Dictionary Optimization for Block-Sparse Representations *

Kevin Rosenblum and **Lihi Zelnik-Manor**
kevin@tx.technion.ac.il
lihi@ee.technion.ac.il

Yonina C. Eldar
yonina@ee.technion.ac.il

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. 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.

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 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 (Candes, Romberg, and Tao 2006; Donoho 2006) 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) (Chen, Donoho, and Saunders 1999) and Orthogonal Matching Pursuit (OMP) (Tropp 2004; Mallat and Zhang 1993).

Recent work (Aharon, Elad, and Bruckstein 2006b; Engan, Aase, and Hakon-Husoy 1999; Olshausen and Field 1996; Lesage et al. 2005; Duarte-Carvajalino and Sapiro 2008) has demonstrated that adapting the dictionary D to fit a given set of signal examples leads to improved signal reconstruction. At the price of being slow, 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 a unique sparsity 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 (Eldar and Mishali 2009; Gedalyahu and Eldar 2010). This occurs naturally, for example, in face recognition (Basri and Jacobs 2003; Yang et al. 2007), motion segmentation (Vidal and Ma 2006), multiband signals (Mishali and Eldar 2009; 2010; Landau 1967), measurements of gene expression levels (Parvaresh et al. 2008), 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) (Eldar and Mishali 2009; Stojnic, Parvaresh, and Hassibi 2009; Rauhut and Eldar 2009) and *Block OMP* (BOMP) (Eldar, Kuppinger, and Bölcskei 2010; Eldar and Bölcskei 2009) 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 infer the block structure from the data while adapting the dictionary.

We start by formulating this task as an optimization problem. We then present an algorithm for minimizing the pro-

*The authors are with the Technion - Israel Institute of Technology, Haifa, Israel.

posed 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 (Vidal, Ma, and Sastry 2005; Elhamifar 2009; Ma et al. 2007). 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 algorithm (Aharon, Elad, and Bruckstein 2006b), which yields sparsifying dictionaries by sequentially updating the dictionary atoms, to the case of block structures. In other words, when the blocks are of size 1 our cost function and the algorithm we propose reduce 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 show 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 (Section). In Section we give a detailed description of the algorithm for finding a block structure and in Section we describe the dictionary update part. We evaluate the performance of the proposed algorithms and compare them 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^r_c. 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.

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 (Aharon, Elad, and Bruckstein 2006b; 2006a) and MOD (Method of Optimal Directions) (Engan, Aase, and Hakon-Husoy 1999). We will generalize these methods to block-sparsifying dictionary design in Section .

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

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

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 (1) 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 (1) is non-convex and NP-hard in general. Both MOD and K-SVD attempt to approximate (1) 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 (1) becomes:

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

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.

The MOD algorithm treats the problem in (3) 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 (3) 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

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

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.

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.

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 (Eldar and Mishali 2009; Stojnic, Parvaresh, and Hassibi 2009). Instead, in this paper we address the more general case where the block structure is unknown and the blocks can be of varying sizes. The only assumption we make on the block structure is 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. 1 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:

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

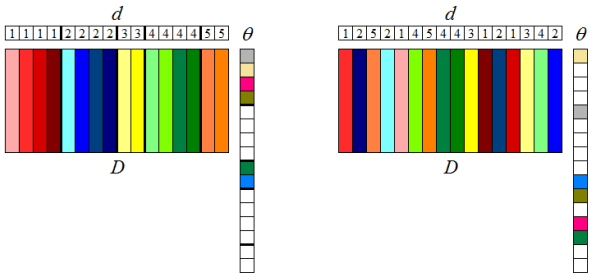


Figure 1: Two equivalent examples of dictionaries D and block structures d with 5 blocks, together with 2-block-sparse representations θ. Both examples represent the same signal, since the atoms in D and the entries of d and θ are permuted in the same manner.

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

The case when there is no underlying block structure or when the block structure is ignored, is equivalent to setting $s = 1$ and $d = [1,\ldots,K]$. Substituting this into (5), reduces it to (1). 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 (5) via the proposed framework improves recovery results and lowers the representation errors with respect to (1).

Algorithm Preview

In this section, we propose a framework for solving (5). 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 (5) for d and Θ while keeping $D^{(n-1)}$ fixed:

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

   An exact solution would require a combinatorial search over all feasible d and Θ. Instead, we propose a tractable approximation to (6) in Section , referred to as *Sparse Agglomerative Clustering (SAC)*. Agglomerative clustering builds blocks by progressively merging the closest atoms according to some distance metric (Duda and Hart 2001). SAC uses the l_0-norm for this purpose.

2. Fit the dictionary $D^{(n)}$ to the data by solving (5) for D and Θ while keeping $d^{(n)}$ fixed:

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

In Section we propose an algorithm, referred to as *Block K-SVD (BK-SVD)*, for solving (8). 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.

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. $n = n + 1$.

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. 2(a). The suggested method is based on the coordinate relaxation technique to solve (6) 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. $d = [1, \ldots, K]$. To initialize Θ we keep d fixed and solve (6) 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 (Eldar, Kuppinger, and Bölcskei 2010). 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 (6) while keeping Θ fixed. Although the objective does not depend on d, the constraints 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:

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

Before we describe how (8) 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. Therefore, 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 (8) 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 (8) can now be rewritten as:

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

where $|\omega_j|$ denotes the size of the list ω_j. We propose using a sub-optimal tractable agglomerative clustering algorithm (Johnson 1967) 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. We allow merging blocks as long as the maximum block size s is not exceeded.

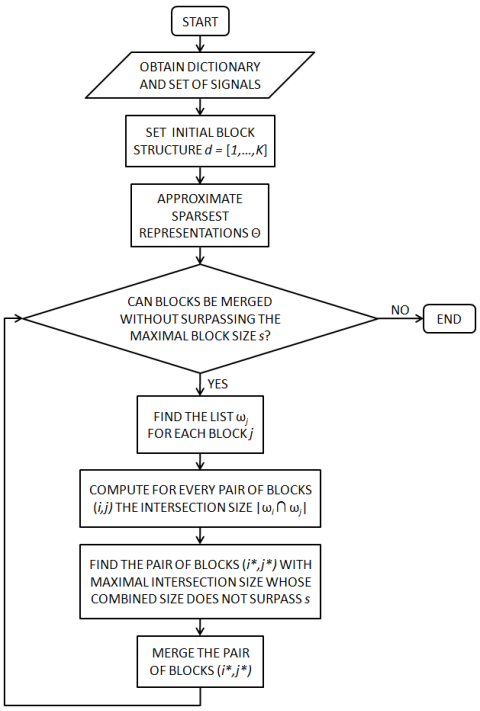
More specifically, at each step we find the pair of blocks (j_1^*, j_2^*) such that:

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

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 (8); 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. 2(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 (8) 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 (8) 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 (8).

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



(a)

$$\Theta_{4\times 4} = \begin{pmatrix} 1.2 & 0 & 0 & -3.5 \\ 0 & 0 & 0.2 & 0 \\ 1.9 & 0 & 0 & 2.3 \\ 0 & 4.7 & -0.8 & 0 \end{pmatrix}$$

$s = 2$

$$d=[1,2,3,4] \Rightarrow \begin{cases} \omega_1 = \{1,4\} \\ \omega_2 = \{3\} \\ \omega_3 = \{1,4\} \\ \omega_4 = \{2,3\} \end{cases} \Rightarrow \sum_{i=1}^{4} \|\Theta_i\|_{0,d} = 7$$

Step 1: $|\omega_1 \cap \omega_3| = 2 \Rightarrow d[3] = 1$

$\Downarrow$

$$d=[1,2,1,4] \Rightarrow \begin{cases} \omega_1 = \{1,4\} \\ \omega_2 = \{3\} \\ \omega_4 = \{2,3\} \end{cases} \Rightarrow \sum_{i=1}^{4} \|\Theta_i\|_{0,d} = 5$$

Step 2: $|\omega_2 \cap \omega_4| = 1 \Rightarrow d[4] = 2$

$\Downarrow$

$$d=[1,2,1,2] \Rightarrow \begin{cases} \omega_1 = \{1,4\} \\ \omega_2 = \{2,3\} \end{cases} \Rightarrow \sum_{i=1}^{4} \|\Theta_i\|_{0,d} = 4$$

(b)

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

and are therefore not suited for highly sparse data and the l_0-norm metric. Moreover, since oversized blocks are unwanted, it is preferable to limit the block size rather than the number of 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.

Block K-SVD Algorithm

We now propose the BK-SVD algorithm for recovering the dictionary D and the representations Θ by optimizing (8) 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 (8) which reduces to

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

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

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

Inspired by the K-SVD algorithm, the blocks in $D^{(m-1)}$ are updated sequentially, along with the corresponding nonzero coefficients in $\Theta^{(m)}$. For every block $j \in d$, 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, the solution is unique up to the 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.

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.

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.

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.

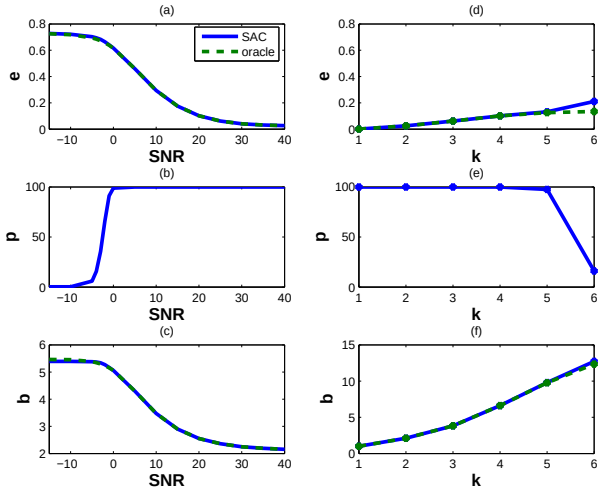


Figure 3: Simulation results of the SAC algorithm. The graphs show e, p and b as a function of the SNR of the data signals for $k = 2$ (a, b, c), and as a function of k in a noiseless setting (d, e, f).

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. 3 also shows the objective of (8), 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. 3.(b)), the representation error e drops to zero (Fig. 3.(a)) and the block-sparsity b drops to the lowest possible value $k = 2$ (Fig. 3.(c)). Fig. 3.(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. 3.(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.

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, updating Θ using block-sparse approximation and updating

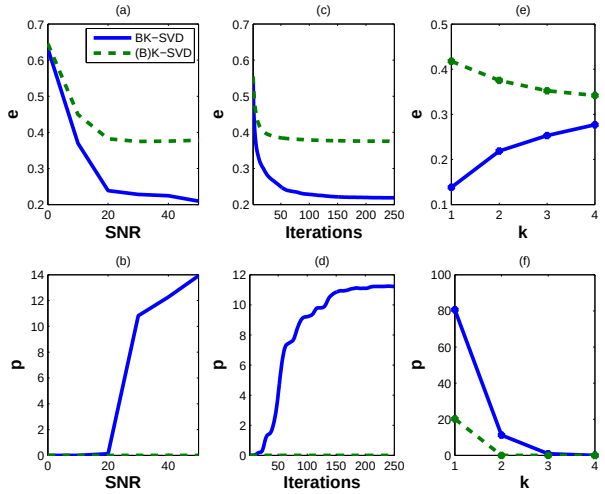


Figure 4: Simulation results of the BK-SVD and (B)K-SVD algorithms. The graphs show the reconstruction error e and the recovery percentage p as a function of the SNR of the data signals for $k = 2$ and after 250 iterations (a, b), as a function of the number of iterations for $k = 2$ in a noiseless setting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

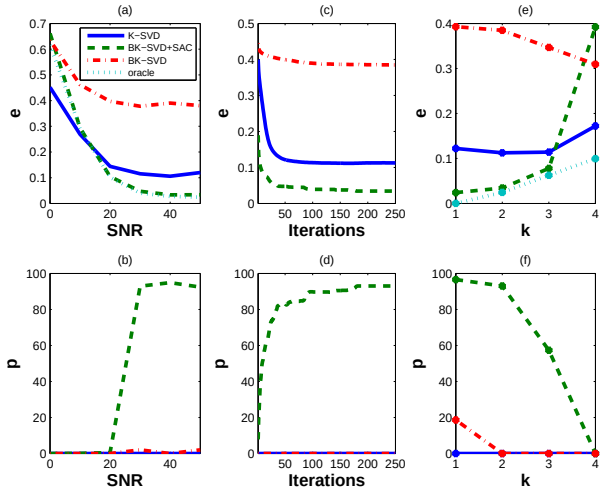


Figure 5: Simulation results of our overall algorithm (BK-SVD+SAC), the BK-SVD algorithm and the K-SVD algorithm. The graphs show the reconstruction error e and the recovery percentage p as a function of the SNR of the data signals for $k = 2$ after 250 iterations (a, b), as a function of the number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

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. 4 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.

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. 5).

Our experiments show that for SNR > 10[dB], the proposed block-sparsifying dictionary design algorithm yields lower reconstruction errors (see Fig. 5.(a)) and a higher percentage of correctly reconstructed blocks (see Fig. 5.(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. 5.(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 the 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. 5 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 $k <= 3$, 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 $k >= 4$.

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. 5 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.

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 setting the maximal block size s to be s_h.

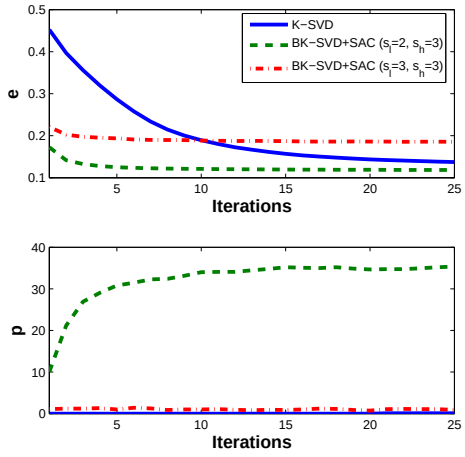
In Fig. 6(a), we evaluate the ability of our block sparsifying dictionary design algorithm to recover the optimal dictionary, which contains 12 blocks of size 3, and 12 blocks of size 2. As expected, better results are obtained when choosing $s_l = 2$. In Fig. 6(b), the underlying block subspaces are 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.

Conclusions

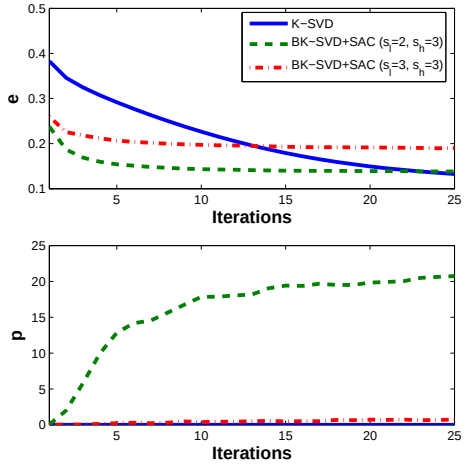
In this paper, we proposed a framework for the design of a block-sparsifying dictionary given a set of signals and a maximal block size. The algorithm consists of two steps: a block structure update step (SAC) and a dictionary update step (BK-SVD). When the maximal block size is chosen to be 1, the algorithm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



(a)



(b)

Figure 6: 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. (a) contains 12 blocks of size 2 and 12 block of size 3. (b) contains 30 blocks of size 2.

Acknowledgements

The research of Lihi Zelnik-Manor is supported by Marie Curie IRG-208529.

References

Aharon, M.; Elad, M.; and Bruckstein, A. M. 2006a. On the uniqueness of overcomplete dictionaries and a practical way to retrieve them. *Journal of Linear Algebra and Applications* 416:4867.

Aharon, M.; Elad, M.; and Bruckstein, A. M. 2006b. The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representations. *IEEE Trans. SP* 54(11).

Basri, R., and Jacobs, D. 2003. Lambertian refelectances and linear subspaces. *IEEE Transactions On Pattern Analysis And Machine Intelligence* 25(2):383–390.

Candes, E.; Romberg, J.; and Tao, T. 2006. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inform. Theory* 52:489–509.

Chen, S. S.; Donoho, D. L.; and Saunders, M. A. 1999. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comput.* 20(1):33–61.

Donoho, D. 2006. Compressed sensing. *IEEE Trans. Inform. Theory* 52(4):1289–1306.

Duarte-Carvajalino, J. M., and Sapiro, G. 2008. Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization. *IMA Preprint Series* (2211).

Duda, H., and Hart, P. 2001. Stork, Pattern Classification.

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

Eldar, Y. C., and Mishali, M. 2009. Robust recovery of signals from a structured union of subspaces. *IEEE Trans. Inform. Theory* 55(11):5302–5316.

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

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

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

Gedalyahu, K., and Eldar, Y. C. 2010. Time delay estimation from low rate samples: A union of subspaces approach. *arXiv.org 0905.2429; to appear in IEEE Trans. Signal Process.*

Johnson, S. C. 1967. Hierarchical clustering schemes. *Psychometrika* 32:241–254.

Landau, H. J. 1967. Necessary density conditions for sampling and interpolation of certain entire functions. *Acta Math.* 117(1):37–52.

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

Ma, Y.; Derksen, H.; Hong, W.; and Wright, J. 2007. Segmentation of multivariate mixed data via lossy coding and compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(9):15461562.

Mallat, S. G., and Zhang, Z. 1993. Matching pursuits and time-frequency dictionaries. *IEEE Trans. Sig. Proc.* 41(12):33973415.

Mishali, M., and Eldar, Y. C. 2009. Blind multiband signal reconstruction: Compressed sensing for analog signals. *IEEE Trans. Sig. Proc.* 57(3):993–1009.

Mishali, M., and Eldar, Y. C. 2010. From theory to practice: Sub-nyquist sampling of sparse wideband analog signals. *IEEE Journal of Selected Topics in Signal Processing* 4(2):375 – 391.

Olshausen, B., and Field, D. 1996. Natural image statistics and efficient coding. *Network: Comput. Neural Syst.* 2(7):333–339.

Parvaresh, F.; Vikalo, H.; Misra, S.; and Hassibi, B. 2008. Recovering sparse signals using sparse measurement matrices in compressed dna microarrays. *IEEE Journal of Selected Topics in Signal Processing*.

Rauhut, H., and Eldar, Y. 2009. Average case analysis of multichannel sparse recovery using convex relaxation. *preprint*.

Stojnic, M.; Parvaresh, F.; and Hassibi, B. 2009. On the reconstruction of block-sparse signals with an optimal number of measurements. *IEEE Trans. Sig. Proc.* 57(8):3075–3085.

Tropp, J. 2004. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. Inform. Theory* 50(10):2231–2242.

Vidal, R., and Ma, Y. 2006. A unified algebraic approach to 2-D and 3-D motion segmentation and estimation. *Journal of Mathematical Imaging and Vision* 25(3):403–421.

Vidal, R.; Ma, Y.; and Sastry, S. 2005. Generalized principal component analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(11).

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