Dictionary Optimization for Block-Sparse Representations

Lihi Zelnik-Manor, Kevin Rosenblum, Yonina C. Eldar

***Abstract*—Recent work has demonstrated that using a carefully designed dictionary instead of a predefined one, can improve the sparsity in jointly representing a class of signals. This has motivated the derivation of learning methods for designing a dictionary which leads to the sparsest representation for a given set of signals. In some applications, the signals of interest can have further structure, so that they can be well approximated by a union of a small number of subspaces (e.g., face recognition and motion segmentation). This implies the existence of a dictionary which enables *block-sparse* representations of the input signals once its atoms are properly sorted into blocks. In this paper, we propose an algorithm for learning a *block-sparsifying dictionary* of a given set of signals. We do not require prior knowledge on the association of signals into groups (subspaces). Instead, we develop a method that automatically detects the underlying block structure given the maximal size of those groups. This is achieved by iteratively alternating between updating the block structure of the dictionary and updating the dictionary atoms to better fit the data. Our experiments show that for block-sparse data the proposed algorithm significantly improves the dictionary recovery ability and lowers the representation error compared to dictionary learning methods that do not employ block structure.**

I. Introduction

The framework of sparse coding aims at recovering an unknown vector $\theta \in R^K$ from an under-determined system of linear equations $x = D\theta$, where $D \in R^{N\times K}$ is a given dictionary, and $x \in R^N$ is an observation vector with $N < K$. Since the system is under-determined, θ can not be recovered without additional information. The framework of compressed sensing [**?**], [**?**] exploits sparsity of θ in order to enable recovery. Specifically, when θ is known to be sparse so that it contains few nonzero coefficients, and when D is chosen properly, then θ can be recovered uniquely from $x = D\theta$. Recovery is possible irrespectively of the locations of the nonzero entries of θ. This result has given rise to a multitude of different recovery algorithms. Most prominent among them are Basis Pursuit (BP) [**?**], [**?**], [**?**] and Orthogonal Matching Pursuit (OMP) [**?**], [**?**].

Recent work [**?**], [**?**], [**?**], [**?**], [**?**], [**?**] has demonstrated that adapting the dictionary D to fit a given set of signal examples leads to improved signal reconstruction. These learning algorithms attempt to find a dictionary that leads to optimal sparse representations for a certain class of signals. These methods show impressive results for representations with arbitrary sparsity structures. In some applications, however, the representations have further structure that can be exploited. Our interest is in the case of signals that are known to be drawn from a union of a small number of subspaces [**?**], [**?**], [**?**]. This occurs naturally, for example, in face recognition [**?**], [**?**], motion segmentation [**?**], multiband signals [**?**], [**?**], [**?**], measurements of gene expression levels [**?**], and more. For such signals, sorting the dictionary atoms according to the underlying subspaces leads to sparse representations which exhibit a block-sparse structure, i.e., the nonzero coefficients occur in clusters of varying sizes. Several methods, such as *Block BP* (BBP) [**?**], [**?**], [**?**] and *Block OMP* (BOMP) [**?**], [**?**] and group Lasso [**?**], [**?**] have been proposed to take advantage of this structure in recovering the block-sparse representation θ. These methods typically assume that the dictionary is predetermined and the block structure is known.

In this paper we propose a method for designing a *block-sparsifying dictionary* for a given set of signals. In other words, we wish to find a dictionary that provides block-sparse representations best suited to the signals in a given set. To take advantage of the block structure via block-sparse approximation methods, it is necessary to know the block structure of the dictionary. We do not assume that it is known a-priori. Instead, we only assume all blocks have a known maximal size and infer the block structure from the data accordingly while adapting the dictionary. If we were not constraining the maximal block size, we would eventually end up with one block which contains all the dictionary atoms.

We start by formulating this task as an optimization problem. We then present an algorithm for minimizing the proposed objective, which iteratively alternates between updating the block structure and updating the dictionary. The block structure is inferred by the agglomerative clustering of dictionary atoms that induce similar sparsity patterns. In other words, after finding the sparse representations of the training signals, the atoms are progressively merged according to the similarity of the sets of signals they represent. A variety of segmentation methods through subspace modeling have been proposed recently [**?**], [**?**], [**?**]. These techniques learn an underlying collection of subspaces based on the assumption that each of the samples lies close to one of them. However, unlike our method, they do not treat the more general case where the signals are drawn from a union of several subspaces.

The dictionary blocks are then sequentially updated to minimize the representation error at each step. The proposed algorithm is an intuitive extension of the K-SVD method [**?**], which yields sparsifying dictionaries by sequentially updating the dictionary atoms, to the case of block structures: When the blocks are of size 1 our cost function and algorithm reduce

The authors are with the Technion - Israel Institute of Technology, Haifa, Israel. Email: kevin@tx.technion.ac.il, lihi@ee.technion.ac.il, yonina@ee.technion.ac.il.

to K-SVD. Our experiments show that updating the dictionary block by block is preferred over updating the atoms in the dictionary one by one, as in K-SVD.

We demonstrate empirically that both parts of the algorithm are indispensable to obtain high performance. While fixing a random block structure and applying only the dictionary update part leads to improved signal reconstruction compared to K-SVD, combining the two parts leads to even better results. Furthermore, our experiments show that K-SVD often fails to recover the underlying block structure. This is in contrast to our algorithm which succeeds in detecting most of the blocks.

We begin by reviewing previous work on dictionary design in Section **??**. In Section **??** we present an objective for designing block-sparsifying dictionaries. We show that this objective is a direct extension of the one used by K-SVD. We then propose an algorithm for minimizing the proposed cost function in Section **??**. Section **??** provides a detailed description of the proposed algorithm. We evaluate the performance of our method and compare it to previous work in Section **??**.

Throughout the paper, we denote vectors by lowercase letters, e.g., x, and matrices by uppercase letters, e.g., A. The jth column of a matrix A is written as A_j, and the ith row as A^i. The sub-matrix containing the entries of A in the rows with indices r and the columns with indices c is denoted A_c^r. The Frobenius norm is defined by $\|A\|_F \equiv \sqrt{\sum_j \|A_j\|_2^2}$. The ith element of a vector x is denoted $x[i]$. $\|x\|_p$ is its l_p-norm and $\|x\|_0$ counts the number of non-zero entries in x.

II. Prior work on dictionary design

The goal in dictionary learning is to find a dictionary D and a representation matrix Θ that best match a given set of vectors X_i that are the columns of X. In addition, we would like each vector Θ_i of Θ to be sparse. In this section we briefly review two popular sparsifying dictionary design algorithms, K-SVD [**?**] and MOD (Method of Optimal Directions) [**?**]. We will generalize these methods to block-sparsifying dictionary design in Section **??**.

To learn a dictionary, both MOD and K-SVD attempt to optimize the same cost function for a given sparsity measure k:

$$\begin{aligned} \min_{D,\Theta} \quad & \|X - D\Theta\|_F \\ \text{s.t.} \quad & \|\Theta_i\|_0 \le k,\ i = 1, \ldots, L \end{aligned} \tag{1}$$

where $X \in R^{N\times L}$ is a matrix containing L given input signals, $D \in R^{N\times K}$ is the dictionary and $\Theta \in R^{K\times L}$ is a sparse representation of the signals. Note that the solution of (**??**) is never unique due to the invariance of D to permutation and scaling of columns. This is partially resolved by requiring normalized columns in D. We will therefore assume throughout the paper that the columns of D are normalized to have l_2-norm equal 1.

Problem (**??**) is non-convex and NP-hard in general. Both MOD and K-SVD attempt to approximate (**??**) using a relaxation technique which iteratively fixes all the parameters but one, and optimizes the objective over the remaining variable. In this approach the objective decreases (or is left unchanged) at each step, so that convergence to a local minimum is guaranteed. Since this might not be the global optimum both approaches are strongly dependent on the initial dictionary $D^{(0)}$. The convention [**?**] is to initialize $D^{(0)}$ as a collection of K data signals from the same class as the training signals X.

The first step of the nth iteration in both algorithms optimizes Θ given a fixed dictionary $D^{(n-1)}$, so that (**??**) becomes:

$$\begin{aligned} \Theta^{(n)} = \arg\min_{\Theta} \quad & \|X - D^{(n-1)}\Theta\|_F \\ \text{s.t.} \quad & \|\Theta_i\|_0 \le k,\ i = 1, \ldots, L. \end{aligned} \tag{2}$$

This problem can be solved approximately using sparse coding methods such as BP or OMP for each column of Θ, since the problem is separable in these columns. Next, $\Theta^{(n)}$ is kept fixed and the representation error is minimized over D:

$$D^{(n)} = \arg\min_{D} \|X - D\Theta^{(n)}\|_F. \tag{3}$$

The difference between MOD and K-SVD lies in the choice of optimization method for $D^{(n)}$. While K-SVD converges faster than MOD, both methods yield similar results (i.e., similar reconstruction errors).

The MOD algorithm treats the problem in (**??**) directly. This problem has a closed form solution given by the pseudo-inverse:

$$D^{(n)} = X\Theta'^{(n)}(\Theta^{(n)}\Theta'^{(n)})^{-1}. \tag{4}$$

Here we assume for simplicity that $\Theta^{(n)}\Theta'^{(n)}$ is invertible. The K-SVD method solves (**??**) differently. The columns in $D^{(n-1)}$ are updated sequentially, along with the corresponding non-zero coefficients in $\Theta^{(n)}$. This parallel update leads to a significant speedup while preserving the sparsity pattern of $\Theta^{(n)}$. For $j = 1, \ldots, K$, the update is as follows. Let $\omega_j \equiv \{i \in 1, \ldots, L | \Theta_i^j \neq 0\}$ be the set of indices corresponding to columns in $\Theta^{(n)}$ that use the atom D_j, i.e., their ith row is non-zero. Denote by $R_{\omega_j} = X_{\omega_j} - \sum_{i\neq j}(D_i\Theta^i_{\omega_j})$ the representation error of the signals X_{ω_j} excluding the contribution of the jth atom. The representation error of the signals with indices ω_j can then be written as $\|R_{\omega_j} - D_j\Theta^j_{\omega_j}\|_F$. The goal of the update step is to minimize this representation error, which is accomplished by choosing

$$D_j = U_1, \qquad \Theta^j_{\omega_j} = \Delta_1^1 V_1'.$$

Here $U\Delta V'$ is the Singular Value Decomposition (SVD) of R_{ω_j}. Note, that the columns of D remain normalized after the update. The K-SVD algorithm obtains the dictionary update by K separate SVD computations, which explains its name.

III. Block-Sparsifying Dictionary optimization

We now formulate the problem of block-sparsifying dictionary design. We then propose an algorithm which can be seen as a natural extension of K-SVD for the case of signals with block sparse representations. Our method involves an additional clustering step in order to determine the block structure.

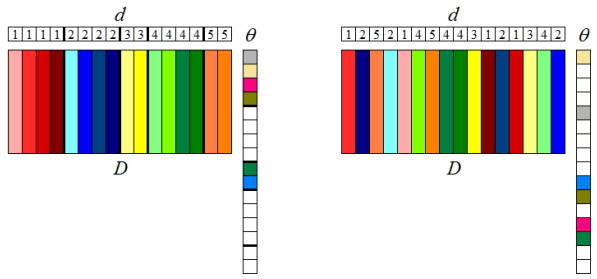


Fig. 1. Two equivalent examples of dictionaries D and block structures d with 5 blocks, together with 2-block-sparse representations θ. Both examples represent the same signal, since the atoms in D and the entries of d and θ are permuted in the same manner (in both cases $\|\theta\|_{0,d} = 2$).

A. Problem Definition

For a given set of L signals $X = \{X_i\}_{i=1}^{L} \in R^N$, we wish to find a dictionary $D \in R^{N\times K}$ whose atoms are sorted in blocks, and which provides the most accurate representation vectors whose non-zero values are concentrated in a fixed number of blocks. In previous works dealing with the block-sparse model, it is typically assumed that the block structure in D is known a-priori, and even more specifically, that the atoms in D are sorted according to blocks [**?**], [**?**]. Instead, in this paper we address the more general case where the association of dictionary atoms into blocks is not known a-priori. We do, however, make the assumption that the maximal block size, denoted by s, is known.

More specifically, suppose we have a dictionary whose atoms are sorted in blocks that enable *block-sparse* representations of the input signals. Assume that each block is given an index number. Let $d \in R^K$ be the vector of block assignments for the atoms of D, i.e., $d[i]$ is the block index of the atom D_i. We say that a vector $\theta \in R^K$ is k-block-sparse over d if its non-zero values are concentrated in k blocks only. This is denoted by $\|\theta\|_{0,d} = k$, where $\|\theta\|_{0,d}$ is the l_0-norm over d and counts the number of non-zero blocks as defined by d. Fig. **??** presents examples of two different block structures and two corresponding block-sparse vectors and dictionaries.

Our goal is to find a dictionary D and a block structure d, with maximal block size s, that lead to optimal k-block sparse representations $\Theta = \{\Theta_i\}_{i=1}^{L}$ for the signals in X:

$$\begin{aligned} \min_{D,d,\Theta} \quad & \|X - D\Theta\|_F \\ \text{s.t.} \quad & \|\Theta_i\|_{0,d} \le k,\ i = 1,\dots,L \\ & |d_j| \le s,\ j \in [1,B] \end{aligned} \tag{5}$$

where $d_j = \{i \in 1,\dots,K | d[i] = j\}$ is the set of indices belonging to block j (i.e., the list of atoms in block j), and B is the number of blocks.

The case when there is no underlying block structure or when the block structure is ignored, is equivalent to setting $s = 1$ and $d = [1,\dots,K]$. Substituting this into (**??**), reduces it to (**??**). In this setting, the objective and the algorithm we propose coincide with K-SVD. In Section **??** we demonstrate through simulations that when an underlying block structure exists, optimizing (**??**) via the proposed framework improves recovery results and lowers the representation errors with respect to (**??**).

B. Algorithm Preview

In this section, we propose a framework for solving (**??**). Since this optimization problem is non-convex, we adopt the coordinate relaxation technique. We initialize the dictionary $D^{(0)}$ as the outcome of the K-SVD algorithm (using a random collection of K signals leads to similar results, but slightly slower convergence). Then, at each iteration n we perform the following two steps:

1) Recover the block structure by solving (**??**) for d and Θ while keeping $D^{(n-1)}$ fixed:

$$\begin{aligned} [d^{(n)},\Theta^{(n)}] = \underset{d,\Theta}{\text{argmin}} \quad & \|X - D^{(n-1)}\Theta\|_F \\ \text{s.t.} \quad & \|\Theta_i\|_{0,d} \le k,\ i = 1,\dots,L \\ & |d_j| \le s,\ j \in [1,B]. \end{aligned} \tag{6}$$

   An exact solution would require a combinatorial search over all feasible d and Θ. Instead, we propose a tractable approximation to (**??**) in Section **??**, referred to as *Sparse Agglomerative Clustering (SAC)*. Agglomerative clustering builds blocks by progressively merging the closest atoms according to some distance measure [**?**], [**?**]. SAC uses the l_0-norm for this purpose.
2) Fit the dictionary $D^{(n)}$ to the data by solving (**??**) for D and Θ while keeping $d^{(n)}$ fixed:

$$\begin{aligned} [D^{(n)},\Theta^{(n)}] = \underset{D,\Theta}{\text{argmin}} \quad & \|X - D\Theta\|_F \\ \text{s.t.} \quad & \|\Theta_i\|_{0,d^{(n)}} \le k,\ i = 1,\dots,L. \end{aligned} \tag{7}$$

   In Section **??** we propose an algorithm, referred to as *Block K-SVD (BK-SVD)*, for solving (**??**). This technique can be viewed as a generalization of K-SVD since the blocks in $D^{(n)}$ are sequentially updated together with the corresponding non-zero blocks in $\Theta^{(n)}$.

In the following sections we describe in detail the steps of this algorithm. The overall framework is summarized in Algorithm 1.

C. Block Structure Recovery: Sparse Agglomerative Clustering

In this section we propose a method for recovering the block structure d given a fixed dictionary D, as outlined in Fig. **??**.(a). The suggested method is based on the coordinate relaxation technique to solve (**??**) efficiently. We start by initializing d and Θ. Since we have no prior knowledge on d it is initialized as K blocks of size 1, i.e. $d = [1,\dots,K]$. To initialize Θ we keep d fixed and solve (**??**) over Θ using OMP with $k \times s$ instead of k non-zero entries, since the signals are known to be combinations of k blocks of size s. Based on the obtained Θ, we first update d as described below and then again Θ using BOMP [**?**]. The BOMP algorithm sequentially selects the dictionary blocks that best match the input signals X_i, and can be seen as a generalization of the OMP algorithm to the case of blocks.

To update d we wish to solve (**??**) while keeping Θ fixed. Although the objective does not depend on d, the constraints

Algorithm 1 Block-Sparse Dictionary Design

Input: A set of signals X, block sparsity k and maximal block size s.

Task: Find a dictionary D, block structure d and the corresponding sparse representation Θ by optimizing:

$$\begin{aligned} \min_{D,d,\Theta} \quad & \|X - D\Theta\|_F \\ \text{s.t.} \quad & \|\Theta_i\|_{0,d} \leq k,\ i = 1, \ldots, L \\ & |d_j| \leq s,\ j \in d. \end{aligned}$$

Initialization: Set the initial dictionary $D^{(0)}$ as the outcome of K-SVD.

Repeat from $n = 1$ *until convergence*:

1) Fix $D^{(n-1)}$, and update $d^{(n)}$ and $\Theta^{(n)}$ by applying Sparse Agglomerative Clustering.
2) Fix $d^{(n)}$, and update $D^{(n)}$ and $\Theta^{(n)}$ by applying BK-SVD.
3) $n = n + 1$.

do. Therefore, the problem becomes finding a block structure with maximal block size s that meets the constraint on the block-sparsity of Θ. To this end, we seek to minimize the block-sparsity of Θ over d:

$$\min_d \sum_{i=1}^{L} \|\Theta_i\|_{0,d} \quad \text{s.t.} \quad |d_j| \leq s,\ j \in [1, B]. \tag{8}$$

Before we describe how (**??**) is optimized we first wish to provide some insight. When a signal X_i is well represented by the unknown block d_j, then the corresponding rows in Θ_i are likely to be non-zero. Therefore, *rows* of Θ that exhibit a similar pattern of non-zeros are likely to correspond to *columns* of the same dictionary block. Consequently, grouping dictionary columns into blocks is equivalent to grouping rows of Θ according to their sparsity pattern. To detect rows with similar sparsity patterns we next rewrite the objective of (**??**) as a function of the pattern on non-zeros.

Let $\omega_j(\Theta, d)$ denote the list of columns in Θ that have non-zero values in rows corresponding to block d_j, i.e., $\omega_j(\Theta, d) = \{i \in 1, \ldots, L |\ \|\Theta_i^{d_j}\|_2 > 0\}$. Problem (**??**) can now be rewritten as:

$$\min_d \sum_{j \in [1,B]} |\omega_j(\Theta, d)| \quad \text{s.t.} \quad |d_j| \leq s,\ j \in [1, B] \tag{9}$$

where $|\omega_j|$ denotes the size of the list ω_j, and B is the current number of blocks. We propose using a sub-optimal tractable agglomerative clustering algorithm [**?**] to minimize this objective. At each step we merge the pair of blocks that have the most similar pattern of non-zeros in Θ, leading to the steepest descent in the objective. In our implementation we allow merging blocks as long as the maximum block size s is not exceeded. While this aims at obtaining blocks of size s, some of the blocks could be smaller since, for example, two blocks of size $s - 1$ will not be further merged as their joint dimension exceeds s.

More specifically, at each step we find the pair of blocks (j_1^*, j_2^*) such that:

$$[j_1^*, j_2^*] = \arg\max_{j_1 \neq j_2} |\omega_{j_1} \cap \omega_{j_2}| \quad \text{s.t.} \quad |d_{j_1}| + |d_{j_2}| \leq s.$$

We then merge j_1^* and j_2^* by setting $\forall i \in d_{j_2} : d[i] \leftarrow j_1$, $\omega_{j_1} \leftarrow \{\omega_{j_1} \cup \omega_{j_2}\}$, and $\omega_{j_2} \leftarrow \emptyset$. This is repeated until no blocks can be merged without breaking the constraint on the block size. We do not limit the intersection size for merging blocks from below, since merging is always beneficial. Merging blocks that have nothing in common may not reduce the objective of (**??**); however, this can still lower the representation error at the next BK-SVD iteration. Indeed, while the number of blocks k stays fixed, the number of atoms that can be used to reduce the error increases.

Fig. **??**.(b) presents an example that illustrates the notation and the steps of the algorithm. In this example the maximal block size is $s = 2$. At initialization the block structure is set to $d = [1, 2, 3, 4]$, which implies that the objective of (**??**) is $\sum_{i=1}^{L} \|\Theta_i\|_{0,d} = 2 + 1 + 2 + 2 = 7$. At the first iteration, ω_1 and ω_3 have the largest intersection. Consequently, blocks 1 and 3 are merged. At the second iteration, ω_2 and ω_4 have the largest intersection, so that blocks 2 and 4 are merged. This results in the block structure $d = [1, 2, 1, 2]$ where no blocks can be merged without surpassing the maximal block size. The objective of (**??**) is reduced to $\sum_{i=1}^{L} \|\Theta_i\|_{0,d} = 4$, since all 4 columns in Θ are 1-block-sparse. Note that since every column contains non-zero values, this is the global minimum and therefore the algorithm succeeded in solving (**??**).

While more time-efficient clustering methods exist, we have selected agglomerative clustering because it provides a simple and intuitive solution to our problem. Partitional clustering methods, such as K-Means, require initialization and are therefore not suited for highly sparse data and the l_0-norm metric. Moreover, it is preferable to limit the maximal block size rather than to determine the minimal number of blocks. This is since the latter could lead to degeneracies such as obtaining a single high-dimensional block and multiple one-dimensional blocks. It is important to note that due to the iterative nature of our dictionary design algorithm, clustering errors can be corrected in the following iteration, after the dictionary has been refined.

The computational complexity of the SAC algorithm is $L \cdot K^2$ since we need to calculate the distances between all pairs of atoms and each distance estimation is of complexity L. Prior to clustering we employ OMP whose complexity is $L((ks)^2 K + 2NK)$ (see [**?**]).

D. Block K-SVD Algorithm

We now propose the BK-SVD algorithm for recovering the dictionary D and the representations Θ by optimizing (**??**) given a block structure d and input signals X.

Using the coordinate relaxation technique, we solve this problem by minimizing the objective based on alternating Θ and D. At each iteration m, we first fix $D^{(m-1)}$ and use BOMP to solve (**??**) which reduces to

$$\begin{aligned} \Theta^{(m)} = \arg\min_{\Theta} \quad & \|X - D^{(m-1)}\Theta\|_F \\ \text{s.t.} \quad & \|\Theta_i\|_{0,d} \leq k,\ i = 1, \ldots, L. \end{aligned} \tag{10}$$

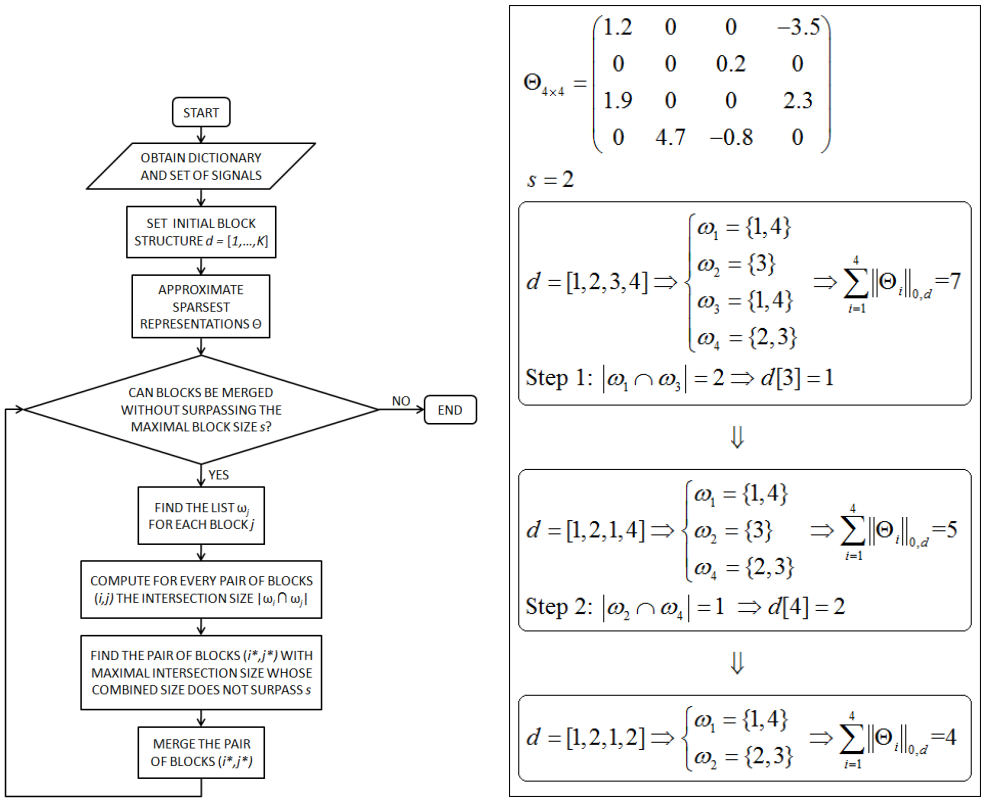


Fig. 2. (a) A flow chart describing the SAC algorithm. (b) A detailed example of the decision making process in the SAC algorithm.

Next, to obtain $D^{(m)}$ we fix $\Theta^{(m)}, d$ and X, and solve:

$$D^{(m)} = \arg\min_{D} \|X - D\Theta^{(m)}\|_F. \quad (11)$$

Inspired by the K-SVD algorithm, the blocks in $D^{(m-1)}$ are updated sequentially, along with the corresponding non-zero coefficients in $\Theta^{(m)}$. For every block $j \in [1, B]$, the update is as follows. Denote by $R_{\omega_j} = X_{\omega_j} - \sum_{i \neq j} D_{d_i} \Theta^{d_i}_{\omega_j}$ the representation error of the signals X_{ω_j} excluding the contribution of the jth block. Here ω_j and d_j are defined as in the previous subsection. The representation error of the signals with indices ω_j can then be written as $\|R_{\omega_j} - D_{d_j}\Theta^{d_j}_{\omega_j}\|_F$. Finally, the representation error is minimized by setting $D_{d_j}\Theta^{d_j}_{\omega_j}$ equal to the matrix of rank $|d_j|$ that best approximates R_{ω_j}. This can obtained by the following updates:

$$D_{d_j} = [U_1, \ldots, U_{|d_j|}]$$

$$\Theta^{d_j}_{\omega_j} = [\Delta^1_1 V_1, \ldots, \Delta^{|d_j|}_{|d_j|} V_{|d_j|}]'$$

where the $|d_j|$ highest rank components of R_{ω_j} are computed using the SVD $R_{\omega_j} = U\Delta V'$. The updated D_{d_j} is now an orthonormal basis that optimally represents the signals with indices ω_j. Note that the representation error is also minimized when multiplying D_{d_j} on the right by W and $\Theta^{d_j}_{\omega_j}$ on the left by W^{-1}, where $W \in R^{|d_j| \times |d_j|}$ is an invertible matrix. However, if we require the dictionary blocks to be orthonormal subspaces, then the solution is unique up to a permutation of the atoms. It is also important to note that if $|d_j| > |\omega_j|$, then $|d_j| - |\omega_j|$ superfluous atoms in block j can be discarded without any loss of performance.

This dictionary update minimizes the representation error while preserving the sparsity pattern of $\Theta^{(m)}$, as in the K-SVD dictionary update step. However, the update step in the BK-SVD algorithm converges faster thanks to the simultaneous optimization of the atoms belonging to the same block. Our simulations show that it leads to smaller representation errors as well. Moreover, the dictionary update step in BK-SVD requires about s times less SVD computations, which makes the proposed algorithm significantly faster than K-SVD.

As was shown in [**?**] the complexity of K-SVD is dominated by that of OMP and is hence $L((ks)^2K+2NK)$ (assuming the target sparsity is ks). Similarly, for BK-SVD the complexity is $L((k)^2K + 2NK)$.

We next present a simple example illustrating the advantage of the BK-SVD dictionary update step, compared to the K-SVD update. Let D_1 and D_2 be the atoms of the same block, of size 2. A possible scenario is that $D_2 = U_1$ and $\Theta^2_{\omega_j} = -\Delta(1,1)V_1'$. In K-SVD, the first update of D is $D_1 \leftarrow U_1$ and $\Theta^1_{\omega_j} \leftarrow \Delta(1,1)V_1'$. In this case the second update would leave D_2 and $\Theta^2_{\omega_j}$ unchanged. As a consequence, only the highest rank component of R_{ω_j} is removed. Conversely, in the proposed BK-SVD algorithm, the atoms D_1 and D_2 are updated simultaneously, resulting in the two highest rank components of R_{ω_j} being removed.

IV. Experiments

In this section, we evaluate the contribution of the proposed block-sparsifying dictionary design framework empirically.

We also examine the performance of the SAC and the BK-SVD algorithms separately. We perform experiments on synthetic data as well as real-world image data.

For each simulation, we repeat the following procedure 50 times: We randomly generate a dictionary D^* of dimension 30×60 with normally distributed entries and normalize its columns. The block structure is chosen to be of the form:

$$d^* = [1, 1, 1, 2, 2, 2, \ldots, 20, 20, 20]$$

i.e. D^* consists of 20 subspaces of size $s = 3$. We generate $L = 5000$ test signals X of dimension $N = 30$, that have 2-block sparse representations Θ^* with respect to D^* (i.e. $k = 2$). The generating blocks are chosen randomly and independently and the coefficients are i.i.d. uniformly distributed. White Gaussian noise with varying SNR was added to X.

We perform three experiments:

1) Given D^* and X, we examine the ability of SAC to recover d^*.
2) Given d^* and X, we examine the ability of BK-SVD to recover D^*.
3) We examine the ability of BK-SVD combined with SAC to recover D^* and d^* given only X.

We use two measures to evaluate the success of the simulations based on their outputs D, d and Θ:

- The normalized representation error $e = \frac{\|X - D\Theta\|_F}{\|X\|_F}$.
- The percentage p of successfully recovered blocks. For every block in D, we match the closest block in D^* without repetition, where the (normalized) distance between two blocks S_1 and S_2 (of sizes s_1 and s_2) is measured by [**?**]:

$$\text{Dist}(S_1, S_2) \equiv \sqrt{\left(1 - \frac{\|S_1' S_2\|_F^2}{\max(s_1, s_2)}\right)}$$

assuming that both blocks are orthonormalized. If the distance between the block in D and its matched block in D^* is smaller than 0.01, we consider the recovery of this block as successful.

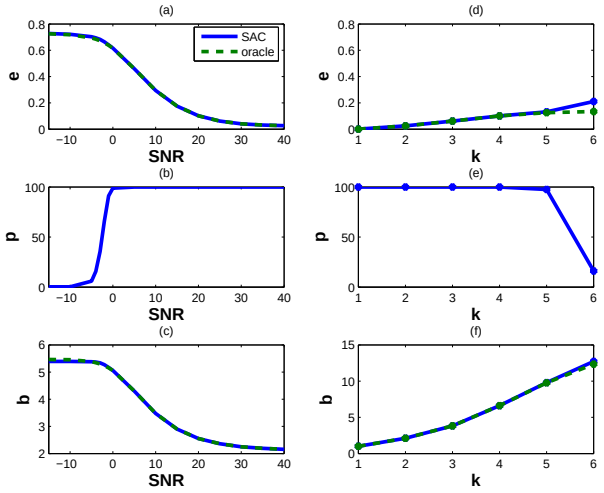


Fig. 3. Simulation results of the SAC algorithm. The graphs show e, p and b as a function of the SNR of the data signals for $k = 2$ (a, b, c), and as a function of k in a noiseless setting (d, e, f).

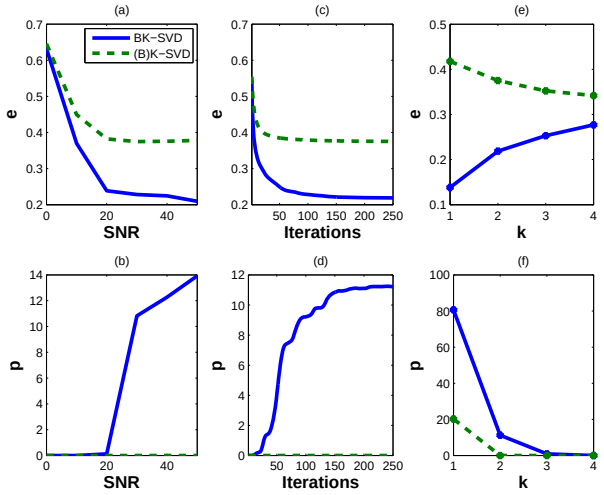


Fig. 4. Simulation results of the BK-SVD and (B)K-SVD algorithms. The graphs show the reconstruction error e and the recovery percentage p as a function of the SNR of the data signals for $k = 2$ and after 250 iterations (a, b), as a function of the number of iterations for $k = 2$ in a noiseless setting (c, d), and as a function of k in a noiseless setting after 250 iterations (e, f).

A. Evaluating SAC

To evaluate the performance of the SAC algorithm, we assume that D^* is known, and use SAC to reconstruct d^* and then BOMP to approximate Θ^*. The SAC algorithm is evaluated as a function of the SNR of the signals X for $k = 2$, and as a function of k in a noiseless setting. In addition to e and p, Fig. **??** also shows the objective of (**??**), which we denote by b. We compare our results with those of an "oracle" algorithm, which is given as input the true block structure d^*. It then uses BOMP to find Θ. The oracle's results provide a lower bound on the reconstruction error of our algorithm (we cannot expect our algorithm to outperform the oracle). It can be seen that for SNR higher than -5[dB], the percentage p of successfully recovered blocks quickly increases to 100% (Fig. **??**.(b)), the representation error e drops to zero (Fig. **??**.(a)) and the block-sparsity b drops to the lowest possible value $k = 2$ (Fig. **??**.(c)). Fig. **??**.(e) shows that the block structure d^* is perfectly recovered for $k < 6$. However, for $k = 6$, SAC fails in reconstructing the block structure d^*, even though the block sparsity b reaches the lowest possible value (Fig. **??**.(f)). This is a consequence of the inability of OMP to recover the sparsest approximation of the signals X with $k \times s = 12$ nonzero entries. In terms of e and b, our algorithm performs nearly as good as the oracle.

B. Evaluating BK-SVD

To evaluate the performance of the BK-SVD algorithm we assume that the block structure d^* is known. We initialize the dictionary $D^{(0)}$ by generating 20 blocks of size 3 where each block is a randomly generated linear combination of 2 randomly selected blocks of D^*. We then evaluate the contribution of the proposed BK-SVD algorithm. Recall that dictionary design consists of iterations between two steps,

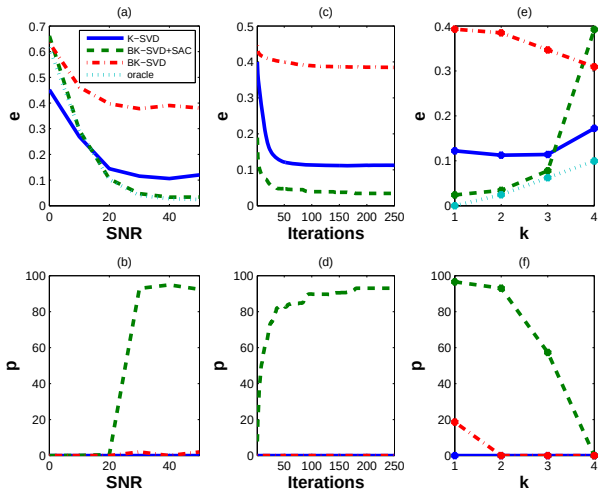


Fig. 5. Simulation results of our overall algorithm (BK-SVD+SAC), the BK-SVD algorithm and the K-SVD algorithm. The graphs show the reconstruction error e and the recovery percentage p as a function of the SNR of the data signals for $k = 2$ after 250 iterations (a, b), as a function of the number of iterations for $k = 2$ in a noiseless setting (c, d), and as a function of k in a noiseless setting after 250 iterations (e, f).

updating Θ using block-sparse approximation and updating the blocks in D and their corresponding non-zero representation coefficients. To evaluate the contribution of the latter step, we compare its performance with that of applying the same scheme, but using the K-SVD dictionary update step. We refer to this algorithm as (B)K-SVD. The algorithms are evaluated as a function of the SNR of the signals X for $k = 2$ after 250 iterations, as a function of the number of iterations for $k = 2$ in a noiseless setting, and as a function of k in a noiseless setting after 250 iterations. It is clear from Fig. **??** that the simultaneous update of the atoms in the blocks of D is imperative and does not only serve as a speedup of the algorithm.

C. Evaluating the Overall Framework

To evaluate the performance of the overall block-sparsifying dictionary design method, we combine SAC and BK-SVD. At each iteration we only run BK-SVD once instead of waiting for it to converge, improving the ability of the SAC algorithm to avoid traps. Our results are compared with those of K-SVD (with a fixed number of 8 coefficients) and with those of BK-SVD (with a fixed block structure) as a function of the SNR, as a function of the number of iterations. The algorithms are evaluated as a function of the SNR of the signals X for $k = 2$ after 250 iterations, as a function of the number of iterations for $k = 2$ in a noiseless setting, and as a function of k in a noiseless setting after 250 iterations (Fig. **??**).

Our experiments show that for SNR > 10[dB], the proposed block-sparsifying dictionary design algorithm yields lower reconstruction errors (see Fig. **??**.(a)) and a higher percentage of correctly reconstructed blocks (see Fig. **??**.(b)), compared to K-SVD. Moreover, even in a noiseless setting, the K-SVD algorithm fails to recover the sparsifying dictionary, while our algorithm succeeds in recovering 93% of the dictionary blocks, as shown in Fig. **??**.(d).

For SNR ≤ 10[dB] we observe that K-SVD reaches lower reconstruction error compared to our block-sparsifying dictionary design algorithm. This is since when the SNR is low the block structure is no longer present in the data and the use of block-sparse approximation algorithms is unjustified. To verify this is indeed the cause for failure of our algorithm, we further compare our results with those of an oracle algorithm, which is given as input the true dictionary D^* and block structure d^*. It then uses BOMP to find Θ. Fig. **??** shows that for all noise levels, our algorithm performs nearly as good as the oracle. Furthermore, for SNR ≤ 10[dB] we observe that K-SVD outperforms the oracle, implying that the use of block-sparsifying dictionaries is unjustified. For $k \leq 3$, in a noiseless setting, the performance of our algorithm lies close to that of the oracle, and outperforms the K-SVD algorithm. However, we note that this is not the case for $k \geq 4$.

Finally, we wish to evaluate the contribution of the SAC algorithm to the overall framework. One could possibly fix an initial block structure and then iteratively update the dictionary using BK-SVD, in hope that this will recover the block structure. Fig. **??** shows that the representation error e is much lower when including SAC in the overall framework. Moreover, BK-SVD consistently fails in recovering the dictionary blocks. Note, that for the task of dictionary design it is always beneficiary to aggregate atoms into blocks. Hence, the SAC algorithm continues grouping atoms until the maximal block size has been reached, or no further merges can occur (e.g., merging any pair of blocks will result in exceeding s). The resulting block structure has blocks of size s.

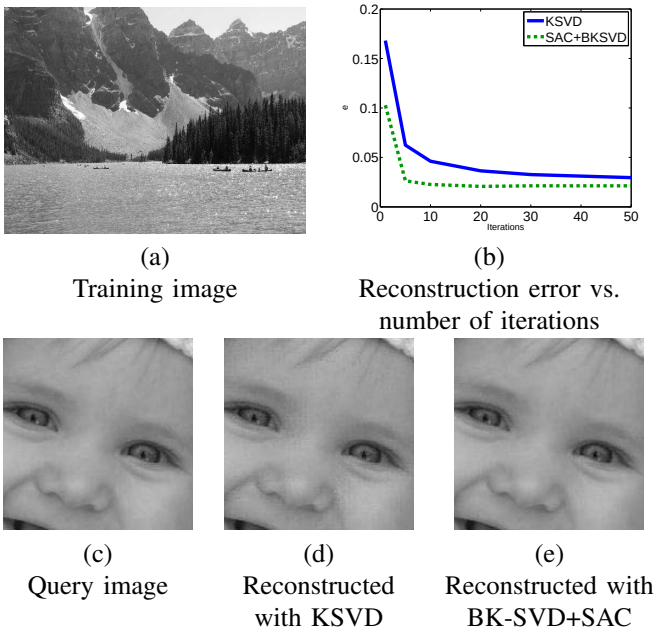
D. Experiments on Real-World Data

To further show the usefulness of the proposed approach we test it on real image data. Many algorithms in image processing, e.g., denoising, super-resolution, inpainting and object recognition, are based on decomposing the query image into patches and replacing each patch with a combination of the most similar ones in a given database of patches [**?**], [**?**], [**?**]. For example, in [**?**] object recognition is performed using a database of labeled image patches, i.e., the category from which each patch originated is known. For each patch from the query image one finds the most similar ones in the database and extracts their labels. The query image is then classified according to the labels of the matches from the database using, e.g., majority voting. In all of these patch-based applications, the success of the algorithm depends on finding correctly the nearest neighbors of each patch. This is achieved when the database of patches provides a good coverage of the space of all image patches.

Here, we compare the ability of K-SVD and our SAC+BK-SVD to cover the space of image patches, via the following experiment. We randomly generate a dictionary D of dimension 25×75 with normally distributed entries and normalize its columns. We further set the maximal block size to $s = 3$ and block sparsity to $k = 3$. We extract all 5×5 non-overlapping image patches from a training image and reshape them into $L = 1800$ column vectors of dimension $N = 25$, which are used as training signals in X. We then use both K-SVD and the proposed SAC+BK-SVD for optimizing the dictionary.

Next, we take a different image for testing, extract all its non-overlapping patches and find for each patch its sparse representation. We find a $k \cdot s$ sparse solution when using the K-SVD dictionary and a k-block sparse solution when using the dictionary trained by SAC+BK-SVD. Finally, we compute the mean reconstruction error over all image patches. We have repeated this experiment 50 times and report the mean errors over these 50 trials. The results are presented in Fig. **??** as a function of the number of iterations in training the dictionary. As can be seen, the dictionary learned by SAC+BK-SVD leads to smaller errors, and converges faster than K-SVD.

We have further repeated this experiment, but using the output of K-SVD as the initialization of SAC+BK-SVD instead of random initialization. The results were highly similar.



(a) Training image (b) Reconstruction error vs. number of iterations

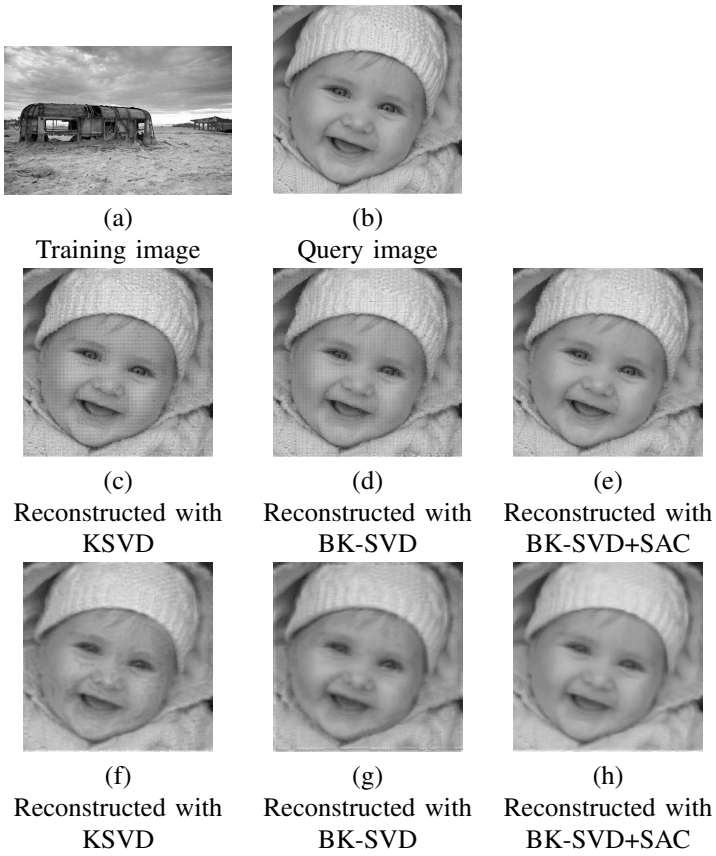
(c) Query image (d) Reconstructed with KSVD (e) Reconstructed with BK-SVD+SAC

Fig. 6. Each 5×5 patch in the query image was replaced with its corresponding sparse approximation using the dictionaries trained by K-SVD and SAC+BK-SVD. The latter leads to smaller errors and faster convergence.

Similar to [**?**] we also experimented with super-resolution. We extracted $L = 2300$ training patches of size 8×8 from the image in Fig. **??**(a). These patches were used to train a dictionary of size $N \times K = 64 \times 96$, initialized with random gaussian entries. The allowed size of each block is $s = 3$, and the allowed block sparsity was $k = 2$. We trained three dictionaries. The first using KSVD with $k \cdot s = 6$ nonzero elements, the second using SAC+BK-SVD with k nonzero blocks, and the third using BKSVD (with fixed block structure) with k nonzero blocks. The mean reconstruction errors over the training signals were for KSVD $e = 0.036274$, for SAC+BK-SVD $e = 0.032858$, and for BKSVD $e = 0.038445$.

Next, we extracted all 8×8 non-overlapping image patches of the image in Figure **??**.(b). The corresponding 64 long vectors were compressed into 16 dimensions using a sensing matrix trained by the algorithm of [**?**]. We then used each of the trained dictionaries to reconstruct the 64 dimensional patches. The reconstruction results are presented in Fig. **??**(c)-(e). It can be seen that the proposed SAC+BK-SVD leads to the smallest errors out of the three.

Finally, we have further extracted all 8×8 image patches with overlaps, of the image in Figure **??**.(b). The corresponding 64 long vectors were again compressed into 16 dimensions using a sensing matrix trained by the algorithm of [**?**]. We then used each of the trained dictionaries to reconstruct the 64 dimensional patches. The image was reconstructed by averaging at each pixel the corresponding values from the reconstructed patches that include it. The reconstruction results are presented in Fig. **??**(f)-(h). It can be seen again, that the proposed SAC+BK-SVD leads to the smallest errors out of the three.



(a) Training image (b) Query image

(c) Reconstructed with KSVD (d) Reconstructed with BK-SVD (e) Reconstructed with BK-SVD+SAC

(f) Reconstructed with KSVD (g) Reconstructed with BK-SVD (h) Reconstructed with BK-SVD+SAC

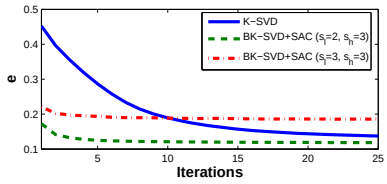
Fig. 7. Super-resolution experiment. Dictionaries were trained by K-SVD, BK-SVD and SAC+BK-SVD using 8×8 patches from the training image in (a). Then, each 8×8 patch in the query image (b) was compressed into 16 dimensions. A 64 dimensional approximation was obtained for each compressed patch using the three dictionaries. (c)-(e) show the obtained reconstruction with non-overlapping patches. (f)-(h) show the obtained reconstruction after averaging all overlapping patches. SAC+BK-SVD leads to smallest errors.

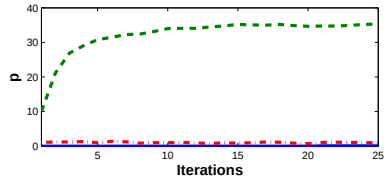
E. Choosing the Maximal Block Size

We now consider the problem of setting the maximal block size in the dictionary, when all we are given is that the sizes of the blocks are in the range $[s_l \ s_h]$. This also includes the case of varying block sizes. Choosing the maximal block size s to be equal to s_l will not allow to successfully reconstruct blocks containing more than s_l atoms. On the other hand, setting $s = s_h$ will cause the initial sparse representation matrix Θ,

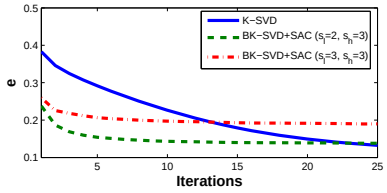
obtained by the OMP algorithm, to contain too many non-zero coefficients. This is experienced as noise by the SAC algorithm, and may prevent it from functioning properly. It is therefore favorable to use OMP with $k \times s_l$ non-zero entries only, and set the maximal block size s to be s_h.

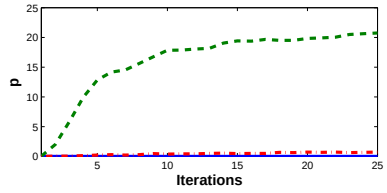
In Fig. **??**, we evaluate the ability of our block sparsifying dictionary design algorithm to recover the optimal dictionary, which contains 12 blocks of size 3, and 12 blocks of size 2. As expected, better results are obtained when choosing $s_l = 2$. In Fig. **??**, the underlying block subspaces are all of dimension 2, but s_h is erroneously set to be 3. We see that when $s_l = 2$, we succeed in recovering a considerable part of the blocks, even though blocks of size 3 are allowed. In both simulations, K-SVD uses $k \times s_h$ non-zero entries, which explains why it is not significantly outperformed by our algorithm in terms of representation error. Moreover, the percentage of reconstructed blocks by our algorithm is relatively low compared to the previous simulations, due to the small block sizes.





(a)





(b)

Fig. 8. Simulation results of our overall algorithm (BK-SVD+SAC) and the K-SVD algorithm, with maximal block size $s_h = 3$. The graphs show the reconstruction error e and the recovery percentage p as a function of the number of iterations (recall that given our definition of p we declare a correct recovery when the distance between the recovered block and its best match is small, i.e., they need not be of the same dimension). (a) The input data contains 12 blocks of size 2 and 12 block of size 3. (b) The input data contains 30 blocks of size 2.

V. Conclusion

In this paper, we proposed a framework for the design of a block-sparsifying dictionary given a set of signals and a maximal block size. The algorithm consists of two steps: a block structure update step (SAC) and a dictionary update step (BK-SVD). When the maximal block size is chosen to be 1, the algorithm reduces to K-SVD.

We have shown via experiments that the block structure update step (SAC) provides a significant contribution to the dictionary recovery results. We have further shown that for $s > 1$ the BK-SVD dictionary update step is superior to the K-SVD dictionary update. Moreover, the representation error obtained by our dictionary design method lies very close to the lower bound (the oracle) for all noise levels. This suggests that our algorithm has reached its goal in providing dictionaries that lead to accurate sparse representations for a given set of signals.

To further improve the proposed approach one could try and make the dictionary design algorithm less susceptible to local minimum traps. Another refinement could be replacing blocks in the dictionary that contribute little to the sparse representations (i.e. "unpopular blocks") with the least represented signal elements. This is expected to only improve reconstruction results. Finally, we may replace the time-efficient BOMP algorithm, with other block-sparse approximation methods. We leave these issues for future research.

VI. Acknowledgements

The research of Lihi Zelnik-Manor is supported by Marie Curie IRG-208529 and by the Ollendorf foundation. The work of Yonina Eldar was supported in part by a Magneton grant from the Israel Ministry of Industry and Trade, and by the Israel Science Foundation under Grant 170/10. We wish to thank Ezri Son for his assistance in the empirical evaluation.

References

[1] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inform. Theory*, vol. 52, pp. 489–509, Feb. 2006.

[2] D. Donoho, "Compressed sensing," *IEEE Trans. Inform. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[3] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1999.

[4] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Royal Statist. Soc B*, vol. 58, no. 1, pp. 267–288, 1996.

[5] J. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Inform. Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.

[6] S. G. Mallat and Z. Zhang, "Matching pursuits and time-frequency dictionaries," *IEEE Trans. Sig. Proc.*, vol. 41, no. 12, p. 33973415, Dec. 1993.

[7] M. Aharon, M. Elad, and A. M. Bruckstein, "The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representations," *IEEE Trans. SP*, vol. 54, no. 11, 2006.

[8] K. Engan, S. O. Aase, and J. H. Hakon-Husoy, "Method of optimal directions for frame design," *IEEE Int. Conf. Acoust., Speech, Signal Process*, vol. 5, pp. 2443–2446, 1999.

[9] B. Olshausen and D. Field, "Natural image statistics and efficient coding," *Network: Comput. Neural Syst.*, vol. 2, no. 7, pp. 333–339, 1996.

[10] S. Lesage, R. Gribonval, F. Bimbot, and L. Benaroya, "Learning unions of orthonormal bases with thresholded singular value decomposition," *IEEE Conf. on Acoustics, Speech and Signal Processing*, 2005.

[11] J. M. Duarte-Carvajalino and G. Sapiro, "Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization," *IMA Preprint Series*, no. 2211, May 2008.

[12] K. K.-D. J. F. M. B. D. R. K. E. T. W. Lee and T. J. Senowski, "Dictionary learning algorithms for sparse representation," *Neural Computation*, vol. 15, no. 2, pp. 349–396, 2003.

[13] Y. C. Eldar and M. Mishali, "Robust recovery of signals from a structured union of subspaces," *IEEE Trans. Inform. Theory*, vol. 55, no. 11, pp. 5302–5316, Nov. 2009.

[14] K. Gedalyahu and Y. C. Eldar, "Time delay estimation from low rate samples: A union of subspaces approach," *IEEE Trans. Signal Process.*, vol. 58, no. 6, pp. 3017–3031, Jun. 2010.

[15] Y. Lu and M. Do, "Sampling signals from a union of subspaces," vol. 25, no. 2, pp. 41–47, Mar. 2008.

[16] R. Basri and D. Jacobs, "Lambertian refelectances and linear subspaces," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 25, no. 2, pp. 383–390, Feb 2003.

[17] A. Y. Yang, J. Wright, Y. Ma, and S. Sastry, "Feature selection in face recognition: A sparse representation perspective," *UC Berkeley Tech Report*, Aug 2007.

[18] R. Vidal and Y. Ma, "A unified algebraic approach to 2-D and 3-D motion segmentation and estimation," *Journal of Mathematical Imaging and Vision*, vol. 25, no. 3, pp. 403–421, Oct. 2006.

[19] M. Mishali and Y. C. Eldar, "Blind multiband signal reconstruction: Compressed sensing for analog signals," *IEEE Trans. Sig. Proc.*, vol. 57, no. 3, pp. 993–1009, Mar. 2009.

[20] ——, "From theory to practice: Sub-Nyquist sampling of sparse wideband analog signals," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 375 – 391, Apr. 2010.

[21] H. J. Landau, "Necessary density conditions for sampling and interpolation of certain entire functions," *Acta Math.*, vol. 117, no. 1, pp. 37–52, 1967.

[22] F. Parvaresh, H. Vikalo, S. Misra, and B. Hassibi, "Recovering sparse signals using sparse measurement matrices in compressed dna microarrays," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 3, pp. 275–285, Jun. 2008.

[23] M. Stojnic, F. Parvaresh, and B. Hassibi, "On the reconstruction of block-sparse signals with an optimal number of measurements," *IEEE Trans. Sig. Proc.*, vol. 57, no. 8, pp. 3075–3085, Aug. 2009.

[24] H. Rauhut and Y. Eldar, "Average case analysis of multichannel sparse recovery using convex relaxation," *IEEE Trans. Inform. Theory*, vol. 56, no. 1, pp. 505–519, Jan. 2010.

[25] Y. C. Eldar, P. Kuppinger, and H. Bölcskei, "Block-sparse signals: Uncertainty relations and efficient recovery," *IEEE Trans. Sig. Proc.*, Apr. 2010.

[26] Y. C. Eldar and H. Bölcskei, "Block-sparsity: Coherence and efficient recovery," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 0, pp. 2885–2888, 2009.

[27] L. Meier, S. V. D. Geer, and P. B
"uhlmann, "The group lasso for logistic regression," *Journal of the Royal Statistical Society: Series B (Statistical Methodology*, vol. 70, no. Part 1, pp. 53–71, 2008.

[28] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology*, vol. 68, no. 1, pp. 49 – 67, Feb. 2006.

[29] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (GPCA)," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, 2005.

[30] R. V. E. Elhamifar, "Sparse subspace clustering," *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[31] Y. Ma, H. Derksen, W. Hong, and J. Wright, "Segmentation of multivariate mixed data via lossy coding and compression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, p. 15461562, Sep 2007.

[32] H. Duda and P. Hart, "Stork, Pattern Classification," 2001.

[33] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, pp. 241–254, Sep. 1967.

[34] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit," *Techninal report - CS Technion*, 2008.

[35] X. Sun and Q. Cheng, "On subspace distance," in *ICIAR (2)*, 2006, pp. 81–89.

[36] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[37] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 2272–2279.

[38] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. Le-Cun, "Learning invariant features through topographic filter maps," pp. 1605–1612, 2009.

[39] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008.* IEEE, 2008, pp. 1–8.

Lihi Zelnik-Manor Lihi Zelnik-Manor received the BSc degree in mechanical engineering from the Technion in 1995, where she graduated summa cum laude, and the MSc (with honors) and PhD degrees in computer science from the Weizmann Institute of Science in 1998 and 2004, respectively. After graduating, she worked as a postdoctoral fellow in the Department of Engineering and Applied Science at the California Institute of Technology (Caltech). Since 2007, she has been a senior lecturer in the Electrical Engineering Department at the Technion. Her research focuses on the analysis of dynamic visual data, including video analysis and visualizations of multiview data. Her awards and honors include the Israeli high-education planning and budgeting committee (Vatat) three-year scholarship for outstanding PhD students, and the Sloan-Swartz postdoctoral fellowship. She also received the best Student Paper Award at the IEEE Shape Modeling International Conference 2005 and the AIM@SHAPE Best Paper Award 2005. She is a member of the IEEE.

Kevin Rosenblum Kevin Rosenblum received the B.Sc. degree in electrical engineering and the B.A. degree in physics in 2008, and the M.Sc. degree in electrical engineering in 2011 from the Technion-Israel Institute of Technology, Haifa, Israel. He is currently an algorithm engineer with Mobileye's Vehicle Detection division in Jerusalem, and he is pursuing the M.B.A. degree at the Tel-Aviv University. His main areas of interests include signal processing algorithms, image processing, and computer vision.

Yonina C. Eldar Yonina C. Eldar received the B.Sc. degree in physics and the B.Sc. degree in electrical engineering both from Tel-Aviv University (TAU), Tel-Aviv, Israel, in 1995 and 1996, respectively, and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, in 2002.

From January 2002 to July 2002, she was a Postdoctoral Fellow at the Digital Signal Processing Group at MIT. She is currently a Professor in the Department of Electrical Engineering at the TechnionIsrael Institute of Technology, Haifa. She is also a Research Affiliate with the Research Laboratory of Electronics at MIT and a Visiting Professor at Stanford University, Stanford, CA. Her research interests are in the broad areas of statistical signal processing, sampling theory and compressed sensing, optimization methods, and their applications to biology and optics.

Dr. Eldar was in the program for outstanding students at TAU from 1992 to 1996. In 1998, she held the Rosenblith Fellowship for study in electrical engineering at MIT, and in 2000, she held an IBM Research Fellowship. From 2002 to 2005, she was a Horev Fellow of the Leaders in Science and Technology program at the Technion and an Alon Fellow. In 2004, she was awarded the Wolf Foundation Krill Prize for Excellence in Scientific Research, in 2005 the Andre and Bella Meyer Lectureship, in 2007 the Henry Taub Prize for Excellence in Research, in 2008 the Hershel Rich Innovation Award, the Award for Women with Distinguished Contributions, the Muriel & David Jacknow Award for Excellence in Teaching, and the Technion Outstanding Lecture Award, in 2009 the Technions Award for Excellence in Teaching, and in 2010 the Michael Bruno Memorial Award from the Rothschild Foundation. She is a member of the IEEE Signal Processing Theory and Methods technical committee and the Bio Imaging Signal Processing technical committee, an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the EURASIP Journal of Signal Processing, the SIAM Journal on Matrix Analysis and Applications, and the SIAM Journal on Imaging Sciences, and on the Editorial Board of Foundations and Trends in Signal Processing.