse coding problem. The effectiveness of the proposed method is demonstrated with several recognition tasks, and the experimental results show that our method is very competitive with some state-of-the-art approaches.

Keywords Sparse coding · Label consistency · Structured sparsity · Image classification

Yuhui Quan would like to thank the support by National Natural Science Foundation of China (Grand No. 61602184), Science and Technology Planning Project of Guangdong Province (Grant No. 2017A030313376), Science and Technology Program of Guangzhou (Grand No. 201707010147).

All the authors are with School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China.
* The corresponding author.

1 Introduction

In the last decade, sparse representation has shown its great advantages in discovering the underlying structures of high-dimensional data. With such advantages, sparse representation has become a promising tool in many recognition systems, e.g. face recognition, scene classification, object classification, action recognition; see e.g. [12, 25, 7, 41, 34, 2, 9]. Given a set of input patterns, most existing sparse representation methods aim at finding a small number of *atoms* (i.e. representative patterns) whose linear combinations approximate those input patterns well.

More specifically, given a set of vectors $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_P\} \subset \mathbb{R}^N$, sparse representation aims at determining a set of coefficient vectors $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_P\} \subset \mathbb{R}^M$ with most elements close to zero, together with a set of atoms $\{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_M\} \subset \mathbb{R}^N$, so that each input vector $\boldsymbol{y}_j$ can be well approximated by the linear combination

$$\boldsymbol{y}_j \approx \sum_{m=1}^{M} \boldsymbol{c}_j(m)\boldsymbol{d}_m. \tag{1}$$

By stacking vectors as matrices, we can rewrite (1) in the matrix form as follows:

$$\boldsymbol{Y} \approx \boldsymbol{D}\boldsymbol{C}, \tag{2}$$

where $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_P] \in \mathbb{R}^{N\times P}$ denotes input signals, $\boldsymbol{D} = [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_M] \in \mathbb{R}^{N\times M}$ contains the universal set of atoms and is called a *dictionary*, and $\boldsymbol{C} = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_P] \in \mathbb{R}^{M\times P}$ is often referred to the *sparse code* of input data. Basically speaking, sparse representation assumes data lies at some union of low-dimensional subspaces and represents the subspace of $\boldsymbol{y}_j$ by the span of the activated atoms.

In general, existing sparse representation methods determine the dictionary $\boldsymbol{D}$ and the sparse code $\boldsymbol{C}$ by solving the

following optimization problem:

$$\min_{\boldsymbol{D},\boldsymbol{C}} \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{C}\|_F^2+\phi(\boldsymbol{D})+\psi(\boldsymbol{C}) \tag{3}$$

where $\phi(\cdot)$ and $\psi(\cdot)$ are the regularization functions on dictionary and sparse code respectively. The function ψ can be further expanded as

$$\psi(\boldsymbol{C})=\psi^s(\boldsymbol{C})+\sum_{i=1}^{z}\psi_i^d(\boldsymbol{C}), \tag{4}$$

where $\psi^s(\cdot)$ is a sparsity-promoting penalty function and $\{\psi_i^d(\boldsymbol{C})\}_{i=1}^z$ are some discriminability-promoting penalty functions on sparse code. Classic sparse representation methods, such as KSVD [1], SRC [42] and L0DL [3], set $\psi(\cdot)$ to be a sparsity-promoting penalty function (i.e. $\psi(\cdot)=\psi^s(\cdot)$) and implement it with the ℓ_0 pseudo-norm or ℓ_1-norm. However, both the ℓ_0 pseudo-norm and ℓ_1-norm primarily encourage sparse solutions, regardless of the potential structural relationships (e.g. spatial, temporal, hierarchical or supervised) existing among variables. To overcome such a weakness, there are two directions of recent approaches. The first direction leads to the so-called *structured sparse coding* approaches [13, 17, 18, 24, 28, 37, 14, 38, 6, 2] seeking for new sparsity-promoting penalty $\psi^s(\cdot)$ which is capable of inducing structured sparse representation where more interesting sparsity patterns are likely to appear.[1] The second direction yields the so-called *supervised sparse coding* approaches [26, 32, 35, 34, 27, 52, 16] focusing on designing effective discrimination penalty $\psi^d(\boldsymbol{C})$ which can fully exploit labels of signals in the sparse coding process for enhancing the discriminability of sparse code. Note that structured sparse coding and supervised sparse coding have overlap when the penalty for promoting structured sparsity is defined based on the labels of signals.

Next, we first have a review on recent sparse coding methods from both of the aforementioned directions. Then we present the motivations and contributions of the proposed method in this paper.

1.1 Related Works

1.1.1 Structured sparse coding

Aiming at developing new sparsity-promoting penalty functions, structured sparse coding is capable of generating interesting sparsity patterns that are beneficial to many applications, e.g., in image processing, wavelet coefficients of natural images can be better modeled with hierarchical sparsity [33], and, in classification, both the stability and separability of sparse code can be enhanced by enforcing distinct inter-class sparsity patterns and similar intra-class sparsity patterns [38]. The basic idea of structured sparse coding is that, dictionary atoms are not only used as singletons but also grouped, and a few groups of atoms are activated at a time. In the past, various kinds of groups have been exploited, including the disjoint groups [14, 38] which assume independencies between groups, the overlapping groups [13, 37, 6] which favor the compactness of dictionary due to the reuse of dictionary atoms in different groups, the tree groups [18, 24] which emphasize hierarchical structured sparsity and are often-seen in wavelet or pyramid representation of images, etc. All these methods define sparsity-promoting penalties on the well-designed groups for inducing structured sparsity directly. Yang et al. [48] proposed an effective scheme to automatically determine the atom groups. In [17], the sparse code is implicitly structured by encouraging the code to exhibit label-consistent patterns under certain transform, which yields a general form of $\psi^s(\cdot)$ as follows:

$$\psi^s(\boldsymbol{C})=\|\boldsymbol{Q}-\boldsymbol{A}\boldsymbol{C}\|_F^2+\mathcal{I}_{\mathcal{C}}(\boldsymbol{C}), \tag{5}$$

where $\mathcal{I}_{\mathcal{C}}(\boldsymbol{C})$ is an indicator function that yields ∞ if $\boldsymbol{C}\notin\mathcal{C}$ and 0 otherwise, $\mathcal{C}$ is the feasible set of sparse code which is often set to be $\{\boldsymbol{C}\in\mathbb{R}^{M\times P}:\|\boldsymbol{c}_i\|_0\le T, i=1,2,\ldots,P\}$ for plain sparsity, $\boldsymbol{A}$ is a linear transformation matrix that transforms the sparse code $\boldsymbol{C}$ to be the label consistency form $\boldsymbol{Q}=[\boldsymbol{q}_1,\ldots,\boldsymbol{q}_P]\in\mathbb{R}^{M\times P}$, and $\boldsymbol{Q}$ is defined as a binary matrix in which nonzero occurs at the entry of kth row and ith column if the atom $\boldsymbol{d}_k$ is expected to share class labels with the signal $\boldsymbol{y}_i$. In other words, $\boldsymbol{q}_i$ is a binary vector with the form $\boldsymbol{q}_i=[0,0,\ldots,1,\ldots,1,\ldots,0,0]^\top\in\mathbb{R}^M$ in which the nonzero values occur at the indices where the input signal $\boldsymbol{y}_i$ and the atoms share the same category label. In other words, $\boldsymbol{Q}$ associates each dictionary atom with a category label.

1.1.2 Supervised sparse coding

By incorporating label information into the coding process, supervised sparse coding introduces discrimination penalty to sparse coding for improving the discriminability of sparse code, which has shown noticeable performance improvement in classification; see e.g. [5, 19, 26, 28, 32, 52, 17, 34, 39, 49, 50].

For utilizing class labels, it is straightforward to conduct sparse coding with a series of adaptive class-specific dictionaries. In other words, $\psi_i^d(\boldsymbol{C})$ in (4) is defined based on class-specific reconstruction error. Then a test signal is assigned to the class for which the best sparse reconstruction is obtained. Such a scheme has been used in [26, 47, 48]. However, purely class-specific dictionaries might have ambiguities across classes, as different classes of signals are likely to have similar components. Regarding this issue, Zhou et al. [54] proposed to additionally learn a globally-shared dictionary for extracting intra-class similar components. Ramirez et al. [36] proposed to decrease the mutual

[1] Such a kind of sparsity is often referred to as *structured sparsity*.

coherence of class-specific dictionaries for reducing the ambiguities among class-specific dictionaries.

When used for classification task, sparse coding can be improved by unifying dictionary learning and classification feedback in a joint framework. In other words, $\psi_i^d(\boldsymbol{C})$ is defined by some classification penalty, which leads to some classification-driven sparse coding schemes. There are various types of classification penalties that have been exploited in existing methods, e.g. softmax penalty function in [26, 27], Fisher discrimination penalty in [12, 47, 54], linear predictive penalty in [32, 52, 17], hinge loss in [22, 45], and logistic loss in [27, 23].

1.2 Motivations and Contributions

Label consistency, as discussed in Section 1.1, is undoubtedly a useful concept for structured sparse coding in classification. Although label consistency has been exploited in some existing methods (e.g. [15, 17]), it remains plenty of room for improvement. In [34], it is shown that the label consistency regularization corresponds to some specific type of ensemble classification penalty which can be generalized by changing the setting of ensemble. Such a generalized version, however, is a supervised sparse coding method instead of the structured one. It is also shown in [20] that the aforementioned label consistency regularization degenerates to a linear predictive penalty under mild conditions, which implies that label consistency could be further exploited. Inspired by these facts, in this paper, we investigate effective ways for the exploitation of label consistency. Our basic idea is considering the following penalty function for promoting structured sparsity:

$$\psi^s(\boldsymbol{C}) = \|\boldsymbol{Q} - \boldsymbol{C}\|_p + \lambda\|\boldsymbol{C}\|_p, \tag{6}$$

where λ is a scalar for weighting, $\boldsymbol{Q}$ is the aforementioned label consistency matrix in (5), $||\cdot||_p$ is a sparsity-inducing norm with parameter $p = 0$ or 1. The first term is the proposed label consistency penalty used for encouraging that intra-class signals share support (i.e. positions of non-zeros) of sparse codes and that the shared supports are distinct between classes, which yields sparsity patterns with enhanced discrimination. The second term is the classic $\ell_0(\ell_1)$ sparsity promoting term. Supposing $\boldsymbol{c}_i$ is sparse under the regularization of $\|\boldsymbol{c}_i\|_0$, then minimizing $\|\boldsymbol{q}_i - \boldsymbol{c}_i\|_0$ can encourage the positions of nonzeros of $\boldsymbol{c}_i$ to align with those of $\boldsymbol{q}_i$. Thus, by combining the two terms in (6), we can learn a dictionary with label consistency and obtain sparse code with discriminative patterns.

There are two key differences between our proposal (6) and the label consistency regularization [15, 17, 29, 34, 35] defined in (5). Firstly, we directly induce discriminative sparsity patterns without using the transform $\boldsymbol{A}$. In this case, the effect of $\boldsymbol{A}$ is merged into the dictionary. Secondly, regarding the label consistency penalty, we use some sparsity measures instead of the ℓ_2-norm. Such a difference is non-trivial. As described in Section 2.2, using sparsity measure for defining label consistency is more suitable than that defined by the ℓ_2-norm, especially for the case where intra-class samples are distributed in multiple subspaces. Based on the proposed label consistency regularization, we proposed an effective structured sparse coding method for classification. Our method enjoys the theoretical benefits from using the proposed label consistency regularization. Moreover, the computational efficiency of our method is not high, as the proposed label consistency regularization brings very little additional computational cost in solving the related sparse coding problem. The above advantages of the proposed method are demonstrated with several recognition tasks, including face recognition, scene classification, and dynamic texture recognition. The experimental results show the excellent performance of the proposed method in comparison with some state-of-the-art sparse coding approaches.

2 Our Method

2.1 Notations

Bold upper letters are used for matrices, bold lower letters for column vectors, light lower letters for scalars, and calligraphic letters for sets. More specifically, $\boldsymbol{y}_j$ denotes the j-th column of the matrix $\boldsymbol{Y}$, y_i denotes the i-th element of the vector $\boldsymbol{y}$, $Y_{i,j}$ denotes the entry of $\boldsymbol{Y}$ at ith row and jth column, and $\boldsymbol{C}_j^\top$ is the jth column of $\boldsymbol{C}^\top$. For $\boldsymbol{x} \in \mathbb{R}^N$, its ℓ_p norm $\|\boldsymbol{x}\|_p$ ($p \in [1, \infty)$) is defined as $\|\boldsymbol{x}\|_p = (\sum_{j=1}^N |x_j|^p)^{1/p}$, and its ℓ_0 pseudo-norm $\|\boldsymbol{x}\|_0$ is defined as $\|\boldsymbol{x}\|_0 := \#\{j|x_j \neq 0\}$. For $\boldsymbol{X} \in \mathbb{R}^{N\times M}$, its Frobenius norm is defined as $\|\boldsymbol{X}\|_F^2 = (\sum_{i=1}^N \sum_{j=1}^M |X_{i,j}|^2)^{1/2}$. Besides, $\boldsymbol{y}^{(t_0)}$ denotes the t_0-th element of a sequence $\{\boldsymbol{y}^{(t)}\}_{t\in\mathbb{N}}$, $\boldsymbol{I}_M$ denotes the $M\times M$ identity matrix, $\boldsymbol{1}_M$ denotes $\underbrace{[1,\ldots,1]}_{M}^\top$ and $\boldsymbol{0}_M$ denotes $\underbrace{[0,\ldots,0]}_{M}^\top$.

2.2 Sparse Coding with Sparse Label Consistency

Based on (6), we set $p = 0$ and construct our model as follows:

$$\min_{\boldsymbol{D},\boldsymbol{C}} \frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{C}\|_F^2+\lambda\|\boldsymbol{C}\|_0+\beta\|\boldsymbol{Q}-\boldsymbol{C}\|_0, \text{s.t.}\forall j\ \|\boldsymbol{d}_j\|_2^2 = 1 \tag{7}$$

where λ and β are two scalars for weighting the contribution of each term, and $\boldsymbol{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_P] \in \mathbb{R}^{M\times P}$ is a predefined sparse label consistency indicator matrix which can be

defined flexibly. A simple way to design $\boldsymbol{Q}$ is using the definition of $\boldsymbol{Q}$ in (5), i.e., setting $\boldsymbol{Q}$ to be a binary matrix where nonzero occurs at the entry of kth row and ith column if the ith sample $\boldsymbol{y}_i$ is expected to be represented by the atoms which are of the same class to the kth atom. In other words, when the columns of $\boldsymbol{Y}$ are sorted by category, $\boldsymbol{q}_i$ is a binary vector with the form $\boldsymbol{q}_i = [0, 0, \ldots, 1, \ldots, 1, \ldots, 0, 0]^\top \in \mathbb{R}^M$ in which the nonzero values occur at the indices where the input signal $\boldsymbol{y}_i$ and the atom share the same class label. For example, assuming $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_6]$ and $\boldsymbol{D} = [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_5]$, where $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, $\boldsymbol{y}_3$, $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ are from 1st class, $\boldsymbol{y}_4$, $\boldsymbol{y}_5$, $\boldsymbol{y}_6$, $\boldsymbol{d}_3$, $\boldsymbol{d}_4$ and $\boldsymbol{d}_5$ are from 2nd class, the corresponding matrix $\boldsymbol{Q}$ is expressed as

$$\boldsymbol{Q} \equiv \begin{bmatrix} 1\,1\,1\,0\,0\,0 \\ 1\,1\,1\,0\,0\,0 \\ 0\,0\,0\,1\,1\,1 \\ 0\,0\,0\,1\,1\,1 \\ 0\,0\,0\,1\,1\,1 \end{bmatrix}.$$

The last term $\|\boldsymbol{Q} - \boldsymbol{C}\|_0$ in (7) can be interpreted as a label consistency penalty, which is different from the label consistency penalty $\|\boldsymbol{Q} - \boldsymbol{A}\boldsymbol{C}\|_F^2$ that is defined in (5) and used in the LC-KSVD method [15, 17]. The differences lie at that we directly structure sparse code by encouraging it to sparsely approximate the label consistency form, while LC-KSVD encourages the sparse code to approximate the label consistency matrix under Euclidean metric after a transform. The replacement of ℓ_2 norm with ℓ_0 norm is non-trivial. Consider $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ which are the sparse coding vectors of two intra-class samples but have different supports (e.g. $\boldsymbol{c}_1 = [c_1, 0, 0]\top$ and $\boldsymbol{c}_2 = [0, c_2, 0]\top$).[2] In this case, regularizing $\|\boldsymbol{c} - \boldsymbol{q}\|_2^2$ is not suitable, as the constraint $\|\boldsymbol{c} - \boldsymbol{q}\|_2^2 \leq \epsilon^2$ implies $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ lie within a ball. In other words, the $\ell 2$ label consistency cannot well handle the data where intra-class signals are distributed in multiple subclasses. In contrast, using the ℓ_0 label consistency $\|\boldsymbol{c} - \boldsymbol{q}\|_0$ is better for this scenario as $\|\boldsymbol{c} - \boldsymbol{q}\|_0 \leq T$ allows $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are scattered in different subspaces. By using the ℓ_0 label consistency, the nonzero coding values of intra-class signals have the flexibility to occur at different entries while sharing the same group of dictionary atoms. This actually allows the input signals from the same category to share a group of atoms but the used atoms of each signal are not necessarily the same.

The setting of $\boldsymbol{Q}$ can be done using the scheme used in [17]. The number of dictionary atoms used in the ith class, denoted by m_i, is set to the same constant $Z > 0$ for all i, where Z is often set to 1. In other words, all rows of $\boldsymbol{Q}$ have the same number of 1s. In theory, the value m_i is related to the dimensionality of the subspace of ith class. In the cases where we have prior on the dimensionality of the subspace of each class, we can set Q with non-uniform groups based on the prior.

The ℓ_0 minimization problem is very challenging to solve, and it often suffers from the big potential of getting stuck in local minimum [12]. Moreover, it was found in some literature(e.g. [42, 46]) that using ℓ_1 norm can yield better discriminability on the sparse code. Thus, we relaxed the $\ell 0$ label consistency penalty to the $\ell 1$ case and proposed the $\ell 1$ label consistent sparse coding model as follows:

$$\min_{\boldsymbol{D},\boldsymbol{C}} \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{C}\|_F^2 + \lambda\|\boldsymbol{C}\|_1 + \beta\|\boldsymbol{Q} - \boldsymbol{C}\|_1, \text{s.t.} \forall j \|\boldsymbol{d}_j\|_2^2 = 1. \tag{8}$$

Compared with the ℓ_0 norm which only cares the support of sparse code, the ℓ_1 norm takes account of both the support and magnitude in sparse code. It is shown that the ℓ_1 sparse label-consistent sparse coding may yield better performance than the ℓ_0 case in some classification tasks.

2.3 Numerical Solution

Solving (7) and (8) is non-trivial, as both of them are non-smooth and non-convex problems. Motivated by its success in solving a broad spectrum of sparse coding problems [3], we solve both these problems with the alternating proximal scheme. The algorithms are summarized in Algorithm 1. More concretely, the unknown variables $\boldsymbol{D}$ and $\boldsymbol{C}$ are alternately estimated one at a time, which breaks the original problem into two simpler subproblems, and then each subproblem is solved by proximal methods. Note that the update of dictionary $\boldsymbol{D}$ is the same for both problems, while the update of sparse code $\boldsymbol{C}$ involves different thresholding schemes, which is shown in the next. Before presenting the numerical algorithm, we give the definition of the proximal operator as follows:

$$Prox_t^f(\boldsymbol{x}) := \operatorname*{argmin}_{\boldsymbol{u}} f(\boldsymbol{u}) + \frac{t}{2}\|\boldsymbol{u} - \boldsymbol{x}\|_2^2. \tag{9}$$

2.3.1 Update of sparse code

We first initialize the dictionary $\boldsymbol{D} = \boldsymbol{D}^{(0)}$ and start with $k = 1$. At the beginning of the k-th iteration, we fix $\boldsymbol{D} = \boldsymbol{D}^{(k-1)}$ and calculate $\boldsymbol{C}^{(k)}$ by solving

$$\boldsymbol{C}^{(k)} = \operatorname*{argmin}_{\boldsymbol{C}} \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{D}^{(k-1)}\boldsymbol{C}\|_F^2 + \lambda\|\boldsymbol{C}\|_p + \beta\|\boldsymbol{Q} - \boldsymbol{C}\|_p, \tag{10}$$

where $p = 0$ regarding the problem (7) and $p = 1$ regarding the problem (8). Let $\mathcal{H}(\boldsymbol{C}, \boldsymbol{D}; \boldsymbol{Y}) = \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{C}\|_F^2$ and

[2] This is often true if the corresponding signal $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ are from different subclasses.

1. Input: Training data $\boldsymbol{Y}$, label consistency term $\boldsymbol{Q}$
2. Initialization: Set dictionary $\boldsymbol{D}^{(0)}$, $\lambda > 0$, $\beta > 0$.
3. For $k = 1, \ldots, K$
   (a) Update of sparse code:
      i. $t^{(k)} = \max(\rho \|\boldsymbol{D}^{(k)^T} \boldsymbol{D}^{(k)}\|_F, \mu_{min})$;
      ii. $\boldsymbol{C}^* = \boldsymbol{C}^{(k-1)} - \frac{1}{t^{(k)}} \nabla \mathcal{H}(\boldsymbol{C}^{(k-1)}, \boldsymbol{D}^{(k-1)}; \boldsymbol{Y})$;
      iii. $\boldsymbol{C}_{i,j}^{(k)} = \mathcal{S}^p_{\frac{\lambda}{t^{(k)}}, \frac{\beta}{t^{(k)}}, \boldsymbol{Q}_{i,j}}(\boldsymbol{C}^*_{i,j})$, for $p = 0$ or 1.
   (b) Update of dictionary: for $j = 1, \ldots, M$:
      i. $r_j^{(k)} = \max(\rho [\boldsymbol{C}^{(k)} \boldsymbol{C}^{(k)^T}]_{j,j}, \gamma_{min})$;
      ii. $\boldsymbol{v}_j^{(k-1)} = \boldsymbol{d}_j^{(k-1)} - \frac{1}{r_j^{(k-1)}} \nabla_{\boldsymbol{d}_j} \mathcal{H}(\boldsymbol{C}^{(k)}, \widetilde{\boldsymbol{D}}_j^{(k-1)})$;
      iii. $\boldsymbol{d}_j^{(k)} = \boldsymbol{v}_j^{(k-1)} / \|\boldsymbol{v}_j^{(k-1)}\|_2$.
4. Output: The learned dictionary $\boldsymbol{D}$.

Algorithm 1: Sparse label consistency dictionary learning

$\mathcal{F}(\boldsymbol{C}) = \lambda \|\boldsymbol{C}\|_p + \beta \|\boldsymbol{Q} - \boldsymbol{C}\|_p$. The problem (10) is solved by the proximal gradients method as follows:

$$\boldsymbol{C}^{(k)} \in Prox^{\mathcal{F}}_{t^{(k)}}(\boldsymbol{C}^{(k-1)} - \frac{1}{t^{(k)}} \nabla_{\boldsymbol{C}} \mathcal{H}(\boldsymbol{C}^{(k-1)}, \boldsymbol{D}^{(k-1)})), \tag{11}$$

where $t^{(k)}$ is the step size, and $\nabla_{\boldsymbol{C}} \mathcal{H}(\boldsymbol{C}, \boldsymbol{D}) = \boldsymbol{D}^T(\boldsymbol{D}\boldsymbol{C} - \boldsymbol{Y})$. By denoting $\boldsymbol{C}^* = -\frac{1}{t^{(k)}} \nabla \mathcal{H}(\boldsymbol{C}^{(k-1)}, \boldsymbol{D}^{(k-1)}; \boldsymbol{Y}) + \boldsymbol{C}^{(k-1)}$, we have

$$\boldsymbol{C}^{(k)} \in \underset{\boldsymbol{C}}{\operatorname{argmin}} \lambda \|\boldsymbol{C}\|_p + \beta \|\boldsymbol{Q} - \boldsymbol{C}\|_p + \frac{t^{(k)}}{2} \|\boldsymbol{C} - \boldsymbol{C}^*\|_F^2. \tag{12}$$

The above problem, whenever $p = 0$ or $p = 1$, is separable with respect to elements of $\boldsymbol{C}$ and has an explicit solution in the simple form of thresholding. When $p = 0$, the analytic solution of (12) is given by

$$\boldsymbol{C}_{i,j}^{(k)} = \mathcal{S}^0_{\frac{\lambda}{t^{(k)}}, \frac{\beta}{t^{(k)}}, \boldsymbol{Q}_{i,j}}(\boldsymbol{C}^*_{i,j}), \tag{13}$$

where $\mathcal{S}^0_{\lambda_1, \lambda_2, q}(\cdot)$ is a thresholding function defined by

$$\mathcal{S}^0_{\lambda_1, \lambda_2, q}(c) = \begin{cases} 0 & \text{if } -\tau_1^2(c) < c \leq min(\tau_1^2(c), \tau_2) \\ q & \text{if } max(q - \tau_1^1(c), \tau_2) < c \leq q + \tau_1^1(c) \\ c & \text{otherwise} \end{cases}, \tag{14}$$

with $\tau_1^k(x) = \sqrt{2\lambda_1 [x \neq 0] + 2\lambda_2 [x \neq q] - 2\lambda_k [q \neq 0]}$, for k = 1,2, $\tau_2 = \begin{cases} \frac{2\lambda_1 - 2\lambda_2 + q^2}{2q} & q \neq 0 \\ \sqrt{2\lambda_1 + 2\lambda_2} & q = 0 \end{cases}$, and $[\bullet]$ is the Iverson bracket that denotes a number that is 1 if the condition $\bullet$ is satisfied, and 0 otherwise. For $p = 1$, the problem (12) has the analytic solution given by

$$\boldsymbol{C}_{i,j}^{(k)} = \mathcal{S}^1_{\frac{\lambda}{t^{(k)}}, \frac{\beta}{t^{(k)}}, \boldsymbol{Q}_{i,j}}(\boldsymbol{C}^*_{i,j}), \tag{15}$$

where $\mathcal{S}^1_{\lambda_1, \lambda_2, q}(\cdot)$ is a double-header thresholding function defined by

$$\mathcal{S}^1_{\lambda_1, \lambda_2, q}(c) = \begin{cases} c + \lambda_1 + \lambda_2 & \text{if } c \leq -\lambda_1 - \lambda_2 \\ 0 & \text{if } -\lambda_1 - \lambda_2 < c \leq \lambda_1 - \lambda_2 \\ c - \lambda_1 + \lambda_2 & \text{if } \lambda_1 - \lambda_2 < c \leq \lambda_1 - \lambda_2 + q \\ q & \text{if } \lambda_1 - \lambda_2 + q < c \leq \lambda_1 + \lambda_2 + q \\ c - \lambda_1 - \lambda_2 & \text{if } c > \lambda_1 + \lambda_2 + q \end{cases}.$$

2.3.2 *Update of dictionary*

After calculating the sparse code at the k-th iteration, we fix $\boldsymbol{C} = \boldsymbol{C}^{(k)}$ and calculate $\boldsymbol{D}^{(k)}$ by solving

$$\boldsymbol{D}^{(k)} = \underset{\boldsymbol{D}}{\operatorname{argmin}} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{C}^{(k)}\|_F^2, \quad \text{s.t. } \forall j \ \|\boldsymbol{d}_j\| = 1. \tag{16}$$

For solving (16), we denote $\mathcal{G}(\boldsymbol{D}) = \mathcal{I}_{\mathcal{X}}(\boldsymbol{D})$ where $\mathcal{X} = \{\boldsymbol{D} \in \mathbb{R}^{N \times M} : \|\boldsymbol{d}_j\|_2^2 = 1, j = 1, 2, \ldots, M\}$ and $\mathcal{I}_{\mathcal{X}}(\boldsymbol{D})$ is the indicator function of $\boldsymbol{D}$ which satisfies $\mathcal{I}_{\mathcal{X}}(\boldsymbol{D}) = 0$ if $\boldsymbol{D} \in \mathcal{X}$ and $+\infty$ otherwise, and then $\boldsymbol{D}^{(k)}$ is sequentially updated column by column using the proximal gradient method, that is,

$$\boldsymbol{d}_j^{(k)} \in Prox^{\mathcal{G}(\hat{\boldsymbol{D}}_j^{(k-1)})}_{r_j^{(k-1)}}(\boldsymbol{d}_j^{(k-1)} - \frac{1}{r_j^{(k-1)}} \nabla_{\boldsymbol{d}_j} \mathcal{H}(\boldsymbol{C}^{(k)}, \widetilde{\boldsymbol{D}}_j^{(k-1)})), \tag{17}$$

where $r_j^{(k-1)}$ is a step size, $\nabla_{\boldsymbol{d}_j} \mathcal{H}(\boldsymbol{C}, \boldsymbol{D}) = (\boldsymbol{D}\boldsymbol{C} - \boldsymbol{Y})\boldsymbol{C}_j^\top$, and

$$\begin{cases} \hat{\boldsymbol{D}}_j^{(k)} = [\boldsymbol{d}_1^{(k+1)}, \ldots, \boldsymbol{d}_{(j-1)}^{(k+1)}, \boldsymbol{d}_j, \boldsymbol{d}_{(j+1)}^{(k)}, \ldots, \boldsymbol{d}_M^{(k)}] \\ \widetilde{\boldsymbol{D}}_j^{(k)} = [\boldsymbol{d}_1^{(k+1)}, \ldots, \boldsymbol{d}_{(j-1)}^{(k+1)}, \boldsymbol{d}_j^{(k)}, \boldsymbol{d}_{(j+1)}^{(k)}, \ldots, \boldsymbol{d}_M^{(k)}] \end{cases}.$$

Denote $\boldsymbol{v}_j^{(k-1)} = \boldsymbol{d}_j^{(k-1)} - \frac{1}{r_j^{(k-1)}} \nabla_{\boldsymbol{d}_j} \mathcal{H}(\boldsymbol{C}^{(k)}, \widetilde{\boldsymbol{D}}_j^{(k-1)})$, we have

$$\boldsymbol{d}_j^{(k)} \in \underset{\|\boldsymbol{d}\|_2 = 1}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{d} - \boldsymbol{v}_j^{(k-1)}\|_F^2, \ j = 1, \ldots, M, \tag{18}$$

whose solution can be directly obtained by

$$\boldsymbol{d}_j^{(k)} = \boldsymbol{v}_j^{(k-1)} / \|\boldsymbol{v}_j^{(k-1)}\|_2$$

for all j.

Using the results of [3], it can be proved that, the solution sequence $\{\boldsymbol{C}^{(k)}, \boldsymbol{D}^{(k)}\}_k$ generated by the iteration procedure in (11) and (17) is a Cauchy sequence and converges to a critical point of (7) or (8).[3] It is noted that, the values of the step-sizes $t^{(k)}$ and $r_j^{(k)}$ in (11) and (17) are determined during the solving procedure. To this end, we need to calculate the Lipschitz constants $L_t^{(k)}$ and $L_{r_j}^{(k)}$ which satisfy $\|\nabla_C \mathcal{H}(\boldsymbol{C}_1, \boldsymbol{D}^{(k)}) - \nabla_C \mathcal{H}(\boldsymbol{C}_2, \boldsymbol{D}^{(k)})\|_2 \leq L_t^{(k)} \|\boldsymbol{C}_1 - \boldsymbol{C}_2\|_2$ and $\|\nabla_{\boldsymbol{d}_j} \mathcal{H}(\boldsymbol{C}^{(k)}, \boldsymbol{d}_{j_1}^{(k)}) - \nabla_{\boldsymbol{d}_j} \mathcal{H}(\boldsymbol{C}^{(k)}, \boldsymbol{d}_{j_2}^{(k)})\|_2 \leq$

[3] For rigorous proof, the sequence $\{\boldsymbol{C}^{(k)}\}_k$ is required to be bounded, which can be guaranteed by simple projection of $\boldsymbol{C}^{(k)}$ at each iteration.

$L_{r_j}^{(k)}\|\boldsymbol{d}_{j_1}^{(k)} - \boldsymbol{d}_{j_2}^{(k)}\|_2$, respectively. By simple calculation, we can select $L_t^{(k)} = \|\boldsymbol{D}^{(k)^T}\boldsymbol{D}^{(k)}\|_F$ and $L_{r_j}^{(k)} = [\boldsymbol{C}^{(k)}\boldsymbol{C}^{(k)^T}]_{j,j}$, $\forall j = 1, 2, \ldots, M$. After obtaining the values of $L_t^{(k)}$ and $L_{r_j}^{(k)}$, we can determine $t^{(k)} = \max(\rho L_t^{(k)}, \mu_{min})$ and $r_j^{(k)} = \max(\rho L_{r_j}^{(k)}, \gamma_{min})$, where $\rho > 1$, μ_{min} is a predefined upper bound, and γ_{min} is a predefined scalar.

2.3.3 Complexity analysis

We now discuss the complexity of the proposed algorithm. It can be seen in Algorithm 1 that for each iteration, e.g. the k-th iteration, the update of sparse code includes

- the estimation of the step-size $t^{(k)}$, with $\boldsymbol{D}^{(k)^T}\boldsymbol{D}^{(k)}$ as highest cost part that requires $\mathcal{O}(NM^2)$ operations;
- the calculation of $\boldsymbol{C}^*$, which mainly takes $\mathcal{O}(MNP)$ operations;
- the thresholding operation (either $\mathcal{S}^0$ or $\mathcal{S}^1$), which requires $\mathcal{O}(MP)$ operations.

The update of dictionary includes

- the estimation of the step-size $r_j^{(k)}$, which requires $\mathcal{O}(P)$ operations;
- the calculation of $\boldsymbol{v}_j^{(k-1)}$, which requires $\mathcal{O}(NP)$ operations;
- the update of $\boldsymbol{d}_j^{(k)}$ for each atom $\boldsymbol{d}_j$, which requires $\mathcal{O}(N)$ operations respectively.

Totally speaking, the dominant operation of sparse code update is $\mathcal{O}(NM^2 + MNP)$, and the main operation of dictionary update is $\mathcal{O}(MNP)$. Combining these results, the overall complexity of the proposed algorithm is $\mathcal{O}(KMNP + KNM^2)$, which is linear with the size of dataset. Thus, the proposed method is suitable for handling large-scale data.

2.4 Classification scheme

Once the dictionary $\boldsymbol{D}$ is learned via (7) or (8), we conduct sparse coding on the training data using the learned dictionary, resulting in the sparse code $\boldsymbol{C}_{\text{train}}$. Then we use $\boldsymbol{C}_{\text{train}}$ to train a linear subspace classifier $\boldsymbol{W}$ as follows:

$$\underset{\boldsymbol{W}}{\operatorname{argmin}} ||\boldsymbol{L} - \boldsymbol{W}\boldsymbol{C}_{\text{train}}||_F^2 + \gamma||\boldsymbol{W}||_2^2, \tag{19}$$

where $\boldsymbol{L}$ is the corresponding binary label matrix where nonzeros occurs at i-th row and j-th column when j-th sample is belong to the i-th class.

When a test sample $\boldsymbol{y}_{\text{test}}$ arrives, we calculate its sparse code $\boldsymbol{c}_{\text{test}}$ via solving

$$\boldsymbol{c}_{\text{test}} = \underset{\boldsymbol{c}}{\operatorname{argmin}} ||\boldsymbol{y}_{\text{test}} - \boldsymbol{D}\boldsymbol{c}||_F^2 + \alpha||\boldsymbol{c}||_p, \tag{20}$$

where $p = 0$ or 1 which is set the same as that of the dictionary learning model, α is a scalar for weighting. This problem is solved by the proximal gradient method [3]. Then we input $\boldsymbol{c}_{\text{test}}$ to the trained classifier $\boldsymbol{W}$ and predict the label vector $\boldsymbol{l}_{\text{test}}$ by calculating

$$\boldsymbol{l}_{\text{test}} = \boldsymbol{W}\boldsymbol{c}_{\text{test}}. \tag{21}$$

The final label of the test sample is assigned to the class i where the biggest value occurs at the i-th dimension of $\boldsymbol{l}_{\text{test}}$.

3 Experiments

In this section, we show the effectiveness of our method using both synthetic data and real datasets. On the synthetic data, we tested the coding results as well as the convergent behaviors of the proposed method. For the real dataset, the classification performances in several recognition tasks including face recognition, scene classification and dynamic texture classification are evaluated. For convenience, we denote the ℓ_1-sparse label consistency (SLC) model (8) by SLC-1, and denote the ℓ_0-SLC model (7) by SLC-0. The methods for comparison mainly include the sparsity-based coding methods K-SVD [1], LC-KSVD [17], D-KSVD [52], LLC [40], Joint [32], and MCDL [34], which are the classic or recent sparse coding methods for classification.

3.1 Evaluation on Synthetic Data

The generation of the synthetic data is as follows. A dictionary $\boldsymbol{D} \in \mathbb{R}^{N\times M}$ is generated by the product of the left and right components of the singular value decomposition of a random Gaussian matrix. The ground truth of sparse code $\boldsymbol{C} \in \mathbb{R}^{M\times P}$ is a random block-diagonal matrix with K diagonal blocks (i.e., to simulate K categories), and each block is of size $\frac{M}{K} \times \frac{P}{K}$. In each block, around 15% are randomly set to 0. Then the synthetic signals are generated by $\boldsymbol{Y} = \boldsymbol{D}\boldsymbol{C} + \boldsymbol{E}$, where $\boldsymbol{E}$ is a random Gaussian matrix with zero mean and standard derivation σ and used for simulating noises. Our evaluation is to estimate $\boldsymbol{D}$ and $\boldsymbol{C}$ from the synthetic $\boldsymbol{Y}$.

Figure 1 shows the results on the synthetic data with $N = 64$, $M = 64$, $P = 3200$, $K = 8$ and $\sigma = 0.1$. The corresponding model parameters are set as follows: $\lambda = 7e-2$ and $\beta = 3e-2$ for the SLC-0 method and $\lambda = 5e-2$ and $\beta = 5e-2$ for the SLC-1 method. The generated synthetic data is shown in Figure 1 (a), and the ground truth of sparse code is shown in Figure 1 (e). Figure 1 (b) and (f) are the sparse coding results of SLC-0 and SLC-1 from the synthetic data respectively. As can be seen, obvious sparsity patterns can be observed in the coding results, and such patterns are very useful for classification as they are discriminative between classes and similar within the same class. These results have demonstrated the power of our method in capturing the intrinsic structures of data. The convergent behaviors of SLC-0 and SLC-1 are shown in Fig 1 (c),(d),(g),(f), which

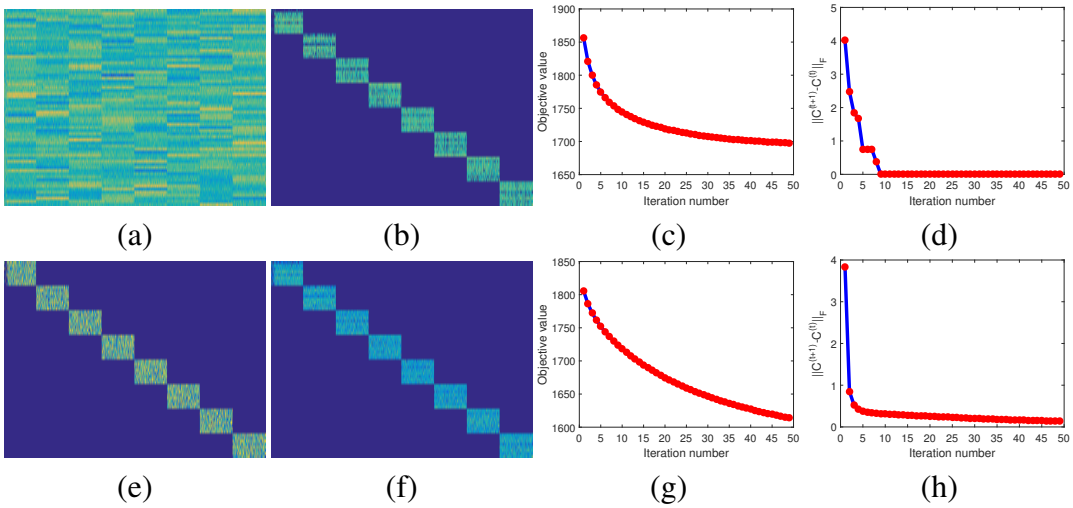

Fig. 1: Results of SLC-0 and SLC-1 on the synthetic data. (a) Input synthetic data; (b) Sparse coding results of SLC-0 from the synthetic data; (c) Objective function value decay of SLC-0; (d) Increment of Frobenius norm of $\boldsymbol{C}^{(t+1)}$ and $\boldsymbol{C}^{(t)}$ in SLC-0; (e) The ground truth of sparse code; (f) Sparse coding results of SLC-1 from the synthetic data; (g) Objective function value decay of SLC-1; (h) Increment of Frobenius norm of $\boldsymbol{C}^{(t+1)}$ and $\boldsymbol{C}^{(t)}$ in SLC-1.

illustrate the effectiveness and correctness of our algorithm. The run time is around 0.08s per iteration using the Matlab implementation on a PC with Core-i7 CPU with 16Gb RAM.

3.2 Evaluation on Real Datasets

We evaluated the proposed method with face recognition, scene classification, and dynamic texture classification. The benchmark datasets for different classification tasks are as follows:

- Face classification: the AR-Face dataset [30];
- Scene classification: the Scene-15 dataset [21];
- Dynamic texture classification: the Dyntex++ dataset [31].

3.2.1 Face recognition

Over the last decade, face recognition has become a popular area of research in computer vision and one of the most successful applications of image analysis and understanding [51, 9, 25, 41]. The evaluation scheme on the AR-Face dataset in the paper follows the standard evaluation procedure in [17, 52]. At the beginning, one subset including 2600 images of 100 subjects [30] is extracted from the AR-Face dataset. See some sample images in Figure 2. Then random face is used as the feature descriptor to represent each face image. More specifically, first the selected images in the AR-Face dataset are cropped to images of size 165×120, and then each cropped image is projected into a 540 dimensional feature vector by using a random matrix of zero-mean normal distribution. 20 images per person are randomly picked up for training and the remaining images for test, and the size of the dictionary is chosen as $M = 600$. We set the parameters $\lambda = 1e-3, \beta = 1e-3$ on the dataset and compared our method with KSVD [1], D-KSVD [52], LC-KSVD [17], LLC [40], Joint [32], MCDL [34]. The experimental results are summarized in Table 1. It can be seen that our method performs best among the compared methods in the AR-Face dataset.

Fig. 2: Some sample images from the AR-Face dataset.

3.2.2 Scene classification

Scene classification is a hot topic in computer vision, which provides some basic knowledges of the presence of surfaces, actions, objects, as well as the layout information such as position. The Scene-15 dataset contains 15 scene categories,

and the number of images in each class varies from 210 to 410. Some sample images in Scene-15 are shown in Figure 3. The feature used for the Scene-15 dataset is the 3000-dimensional SIFT based spatial pyramid descriptor [21, 15, 17]. Same as the experimental setting in [21, 15, 17], 100 images per class are randomly selected for training and the remaining for test. In addition, we set the parameters of our method on the Scene-15 dataset as $\lambda = 1e-3$, $\beta = 5e-2$ and $M = 300$.

We conducted our method on scene classification and compared the performance with other sparse coding methods which include D-KSVD [52], LC-KSVD [17], LLC [40], MCDL [34]. Besides some effective scene classification methods, such as Lazebnik et al. [21], Boureau et al. [4], Yang et al. [44], Gao et al. [8], Bao et al. [3], are also included for comparison. The experimental results are shown in Table 2. As can be seen, the SLC-1 method performs the best among all the compared methods, while the SLC-0 method is just slightly worse than the SLC-1 method but better than the other compared methods.

Fig. 3: Some sample images in the Scene-15 dataset.

3.2.3 Dynamic texture classification

Dynamic texture classification is to classify the videos which contain both spatial and temporal textures. It is a basic part for dynamic scene understanding. The Dyntex++ dataset [10] for evaluation contains 36 DT categories, each with 100 video sequences of size $50 \times 50 \times 50$ cropped from the original sequences from the Dyntex database [31]. Some key frames extracted from the DT sequences of the dataset are shown in Fig 4. The 2700-dimensional wavelet coefficient histogram is chosen as the input for our method. One half of the samples are used for training and the rest for test. The parameters λ, β and M on the Dyntex++ dataset are set to be $1e-2$, $1e-2$ and 3000 respectively. The compared methods include KSVD [1], Joint [42], D-KSVD [52], LC-KSVD [17], Ghanem et al. [11], Zhao et al. [53], MCDL [34], Xu et al. [43]. The results are listed in Table 3. It can be seen that both the SLC-0 and SLC-1 methods perform better than other compared methods.

Fig. 4: Keyframes from the DynTex database.

3.2.4 Result Analysis

From the aforementioned experimental results, the proposed methods, both SLC-0 and SLC-1, achieved better results than the compared methods. Particularly, the superior performance of the proposed methods to the LC-KSVD method has demonstrated that the proposed $\ell_0(\ell_1)$ label consistent regularization method can better characterize the underlying structures of data and is more effective than the traditional label consistency penalty. Meanwhile, the proposed methods slightly outperform the MCDL method which is a generalized version from LC-KSVD with discriminative ensemble classifiers. Note that the MCDL method need to solve a set of linear classifiers during learning, which is much more complex than the proposed model. All these results demonstrate the effectiveness of our methods. It is also noted that SLC-1 consistently performs slightly better than SLC-0. This is unsurprising as the ℓ_1 label consistency term is more discriminative to the ℓ_0 term, and the ℓ_0 model might be easier to stuck in local minimizer than the ℓ_1 case.

3.2.5 Influence of Parameter Selection

The regularization parameters λ and β are two important parameters in both of the proposed methods. In order to test the sensitivity of the proposed methods to these two parameters, we conducted a series of experiments to analyze the influence of these parameters to the performance of the proposed methods. In detail, we adjusted the regularization parameter λ (β) while fixing β (λ), to check the performance of the proposed methods on different datasets. See Fig 5 for the results. It can be observed that the classification results are not sensitive to the changes of the parameters within a moderate range.

The dictionary size is also a critical parameter that affects the performance of our methods, as it controls the representational power of the models. Thus, we also analyze

Table 1: The recognition accuracies (%) of the compared methods on the AR-Face dataset.

Dataset	KSVD [1]	D-KSVD [52]	LC-KSVD [17]	LLC [40]	Joint [32]	MCDL [34]	SLC-1	SLC-0
AR-Face	86.50	88.80	93.70	88.70	88.24	95.21	97.17	97.16

Table 2: Classification accuracies (%) of the compared method on the Scene-15 dataset.

Methods	Accuracy	Methods	Accuracy	Methods	Accuracy
Lazebnik et al. [21]	81.40	Gao et al. [8]	89.75	LC-KSVD [17]	92.90
Boureau et al. [4]	84.30	LLC [40]	89.20	MCDL [34]	97.35
Yang et al. [44]	80.28	KSVD [1]	86.70	SLC-1	98.10
Bao et al. [3]	93.10	D-KSVD [52]	89.10	SLC-0	97.74

Table 3: Classification accuracies (%) on the DynTex++ dataset.

KSVD [1]	Joint [42]	D-KSVD [52]	LC-KSVD [17]	Ghanem et al. [11]	Zhao et al. [53]	Xu et al. [43]	MCDL [34]	SLC-1	SLC-0
89.31	89.40	89.27	89.67	63.70	89.80	89.90	90.35	90.53	90.52

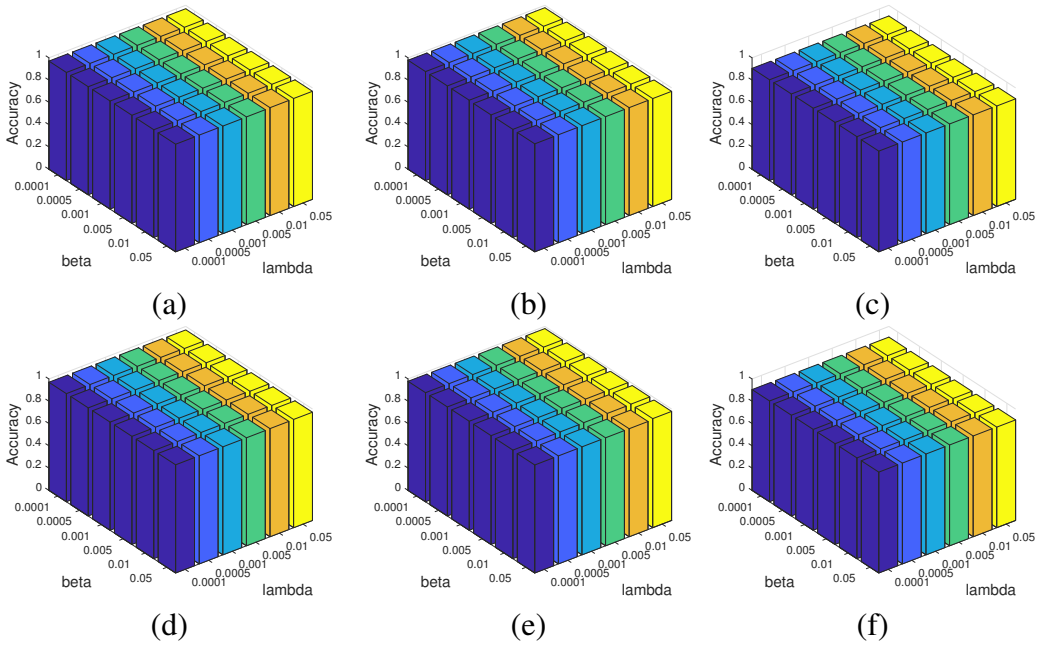

Fig. 5: Influence of parameter selection of λ and β on the recognition accuracy on datasets. (a),(b) and (c) correspond to the results using SLC-0 on AR-Face, Scene-15 and Dyntex++, respectively. (d), (e) and (f) correspond to the results using SLC-1 on AR-Face, Scene-15 and Dyntex++ respectively.

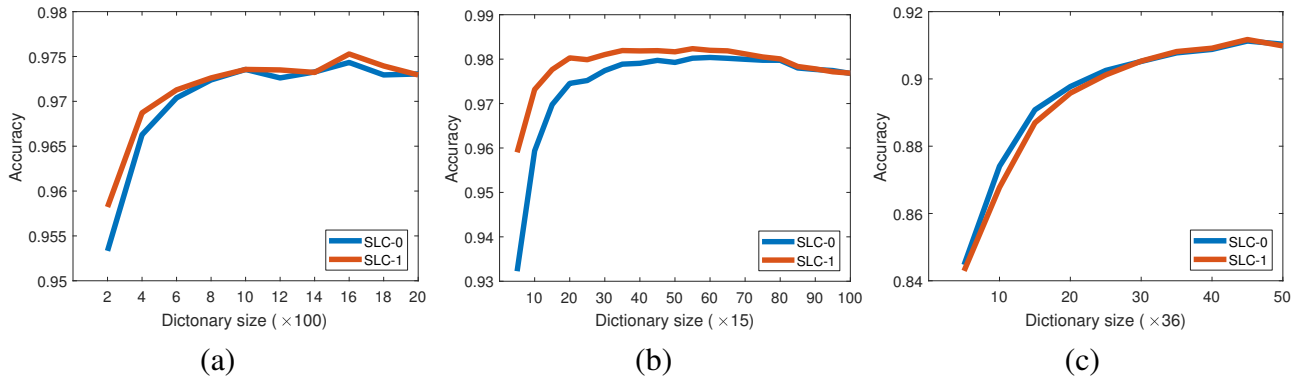

Fig. 6: Influence of parameter selection of T on the recognition accuracy on the test datasets. (a) AR-Face; (b) Scene-15; and (c) Dyntex++.

the impact of dictionary size to the classification accuracy. With this purpose, we gradually increased the dictionary size while keeping the other parameters fixed and tested the classification accuracies on the test datasets. The results are shown in 6. From the results we can see that when the dictionary size is too small, the performance is poor on all the datasets. The reason is that the representational power of such small dictionaries is insufficient for the tasks. When the dictionary size is increased to a moderate value, the performance of the proposed methods saturates and even sightly decreases when the dictionary size goes beyond certain threshold. The reason is that overfitting is likely to appear if the dictionary is too large.

4 Conclusions

Label consistency is very useful for structured sparse coding, especially in classification. However, existing label consistent sparse coding methods still have room for improvement. In this paper, we developed a sparse label consistency penalty for structured sparse coding, which is defined by the sparse approximation of the sparse code to the label consistency matrix. Built upon the sparse label consistency penalty, we proposed an effective and efficient sparse coding method for classification. The advantages of the proposed method are demonstrated using both synthetic data and several real datasets.

References

1. Aharon M, Elad M, Bruckstein A (2006) K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. Transactions on Signal Processing 54(11):4311–4322
2. Bagheri MA, Gao Q, Escalera S, Moeslund TB, Ren H, Etemad E (2017) Locality regularized group sparse coding for action recognition. Computer Vision and Image Understanding 158:106–114
3. Bao C, Ji H, Quan Y, Shen Z (2016) Dictionary learning for sparse coding: Algorithms and convergence analysis. Transactions on Pattern Analysis and Machine Intelligence 38(7):1356–1369
4. Boureau YL, Bach F, LeCun Y, Ponce J (2010) Learning mid-level features for recognition. In: Conference on Computer Vision and Pattern Recognition, IEEE, pp 2559–2566
5. Cai S, Zuo W, Zhang L, Feng X, Wang P (2014) Support vector guided dictionary learning. In: ECCV, Springer, pp 624–639
6. Chi YT, Ali M, Rajwade A, Ho J (2013) Block and group regularized sparse modeling for dictionary learning. In: Conference on Computer Vision and Pattern Recognition, IEEE, pp 377–382
7. Gao Q, Wang Q, Huang Y, Gao X, Hong X, Zhang H (2015) Dimensionality reduction by integrating sparse representation and fisher criterion and its applications. Transactions on Image Processing 24(12):5684–5695
8. Gao S, Tsang IW, Chia LT, Zhao P (2010) Local features are not lonely–laplacian sparse coding for image classification. In: Conference on Computer Vision and Pattern Recognition, IEEE, pp 3555–3561
9. Gao Y, Ma J, Yuille AL (2017) Semi-supervised sparse representation based classification for face recognition with insufficient labeled samples. Transactions on Image Processing 26(5):2545–2560
10. Ghanem B, Ahuja N (2010) Maximum margin distance learning for dynamic texture recognition. In: European Conference on Computer Vision, Springer, pp 223–236
11. Ghanem B, Ahuja N (2010) Maximum margin distance learning for dynamic texture recognition. In: European Conference on Computer Vision, Springer, pp 223–236

12. Huang K, Aviyente S (2006) Sparse representation for signal classification. In: Advances in Neural Information Processing Systems, pp 609–616
13. Jacob L, Obozinski G, Vert JP (2009) Group lasso with overlap and graph lasso. In: International Conference on Machine Learning, ACM, pp 433–440
14. Jenatton R, Audibert JY, Bach F (2011) Structured variable selection with sparsity-inducing norms. The Journal of Machine Learning Research 12:2777–2824
15. Jiang Z, Lin Z, Davis LS (2011) Learning a discriminative dictionary for sparse coding via label consistent K-SVD. In: Conference on Computer Vision and Pattern Recognition, IEEE, pp 1697–1704
16. Jiang Z, Zhang G, Davis LS (2012) Submodular dictionary learning for sparse coding. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp 3418–3425
17. Jiang Z, Lin Z, Davis L (2013) Label consistent K-SVD: Learning a discriminative dictionary for recognition. Transactions on Pattern Analysis and Machine Intelligence 35(11):2651–2664
18. Kim S, Xing EP (2010) Tree-guided group lasso for multi-task regression with structured sparsity. In: International Conference on Machine Learning, pp 543–550
19. Kong S, Wang D (2012) A dictionary learning approach for classification: separating the particularity and the commonality. In: ECCV, Springer, pp 186–199
20. Kviatkovsky I, Gabel M, Rivlin E, Shimshoni I (2017) On the equivalence of the lc-ksvd and the d-ksvd algorithms. Transactions on Pattern Analysis and Machine Intelligence 39(2):411–416
21. Lazebnik S, Schmid C, Ponce J (2006) Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: Conference on Computer Vision and Pattern Recognition, IEEE, vol 2, pp 2169–2178
22. Lian XC, Li Z, Lu BL, Zhang L (2010) Max-margin dictionary learning for multiclass image categorization. In: European Conference on Computer Vision, Springer, pp 157–170
23. Lian XC, Li Z, Wang C, Lu BL, Zhang L (2010) Probabilistic models for supervised dictionary learning. In: Conference on Computer Vision and Pattern Recognition, IEEE, pp 2305–2312
24. Liu J, Ye J (2010) Moreau-yosida regularization for grouped tree structure learning. In: Advances in Neural Information Processing Systems, vol 23, pp 1459–1467
25. Lu J, Wang G, Deng W, Moulin P (2014) Simultaneous feature and dictionary learning for image set based face recognition. In: European Conference on Computer Vision, Springer, pp 265–280
26. Mairal J, Bach F, Ponce J, Sapiro G, Zisserman A (2008) Discriminative learned dictionaries for local image analysis. In: Conference on Computer Vision and Pattern Recognition, IEEE, pp 1–8
27. Mairal J, Ponce J, Sapiro G, Zisserman A, Bach FR (2009) Supervised dictionary learning. In: Advances in Neural Information Processing Systems, pp 1033–1040
28. Majumdar A (2015) Discriminative label consistent dictionary learning. In: Image Processing (ICIP), 2015 IEEE International Conference on, IEEE, pp 1016–1020
29. Marsousi M, Li X, Plataniotis KN (2016) Shape-included label-consistent discriminative dictionary learning: An approach to detect and segment multi-class objects in images. In: International Conference on Image Processing, IEEE, pp 729–733
30. Martinez AM (1998) The ar face database. CVC Technical Report 24
31. Péteri R, Fazekas S, Huiskes MJ (2010) DynTex : A Comprehensive Database of Dynamic Textures. Pattern Recognition Letters 31:1627–1632
32. Pham DS, Venkatesh S (2008) Joint learning and dictionary construction for pattern recognition. In: Conference on Computer Vision and Pattern Recognition, IEEE, pp 1–8
33. Quan Y, Ji H, Shen Z (2014) Data-driven multi-scale non-local wavelet frame construction and image recovery. Journal of Scientific Computing pp 1–23
34. Quan Y, Xu Y, Sun Y, Huang Y (2016) Supervised dictionary learning with multiple classifier integration. Pattern Recognition 55:247–260
35. Quan Y, Xu Y, Sun Y, Huang Y, Ji H (2016) Sparse coding for classification via discrimination ensemble. In: Conference on Computer Vision and Pattern Recognition, pp 5839–5847
36. Ramirez I, Sprechmann P, Sapiro G (2010) Classification and clustering via dictionary learning with structured incoherence and shared features. In: Conference on Computer Vision and Pattern Recognition, IEEE, pp 3501–3508
37. Rosenblum K, Zelnik-Manor L, Eldar Y (2010) Dictionary optimization for block-sparse representations. In: Association for the Advancement of Artificial Intelligence Fall Symposium: Manifold Learning and Its Applications, pp 50–58
38. Sprechmann P, Ramirez I, Sapiro G, Eldar YC (2011) C-hilasso: A collaborative hierarchical sparse modeling framework. Transactions on Signal Processing 59(9):4183–4198
39. Sun Y, Liu Q, Tang J, Tao D (2014) Learning discriminative dictionary for group sparse representation. Transactions on Image Processing 23(9):3816–3828
40. Wang J, Yang J, Yu K, Lv F, Huang T, Gong Y (2010) Locality-constrained linear coding for image classification. In: Conference on Computer Vision and Pattern

Recognition, IEEE, pp 3360–3367
41. Wang X, Yang M, Shen L (2016) Structured regularized robust coding for face recognition. Neurocomputing 216:18–27
42. Wright J, Yang AY, Ganesh A, Sastry SS, Ma Y (2009) Robust face recognition via sparse representation. Transactions on Pattern Analysis and Machine Intelligence 31(2):210–227
43. Xu Y, Quan Y, Ling H, Ji H (2011) Dynamic texture classification using dynamic fractal analysis. In: International Conference on Computer Vision, IEEE, pp 1219–1226
44. Yang J, Yu K, Gong Y, Huang T (2009) Linear spatial pyramid matching using sparse coding for image classification. In: Conference on Computer Vision and Pattern Recognition, IEEE, pp 1794–1801
45. Yang J, Yu K, Huang T (2010) Supervised translation-invariant sparse coding. In: Conference on Computer Vision and Pattern Recognition, IEEE, pp 3517–3524
46. Yang J, Zhang L, Xu Y, Yang Jy (2012) Beyond sparsity: The role of l 1-optimizer in pattern classification. Pattern Recognition 45(3):1104–1118
47. Yang M, Zhang D, Feng X (2011) Fisher discrimination dictionary learning for sparse representation. In: International Conference on Computer Vision, IEEE, pp 543–550
48. Yang M, Dai D, Shen L, Van Gool L (2014) Latent dictionary learning for sparse representation based classification. In: Conference on Computer Vision and Pattern Recognition, pp 4138–4145
49. Yang M, Zhang L, Feng X, Zhang D (2014) Sparse representation based fisher discrimination dictionary learning for image classification. International Journal of Computer Vision 109(3):209–232
50. Zhang D, Liu P, Zhang K, Zhang H, Wang Q, Jing X (2015) Class relatedness oriented-discriminative dictionary learning for multiclass image classification. Pattern Recognition
51. Zhang H, Wu QJ, Chow TW, Zhao M (2012) A two-dimensional neighborhood preserving projection for appearance-based face recognition. Pattern Recognition 45(5):1866–1876
52. Zhang Q, Li B (2010) Discriminative K-SVD for dictionary learning in face recognition. In: Conference on Computer Vision and Pattern Recognition, IEEE, pp 2691–2698
53. Zhao G, Pietikainen M (2007) Dynamic texture recognition using local binary patterns with an application to facial expressions. Transactions on Pattern Analysis and Machine Intelligence 29(6):915–928
54. Zhou N, Shen Y, Peng J, Fan J (2012) Learning inter-related visual dictionary for object recognition. In: Conference on Computer Vision and Pattern Recognition, IEEE, pp 3490–3497