K-SVD: DESIGN OF DICTIONARIES FOR SPARSE REPRESENTATION

Michal Aharon *Michael Elad* *Alfred Bruckstein*

Technion—Israel Institute of Technology, Haifa 32000, Israel

ABSTRACT

In recent years there is a growing interest in the study of sparse representation for signals. Using an overcomplete dictionary that contains prototype signal-atoms, signals are described by sparse linear combinations of these atoms. Recent activity in this field concentrated mainly on the study of pursuit algorithms that decompose signals with respect to a given dictionary.

In this paper we propose a novel algorithm – the K-SVD algorithm – generalizing the K-Means clustering process, for adapting dictionaries in order to achieve sparse signal representations.

We analyze this algorithm and demonstrate its results on both synthetic tests and in applications on real data.

1. INTRODUCTION

Recent years have witnessed a growing interest in the use for sparse representations for signals. Using an overcomplete dictionary matrix $\mathbf{D} \in \mathbb{R}^{n\times K}$ that contains K atoms, $\{\mathbf{d}_j\}_{j=1}^K$, as its columns, it is assumed that a signal $\mathbf{y} \in \mathbb{R}^n$ can be represented as a sparse linear combination of these atoms. The representation of $\mathbf{y}$ may either be exact $\mathbf{y} = \mathbf{D}\mathbf{x}$, or approximate, $\mathbf{y} \approx \mathbf{D}\mathbf{x}$, satisfying $\|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2 \leq \epsilon$. The vector $\mathbf{x} \in \mathbb{R}^K$ displays the representation coefficients of the signal $\mathbf{y}$. This, sparsest representation, is the solution of either

$$(P_0) \quad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{y} = \mathbf{D}\mathbf{x}, \tag{1}$$

or

$$(P_{0,\epsilon}) \quad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2 \leq \epsilon, \tag{2}$$

where $\|\cdot\|_0$ is the l^0 norm, counting the non zero entries of a vector.

Applications that can benefit from the sparsity and overcompleteness concepts (together or separately) include compression, regularization in inverse problems, feature extraction, and more. In this paper we consider the problem of designing dictionaries based on learning from signal examples. Our goal is to find the dictionary $\mathbf{D}$ that yields sparse representations for a set of training signals. We believe that such dictionaries have the potential to outperform commonly used pre-determined dictionaries. With the ever-growing computational resources that we have access to today, such methods will adapt dictionaries for special classes of signals, and yield better performance in various applications.

2. PRIOR ART

Pursuits Algorithms: In order to use overcomplete and sparse representations in applications, one need to fix a dictionary $\mathbf{D}$, and then find efficient ways to solve (1) or (2). Exact determination of sparsest representations proves to be an NP-hard problem [1]. Hence, approximate solutions are considered instead. In the past decade or so several efficient pursuit algorithms have been proposed. The simplest ones are the *Matching Pursuit* (MP) [2] or the *Orthogonal Matching Pursuit* (OMP) algorithms [3]. These are greedy algorithms that select the dictionary atoms sequentially. A second well known pursuit approach is the *Basis Pursuit* (BP) [4]. It suggests a convexisation of the problems posed in (1) and (2), by replacing the ℓ^0-norm with an ℓ^1-norm. The Focal Under-determined System Solver (FOCUSS) is very similar, using the ℓ^p-norm with $p \leq 1$, as a replacement to the ℓ^0-norm [5]. Here, for $p < 1$ the similarity to the true sparsity measure is better, but the overall problem becomes non-convex, giving rise to local minima that may divert the optimization. Extensive study of these algorithms in recent years has established that if the sought solution, $\mathbf{x}$, is sparse enough, these techniques recover it well [6, 7, 8, 3].

Design of Dictionaries: We now briefly mention the main work done in the field of fitting a dictionary to data examples. A more thorough summary can be found in [9]. Pioneering work by Field and Olshausen set the stage for dictionary training methods, proposing a ML-based objective function to minimize with respect to the desired dictionary [10]. Subsequent work by Lewicki, Olshausen and Sejnowski, proposed several direct extensions of this work [11]. Further important contributions on training of sparse representations dictionaries has been made by the creators of the FOCUSS algorithm, Rao and Kreutz-Delgado, together with Engan [12, 13]. They pointed out the connection between the sparse coding dictionary design and the vector quantization problem, and proposed some type of general-

ization of the well known K-Means algorithm. In this work we present a different approach for this generalization. We regard this recent activity on the subject as a further proof for the importance of this subject, and the prospects it encompasses.

3. THE K-SVD ALGORITHM

In this section we introduce the K-SVD algorithm for training of dictionaries. This algorithm is flexible, and works in conjugation with any pursuit algorithm. It is simple, and designed to be a truly direct generalization of the K-Means. As such, when forced to work with one atom per signal, it trains a dictionary for the Gain-Shape VQ. When forced to have a unit coefficient for this atom, it exactly reproduces the K-Means algorithm.

We start our discussion with a short description of the Vector Quantization (VQ) problem and the K-Means algorithm. In VQ, a codebook $\mathbf{C}$ that includes K codewords is used to represent a wide family of signals $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{N}$ ($N \gg K$) by a nearest neighbor assignment. This leads to an efficient compression or description of those signals, as clusters in $\mathbb{R}^n$ surrounding the chosen codewords. The VQ problem can be mathematically described by

$$\min_{\mathbf{C},\mathbf{X}} \left\{ \|\mathbf{Y} - \mathbf{C}\mathbf{X}\|_F^2 \right\} \quad \text{s.t.} \quad \forall i,\ \mathbf{x}_i = \mathbf{e}_k \text{ for some } k. \quad (3)$$

The *K-Means* algorithm [14] is an iterative method, used for designing the optimal codebook for VQ. In each iteration there are two stages - one for sparse coding that essentially evaluates $\mathbf{X}$ by mapping each signal to its closest atom in $\mathbf{C}$, and the second for updating the codebook, changing sequentially each column $\mathbf{c}_i$ in order to better represent the signals mapped to it.

The sparse representation problem can be viewed as a generalization of VQ objective (3), in which we allow each input signal to be represented by a **linear combination** of codewords, which we now call dictionary elements or atoms. As a result, the minimization described in Equation (3) converts to

$$\min_{\mathbf{D},\mathbf{X}} \left\{ \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 \right\} \quad \text{subject to} \quad \forall i,\ \|\mathbf{x}_i\|_0 \le T_0. \quad (4)$$

In the K-SVD algorithm we solve (4) iteratively, using two stages, parallel to those in K-Means. In the sparse coding stage, we compute the coefficients matrix $\mathbf{X}$, using any pursuit method, and allowing each coefficient vector to have no more than T_0 non zero elements. Then, we update each dictionary element sequentially, changing its content, and the values of its coefficients, to better represent the signals that use it. This is markedly different from the K-Means generalizations that were proposed previously, e.g., [12, 13], since these methods freeze $\mathbf{X}$ while finding a better $\mathbf{D}$, while we change the columns of $\mathbf{D}$ sequentially, and allow changing the relevant coefficients as well. This difference results in a Gauss-Seidel-like acceleration, since the subsequent columns to consider for updating are based on more relevant coefficients. The process of updating each $\mathbf{d}_k$ has a straight forward solution, as it reduces to finding a rank-one approximation to the matrix of residuals,

$$\mathbf{E}_k = \mathbf{Y} - \sum_{j \neq k} \mathbf{d}_j \mathbf{x}^j \quad (5)$$

where $\mathbf{x}^j$ is the j'th row in the coefficient matrix $\mathbf{X}$. This approximation is based on the singular value decomposition (SVD). A full description of the algorithm is given below, while a more detailed one can be found in [9].

Initialization : Set the random normalized dictionary matrix $\mathbf{D}^{(0)} \in \mathbb{R}^{n \times K}$. Set $J = 1$.
Repeat until convergence,
Sparse Coding Stage: Use any pursuit algorithm to compute $\mathbf{x}_i$ for $i = 1, 2, \ldots, N$

$$\min_{\mathbf{x}} \left\{ \|\mathbf{y}_i - \mathbf{D}\mathbf{x}\|_2^2 \right\} \quad \text{subject to} \quad \|\mathbf{x}\|_0 \le T_0.$$

Codebook Update Stage: For $k = 1, 2, \ldots, K$

- Define the group of examples that use $\mathbf{d}_k$, $\omega_k = \{i |\ 1 \le i \le N,\ \mathbf{x}_i(k) \neq 0\}$.
- Compute

$$\mathbf{E}_k = \mathbf{Y} - \sum_{j \neq k} \mathbf{d}_j \mathbf{x}^j,$$

- Restrict $\mathbf{E}_k$ by choosing only the columns corresponding to those elements that initially used $\mathbf{d}_k$ in their representation, and obtain $\mathbf{E}_k^R$.
- Apply SVD decomposition $\mathbf{E}_k^R = \mathbf{U} \quad \mathbf{V}^T$. Update: $\mathbf{d}_k = \mathbf{u}_1$, $\mathbf{x}_R^k = \quad (1,1) \cdot \mathbf{v}_1$

Set $J = J + 1$.

The K-SVD Algorithm

We call this algorithm "K-SVD" to parallel the name K-Means. While K-Means applies K mean calculations to evaluate the codebook, the K-SVD obtains the updated dictionary by K SVD operations, each producing one column.

Similar to the K-Means, we can propose a variety of techniques to further improve the K-SVD algorithm. Most appealing on this list are multi-scale approaches, and tree-based training where the number of columns K is allowed to increase during the algorithm. We leave these matters for future work.

Synthetic Experiments: In order to demonstrate the K-SVD, we conducted a number of synthetic tests, in which we randomly chose a dictionary $\mathbf{D} \in R^{20 \times 50}$ and multiplied it with randomly chosen sparse coefficient vectors

$\{\mathbf{x}_i\}_{i=1}^{1500}$. Finally, we added white noise with varying strength to those signals. The K-SVD was executed for a maximum number of 80 iterations and generated a new dictionary $\tilde{\mathbf{D}}$. We then compared between those two dictionaries, using a similar method to the one reported by [13]. The rate of detected atoms was 92%, 95%, 96% and 96% for SNR levels of 10dB, 20db, 30db and no noise case, respectively. These results are superior than those achieved by both the MOD and the MAP-based algorithm (See [9] for more details).

4. APPLICATIONS TO IMAGE PROCESSING

We carried out several experiments on natural image data, trying to show the practicality of the proposed algorithm and the general sparse coding theme. The training data was constructed as a set of $11,000$ examples of block patches of size 8×8 pixels, taken from a database of face images (in various locations).

Working with real images data we preferred that all dictionary elements except one has a zero mean. The same measure was practiced by previous work [10]. For this purpose, the first dictionary element, denoted as the DC, was set to include a constant value in all its entries, and was not changed afterwards. The DC takes part in all representations, and as a result, all other dictionary elements remain with zero mean during all iterations.

We applied the K-SVD, training a dictionary of size 64×441. The choice $K = 441$ came from our attempt to compare the outcome to the overcomplete Haar dictionary of the same size (see the following section). The coefficients were computed using the OMP with fixed number of coefficients, were the maximal number of coefficients is 10. Note that better performance can be obtained by switching to FOCUSS. We concentrated on OMP because of its simplicity and fast execution. The trained dictionary is presented in Figure 1.

The trained dictionary was compared with the overcomplete Haar dictionary which includes separable basis functions, having steps of various sizes and in all locations (total of 441 elements). In addition, we build an overcomplete separable version of the DCT dictionary by sampling the cosine wave in different frequencies to result a total of 441 elements.

Filling in missing pixels: We chose one random full face image, which consists of 594 non-overlapping blocks (none of which were used for training). For each block, the following procedure was conducted for r in the range $\{0.2, 0.9\}$: A fraction r of the pixels in each block, in random locations, were deleted (set to zero). The coefficients of the corrupted block under the learned dictionary, the overcomplete Haar dictionary, and the overcomplete DCT dictionary were found using OMP with an error bound equivalent to 5 gray-values. All projections in the OMP algorithm included only the non-corrupted pixels, and for this purpose, the dictionary elements were normalized so that the non-corrupted indices in each dictionary element have a unit norm. The resulting coefficient vector of the block $\mathbf{B}$ is denoted $\mathbf{x_B}$. The reconstructed block $\tilde{\mathbf{B}}$ was chosen as $\tilde{\mathbf{B}} = \mathbf{D} \cdot \mathbf{x_B}$. The reconstruction error was set to: $\sqrt{\|\mathbf{B} - \tilde{\mathbf{B}}\|_F^2/64}$ (64 is the number of pixels in each block. Notice all values are in the range [0,1]).

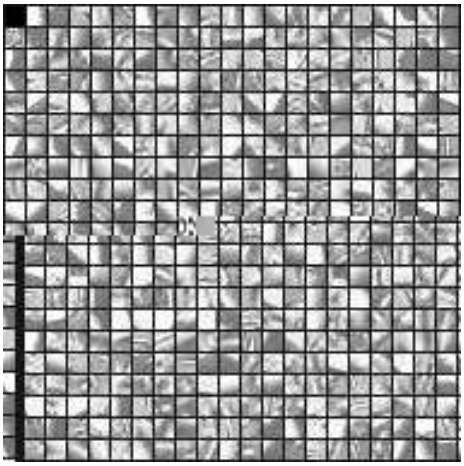


Fig. 1. K-SVD resulted dictionary

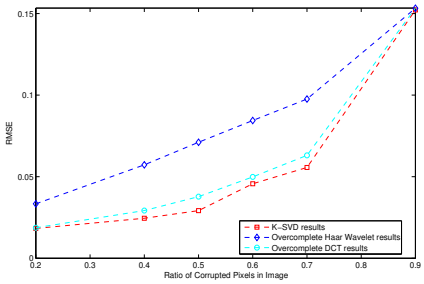
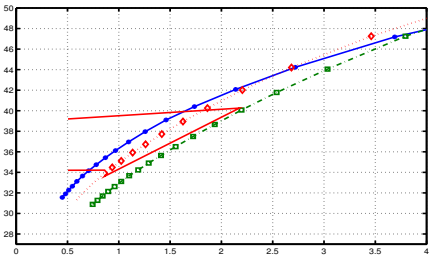


Fig. 2. Filling in - RMSE versus the relative number of missing pixels.

The mean reconstruction errors (for all blocks and all corruption rates) were computed, and are displayed in Figure 2. Two corrupted images and their reconstructions can be seen in Figure 3. As can be seen, higher quality recovery is obtained using the learned dictionary.

Compression: A compression comparison was conducted between the three dictionaries mentioned before, all of size 64×441. In addition, we compared to the regular (unitary) DCT dictionary (used by the JPEG algorithm). The resulted rate-distortion graph is presented in Figure 4. In this com-

Fig. 3. Filling in Results.



pression test, the face image was partitioned (again) into 594 disjoint 8×8 blocks. All blocks were coded in various rates (bits-per-pixel values), and the PSNR was measured. Let $\mathbf{I}$ be the original image and $\tilde{\mathbf{I}}$ be the coded image, combined by all the coded blocks. We denote e^2 as the mean squared error between $\mathbf{I}$ and $\tilde{\mathbf{I}}$, and calculate $PSNR = 10 \cdot \log_{10}\left(\frac{1}{e^2}\right)$.

In each test we set an error goal ϵ, and fixed the number of bits-per-coefficient Q. For each such pair of parameters, all blocks were coded in order to achieve the desired error goal, and the coefficients were quantized to the desired number of bits (uniform quantization, using upper and lower bounds for each coefficient in each dictionary based on the training set coefficients). For the overcomplete dictionaries, we used the OMP coding method. The rate value was defined as

$$R = \frac{a \cdot \sharp Blocks + \sharp coefs \cdot (b + Q)}{\sharp pixels}, \qquad (6)$$

where a holds the required number of bits to code the number of coefficients for each block. b holds the required number of bits to code the index of the representing atom. Both a and b values were calculated using an entropy coder. $\sharp Blocks$ is the number of blocks in the image (594). $\sharp coefs$ is the total number of coefficients required to represent the whole image, and $\sharp pixels$ is the number of pixels in the image ($= 64 \cdot \sharp Blocks$).

In the unitary DCT dictionary we picked the coefficients in a zig-zag order, as done by JPEG, until the error bound ϵ is reached. Therefore, the index of each atom should not be coded, and the rate was defined accordingly.

By sweeping through various values of ϵ and Q we get per each dictionary several curves in the R-D plane. Figure 4 presents the best obtained R-D curves for each dictionary. As can be seen, the K-SVD dictionary outperforms all other dictionaries, and achieves up to $1 - 2dB$ better for bit rates less than 1.5 bits-per-pixel (where the sparsity model holds true).

5. CONCLUSIONS

In this paper we have presented the K-SVD – an algorithm for designing an overcomplete dictionary that best suits a