K-SVD and its Non-Negative Variant for Dictionary Design

Michal Aharon Michael Elad Alfred M. Bruckstein
Department of Computer Science
Technion—Israel Institute of Technology
Technion City, 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 as 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. Designing dictionaries to better fit the above model can be done by either selecting pre-specified transforms, or by adapting the dictionary to a set of training signals. Both these techniques have been considered in recent years, however this topic is largely still open. In this paper we address the latter problem of designing dictionaries, and introduce the K-SVD algorithm for this task. We show how this algorithm could be interpreted as a generalization of the K-Means clustering process, and demonstrate its behavior in both synthetic tests and in applications on real data. Finally, We turn to describe its generalization to nonnegative matrix factorization problem that suits signals generated under an additive model with positive atoms. we present a simple and yet efficient variation of the K-SVD that handles such extraction of non-negative dictionaries.

1 Introduction

Recent years have witnessed a growing interest in the use of 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 $\mathbf{y} \in \mathbb{R}^n$ ($n \ll K$) 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 $\mathbf{x} \in \mathbb{R}^K$ contains the representation coefficients of the signal $\mathbf{y}$. This sparsest representation is the solution of either

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

or

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

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 order to use overcomplete and sparse representations in applications, one needs 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].

All the above is done with the assumption that the dictionary is given. Designing dictionaries to better fit the above model can be done by either selecting pre-specified transforms, or by adapting the dictionary to a set of training signals. Both these techniques have been considered in recent years, however this topic is largely still open.

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. 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.

A 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 [9]. Subsequent work by Lewicki, Olshausen and Sejnowski, proposed several direct extensions of this work [10]. 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 [11, 12]. They pointed out the connection between the sparse coding dictionary design and the vector quantization problem, and proposed some type of generalization 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. A more thorough summary of the above activity can be found in [13].

In this paper we present a novel algorithm for dictionary training – the K-SVD. We show how this algorithm could be interpreted as a generalization of the K-Means clustering process, and demonstrate its behavior in both synthetic tests and in applications on real data. Finally, We turn to the describe its generalization to nonnegative matrix factorization that suits signals generated under an additive model with positive atoms. we present a simple and yet efficient variation of the K-SVD that handles such extraction of non-negative dictionaries.

2 The K-SVD Algorithm

We now turn to introduce the K-SVD 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 described as a programming task

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

where $\mathbf{e}_k$ is a vector from the standard basis, having all zero entries except one, in the k-th entry, being 1. The *K-Means* algorithm [14] is an iterative method, used for designing the optimal codebook for VQ. At 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

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

A similar objective could alternatively be posed as

$$\min_{\mathbf{D},\mathbf{X}} \sum_i \|\mathbf{x}_i\|_0 \quad \text{subject to} \quad \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 \leq \epsilon, \tag{5}$$

for a fixed value ϵ. In this paper we mainly discuss the first problem (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., [11, 12], 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. We also note that the computational complexity of the K-SVD is equal to the previously reported algorithms for this task [12].

We now describe the process of updating each atom $\mathbf{d}_k$ and its corresponding coefficients, which are located in the k-th row of the coefficient matrix $\mathbf{X}$, denoted as $\mathbf{x}^k$. We first find the matrix of residuals,

$$\mathbf{E}_k = \mathbf{Y} - \left(\mathbf{D}\mathbf{X} - \mathbf{d}_k\mathbf{x}^k\right) \tag{6}$$

and restrict this matrix only to the columns that correspond to the signals that initially use the currently improved atom. Let ω be the set of indices of these signals,

$$\omega = \{i \mid\ 1 \leq i \leq N, \mathbf{x}^k(i) \neq 0\} = \{i \mid 1 \leq i \leq N,\ \mathbf{x}_i(k) \neq 0\}. \tag{7}$$

Similarly, denote $\mathbf{E}_k^{\omega}$ as the restricted residual matrix, which we would now like to approximate using a multiplication of the two updated vectors $\mathbf{d}_k$ and $\mathbf{x}^k$, i.e. we seek for a rank-one approximation. Clearly, this approximation is based on the singular value decomposition (SVD), taking the first left and right singular vectors, together with the first singular value. A full description of the algorithm is given in Figure 2.

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 $i = 1, 2, \ldots, N$

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

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

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

$$\mathbf{E}_k = \mathbf{Y} - \left(\mathbf{D}\mathbf{X} - \mathbf{d}_k \mathbf{x}^k \right),$$

- Restrict $\mathbf{E}_k$ by choosing only the columns corresponding to ω_k, and obtain $\mathbf{E}_k^{\omega_k}$.
- Apply SVD decomposition $\mathbf{E}_k^{\omega_k} = \mathbf{U}\boldsymbol{\Delta}\mathbf{V}^T$. Update: $\mathbf{d}_k = \mathbf{u}_1$, $\mathbf{x}^k = \boldsymbol{\Delta}(1,1) \cdot \mathbf{v}_1$

Set $J = J + 1$.

Figure 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. This algorithm is flexible, and can be combined with any pursuit method. 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. Furthermore, its simplicity enables several variations for this algorithm in order for it to adapt for similar problem, such as the non-negative matrix factorization, as will be shown hereafter.

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.

Just like in K-Means, convergence of the K-SVD to the global minimum solution cannot be guaranteed. However, if we assume that the pursuit method used is guaranteed to succeed (which is not trivial, but can be claimed under some conditions) a reduction of the cost function (4) in each iteration is promised, and therefor, convergence into a local minimum point.

3 Experiments

Synthetic Experiments:

In order to demonstrate the K-SVD, we conducted a number of synthetic tests, in which we randomly chose a dictionary $\mathbf{D} \in R^{20\times 30}$ and multiplied it with randomly chosen sparse coefficient vectors $\{\mathbf{x}_i\}_{i=1}^{1000}$ containing 7 non-zeros each. White noise with varying strength was added to those signals, and the K-SVD was executed on this data for a maximum number of 200 iterations (most tests ended after $100-150$ iterations). The resulting dictionary $\tilde{\mathbf{D}}$ was then compared to the original one, using a similar method to the one reported by [12]. The rate of detected atoms was between 94% to 100%, for SNR levels of 20 dB and beyond. Similar results were reported in [12], but required about 3-5 times more iterations.

Experiments on Real Data:

We did several experiments that involve true image data, trying to show the practicality of the proposed algorithm and the general sparse coding theme. We should emphasize that our tests here come only to demonstrate the concept of using such dictionaries with sparse representations, and further work is required to fully deploy those ideas in actual applications.

First, we used the K-SVD algorithm in order to find a best overcomplete dictionary for a training set containing $11,000$ 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. Therefore, the first dictionary element was set to include a constant value in all its entries, and was not changed afterwards. This element 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 undecimated overcomplete Haar dictionary of the same size. This dictionary has separable basis functions, having steps of various sizes and in all locations. The trained dictionary elements, as also the overcomplte Haar dictionary, are shown in Figure 2. The coefficients were computed using the OMP, where the maximal number of coefficients is $T_0 = 10$. Note that better performance can be obtained by switching to BP or FOCUSS. We concentrated on OMP because of its simplicity and fast execution.
Given the two possible dictionaries described above, we performed two experiments:

- **Filling in missing pixels:** We chose one random full face image, which consists of ~ 600 blocks (all of which were not used for training). On each block, r of the pixels, in random locations, were discarded. The corrupted blocks were decomposed under the learned dictionary and the Haar dictionary using OMP with error bound of 5 gray levels. 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 computed representation defined the reconstructed blocks, after being multiplied in their corresponding dictionary elements.

  The mean reconstruction errors (for all blocks and all corruption rates) were computed, and are displayed in the upper right side of Figure 3. One test image and its reconstruction can be seen

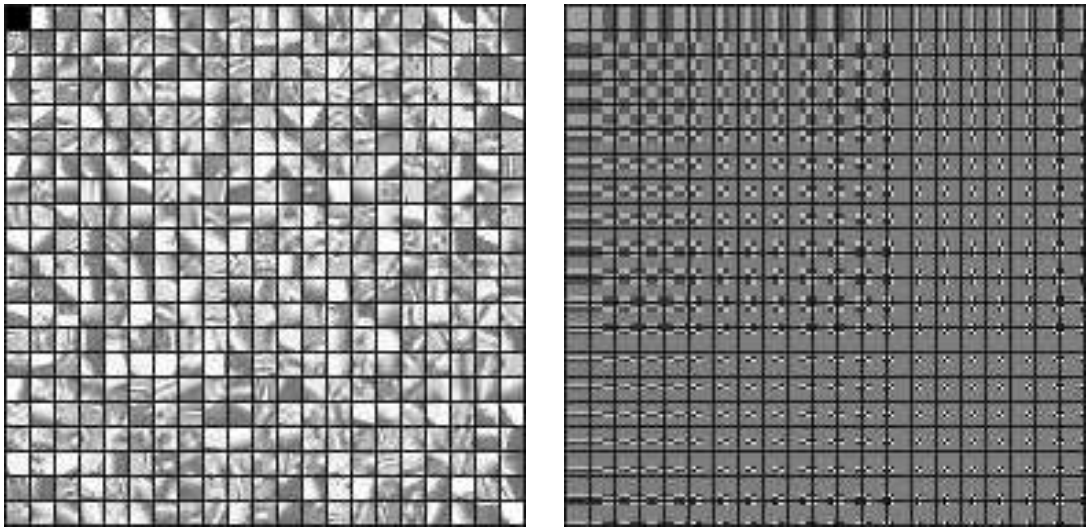

Figure 2: On the left, the K-SVD resulted dictionary. On the right, the overcomplete Haar dictionary.

on the bottom right side, where the left image is the corrupted one, having 50% of its pixels missing, and the middle and right images present the reconstructed images by the learned and Haar dictionaries, respectively. As can be seen, high quality recovery is obtained, and with substantial advantage to the learnt dictionary. Note that in using bigger blocks the performance would have been further improved.

- **Compression:** A compression test was conducted, comparing between the learned dictionary, the overcomplete Haar dictionary, and the complete DCT dictionary, which is being used by the JPEG algorithm. Each 8×8 block was compressed independently, using OMP with an error bound (that depend on the desired SNR value). The compression was measured in bit per pixel (BPP), assuming 10 bits per each coefficient. In the overcomplete dictionaries, elements with arbitrary index were allowed, and therefore, the BPP value was set as $BPP = C(10 + logK)/64$ (where C is the number of required coefficients). In DCT, we used the leading coefficient (as done by JPEG), resulting $BPP = C \cdot 10/64$.

  A summary rate-distortion graph is presented on the upper left side of Figure 3, and a sample compressed image can be seen on the lower left side. We can see that the learned dictionary outperforms the other two alternatives in all compression rates below 2.7 BPP, where sparsity of the representation is still true.

4 Non-Negative Sparse Coding via K-SVD

General:

For some applications, using sparse representations and overcomplete dictionaries together with forcing non-negativity on both the dictionary and the coefficients, may lead to revealing the 'ingredients'

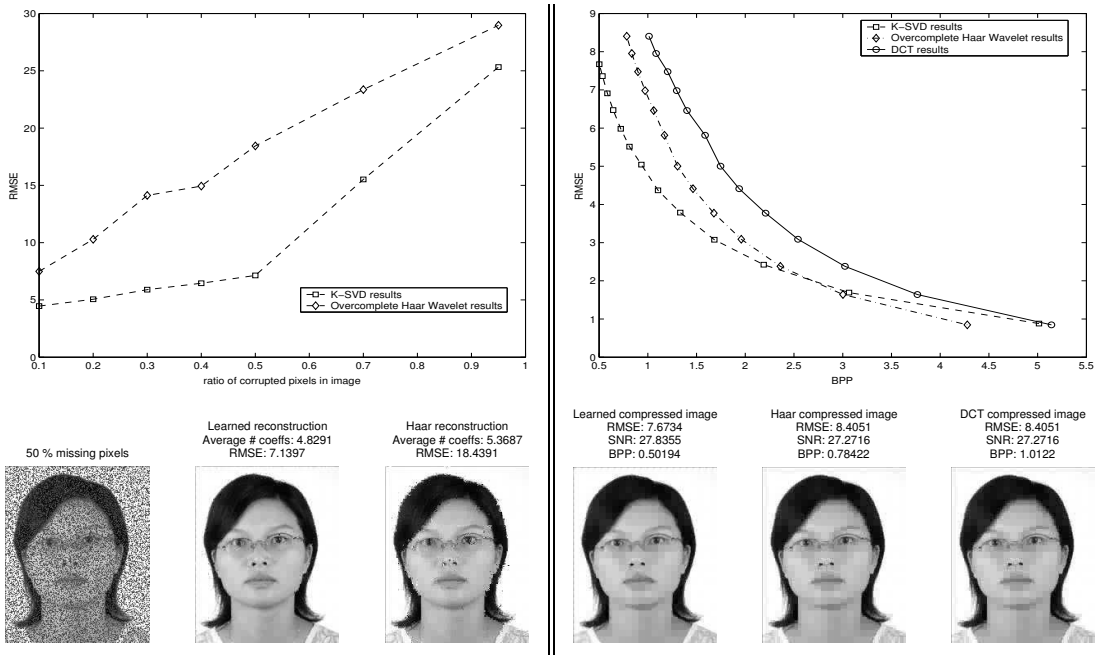

Figure 3: On the right - Filling in results. On top: RMSE versus the relative number of missing pixels, at the bottom: sample images. On the left - Compression test results. On top: results in graph, at the bottom: sample images.

from which all training signals are built of [15, 16]. The inability to subtract values from the linear combination force the dictionary elements to become sparser, and converge to the building blocks of the training signals. This subject is often referred to in literature as '*Non-Negative Matrix Factorization*', or NMF, computing both the dictionary and the coefficient matrices, whose product approximates the signal matrix $\mathbf{Y} \approx \mathbf{DX}$. Application for NMF are many and include dimensionality reduction [17] and analysis of data such as audio [18], text [19] and even data obtained from astronomical spectrometers [20].

Non-negative Decomposition:

Pursuit methods with non-negativity constraints are similar to those presented earlier. A non-negative version of BP minimizes the following convex function [15],

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 + \lambda \sum_i \mathbf{x}_i, \quad subject\ to\ \ \forall_i\, \mathbf{x}_i \geq 0. \tag{8}$$

The non-negative constraint reduces the need for an absolute value over the entries of the coefficient vector $\mathbf{x}$. The derived iterative technique is the following [15],

$$x^{t+1} = \mathbf{x}^t . * (\mathbf{D}^T\mathbf{y}) ./ (\mathbf{D}^T\mathbf{D}\mathbf{x}^t + \lambda), \tag{9}$$

where $.*$ and $./$ represent entry-wise multiplication and division. Kreutz-Delgado and Murray showed a non-negative version of the FOCUSS algorithm [21], to which they called FOCUSS+. They proposed to

project the results after each iteration onto the positive surface, by setting to zero all negative elements. A thorough analysis concerning the linear programming solution of the convex problem,

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{subject to} \quad \mathbf{y} = \mathbf{D}\mathbf{x}, \mathbf{x} \geq 0 \tag{10}$$

was recently given by Donoho and Tanner [22]. They studied the connection between the true sparsest solution and the approximated one derived from the solution of (10). Considering the intrinsic properties of the dictionary $\mathbf{D}$, and in particular, the convex hull of the point–set that contains the columns of $\mathbf{D}$, conclusions regarding the equivalence between the two problems were drawn.

Design of Non-Negative Dictionaries - Prior Art:

A simple method for non-negative matrix factorization, that finds iteratively both the dictionary and the coefficient matrices was introduced by Lee and Seung in [23]. However, this method does not encourage coefficients' sparsity, and therefore is not designed for finding overcomplete dictionaries. In [24] they introduced their algorithm as a method for revealing the parts constructing the training signals, and presented their results working on a set of face images. The corresponding dictionary elements became localized, and each element contained different parts of the face. Hoyer [15, 16] developed an improvement for Lee and Seung's algorithm, by enforcing sparsity constraints, therefore allowing the work with overcomplete dictionaries. He repeated the same tests with similar results.

K-SVD Variation For Non-Negative Dictionaries - NN-K-SVD

In order to adapt the K-SVD for producing non-negative dictionaries (and coefficient matrices) two slight changes should be done. In the sparse coding stage, an adequate pursuit method must be used, forcing non-negative coefficients, as described above. One possibility is to use the Non-negative variation for Focuss [21]. We preferred to use the iterative method presented in [15], also described in Equation (9), which is a variation of BP for non-negative decomposition. Furthermore, we added one change for this method, in order to allow finding a decomposition with a pre-specified number of coefficients, L. After a couple of iterations are done, the indices of the L largest coefficients are selected, and the data is approximated by those element alone, under least squares sense with non-negativity constraint on the coefficients. If we denote $\mathbf{D}^L$ as the sub-matrix that include the L selected elements, we solved

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{D}^L\mathbf{x}\| \;\; s.t. \;\; \mathbf{x} \geq 0. \tag{11}$$

using Matlab's function, which uses the algorithm described in [25].

In the dictionary update stage, we must force the dictionary matrix to stay positive after each atom optimization. Our problem is

$$min_{d_k, x^k} \|\mathbf{E}_r^{\omega_k} - d_k x^k\| \;\; s.t. \;\; d_k, x^k \geq 0. \tag{12}$$

Actually it reduces to finding the best positive rank-one matrix that approximate the error matrix $\mathbf{E}_r^{\omega_k}$ (which might include both positive and negative values). This problem has the same complexity as the original SVD step, but in order to reach a local minima an iterative technique is required. We chose to use the iterative technique described in Figure 4, for $\mathbf{A} = \mathbf{E}_r^{\omega_k}$. The initial solution for this method is

chosen as the SVD solution, truncated to null the negative entries. Note that the first singular vectors can both be multiplied by (-1) without changing the overall rank-one approximation, and therefore both options should be tested and compared. A full description is presented in figure 4

Initialization: Set

$$\mathbf{d}(i) = \begin{cases} 0 & \mathbf{u}_1(i) < 0 \\ \mathbf{u}_1(i) & otherwise \end{cases}, \quad \mathbf{x}(i) = \begin{cases} 0 & \mathbf{v}_1(i) < 0 \\ \mathbf{v}_1(i) & otherwise \end{cases},$$

where $\mathbf{u}_1$ and $\mathbf{v}_1$ are the first singular vectors of $\mathbf{A}$.
Repeat for J times:

1. Set: $\mathbf{d} = \frac{\mathbf{Ax}}{\mathbf{x'x}}$. Project: $\mathbf{d}(i) = \begin{cases} 0 & \mathbf{d}(i) < 0 \\ \mathbf{d}(i) & otherwise \end{cases}$
2. Set: $\mathbf{x} = \frac{\mathbf{d'A}}{\mathbf{d'd}}$. Project: $\mathbf{x}(i) = \begin{cases} 0 & \mathbf{x}(i) < 0 \\ \mathbf{x}(i) & otherwise \end{cases}$

Figure 4: Finding a positive rank one approximation for a matrix $\mathbf{A} = \mathbf{dx}'$

We often found that the true local minima is only slightly different from the initial solution supplied by the SVD projection to the non-negative space, and therefore, we decided to skip the iterative method in cases where the initial solution supply a sufficient reduction of the error. Notice that setting the negative values in the error matrix to zero, and applying SVD, also ensures us positive updated elements, but this produces worse results.

At the end of this iterative procedure, the vector $\mathbf{d}_k$ should be normalized by dividing it by a scalar, as it construct a dictionary element, and $\mathbf{x}^k$ should be multiplied in the same scalar. The full K-SVD variation for non-negative factorization, denoted as *NN-K-SVD* is presented in Figure 5.

Experiment:

The following synthetic experiment was done with the NN-K-SVD. We manually generated 10 dictionary elements of size 8×8, containing the images of the 10 decimal digits. Each digit was then translated by 2 pixels to the right/left, and 1 pixel up/down, to construct 9 possible configurations, resulting with a total of 90 dictionary elements. This dictionary is presented on the upper left side of Figure 6. 3000 training signals were generated as random combinations of 5 such atoms, with random positive coefficients. At first, the tests were conducted without noise, and afterward, a noise level of 15 SNR was added. Those 3000 training signals were given as input to the NN-K-SVD, which resulted with the positive dictionaries presented in the two upper right images (for the two tests) of Figure 6. The NN-K-SVD run was stopped after 60 and 100 iterations respectively. We also used the same data with Hoyer's algorithm [15], which was stopped after 1500 iterations. The resulted dictionaries are presented in the second row of Figure 6. Information about each image is given in the Figure. Note that in this of test the NN-K-SVD had an advantage in using the exact number of coefficients, while Hoyer's algorithm was executed as is with a sparsity factor of 0.8 (see [16]) on the coefficients.

Initialization : Set the non-negative 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 for non-negative decomposition to compute $\mathbf{x}_i$ for $i = 1, 2, \ldots, N$

$$\min_{\mathbf{x}} \left\{ \|\mathbf{y}_i - \mathbf{D}\mathbf{x}\|_2^2 \right\} \quad \text{subject to} \quad \|\mathbf{x}\|_0 \leq T_0 \wedge \forall_i \, \mathbf{x}_i \geq 0.$$

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

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

$$\mathbf{E}_k = \mathbf{Y} - \left(\mathbf{D}\mathbf{X} - \mathbf{d}_k \mathbf{x}^k \right),$$

- Restrict $\mathbf{E}_k$ by choosing only the columns corresponding to ω_k, and obtain $\mathbf{E}_k^{\omega_k}$.
- calculate $\mathbf{d}_k$ and $\mathbf{x}^k$ as described in Figure 4. Normalize $\mathbf{d}_k$.

Set $J = J + 1$.

Figure 5: NN-K-SVD

The NN-K-SVD results for applications on real image data have not been thoroughly examined yet, and we hope to report of those soon. Initial experiments with face images resulted with elements similar in nature to those achieved by Hoyer in [16], although further work is required to tie these to a practical application.

5 Conclusions

In this paper we presented the K-SVD – an algorithm for designing an overcomplete dictionary that best suits a set of given signals, giving a sparse representation per each. We also described a variation of this algorithm in order for it to apply to non-negative matrix factorization problems. We have demonstrated the results of the K-SVD in both synthetic and real images tests.

We believe this kind of dictionary design could successfully replace popular representation methods, used in image enhancement, compression, parts extraction and more. Further work is required to establish such belief.

References

[1] G. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *Journal of Constructive Approximation*, 13:57–98, 1997.

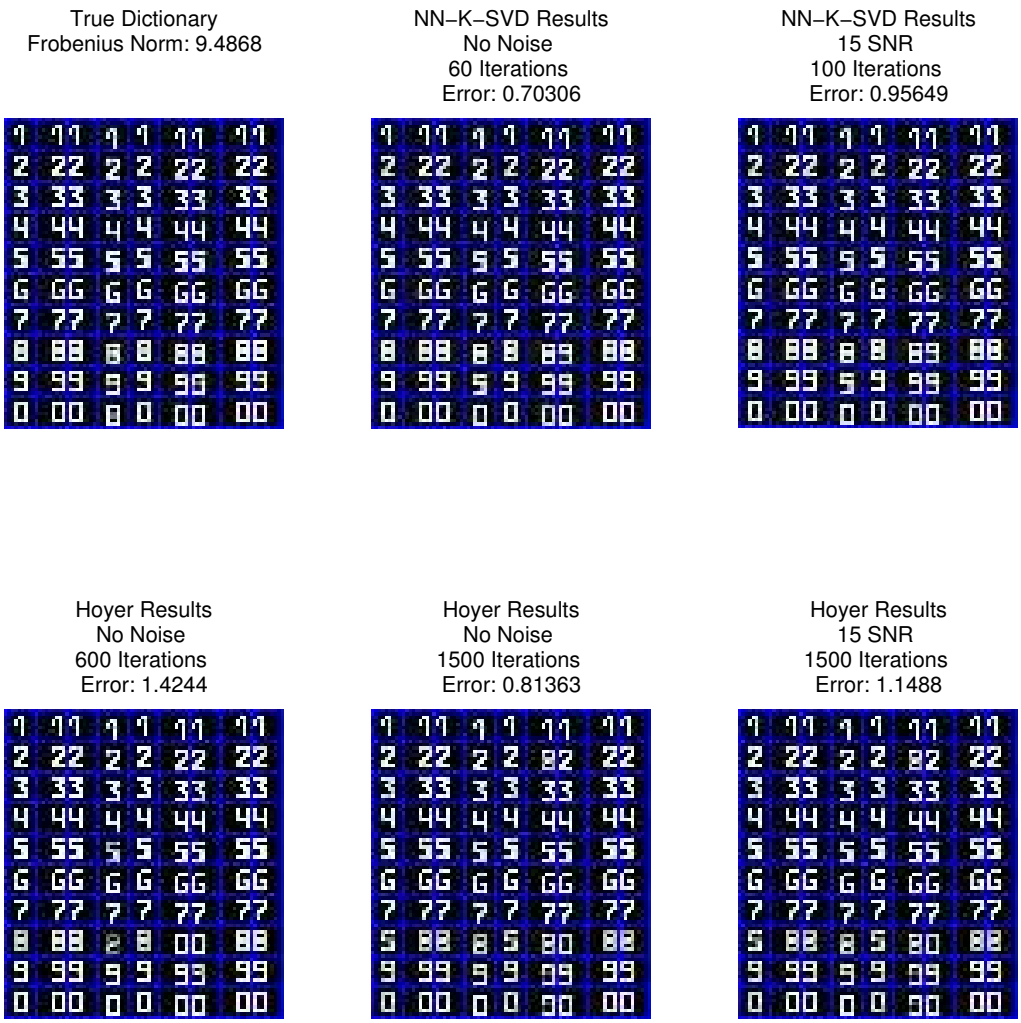

Figure 6: On top, from left to right: True initial dictionary, K-SVD results after 60 iterations in the no-noise test, and after 100 iterations when the noise level was 15 SNR. On the bottom, from left to right: Hoyer's algorithm results in the no-noise case after 600 and after 1500 iterations, and after 1500 iterations in the test with noise level of 15 SNR.

[2] S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Processing.*, 41(12):3397–3415, 1993.

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

[4] S.S. Chen, D.L. Donoho, and M.A. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159, 2001.

[5] B.D. Rao and K. Kreutz-Delgado. An affine scaling methodology for best basis selection. *IEEE Transactions on signal processing*, 47(1):187–200, 1999.

[6] D.L. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition . *IEEE Trans. On Information Theory*, 47(7):2845–62, 1999.

[7] D.L. Donoho and M. Elad. Optimally sparse representation in general (non-orthogonal) dictionaries via l_1 minimization. *PNAS*, 100(5):2197–2202, 2003.

[8] R. Gribonval and M. Nielsen. Sparse decompositions in unions of bases. *IEEE Transactions on Information Theory*, 49(12):3320–3325, 2003.

[9] B.A. Olshausen and D.J. Field. Natural image statistics and efficient coding. *Network: Computation in Neural Systems*, 7(2):333–9, 1996.

[10] M.S. Lewicki and T.J. Sejnowski. Learning overcomplete representations. *Neural Comp.*, 12:337–365, 2000.

[11] K. Engan, S.O. Aase, and J.H. $Hus\phi y$. Multi-frame compression: Theory and design,. *EURASIP Signal Processing*, 80(10):2121–2140, 2000.

[12] K. Kreutz-Delgado, J.F. Murray, B.D. Rao, K. Engan, T. Lee, and T.J. Sejnowski. Dictionary leaning algorithms for sparse representation. *Neural Computation*, 15(2):349–396, 2003.

[13] M. Elad, M. Aharon, and A.M. Bruckstein. K-svd: An algorithm for designining of overcomplete dictionaries for sparse representation. *submitted to IEEE on Signal Proc.*

[14] A. Gersho and R.M. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, Dordrecht, Netherlands, 1992.

[15] P. O. Hoyer. Non-negative sparse coding. *Neural Networks for Signal Processing XII (Proc. IEEE Workshop on Neural Networks for Signal Processing)*, pages 557–565, 2002.

[16] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, pages 1457–1469, 2004.

[17] J. P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov. Metagenes and molecular pattern discovery usingmatrix factorization. *PNAS*, 101(12):4164–4169, March 2004.

[18] P. Smaragdis and J. C. Brown. Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs. *IEEE workshop on application of signal processing to Audio and Acoustics*, pages 177–180, October 2003.

[19] F. Shahnaz, M. Berry, P. Pauca, and R. Plemmons. Document clustering using nonnegative matrix factorization. *Submitted to the Journal on Information Processing and Management*, August 2004.

[20] P. Pauca, J. Piper, and R. Plemmons. Nonnegative matrix factorization for spectral data analysis. May 2005.

[21] J. F. Murray and K. Kreutz-Delgado. Sparse image coding using learned overcomplete dictionaries. *IEEE International Workshop on Machine Learning for Signal Processing*, September 2004.

[22] Sparse nonnegative solution of underdetermined linear equations by linear programming. *Proc Natl Acad Sci U S A.*, 2005.

[23] D.D. Lee and H.S. Seung. Algorithms for non-negative matrix factorization. *Adv. Neural Info. Proc. Syst.*, 13:556–562, 2001.

[24] D. Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Mature*, pages 788–791, 1999.

[25] C.L. Lawson and R.J. Hanson. *Solving Least Squares Problems*. Prentice-Hall, 1974.