Blockwise coordinate descent schemes for efficient and effective dictionary learning

Bao-Di Liu, Yu-Xiong Wang, Bin Shen, Xue Li, Yu-Jin Zhang, Yan-Jiang Wang

1. Introduction

Recently, sparse representation based dictionary learning and its variants have been widely used in large-scale visual data sensing and analysis, such as image classification [40], image inpainting [36], image super-resolution [41], object detection [38], transfer learning [39], and image annotation [42]. Different from traditional decomposition frameworks like principal component analysis, non-negative matrix factorization [35], and low-rank factorization [37], sparse representation allows coding under over-complete bases, and thus makes the attained sparse codes capable of representing the data more adaptively and flexibly. Given that different dictionaries provide quite disparate representation ability, one central issue is how to build dictionary so as to find atoms identifying the best causes of the target data. This has triggered the emergence and boom of the so-called dictionary learning.

The primitive solution to exact sparse representation optimizes a quadratic function with \( \ell_0 \)-norm constraint; it is an \( NP \)-hard problem, though. Therefore, sparse approximation methods are considered instead. Typical approaches are Matching Pursuit (MP) [23] and Orthogonal Matching Pursuit (OMP) [27]. These methods are relatively fast iterative procedures that have been used extensively in practical applications. However, \( \ell_0 \)-norm constrained optimization problem is non-convex with multiple local optima, and the performance of these methods is thus not guaranteed in general. In 2006, Donoho [8] proved that \( \ell_0 \)-norm constraint could be well approximated by convex relaxation of \( \ell_1 \)-norm if sparse enough. For that matter, the solution to sparse approximation becomes a \( \ell_1 \)-regularized least-squares optimization problem (\( \ell_1-\ell_2 \)). At present, effective approaches to solving \( \ell_1-\ell_2 \) include active-set methods, such as Homotopy [26], LARS [29], and feature-sign search [15], gradient methods (also called first-order or iterative soft-thresholding methods), such as Coordinate descent and analysis, such as image classification, its variants have been widely used in large-scale visual data sensing and analysis, such as image classification [40], image inpainting [36], image super-resolution [41], object detection [38], transfer learning [39], and image annotation [42]. Different from traditional decomposition frameworks like principal component analysis, non-negative matrix factorization [35], and low-rank factorization [37], sparse representation allows coding under over-complete bases, and thus makes the attained sparse codes capable of representing the data more adaptively and flexibly. Given that different dictionaries provide quite disparate representation ability, one central issue is how to build dictionary so as to find atoms identifying the best causes of the target data. This has triggered the emergence and boom of the so-called dictionary learning.

The primitive solution to exact sparse representation optimizes a quadratic function with \( \ell_0 \)-norm constraint; it is an \( NP \)-hard problem, though. Therefore, sparse approximation methods are considered instead. Typical approaches are Matching Pursuit (MP) [23] and Orthogonal Matching Pursuit (OMP) [27]. These methods are relatively fast iterative procedures that have been used extensively in practical applications. However, \( \ell_0 \)-norm constrained optimization problem is non-convex with multiple local optima, and the performance of these methods is thus not guaranteed in general. In 2006, Donoho [8] proved that \( \ell_0 \)-norm constraint could be well approximated by convex relaxation of \( \ell_1 \)-norm if sparse enough. For that matter, the solution to sparse approximation becomes a \( \ell_1 \)-regularized least-squares optimization problem (\( \ell_1-\ell_2 \)). At present, effective approaches to solving \( \ell_1-\ell_2 \) include active-set methods, such as Homotopy [26], LARS [29], and feature-sign search [15], gradient methods (also called first-order or iterative soft-thresholding methods), such as
operator-splitting [6], iterative splitting and thresholding [7], and fixed-point iteration [12]. Active-set methods are efficient for small or medium-sized problems, or when requiring very sparse solutions. Gradient methods need more iterations especially when the solution is not very sparse or the initialization is not ideal.

Meanwhile, dictionary learning can be considered to optimize a least-squares problem with quadratic constraints ($\ell_2 - \ell_2$). Currently, effective algorithms involve MOD [9], K-SVD [1], gradient descent [4], ODL [22], and Lagrange dual [15]. MOD updates all the entries of the dictionary simultaneously, but it is not guaranteed to converge. K-SVD sequentially updates the dictionary column-wise together with the corresponding sparse codes using singular value decomposition (SVD), which is computationally expensive. Gradient descent often shows slow convergence. The ODL method processes the variables in a sequential and element-wise fashion.

Here we consider the simultaneous sparse coding and dictionary learning (SC–DL) problem commonly formulated as alternating $\ell_1 - \ell_1$ and $\ell_2 - \ell_1$ optimization. To the best of our knowledge, among the existing approaches, the feature-sign search/Lagrangian dual (FS-LD) algorithm speeds up the sparse coding procedure significantly and achieves optimum performance up to now. However, the cyclical sign feedback and adjustment actually abuse its efficiency in the feature-sign search phase. As for the Lagrangian dual phase, the adopted Newton’s method needs several iterations and shows low convergence rate. Besides, the presence of the matrix inversion will introduce numerical difficulties and instability in some situations as demonstrated later in our experimental evaluation.

In this paper, instead, we recast the problem of SC–DL under a much simpler scheme, i.e., blockwise coordinate descent. Coordinate descent algorithms actually were proposed in solving the sparsity induced least-squares minimization long ago. However, its powerful performance, i.e., efficiency and effectiveness, has not been fully appreciated. We can see how it will be revealed by appropriate variable partition and resorting to simple update rules. In particular, if we focus on one basic single variable, a direct closed-form solution will be obtained based on the property of a much simpler univariate parabolic function. The analytical solutions of several variables can be further unified into a parallel vector formula according to the separability of the objective function. In addition, this optimization scheme is suitable for both $\ell_1 - \ell_1$ and $\ell_2 - \ell_1$ with only slight modifications. Hence, the proposed algorithm is simple, efficient, and effective with theoretically guaranteed convergence. We demonstrate that our algorithm significantly accelerates the solution to SC–DL, and has superior solutions especially in the case of relatively small number of samples or seeking for comparatively much sparser codes.

Our main contributions are four-fold: (1) We explore the potential of simple coordinate descent in solving SC–DL, which has been largely ignored partially due to its too simplicity and poor performance generally. Our approach is different from the typical direction of relying on more sophisticated first or second-order quadratic optimizers. (2) We exploit proper partition and reorganization of variables to be optimized, and generalize the conventional coordinate-wise soft-threshold for SC and ODL for DL into a blockwise fashion, making parallel computation feasible. (3) We treat elements both in dictionary and codes homogeneously, and decouple SC–DL optimization as several blockwise alternate subproblems rather than the current approach by decoupling it first as two subproblems (i.e., alternate SC and DL) using different optimizers and treating elements in dictionary and codes separately. Hence, we unify coordinate descent for both SC and DL under the same framework, which was only addressed for either of them separately. (4) The resulting algorithm is surprisingly competitive in seeking better solutions much faster. To the best of our knowledge, it is the fastest procedure for SC–DL to date.

A preliminary version of this work appeared as Liu et al. [20]. We clarify many technical details omitted in the previous version, present results on substantially extended scenarios (e.g., learning large, highly over-complete representations) and applications (e.g., image classification), and offer an in-depth analysis of how, why, and when the proposed algorithm is superior to the standard approaches.

The rest of this paper is organized as follows. In Section 2 problem statement is reviewed briefly. We elaborate the proposed algorithm in solving $\ell_1 - \ell_1$ and $\ell_2 - \ell_1$ as well as convergence analysis in Section 3. Section 4 shows experimental results and analysis. We provide further application of the proposed algorithm in image classification tasks on standard benchmarks in Section 5. Discussions and conclusions are drawn in Section 6.

2. Problem statement

The goal of sparse representation based dictionary learning is to represent vectors approximately as a weighted linear combination of only few learned basis vectors. Let $X \in \mathbb{R}^{D \times N}$ be the input data matrix, where $D$ and $N$ are the dimension and number of the data vectors, respectively. Let $B \in \mathbb{R}^{K \times D}$ and $S \in \mathbb{R}^{K \times N}$ denote the dictionary (basis matrix) and corresponding sparse codes (also called coefficient matrix), respectively, where $K$ is the size of the dictionary. Sparse representation aims at solving the following optimization problem:

$$
\min_{B, S} \quad \|B \cdot S\|_2^2 + 2\alpha \|S\|_1 \\
\text{s.t.} \quad \|B_i\|_2 \leq 1, \quad \forall i = 1, 2, ..., K.
$$

(1)

Here, $A_n$, and $A_i$ denote the $n$th column and $i$th row vectors of a matrix $A$, respectively. The $\ell_1$-norm regularization term is adopted to enforce sparsity of $S$ and $\alpha$ is the regularization parameter to control the tradeoff between fitting goodness and sparseness.

While the optimization problem (1) is not jointly convex in both $S$ and $B$, it is separately convex in either $S$ (with $B$ fixed) or $B$ (with $S$ fixed). So it can be decoupled into the following two optimization subproblems, which can be solved by alternating minimizations [15].

$$
\ell_1 - \ell_1 \text{minimization problem is as follows:}
$$

$$
\min_{S} \quad \|B \cdot S\|_2^2 + 2\alpha \|S\|_1.
$$

(2)

$$
\ell_2 - \ell_1 \text{minimization problem is as follows:}
$$

$$
\min_{B} \quad \|B \cdot S\|_2^2 \\
\text{s.t.} \quad \|B_i\|_2 \leq 1, \quad \forall i = 1, 2, ..., K.
$$

(3)

3. Proposed algorithm for sparse coding and dictionary learning

Consider the $\ell_1 - \ell_1$ minimization problem first. If we concentrate on the basic element, i.e., one single variable, with the remaining ones fixed at a time, the objective function in Eqn. (2) reduces to a much simpler univariate parabolic function. A direct, closed-form solution minimizing the corresponding cost function can be thus easily obtained based on the convexity and monotonic property of the parabolic function. Moreover, according to the separability of the objective function, the analytical solutions of several independent variables (in this case the entries in the same row) can be unified into a block formula, making parallel computation and further acceleration possible. In other words, upon such block partition mode, an exact blockwise coordinate descent can be carried out effectively. As for the $\ell_2 - \ell_1$ minimization problem, this strategy is also applicable. In this sense, the
solutions to these two optimization problems can be tackled under the same scheme. The specific block partition modes and the corresponding update rules of finding sparse codes and learning dictionary are validated by the following two theorems, respectively.

3.1 $\ell_1$-$\ell_2$ minimization for finding sparse codes

**Theorem 1.** \( \forall k \in \{1, 2, \ldots, K\} \) with \( \{b_{n \neq k} = 1, \ldots, K\} / B_k \) (i.e., all the rows in \( S \) except for the \( k \)th row) and \( B \) fixed, the minimization of Eqn. (2) with respect to the single row has the closed-form solution

\[
S_{i_k} = \arg \min_{S_{i_k}} \left\{ \|X - BS\|_F^2 + 2 \alpha \|S_{i_k}\|_1 \right\} = \max \left\{ \|B_{k_i}\|^2 \right\} + \min \left\{ \|B_{k_i}\|^2 \right\}, \quad \alpha.
\]

(4)

where

\[
S_{k_i}^* = \begin{cases} \{ S_{k_i} \} & p \neq k \\ \{ 0 \} & p = k \end{cases}
\]

**Proof.** The objective function in Eqn. (2) can be rewritten as

\[
f(S) = \|X - BS\|_F^2 + 2 \alpha \|S\|_1 = \text{tr} \left\{ X^T X - 2X^T BS + S^T B^T BS \right\} + 2 \alpha \|S\|_1
\]

\[
= \text{tr} \left\{ X^T X \right\} - 2 \sum_{n=1}^{N} \left\{ X_n^T B_{k_i} S_{k_i} + \sum_{n=1}^{N} S_{k_i} B^T_n S_{k_i} \right\} + 2 \alpha \sum_{k=1}^{K} \|S_{k_i}\|_1,
\]

(5)

where tr(A) represents the trace of matrix A.

Ignoring the constant term \( \text{tr} \{ X^T X \} \), the objective function of \( S_{k_i} \) reduces to Eqn. (6) with \( B \) fixed:

\[
f(S_{k_i}) = S_{k_i}^T B_{k_i}^T B_{k_i} S_{k_i} + 2 \alpha \|S_{k_i}\|_1 + 2 \alpha \left\{ \sum_{k=1}^{K} \|S_{k_i}\|_1 \right\}
\]

(6)

And then the objective function of \( S_{k_i} \) in Eqn. (6) reduces to Eqn. (7) with \( B \) and \( \{S_1, S_2, \ldots, S_K\} \) (i.e., all the elements in \( S_{k_i} \) except \( S_{k_i} \)) fixed.

\[
f(S_{k_i}) = S_{k_i}^T B_{k_i}^T B_{k_i} S_{k_i} + 2 \alpha \|S_{k_i}\|_1 + 2 \alpha \left\{ \sum_{k=1}^{K} \|S_{k_i}\|_1 \right\}
\]

(7)

where \( H_{k_i} = \{B_{k_i} X_{k_i} - \sum_{j=1}^{K} B_{j} S_{j}\} \).

When \( \|B_{k_i}\|_1 > 0 \), \( f(S_{k_i}) \) is piece-wise parabolic function with \( \|B_{k_i}\|_1 \). Based on the convexity and monotonic property of the parabolic function, it is not difficult to know that \( f(S_{k_i}) \) reaches the minimum at the unique point

\[
S_{k_i} = \text{max} \{H_{k_i}, \alpha\} + \min \{H_{k_i}, -\alpha\}.
\]

(8)

Furthermore, given that the optimal value for \( S_{k_i} \) does not depend on the other entries in the same row, each entire row of \( S \) can be optimized simultaneously. That is,

\[
S_{i_k} = \text{max} \{H_{i_k}, \alpha\} + \min \{H_{i_k}, -\alpha\}.
\]

(9)

3.2 $\ell_2$-$\ell_2$ minimization for learning dictionary

**Theorem 2.** \( \forall k \in \{1, 2, \ldots, K\} \) with \( S \) and \( |B_{k \neq k} = 1, \ldots, K\} / B_k \) (i.e., all the columns in \( B \) except for the \( k \)th column) fixed, the constrained minimization problem of Eqn. (3) with respect to the single column has the closed-form solution

\[
B_{i_k} = \arg \min_{B_{i_k}} \|X - BS\|_F^2 = \frac{X[S_{i_k}]^T - B_k^T S_{i_k}^T}{\|X[S_{i_k}]^T - B_k^T S_{i_k}^T\|_2}
\]

(10)

where

\[
\tilde{B}_{i_k} = \begin{cases} B_{i_k} & p \neq k \\ 0 & p = k \end{cases}
\]

**Proof.** Without the sparseness regularization term in Eqn. (2) and additional constraints in Eqn. (3), \( S_{i_k} \) and \( B_{i_k} \) are dual in objective function \( \|X - BS\|_F^2 \) for \( \forall k \in \{1, 2, \ldots, K\} \). Hence, \( \forall d \in \{1, 2, \ldots, D\} \), \( \forall k \in \{1, 2, \ldots, K\} \) with \( |B_{d\neq k} = 1, \ldots, K\} / B_k \) (i.e., all the elements except \( B_k \)) and \( S \) fixed, the unconstrained single variable minimization problem of Eqn. (3) has the closed-form solution

\[
B_{i_k} = \arg \min_{B_{i_k}} \|X - BS\|_F^2 = \frac{X[S_{i_k}]^T - B_k^T S_{i_k}^T}{\|X[S_{i_k}]^T - B_k^T S_{i_k}^T\|_2}
\]

(11)

where \( \|S_{i_k}\|_1 > 0 \).

Similarly, since the optimal value for \( B_{i_k} \) does not depend on the other entries in the same column, the objective function of \( B_{i_k} \) reduces to Eqn. (12) with \( S \) fixed:

\[
f(B_{i_k}) = S_{i_k} [S_{i_k}]^T [B_{i_k}][B_{i_k}] + 2 [B_{i_k}] \{ [B_{i_k} S_{i_k}] - X[S_{i_k}]^T \}
\]

(12)

When imposing the norm constraint, i.e., \( \|B_{i_k}\|_2 = 1 \), Eqn. (12) becomes

\[
f(B_{i_k}) = 2 [B_{i_k}] \{ [B_{i_k} S_{i_k}] - X[S_{i_k}]^T \} + S_{i_k} [S_{i_k}]^T.
\]

(13)

Hence, the original constrained minimization problem becomes a linear programming under a unit norm constraint, whose solution is

\[
\frac{X[S_{i_k}]^T - B_k^T S_{i_k}^T}{\|X[S_{i_k}]^T - B_k^T S_{i_k}^T\|_2}
\]

3.3 Overall algorithm

Blockwise coordinate descent for dictionary learning (BCDDL) algorithm is shown in Algorithm 1. Here, \( \mathbf{1} \in \mathbb{R}^{K 	imes K} \) is a square matrix with all entries, \( \mathbf{1}_f \) is the identity matrix, and \( \odot \) indicates element-wise dot product. By iterating \( S \) and \( B \) alternately, the sparse codes are obtained, and the corresponding dictionary is learned.

**Algorithm 1.** Blockwise coordinate descent for sparse coding and dictionary learning.

**Require:** Data matrix \( X \in \mathbb{R}^{D \times N} \) and \( K \)

1. \( B \leftarrow \text{rand}(D, K), B_{i_k} = \left\{ B_{i_k} \right\}_{i=1,...,K} \backslash k, S \leftarrow \text{zeros}(K, N) \)
2. \( \text{iter} = 0 \)
3. while \((f(\text{iter}) - f(\text{iter} + 1))/f(\text{iter}) > 1e^-6\) do
4. \( \text{iter} \leftarrow \text{iter} + 1 \)
5. \( \text{Update} S \):
6. Compute \( A = (B_i ^T B) \odot (1 - I) \) and \( E = B_i ^T X \)
7. for \( k = 1; k \leq K; k++ \) do
8. \( S_{i_k} = \text{max} \{E_{i_k} - A_{i_k} S_{i_k}, \alpha\} \)
9. \( + \min \{E_{i_k} - A_{i_k} S_{i_k}, -\alpha\} \)
7. end for
10. \( \text{Update} B \):
11. Compute \( G = (S_i ^T) \odot (1 - I), W = X S_i ^T \)
12. for \( k = 1; k \leq K; k++ \) do

Please cite this article as: B.-D. Liu, et al., Blockwise coordinate descent schemes for efficient and effective dictionary learning, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.06.096
13: \[ \mathbf{B}_{a} = \frac{\mathbf{W}_{a} - \mathbf{B}_{c} \mathbf{S}}{\mathbf{W}_{a} - \mathbf{B}_{c} \mathbf{S}}; \]
14: end for
15: \textbf{Update the objective function:}
16: \[ f(\text{iter}) = \| \mathbf{X} - \mathbf{B}\mathbf{S} \|^2 + 2\alpha \| \mathbf{S} \|_1; \]
17: end while
18: \textbf{return} \( \mathbf{B} \) and \( \mathbf{S} \)

3.4. Analysis and comment

\textbf{Theorem 3.} The objective function in Eqn. (1) is nonincreasing under the update rules given in Eqns. (4) and (10) and the variables in the objective function (\( \mathbf{B} \) and \( \mathbf{S} \)) converge.

\textbf{Proof.} The objective function in Eqn. (1) is nondifferentiable and nonconvex. Under the optimization framework of BCDDL, it is partitioned as 2 K blocks, in which each block is a column of \( \mathbf{B} \) or a row of \( \mathbf{S} \):

\[ f(\mathbf{B}_1, \mathbf{B}_2, \ldots; \mathbf{B}_k, \mathbf{S}_1, \mathbf{S}_2, \ldots; \mathbf{S}_k) = \sum_{k=1}^{K} \left\| \mathbf{X} - \sum_{i=1}^{K} \mathbf{B}_i \mathbf{S}_i \right\|_F^2 + 2\alpha \sum_{i=1}^{K} \| \mathbf{S}_i \|_1. \]

\textbf{s.t.} \[ \| \mathbf{B}_i \|_F^2 \leq 1, \quad \forall k = 1, 2, \ldots, K. \] (14)

Since the exact minimization point is obtained by Eqn. (4) or Eqn. (10), each operation updates \( \mathbf{S}_1, \ldots, \mathbf{S}_k, \mathbf{B}_1, \ldots, \mathbf{B}_k \) alternately and it monotonically decreases the objective function in Eqn. (1). Considering that the objective function is obviously bounded below, it converges. Now consider the convergence of the variables \( \mathbf{B} \) and \( \mathbf{S} \) in the function. Each subproblem with respect to \( \mathbf{B}_1, \mathbf{B}_2, \ldots; \mathbf{B}_k, \mathbf{S}_1, \mathbf{S}_2, \ldots; \mathbf{S}_k \) becomes convex and it satisfies the separability and regularity properties proposed in [30], and the optimal value for a given block of variables is uniquely achieved by the solutions (Eqns. (4) and (10)) due to the strict convexity of subproblems at each iteration. These properties make the subsequence \( (\mathbf{B} \) and \( \mathbf{S} \)) via alternating minimization by blockwise coordinate descent converge to a stationary point. The detailed proof of the convergence of (blockwise) coordinate descent for functions satisfying some mild conditions is referred to as Theorem 1 in [30] and also in [2].

The time complexity of one iteration round in K-SVD, MOD, FS-LD, and BCDDL is shown in Table 1, respectively. In Table 1, \( K \) represents the size of dictionary; \( D \) represents the dimension of features; \( N \) represents the number of features; \( L \) represents the number of nonzero elements in each column of the sparse codes; \( \alpha \) represents the number of iterations at Step 2 in Algorithm 1 of FS-LD [15]; \( \beta \) represents the number of iterations at step 3 in Algorithm 1 of FS-LD [15]. From Table 1, we can see that the time complexity of BCDDL (DL), and sparse coding, i.e., BCDDL (SC), is the same. This is consistent with our homogeneous optimization procedure with respect to both \( \mathbf{B} \) and \( \mathbf{S} \). Moreover, the time complexity of BCDDL is less than the other three algorithms, making BCDDL more efficient. For memory consumption, it is the same for K-SVD, MOD, and FS-LD algorithms, which mainly includes the memory storage of the dictionary \( \mathbf{B} \), sample \( \mathbf{X} \), and their corresponding sparse codes \( \mathbf{S} \) for sequential processing of each sample. By contrast, to process all the data once, BCDDL includes an additional intermediate variable \( \mathbf{C} = \mathbf{B}^{T} \mathbf{X} \), which consumes the same memory as \( \mathbf{S} \).

The proposed algorithm seems simple. However, it demonstrates surprising efficiency and effectiveness. The fundamental reason for such an enhancement might lie in that it fully excite and cohere the sparse natural of the problem. First, for a single variable, the coordinate-wise update is actually a closed-form soft-thresholding operator. If the underlying elements are quite sparse, the soft-thresholding operator is able to efficiently detect the zero by a simple check. Only a small number of updates are thus needed. Second, what the algorithm has done can be viewed as simple recombination and rearrangement of the soft-thresholding operators; whereas, this trick of the appropriate block partition brings in effective parallel computation. Since the solutions are expected to be sparse, which means that the elements of the matrices will be almost disjoint, it makes the coupling between variables in the least-squares subproblems rather low, and thereby allows an exact coordinate descent method to be capable of solving the nearly separable subproblems efficiently. Third, our approach ensures continuous decrease in the objective function, yet not so greedy as other methods like MOD so as to avoid overshooting. Finally, another obvious characteristic is that it adopts the unified or homogeneous update scheme for both dictionary learning and sparse coding, which differs from other existing approaches. It might get extra benefit from this duality perspective of basis and coefficient matrix. Besides, it is parameter free which debases the complexity of usage.

4. Experimental results

The performance of our blockwise coordinate descent for dictionary learning (BCDDL) algorithm is evaluated on three benchmark datasets: natural images [25], Caltech-101 [16], and Scene 15 [14]. All experiments are implemented on a PC with Intel Core i5 M560 2.67 GHz CPU, 8 GB RAM. The software environment is Windows 7 and Matlab 7.12.

For the natural images dataset, patches are randomly selected. In each patch, every pixel value forms a vector as its representation. For Caltech-101 and Scene 15, 16 × 16 size patches are densely sampled from images and are then represented by the SIFT descriptors with grid size 4 × 4. Given that the FS-LD algorithm significantly outperforms other approaches [15], a systematic evaluation and comparison mainly between proposed the proposed method and FS-LD algorithm are carried out with regard to their performance in terms of running time, convergence rate, learning dictionary in certain cases, and the like.

4.1. Running time for learning dictionary and finding sparse codes

In this section, the algorithm is evaluated on the natural images dataset with a set of 1000 input vectors (each 14 × 14 pixels) randomly selected. \( \alpha \) is set to 0.2. The size of dictionary is 512. We compare our algorithm with the FS-LD algorithm. To avoid being influenced by random environmental variations, the experiment is repeated 20 times. In each experiment, 100 iterations are operated to optimize dictionary and sparse codes alternately. Fig. 1 shows the running time per iteration. Fig. 1(a) shows the running
time for $\ell_2 - \ell_1$ s. Fig. 1(b) shows the running time of $\ell_2 - \ell_1$ s. It is obvious that our algorithm runs much faster for both $\ell_2 - \ell_1$ and $\ell_1 - \ell_1$ s per iteration. The running time of FS-LD algorithm is especially long in the first few iterations, because the energy is scattered in the initial stages, which requires more feedback and adjustment to determine the response of sparse codes. The average ratio of running time per iteration of the FS-LD algorithm to that of ours is about 12.98 s and 31.86 s for $\ell_2 - \ell_1$ s and $\ell_1 - \ell_1$ s, respectively.

4.2. Convergence rate

The speed of convergence is another important factor to evaluate algorithms. Here, a set of 1000 input vectors (each $14 \times 14$ pixels) randomly sampled from the natural images dataset are used. $\alpha$ is set to 0.2, the size of dictionary is 512, and 100 iterations are operated to learn dictionary and find sparse codes alternately. Fig. 2(a) shows the objective function values with running time of these two algorithms, where our algorithm converges much faster. Fig. 2(b) shows the objective function values with iterations of these two algorithms. Hence, when combining the experimental results in Figs. 1 and 2, it demonstrates that our algorithm has faster convergence with lower cost per iteration.

4.3. Total time for dictionary learning

With respect to the running time per iteration, our BCDDL algorithm runs much faster than the FS-LD algorithm. We then evaluate the separate stage for learning dictionary on the above three datasets, which is the core concern in training. A set of 1000 input vectors (each $14 \times 14$ pixels for natural images and 128-D SIFT for Caltech-101 and Scene 15, respectively) are chosen. For the natural images dataset, $\alpha$ is set to 0.2 and the size of dictionary is 512; for the Caltech-101 and Scene 15 datasets, $\alpha$ is set to 0.1 and the size of dictionary is also 512. The stopping condition is that the relative change of the objective function value between successive iterations is less than $1e-6$ (i.e., $(\text{fold } - \text{new})/\text{fold} < 1e-6$). The running time of our BCDDL algorithm is 71.29 s, 30.82 s, and 25.53 s in the natural images, Caltech-101, and Scene 15 datasets, respectively, while that of the FS-LD algorithm is 1452.50 s, 231.48 s, and 398.22 s. Hence, the total time for dictionary learning also demonstrates that the speed of our BCDDL algorithm outperforms the FS-LD algorithm significantly.

4.4. The relationship between reconstruction error and sparsity

The effectiveness of our algorithm can be evaluated by the relationship between reconstruction error and sparsity to some extent. Here the sparsity is defined as the average number of nonzero entities in each column of the sparse code matrix. Apart from the remarkable speedup presented above, Fig. 3 shows such variation tendency. The conditions are similar to the previous experimental conditions, except that for the natural images dataset, $\alpha$ is set to 0.6, 0.7, ..., 1.5 and the size of dictionary is 256, while for the Caltech-101 and Scene 15 datasets, $\alpha$ is set to 0.09, 0.11, ..., 0.3 and the size of dictionary is 256.

From Fig. 3, both algorithms achieve the approximately equal reconstruction error when the corresponding codes are not too sparse. However, our algorithm attains less reconstruction error for lower sparsity values, which correspond to higher sparseness. This indicates that our algorithm is capable of finding comparatively much sparser codes while maintaining lower reconstruction error. This property is helpful for some real-world applications demonstrated in the following sections.

4.5. Learning dictionary with extremely sparse codes

Fig. 4 gives a comparison of the learned natural images dictionary between these two algorithms when the codes are extremely sparse. Fig. 4(a) is the result of the FS-LD algorithm and (b) is the result of our BCDDL algorithm. 120,000 input vectors (each $14 \times 14$ pixels) are utilized to infer a set of 256 bases in both cases. $\alpha$ is set to 2.0 and 100 iterations are operated for dictionary learning. Notice that there are several basis images with all zero pixel values in Fig. 4(a) (The regions marked with the red boxes), which implies that the corresponding basis vectors make no sense. On the contrary, our BCDDL algorithm is still adaptive for such situation. So even in the case of extremely sparse codes, our algorithm remains effective.
4.6. Comparison on a synthetic experiment

To demonstrate the effectiveness of our BCDDL algorithm, a synthetic experiment is carried out. A random dictionary (i.i.d. Gaussian entries, normalized columns) of size $20 \times 50$ is generated. From the generated dictionary, 1500 samples are produced by a random combination of 3 atoms, with coefficients drawn from the normal distribution $\mathcal{N}(0, 1)$. Each sample is contaminated by a random zero-mean Gaussian noise with signal-to-noise ratio of 20. We carry out 100 iterations to recover the original dictionary by four common dictionary learning methods. Fig. 5 shows the ratio of recovered atoms. From Fig. 5, we can see that our BCDDL algorithm is capable of recovering 100% atoms in 0.58 s. The FS–LD algorithm recovered 96% atoms in 70.07 s. The K-SVD algorithm recovered 90% atoms in 65.64 s. The MOD algorithm recovered 82% atoms in 47.70 s.

4.7. Learning highly over-complete dictionary of natural images

By using our BCDDL algorithm, highly over-complete dictionary can be learned in reasonable computational time. Figs. 6 and 7 show the learned over-complete natural images dictionary. Fig. 6 is a set of 1024 bases (each 14 × 14 pixels, 400,000 input vectors) and Fig. 7 is a set of 2000 bases (each 20 × 20 pixels, 400,000 input vectors). Fig. 8 shows the first 256 bases in the 2000 bases. $\alpha$ is set to 0.4 and the stopping condition is that the relative change of the objective function value between successive iterations is less than $2 \times 10^{-4}$ (i.e. $(\text{fold}_{\text{new}} - \text{fold}_{\text{old}})/\text{fold}_{\text{old}} < 2 \times 10^{-4}$). Both the learned dictionaries obviously comprise bases showing spatially localized, oriented, and bandpass structures, which identify the most important building blocks in natural images and provide prerequisites for efficient sparse coding. What is more, it is sufficient to account for the principal spatial properties of simple-cell receptive fields.
Table 2 summarizes the running time comparison between FS–LD and BCDDL for learning dictionary with different number of bases and samples. Both algorithms obtain similar and comparable highly over-complete dictionary in these cases. However, our algorithm runs much faster than the FS–LD algorithm. Another noticeable observation is that our algorithm reduces the overall time significantly especially when the number of samples increases. To be specific, for 1024 bases, the ratio of the running time of the FS–LD algorithm to that of ours is 5.69 with 4000 samples, while 8.08 with 40,000 samples. For 2000 bases, the ratios are 5.65 and 9.37. In other words, our algorithm outperforms the FS–LD algorithm in both speed and acceleration.

5. Application

The efficiency and effectiveness of the proposed BCDDL algorithm have been fully illustrated in the previous section. Here we apply it to image classification to further manifest its performance. Image classification attempts to associate images with semantic labels automatically and to help machines understand images readily. It has become a hot topic in computer vision in recent years, with potential applications including content-based image retrieval and scene understanding for robot navigation. The most common framework is the discriminative model [14,34,33]. There are five main steps, including feature extraction, dictionary learning, image coding, image pooling, and SVM-based classification. Our algorithm is applied to dictionary learning and image coding steps for image classification.

5.1. Experimental settings

We evaluate our algorithm on four common datasets: UIUC-Sports dataset [18], Scene 15 dataset [17,14,24], Caltech-101 dataset [16] and Caltech-256 dataset [11]. For each dataset, the data are randomly split into training set and testing set based on published protocols. To make the results more convincing, the experimental process is repeated 8 times, and the mean and standard deviation of the classification accuracy are recorded. Each image is resized with maximum side 300 pixels first. As for the image features, densely sampling patches of $16 \times 16$ are extracted with step size 8 pixels. 128 dimensional SIFT descriptors [21] are obtained with grid size $4 \times 4$, and normalized to 1 with $\ell_2$-norm. The samples used for learning dictionary are 120,000 and the

Table 2

<table>
<thead>
<tr>
<th>Methods</th>
<th>FS–LD (min)</th>
<th>BCDDL (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000 (samples) + 1024 (bases)</td>
<td>35.45</td>
<td>6.23</td>
</tr>
<tr>
<td>40,000 (samples) + 1024 (bases)</td>
<td>262.00</td>
<td>32.42</td>
</tr>
<tr>
<td>4000 (samples) + 2000 (bases)</td>
<td>90.35</td>
<td>15.98</td>
</tr>
<tr>
<td>40,000 (samples) + 2000 (bases)</td>
<td>1362.80</td>
<td>145.40</td>
</tr>
</tbody>
</table>

1 For the UIUC-Sports dataset, we resize the maximum side to 400 due to the high resolution of original images.

Please cite this article as: B.-D. Liu, et al., Blockwise coordinate descent schemes for efficient and effective dictionary learning, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.06.096
5.2. UIUC-Sports dataset

For the UIUC-Sports dataset [18], there are 8 classes with 1579 images in total: rowing (250 images), badminton (200 images), polo (182 images), bocce (137 images), snow boarding (190 images), croquet (236 images), sailing (190 images), and rock climbing (194 images). For each class, the sizes and even the number of instances are quite different, and the poses of the objects vary a lot. The background is also highly clutter and discrepant. Some images from different classes may have similar background. Fig. 9 shows some example images. We follow the standard setup: 70 images per class are randomly selected as the training data, and the rest for testing. Table 3 shows the performance of different methods.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Average classification rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIKVQ [33]</td>
<td>81.87 ± 1.14</td>
</tr>
<tr>
<td>OCSVM [33]</td>
<td>81.33 ± 1.56</td>
</tr>
<tr>
<td>ScSPM [34,10]</td>
<td>82.74 ± 1.46</td>
</tr>
<tr>
<td>LLIC [12]</td>
<td>87.30 ± 0.82</td>
</tr>
<tr>
<td>MSSR [19]</td>
<td>89.77 ± 1.12</td>
</tr>
<tr>
<td>IFV [28]</td>
<td>90.82 ± 0.92</td>
</tr>
<tr>
<td>BCDDL</td>
<td>87.99 ± 0.65</td>
</tr>
</tbody>
</table>

2 All the results of OCSVM and HIKVQ are based on step size 8 and without concatenated Sobel images.

5.3. Scene 15 dataset

For the Scene 15 dataset, there are 15 classes with 4485 images in total. Each class varies from 200 to 400 images. The images contain not only indoor scenes, such as bedroom, living room, PARoffice, kitchen, and store, but also outdoor scenes, such as industrial, forest, mountain, tallbuilding, highway, street, and opencountry. Fig. 10 shows some example images. We use an identical experimental setup as [14]: 100 images per class are randomly selected as the training data and the rest for testing. Table 4 shows the performance of different methods.

5.4. Caltech-101 dataset

The Caltech-101 dataset introduced in [16] contains 102 classes, one of which is the background. After removing the background class, the remaining 101 classes with 8677 images in total are used for classification, with each class varying from 31 to 800 images. We follow the standard experimental setup for this dataset, where 15 and 30 images per category are selected as the training set, and the rest for the testing set (the maximum is 50 images per category for testing). Table 5 shows the performance of different methods.

5.5. Caltech-256 dataset

The Caltech-256 dataset [11] contains 257 classes, one of which is the background. After removing the background class, the remaining 256 classes with a total of 29,780 images are used for classification. We follow the standard experimental setup for this dataset: 15, 30, 45, and 60 training images per category and 15 testing images per category. Table 6 shows the performance of different methods.

Please cite this article as: B.-D. Liu, et al., Blockwise coordinate descent schemes for efficient and effective dictionary learning, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.06.096
Further comparison of four dictionary learning approaches on image classification

The classical and popular dictionary learning (DL) techniques involve MOD, K-SVD, and Lagrange-Dual in the FS-LD algorithm. Here, we give a further comparison among the proposed BCDDL algorithm and the other three methods. Matlab code for the FS-LD algorithm is available at: http://web.eecs.umich.edu/~honglak/softwares/fast_sc.tgz. Matlab codes for the K-SVD and MOD algorithm are available at: http://www.cs.technion.ac.il/~elad/Various/AnalysisKSVDbox.rar.

Both BCDDL and FS-LD solve the alternating $\ell_1-\ell_s$ ($\ell_1$-regularized least-squares) and $\ell_2-\ell_s$ ($\ell_2$-constrained least-squares) optimization problems, while MOD and K-SVD solve the alternating $\ell_0-\ell_s$ ($\ell_0$-regularized least-squares) and $\ell_2-\ell_s$ ($\ell_2$-constrained least-squares) optimization problems. Since the objective functions are not the same for these four algorithms, it is not easy to determine a unified criterion for comparison.

As for $\ell_1-\ell_s$ and $\ell_2-\ell_s$ optimization problems, FS-LD is state-of-the-art in batch mode. There are enough experimental comparisons between FS-LD and the proposed BCDDL method in the previous sections, which demonstrate that ours is more efficient.

Figure 10. Example images from the Scene 15 dataset.

Table 4
Classification rate comparison on the Scene 15 dataset (%).

<table>
<thead>
<tr>
<th>Methods</th>
<th>Average classification rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KSPM [14]</td>
<td>81.4 ± 0.5</td>
</tr>
<tr>
<td>KC [31]</td>
<td>76.7 ± 0.4</td>
</tr>
<tr>
<td>HKBQV [33]</td>
<td>81.77 ± 0.49</td>
</tr>
<tr>
<td>OCSVM [33]</td>
<td>82.02 ± 0.54</td>
</tr>
<tr>
<td>ScSVM [34]</td>
<td>80.28 ± 0.93</td>
</tr>
<tr>
<td>LLC [32]</td>
<td>81.66 ± 0.36</td>
</tr>
<tr>
<td>MSSR [19]</td>
<td>85.18 ± 0.26</td>
</tr>
<tr>
<td>IFV [28]</td>
<td>87.54 ± 0.58</td>
</tr>
<tr>
<td>BCDDL</td>
<td>83.85 ± 0.42</td>
</tr>
</tbody>
</table>

Table 5
Classification rate comparison on the Caltech-101 dataset (%).

<table>
<thead>
<tr>
<th>Methods</th>
<th>15 training</th>
<th>30 training</th>
</tr>
</thead>
<tbody>
<tr>
<td>KSPM [14]</td>
<td>–</td>
<td>64.6 ± 0.8</td>
</tr>
<tr>
<td>KC [31]</td>
<td>–</td>
<td>64.1 ± 1.2</td>
</tr>
<tr>
<td>ScSVM [34]</td>
<td>67.0 ± 0.45</td>
<td>73.2 ± 0.54</td>
</tr>
<tr>
<td>LLC [32]</td>
<td>65.43</td>
<td>73.4</td>
</tr>
<tr>
<td>MSSR [19]</td>
<td>67.97 ± 0.53</td>
<td>76.04 ± 0.67</td>
</tr>
<tr>
<td>IFV [28]</td>
<td>–</td>
<td>80.73 ± 0.82</td>
</tr>
<tr>
<td>M-HMP [3]</td>
<td>–</td>
<td>82.5 ± 0.5</td>
</tr>
<tr>
<td>BCDDL</td>
<td>67.14 ± 0.47</td>
<td>74.61 ± 0.92</td>
</tr>
</tbody>
</table>

Table 6
Classification rate comparison on the Caltech-256 dataset (%).

<table>
<thead>
<tr>
<th>Methods</th>
<th>15 training</th>
<th>30 training</th>
<th>45 training</th>
<th>60 training</th>
</tr>
</thead>
<tbody>
<tr>
<td>KSPM [14]</td>
<td>NA</td>
<td>34.10</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>KC [31]</td>
<td>NA</td>
<td>27.17 ± 0.46</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>ScSVM [34]</td>
<td>27.73 ± 0.51</td>
<td>34.02 ± 0.35</td>
<td>37.46 ± 0.55</td>
<td>40.14 ± 0.91</td>
</tr>
<tr>
<td>LLC [32]</td>
<td>34.36</td>
<td>41.19</td>
<td>45.31</td>
<td>47.68</td>
</tr>
<tr>
<td>MSSR [19]</td>
<td>34.06 ± 0.36</td>
<td>41.14 ± 0.43</td>
<td>44.72 ± 0.42</td>
<td>47.26 ± 0.43</td>
</tr>
<tr>
<td>IFV [28]</td>
<td>38.5 ± 0.2</td>
<td>47.4 ± 0.1</td>
<td>52.1 ± 0.4</td>
<td>54.8 ± 0.4</td>
</tr>
<tr>
<td>M-HMP [3]</td>
<td>42.7</td>
<td>50.7</td>
<td>54.8</td>
<td>58</td>
</tr>
<tr>
<td>BCDDL</td>
<td>32.67 ± 0.28</td>
<td>38.42 ± 0.75</td>
<td>42.07 ± 0.69</td>
<td>44.42 ± 0.45</td>
</tr>
</tbody>
</table>

5.6. Further comparison of four dictionary learning approaches on image classification

The classical and popular dictionary learning (DL) techniques involve MOD, K-SVD, and Lagrange-Dual in the FS-LD algorithm. Here, we give a further comparison among the proposed BCDDL algorithm and the other three methods. Matlab code for the FS-LD algorithm is available at: http://web.eecs.umich.edu/~honglak/softwares/fast_sc.tgz. Matlab codes for the K-SVD and MOD algorithm are available at: http://www.cs.technion.ac.il/~elad/Various/AnalysisKSVDbox.rar.

Both BCDDL and FS-LD solve the alternating $\ell_1-\ell_s$ ($\ell_1$-regularized least-squares) and $\ell_2-\ell_s$ ($\ell_2$-constrained least-squares) optimization problems, while MOD and K-SVD solve the alternating $\ell_0-\ell_s$ ($\ell_0$-regularized least-squares) and $\ell_2-\ell_s$ ($\ell_2$-constrained least-squares) optimization problems. Since the objective functions are not the same for these four algorithms, it is not easy to determine a unified criterion for comparison.

As for $\ell_1-\ell_s$ and $\ell_2-\ell_s$ optimization problems, FS-LD is state-of-the-art in batch mode. There are enough experimental comparisons between FS-LD and the proposed BCDDL method in the previous sections, which demonstrate that ours is more efficient.

Please cite this article as: B.-D. Liu, et al., Blockwise coordinate descent schemes for efficient and effective dictionary learning, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.06.096
and effective. Besides, using Lagrange-Dual in FS–LD, zero atoms may exist when SSᵀ + A is singular in [15]. This is probable especially for much sparser codes and small number of samples. It means that FS–LD learns some meaningless bases, which shrinks the available basis pool and increases the reconstruction error.

As for ℓ₀–ℓ₂ and ℓ₂–ℓ₂ optimization problems, when S is fixed, MOD updates the entire B at once, while ours updates columns of B sequentially and reduces the cost function more. When the ℓ₂-norm normalization constraint is introduced for B, MOD is even not warranted to decrease monotonically. Similar to FS–LD, MOD will also lead to some zero atoms due to the matrix inversion. On the other hand, in the learning dictionary step, K-SVD updates columns of B and the corresponding rows of S simultaneously. However, K-SVD still needs extra OMP or other pursuit algorithm in the sparse coding step, or it will not work well.

To simultaneously compare these four algorithms, an evaluation criterion was given in Section 4.6. Given the ground truth of the underlying dictionary, comparison on a synthetic experiment tested the efficiency and ability of dictionary recovery for these four algorithms.

Here, we give a further comparison by using the results of image classification, i.e., accuracy and running time, as the unified evaluation metric. An experiment is carried out on the Scene 15 dataset for image classification. The features are the same as those used in image classification. 50,000 features are randomly sampled for learning dictionary. The dictionary size is 200, and the nonzero value for each code is about 6. Table 7 lists the comparisons of our methods with other three dictionary learning methods for image classification. Table 8 shows the running time for learning dictionary by the four methods. From Table 8, our approach completes the sparse representation task fastest.

6. Conclusion

In this paper, we proposed BCDDL for simultaneous sparse coding and dictionary learning. To the best of our knowledge, it is the fastest procedure for SC–DL to date. Two highlights distinguish our approach from previous works.

1. Researchers tend to more sophisticated first or second-order quadratic optimizers in solving SC–DL. However, the simplest coordinate descent does work in this case and deserves more attention. Our exhaustive experiments have demonstrated that BCDDL is surprisingly competitive in seeking better solutions much faster.

2. The efficiency of BCDDL not only lies in coordinate descent, but also owes to its proper partition of variables, making parallel computation feasible. This means BCDDL is blockwise rather than coordinate-wise.

In the future, the online sparse coding mechanism will be implemented to further improve the speed and efficiency of the algorithm. Moreover, sparse coding incorporating structure information and locality constraints will also be under consideration.

Acknowledgment

This paper is supported partly by the National Natural Science Foundation of China (Grant nos. 61402535, 61271407), the Natural Science Foundation for Youths of Shandong Province, China (Grant no. ZR2014FQ001), Qingdao Science and Technology Project (No. 14-2-4-111-jch), and the Fundamental Research Funds for the Central Universities, China University of Petroleum (East China) (Grant no. 14CX02169A).

Table 7
Classification rate comparison on the Scene 15 dataset (%).

<table>
<thead>
<tr>
<th>Methods</th>
<th>Average classification rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOD</td>
<td>75.78 ± 0.51</td>
</tr>
<tr>
<td>K-SVD</td>
<td>75.52 ± 0.49</td>
</tr>
<tr>
<td>FS–LD</td>
<td>77.12 ± 0.40</td>
</tr>
<tr>
<td>BCDDL</td>
<td>79.32 ± 0.36</td>
</tr>
</tbody>
</table>

Table 8
Running time of sparse representation on the Scene 15 dataset.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Running time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOD</td>
<td>2231</td>
</tr>
<tr>
<td>K-SVD</td>
<td>2983</td>
</tr>
<tr>
<td>FS–LD</td>
<td>3916</td>
</tr>
<tr>
<td>BCDDL</td>
<td>182</td>
</tr>
</tbody>
</table>

References


Bao-Di Liu was born in Shandong, China. He received the Ph.D. degree in Electronic Engineering from Tsinghua University, China. Currently, he is an assistant professor in College of Information and Control Engineering, China University of Petroleum, China. His research interests include computer vision and machine learning.

Yu-Xiong Wang is a Ph.D. student in the Robotics Institute, School of Computer Science, at Carnegie Mellon University. His research interests include computer vision, image processing, and machine learning.

Bin Shen is now working at Google Research New York. He received Ph.D. degree in Department of Computer Science, Purdue University, West Lafayette, Indiana 47907, US. Before joining Purdue, he got B.S. and M.S. degrees from EE, Tsinghua University, Beijing, in 2007 and 2009, respectively. His research interests include image processing, machine learning and data mining.

Xue Li received the BS degree in Electronic Engineering from Beijing Institute of Technology (BIT), Beijing, China, in 2011. Currently, she is a Ph.D candidate in the Department of Electronic Engineering at Tsinghua University, Beijing, China. Her research interests include image classification, automatic image annotation and machine learning.

Yu-Jin Zhang received the Ph.D. degree in Applied Science from the State University of Liège, Liège, Belgium, in 1989. From 1989 to 1993, he was a post-doc fellow and a research fellow with the Department of Applied Physics and Department of Electrical Engineering at the Delft University of Technology, Delft, the Netherlands. In 1993, he joined the Department of Electronic Engineering at Tsinghua University, Beijing, China, where he is a professor of Image Engineering (since 1997). Since 2014, he is a tenured-professor and the director of Institute of Media Cognition and Intelligent System. He is an active researcher in Image Processing, with current interests on object segmentation from images and video, segmentation evaluation and comparison, moving object detection and tracking, face recognition, facial expression detection/classification, content-based image and video retrieval, information fusion for high-level image understanding, etc. He has authored more than 20 books and published around 400 papers in the areas of image processing, image analysis, and image understanding. Professor Zhang is vice president of China Society of Image and Graphics and director of academic committee of the Society, and a Fellow of SPIE.

Yan-Jiang Wang received the M.S. degree from Beijing University of Aeronautics and Astronautics, Beijing, China, in 1989 and the Ph.D. degree from Beijing Jiaotong University, Beijing, China, in 2001. Now he is a professor of the College of Information and Control Engineering, China University of Petroleum, Qingdao, China. He is also the head of the Institute of Signal and Information Processing, China University of Petroleum. His research interests include pattern recognition, computer vision, and cognitive computation.