Exact Recovery of Sparsely-Used Dictionaries [*]

Daniel A. Spielman[†], Huan Wang[‡], John Wright[§]
Yale University[†‡], Columbia University[§]
U.S.A.
spielman@cs.yale.edu[†], huan.wang@yale.edu[‡], johnwright@ee.columbia.edu[§]

Abstract

We consider the problem of learning sparsely used dictionaries with an arbitrary square dictionary and a random, sparse coefficient matrix. We prove that $O(n \log n)$ samples are sufficient to uniquely determine the coefficient matrix. Based on this proof, we design a polynomial-time algorithm, called Exact Recovery of Sparsely-Used Dictionaries (ER-SpUD), and prove that it probably recovers the dictionary and coefficient matrix when the coefficient matrix is sufficiently sparse. Simulation results show that ER-SpUD reveals the true dictionary as well as the coefficients with probability higher than many state-of-the-art algorithms.

1 Introduction

In the Sparsely-Used Dictionary Learning Problem [M. Aharon and Bruckstein, 2006; Georgiev *et al.*, 2005; Vainsencher *et al.*, 2011], one is given a matrix $\boldsymbol{Y} \in \mathbb{R}^{n \times p}$ and asked to find a pair of matrices $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ and $\boldsymbol{X} \in \mathbb{R}^{m \times p}$ so that $\|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}\|$ is small and so that $\boldsymbol{X}$ is *sparse* – $\boldsymbol{X}$ has only a few nonzero elements. We examine solutions to this problem in which $\boldsymbol{A}$ is a basis, so $m = n$, and without the presence of noise, in which case we insist $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$. Variants of this problem arise in different contexts in machine learning, signal processing, and even computational neuroscience [Olshausen and Field, 1996; Engan *et al.*, 1999; Aharon *et al.*, 2006; Mairal *et al.*, 2009].

In this work [Spielman *et al.*, 2012], we contribute to the understanding of the dictionary learning problem in the case when $\boldsymbol{A}$ is square and nonsingular. We prove that $O(n \log n)$ samples are sufficient to uniquely determine the decomposition with high probability, under the assumption $\boldsymbol{X}$ is generated by a Bernoulli-Gaussian or Bernoulli-Rademacher process.

Our argument for uniqueness suggests a new, efficient dictionary learning algorithm, which we call Exact Recovery of Sparsely-Used Dictionaries (ER-SpUD). This algorithm solves a sequence of linear programs with varying constraints. We prove that under the aforementioned assumptions, the algorithm exactly recovers $\boldsymbol{A}$ and $\boldsymbol{X}$ with high probability. This result holds when the expected number of nonzero elements in each column of $\boldsymbol{X}$ is at most $O(\sqrt{n})$ and the number of samples p is at least $\Omega(n^2 \log^2 n)$. To the best of our knowledge, this result is the first to demonstrate an efficient algorithm for dictionary learning with provable guarantees.

Moreover, we prove that this result is tight to within a $\log$ factor: for the Bernoulli-Gaussian case, when the expected number of nonzeros in each column is $\Omega(\sqrt{n \log n})$, algorithms of this style fail with high probability.

Our algorithm is related to previous proposals by Zibulevsky and Pearlmutter [Zibulevsky and Pearlmutter, 2001] (for source separation) and Gottlieb and Neylon [Gottlieb and Neylon, 2010] (for dictionary learning), but involves several new techniques that seem to be important for obtaining provable correct recovery – in particular, the use of sample vectors in the constraints. Other related recent proposals include [Plumbley, 2007; Jaillet *et al.*, 2010].

2 Notation

We write $\|\boldsymbol{v}\|_p$ for the standard ℓ^p norm of a vector $\boldsymbol{v}$, and we write $\|\boldsymbol{M}\|_p$ for the induced operator norm on a matrix $\boldsymbol{M}$. $\|\boldsymbol{v}\|_0$ denotes the number of non-zero entries in $\boldsymbol{v}$. We denote the Hadamard (point-wise) product by $\odot$. $[n]$ denotes the first n positive integers, $\{1, 2, \ldots, n\}$. For a set of indices I, we let $\boldsymbol{P}_I$ denote the projection matrix onto the subspace of vectors supported on indices I, zeroing out the other coordinates. For a matrix $\boldsymbol{X}$ and a set of indices J, we let $\boldsymbol{X}_J$ ($\boldsymbol{X}^J$) denote the submatrix containing just the rows (columns) indexed by J. We write the standard basis vector that is non-zero in coordinate i as $\boldsymbol{e}_i$. For a matrix $\boldsymbol{X}$ we let $\mathrm{row}(\boldsymbol{X})$ denote the span of its rows. For a set S, $|S|$ is its cardinality.

3 The Probabilistic Models

We analyze the dictionary learning problem under the assumption that $\boldsymbol{A}$ is an arbitrary nonsingular n-by-n matrix, but that $\boldsymbol{X}$ is a random sparse n-by-p matrix with i.i.d. entries. In the Bernoulli(θ)-Gaussian model, the entries of $\boldsymbol{X}$ are independent random variables, each of which has the form $X_{i,j} = \varsigma\tau$, where $\varsigma \sim N(0,1)$ is a standard Gaussian, and τ is 1 with probability θ and 0 with probability $1 - \theta$, independent of ς.

[*] The paper on which this extended abstract is based was the recipient of the best paper award of the 2012 Conference On Learning Theory [Spielman *et al.*, 2012].

We also consider a Bernoulli(θ)-Rademacher model, in which the non-zero entries are chosen uniformly in ± 1.

4 When is the Factorization Unique?

At first glance, it seems the number of samples p required to identify $\boldsymbol{A}$ could be quite large. For example, Aharon *et. al.* view the given data matrix $\boldsymbol{X}$ as having sparse columns, each with at most k nonzero entries. If the given samples $\boldsymbol{y}_j = \boldsymbol{A}\boldsymbol{x}_j$ lie on an arrangement of $\binom{n}{k}$ k-dimensional subspaces $\mathrm{range}(\boldsymbol{A}_I)$, corresponding to possible support sets I, $\boldsymbol{A}$ is identifiable.

On the other hand, the most immediate lower bound on the number of samples required comes from the simple fact that to recover $\boldsymbol{A}$ we need to see at least one linear combination involving each of its columns. The "coupon collection" phenomenon tells us that $p = \Omega(\frac{1}{\theta}\log n)$ samples are required for this to occur with constant probability, where θ is the probability that an element $\boldsymbol{X}_{ij}$ is nonzero. When θ is as small as $O(1/n)$, this means p must be at least proportional to $n \log n$. Our next result shows that, in fact, this lower bound is tight – the problem becomes well-posed once we have observed $cn \log n$ samples.

Theorem 4.1 (Uniqueness) *Under the Bernoulli(θ)-Gaussian and Bernoulli(θ)-Rademacher models, if $1/n \leq \theta \leq 1/C$ and $p > Cn \log n$, then with probability at least $1 - \exp\{-c'p\}$, for any alternative factorization $\boldsymbol{Y} = \boldsymbol{A}'\boldsymbol{X}'$ such that $\max_i \|\boldsymbol{e}_i^T\boldsymbol{X}'\|_0 \leq \max_i \|\boldsymbol{e}_i^T\boldsymbol{X}\|_0$, we have $\boldsymbol{A}' = \boldsymbol{A}\boldsymbol{\Pi}\boldsymbol{\Lambda}$ and $\boldsymbol{X}' = \boldsymbol{\Lambda}^{-1}\boldsymbol{\Pi}^T\boldsymbol{X}$, for some permutation matrix $\boldsymbol{\Pi}$ and nonsingular diagonal matrix $\boldsymbol{\Lambda}$, for some absolute constants C and c'.*

5 Exact Recovery

Theorem 4.1 suggests that we can recover $\boldsymbol{X}$ by looking for sparse vectors in the row space of $\boldsymbol{Y}$. Any vector in this space can be generated by taking a linear combination $\boldsymbol{w}^T\boldsymbol{Y}$ of the rows of $\boldsymbol{Y}$ (here, $\boldsymbol{w}^T$ denotes the vector transpose). We arrive at the optimization problem

$$\text{minimize } \|\boldsymbol{w}^T\boldsymbol{Y}\|_0 \quad \text{subject to} \quad \boldsymbol{w} \neq \boldsymbol{0}.$$

Theorem 4.1 implies that any solution to this problem must satisfy $\boldsymbol{w}^T\boldsymbol{Y} = \lambda\boldsymbol{e}_j^T\boldsymbol{X}$ for some $j \in [n]$, $\lambda \neq 0$. Unfortunately, both the objective and constraint are nonconvex. We therefore replace the ℓ^0 norm with its convex envelope, the ℓ^1 norm, and prevent $\boldsymbol{w}$ from being the zero vector by constraining it to lie in an affine hyperplane $\{\boldsymbol{r}^T\boldsymbol{w} = 1\}$. This gives a linear programming problem of the form

$$\text{minimize } \|\boldsymbol{w}^T\boldsymbol{Y}\|_1 \quad \text{subject to} \quad \boldsymbol{r}^T\boldsymbol{w} = 1. \tag{1}$$

Or equivalently under the change of variables $\boldsymbol{z} = \boldsymbol{A}^T\boldsymbol{w}$, $\boldsymbol{b} = \boldsymbol{A}^{-1}\boldsymbol{r}$,

$$\text{minimize } \|\boldsymbol{z}^T\boldsymbol{X}\|_1 \quad \text{subject to} \quad \boldsymbol{b}^T\boldsymbol{z} = 1. \tag{2}$$

```
ER-SpUD(SC): Exact Recovery of Sparsely-Used Dictionaries using single columns of $\boldsymbol{Y}$ as constraint vectors.
    For $j = 1 \dots p$
        Solve $\min_{\boldsymbol{w}} \|\boldsymbol{w}^T\boldsymbol{Y}\|_1$s.t. $(\boldsymbol{Y}\boldsymbol{e}_j)^T\boldsymbol{w} = 1$, and set $\boldsymbol{s}_j = \boldsymbol{w}^T\boldsymbol{Y}$.
```

5.1 The Algorithms

```
ER-SpUD(DC): Exact Recovery of Sparsely-Used Dictionaries using the sum of two columns of $Y$ as constraint vectors.
  1. Randomly pair the columns of $\boldsymbol{Y}$ into $p/2$ groups $g_j = \{\boldsymbol{Y}\boldsymbol{e}_{j_1}, \boldsymbol{Y}\boldsymbol{e}_{j_2}\}$.
  2. For $j = 1 \dots p/2$
        Let $\boldsymbol{r}_j = \boldsymbol{Y}\boldsymbol{e}_{j_1} + \boldsymbol{Y}\boldsymbol{e}_{j_2}$, where $g_j = \{\boldsymbol{Y}\boldsymbol{e}_{j_1}, \boldsymbol{Y}\boldsymbol{e}_{j_2}\}$.
        Solve $\min_{\boldsymbol{w}} \|\boldsymbol{w}^T\boldsymbol{Y}\|_1$ subject to $\boldsymbol{r}_j^T\boldsymbol{w} = 1$, and set $\boldsymbol{s}_j = \boldsymbol{w}^T\boldsymbol{Y}$.
```

Our algorithms are divided into two stages. In the first stage, we collect many potential rows of $\boldsymbol{X}$ by solving problems of the form (1). In the simpler Algorithm **ER-SpUD(SC)** ("single column"), we do this by using each column of $\boldsymbol{Y}$ as the constraint vector $\boldsymbol{r}$ in the optimization. In the slightly better Algorithm **ER-SpUD(DC)** ("double column"), we pair up all the columns of $\boldsymbol{Y}$ and then substitue the sum of each pair for $\boldsymbol{r}$. In the second stage, we use a greedy algorithm (Algorithm **Greedy**) to select a subset of n of the rows produced. In particular, we choose a linearly independent subset among those with the fewest non-zero elements. From the proof of the uniqueness of the decomposition, we know with high probability that the rows of $\boldsymbol{X}$ are the sparsest n vectors in $\mathrm{row}(\boldsymbol{Y})$. Moreover, for $p \geq \Omega(n \log n)$, Theorems 6.1 and 6.2, along with the coupon collection phenomenon, tell us that a scaled multiple of each of the rows of $\boldsymbol{X}$ is returned by the first phase of our algorithm, with high probability.

```
Greedy: A Greedy Algorithm to Reconstruct $\boldsymbol{X}$ and $\boldsymbol{A}$.
  1. REQUIRE: $\mathcal{S} = \{\boldsymbol{s}_1, \dots, \boldsymbol{s}_T\} \subset \mathbb{R}^p$.
  2. For $i = 1 \dots n$
        REPEAT
            $l \leftarrow \arg\min_{\boldsymbol{s}_l \in \mathcal{S}} \|\boldsymbol{s}_l\|_0$, breaking ties arbitrarily
            $\boldsymbol{x}_i = \boldsymbol{s}_l$
            $\mathcal{S} = \mathcal{S} \backslash \{\boldsymbol{s}_l\}$
        UNTIL rank([$\boldsymbol{x}_1, \dots, \boldsymbol{x}_i$])$= i$
  3. Set $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n]^T$, and $\boldsymbol{A} = \boldsymbol{Y}\boldsymbol{Y}^T(\boldsymbol{X}\boldsymbol{Y}^T)^{-1}$.
```

6 Main Theoretical Results

The intuitive explanations in the previous section can be made rigorous. In particular, under our random models, we can prove that when the number of samples is reasonably large compared to the dimension, (say $p \sim n^2 \log^2 n$), with high probability in $\boldsymbol{X}$ the algorithm will succeed. We conjecture it is possible to decrease the dependency on p to $O(n \log n)$.

Theorem 6.1 (Correct recovery (single-column)) *Suppose $\boldsymbol{X}$ is* $\mathrm{Bernoulli}(\theta)-\mathrm{Gaussian}$. *Then provided $p > c_1 n^2 \log^2 n$, and*

$$\frac{2}{n} \;\le\; \theta \;\le\; \frac{\alpha}{\sqrt{n \log n}}, \qquad (3)$$

*with probability at least $1 - c_f p^{-10}$, the Algorithm **ER-SpUD(SC)** recovers all n rows of $\boldsymbol{X}$. That is, all n rows of $\boldsymbol{X}$ are included in the p potential vectors $\boldsymbol{w}_1^T \boldsymbol{Y}, \ldots, \boldsymbol{w}_p^T \boldsymbol{Y}$. Above, c_1, α and c_f are positive numerical constants.*

The upper bound of $\alpha/\sqrt{n}\log n$ on θ has two sources: an upper bound of $\alpha/\sqrt{n}$ is imposed by the requirement that $\boldsymbol{b}$ be sparse. An additional factor of $\log n$ comes from the need for a gap between $|\boldsymbol{b}|_{(1)}$ and $|\boldsymbol{b}|_{(2)}$ of the k i.i.d. Gaussian random variables. On the other hand, using the sum of two columns of $\boldsymbol{Y}$ as $\boldsymbol{r}$ can save the factor of $\log n$ in the requirement on θ since the "collision" of non-zero entries in the two columns of $\boldsymbol{X}$ creates a larger gap between $|\boldsymbol{b}|_{(1)}$ and $|\boldsymbol{b}|_{(2)}$. More importantly, the resulting algorithm is less dependent on the magnitudes of the nonzero elements in $\boldsymbol{X}$. The algorithm using a single column exploited the fact that there exists a reasonable gap between $|b|_{(1)}$ and $|b|_{(2)}$, whereas the two-column variant **ER-SpUD(DC)** succeeds even if the nonzeros all have the same magnitude.

Theorem 6.2 (Correct recovery (two-column)) *Suppose $\boldsymbol{X}$ is Bernoulli(θ)-Gaussian or Bernoulli(θ)-Rademacher. For some $\alpha > 0$ and for all n larger than some n_0, and $p > c_1 n^2 \log^2 n$, if the probability of non-zero entries θ satisfies*

$$\frac{2}{n} \le \theta \;\le\; \frac{\alpha}{\sqrt{n}}. \qquad (4)$$

*Then with overwhelming probability, the Algorithm **ER-SpUD(DC)** recovers all n rows of $\boldsymbol{X}$. That is, all n rows of $\boldsymbol{X}$ are included in the $p/2$ potential vectors $\boldsymbol{w}_1^T \boldsymbol{Y}, \ldots, \boldsymbol{w}_{p/2}^T \boldsymbol{Y}$.*

Hence, as we choose p to grow faster than $n^2 \log^2 n$, the algorithm will succeed with probability approaching one. That the algorithm succeeds is interesting, perhaps even unexpected. There is potentially a great deal of symmetry in the problem – all of the rows of $\boldsymbol{X}$ might have similar ℓ^1-norm. The vectors $\boldsymbol{r}$ break this symmetry, preferring one particular solution at each step, at least in the regime where $\boldsymbol{X}$ is sparse. To be precise, the expected number of nonzero entries in each column must be bounded by $\sqrt{n \log n}$.

It is natural to wonder whether this is an artifact of the analysis, or whether such a bound is necessary. We can prove that for Algorithm **ER-SpUD(DC)**, the sparsity demands in Theorem 6.2 cannot be improved by more than a factor of $\sqrt{\log n}$. Consider the optimization problem (2). One can show that for each i, $\|\boldsymbol{e}_i^T \boldsymbol{X}\|_1 \approx \theta p$. Hence, if we set $\boldsymbol{z} = \boldsymbol{e}_{j_\star}/b_{j_\star}$, where $j_\star$ is the index of the largest element of $\boldsymbol{b}$ in magnitude, then

$$\|\boldsymbol{z}^T \boldsymbol{X}\|_1 \;=\; \frac{\|\boldsymbol{e}_{j_\star}^T \boldsymbol{X}\|_1}{\|\boldsymbol{b}\|_\infty} \;\approx\; C\frac{\theta p}{\sqrt{\log n}}.$$

If we consider the alternative solution $\boldsymbol{v} = \mathrm{sign}(\boldsymbol{b})/\|\boldsymbol{b}\|_1$, a calculation shows that

$$\|\boldsymbol{v}^T \boldsymbol{X}\|_1 \;\approx\; C' p/\sqrt{n}.$$

Hence, if $\theta > c\sqrt{\log n/n}$ for sufficiently large c, the second solution will have smaller objective function. These calculations are carried through rigorously in the full version, giving:

Theorem 6.3 *For any fixed β and sufficiently large n, and $p \ge C(\beta)n$, the following occurs. If the probability of nonzeros θ satisfies*

$$\theta \ge \sqrt{\frac{\beta \log n}{n}}, \qquad (5)$$

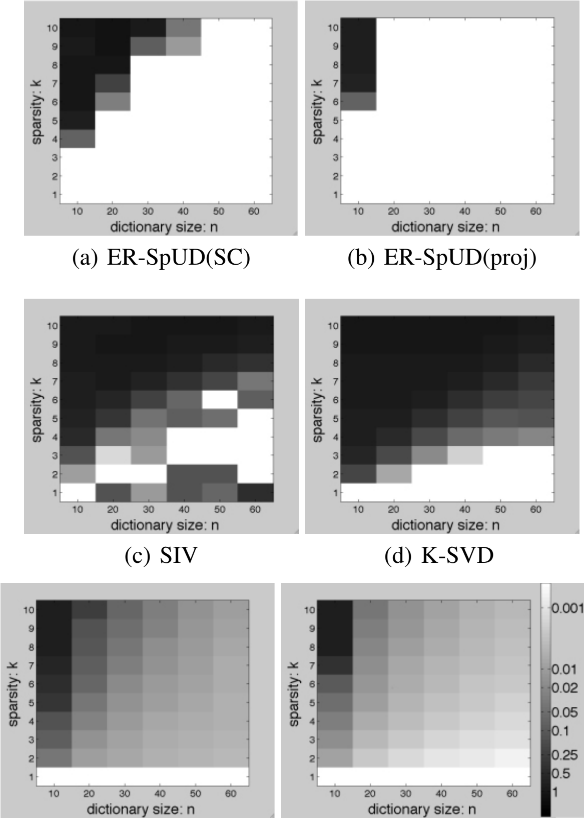
then the probability (in $\boldsymbol{X}$) that solving the optimization problem (1) *with $\boldsymbol{r} = \boldsymbol{Y}\boldsymbol{e}_i$ or $\boldsymbol{r} = \boldsymbol{Y}\boldsymbol{e}_i + \boldsymbol{Y}\boldsymbol{e}_j$ recovers one of the rows of $\boldsymbol{X}$ is at most $n^{-c(\beta)}$, where $c(\beta) > 0$.*

This implies that the result in Theorem 6.1 is nearly the best possible for this algorithm, at least in terms of its demands on θ.

7 Simulations

In this section we systematically evaluate our algorithm, and compare it with the state-of-the-art dictionary learning algorithms, including K-SVD [Aharon *et al.*, 2006], online dictionary learning [Mairal *et al.*, 2009], SIV [Gottlieb and Neylon, 2010], and the relative Newton method for source separation [Zibulevsky, 2003]. The first two methods are not limited to square dictionaries, while the final two methods, like ours, exploit properties of the square case. The method of [Zibulevsky, 2003] is similar in provenance to the incremental nonconvex approach of [Zibulevsky and Pearlmutter, 2001], but seeks to recover all of the rows of $\boldsymbol{X}$ simultaneously, by seeking a local minimum of a larger nonconvex problem. We found in the experiments that a slight variant of the greedy ER-SPUD algorithm, we call the ER-SPUD(proj)[**?**], works even better than the greedy scheme. And thus we also add its result to the comparison list. As our emphasis in this paper is mostly on correctness of the solution, we modify the default settings of these packages to obtain more accurate results (and hence a fairer comparison). For K-SVD, we use high accuracy mode, and switch the number of iterations from 10 to 30. Similarly, for relative Newton, we allow 1,000 iterations. For online dictionary learning, we allow 1,000. We observed diminishing returns beyond these numbers. Since K-SVD and online dictionary learning tend to get stuck at local optimum, for each trial we restart K-SVD and Online learning algorithm 5 times with randomized initializations and report the best performance. We measure accuracy in terms of the relative error, after permutation-scale ambiguity has been removed:

$$\tilde{\mathrm{re}}(\hat{\boldsymbol{A}}, \boldsymbol{A}) \;\doteq\; \min_{\boldsymbol{\Pi},\boldsymbol{\Lambda}} \|\hat{\boldsymbol{A}}\boldsymbol{\Lambda}\boldsymbol{\Pi} - \boldsymbol{A}\|_F / \|\boldsymbol{A}\|_F.$$



(a) ER-SpUD(SC) (b) ER-SpUD(proj)

(c) SIV (d) K-SVD

(e) Online (f) Rel. Newton

Figure 1: Mean relative errors over 10 trials, with varying support k (y-axis, increase from bottom to top) and basis size n(x-axis, increase from left to right). Here, $p = 5n \log_e n$. Our algorithm using a column of Y as r (ER-SpUD(SC)), SIV [Gottlieb and Neylon, 2010], K-SVD [Aharon *et al.*, 2006], online dictionary learning [Mairal *et al.*, 2009], and the relative Newton method for source separation [Zibulevsky, 2003].

Phase transition graph. In our experiments we have chosen $\boldsymbol{A}$ to be a an n-by-n matrix of independent Gaussian random variables. The coefficient matrix $\boldsymbol{X}$ is n-by-p, where $p = 5n \log_e n$. Each column of $\boldsymbol{X}$ has k randomly chosen non-zero entries. In our experiments we have varied n between 10 and 60 and k between 1 and 10. Figure 1 shows the results for each method, with the average relative error reported in greyscale. White means zero error and black is 1. The best performing algorithm is ER-SpUD with iterative projections, which solves almost all the cases except when $n = 10$ and $k \geq 6$. For the other algorithm, When n is small, the relative Newton method appears to be able to handle a denser $\boldsymbol{X}$, while as n grows large, the greedy ER-SpUD is more precise. In fact, empirically the phase transition between success and failure for ER-SpUD is quite sharp – problems below the boundary are solved to high numerical accuracy, while beyond the boundary the algorithm breaks down. In contrast, both online dictionary learning and relative Newton exhibit neither the same accuracy, nor the same sharp transition to failure – even in the black region of the graph, they still return solutions that are not completely wrong. The breakdown boundary of K-SVD is clear compared to online learning and relative Newton. As an active set algorithm, when it reaches a correct solution, the numerical accuracy is quite high. However, in our simulations we observe that both K-SVD and online learning may be trapped into a local optimum even for relatively sparse problems.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation under Grant No. 0915487. JW also acknowledges support from Columbia University.

References

[Aharon *et al.*, 2006] M. Aharon, M. Elad, and A. Bruckstein. The K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[Engan *et al.*, 1999] K. Engan, S. Aase, and J. Hakon-Husoy. Method of optimal directions for frame design. In *ICASSP*, volume 5, pages 2443–2446, 1999.

[Georgiev *et al.*, 2005] P. Georgiev, F. Theis, and A. Cichocki. Sparse component analysis and blind source separation of underdetermined mixtures. *IEEE Transactions on Neural Networks*, 16(4), 2005.

[Gottlieb and Neylon, 2010] L-A. Gottlieb and T. Neylon. Matrix sparsication and the sparse null space problem. *APPROX and RANDOM*, 6302:205–218, 2010.

[Jaillet *et al.*, 2010] F. Jaillet, R. Gribonval, M. Plumbley, and H. Zayyani. An l1 criterion for dictionary learning by subspace identification. In *IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5482–5485, 2010.

[M. Aharon and Bruckstein, 2006] M. Elad M. Aharon and A. Bruckstein. On the uniqueness of overcomplete dictionaries, and a practical way to retrieve them. *Linear Algebra and its Applications*, 416:48–67, 2006.

[Mairal *et al.*, 2009] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 689–696, 2009.

[Olshausen and Field, 1996] B. Olshausen and D. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6538):607–609, 1996.

[Plumbley, 2007] M. Plumbley. Dictionary learning for ℓ^1-exact sparse coding. In *Independent Component Analysis and Signal Separation*, pages 406–413, 2007.

[Spielman *et al.*, 2012] Daniel Spielman, Huan Wang, and John Wright. Exact recovery of sparse-used dictionaries. *25th Annual Conference on Learning Theory*, 23, 2012.

[Vainsencher *et al.*, 2011] D. Vainsencher, S. Mannor, and A. Bruckstein. The sample complexity of dictionary learning. In *Proc. Conference on Learning Theory*, 2011.

[Zibulevsky and Pearlmutter, 2001] M. Zibulevsky and B. Pearlmutter. Blind source separation by sparse decomposition. *Neural Computation*, 13(4), 2001.

[Zibulevsky, 2003] M. Zibulevsky. Blind source separation with relative newton method. *Proceedings ICA*, pages 897–902, 2003.