New Algorithms for Learning Incoherent and Overcomplete Dictionaries

Sanjeev Arora * Rong Ge † Ankur Moitra ‡

August 29, 2013

Abstract

A matrix $A \in \mathbb{R}^{n \times m}$ is said to be μ-incoherent if each pair of columns has inner product at most $\mu/\sqrt{n}$. Starting with the pioneering work of Donoho and Huo [18] such matrices (often called *dictionaries*) have played a central role in signal processing, statistics and machine learning. They allow *sparse recovery*: there are efficient algorithms for representing a given vector as a sparse linear combination of the columns of A (if such a combination exists). However, in many applications ranging from *sparse coding* in machine learning to image denoising, the dictionary is unknown and has to be learned from random examples of the form $Y = AX$ where X is drawn from an appropriate distribution — this is the *dictionary learning* problem. Existing proposed solutions such as the *Method of Optimal Directions* (MOD) [22] or *K-SVD* [2] do not provide any guarantees on their performance nor do they necessarily learn a dictionary for which one can solve sparse recovery problems. The only exception is the recent work of Spielman, Wang and Wright [45] which gives a polynomial time algorithm for dictionary learning when A has *full column rank* (in particular m is at most n). However, in most settings of interest, dictionaries need to be *overcomplete* (i.e., m is larger than n).

Here we give the first polynomial time algorithm for dictionary learning when A is overcomplete. It succeeds under natural conditions on how X is generated, provided that X has at most

$$k \leq c \min(\sqrt{n}/\mu \log n, m^{1/2-\epsilon})$$

non-zero entries (for any $\epsilon > 0$). In other words it can handle almost as many non-zeros as the best sparse recovery algorithms could tolerate *even if one knew the dictionary A exactly.*

*Princeton University, Computer Science Department and Center for Computational Intractability. Email: arora@cs.princeton.edu. This work is supported by the NSF grants CCF-0832797, CCF-1117309, CCF-1302518, DMS-1317308, and Simons Investigator Grant.

†Microsoft Research, New England. Email: rongge@microsoft.com. Part of this work was done while the author was a graduate student at Princeton University and was supported in part by NSF grants CCF-0832797, CCF-1117309, CCF-1302518, DMS-1317308, and Simons Investigator Grant.

‡Massachusetts Institute of Technology, Department of Mathematics and CSAIL. Email: moitra@mit.edu. Part of this work was done while the author was a postdoc at the Institute for Advanced Study and was supported in part by NSF grant No. DMS-0835373 and by an NSF Computing and Innovation Fellowship.

1 Introduction

1.1 Background

A major focus of computational harmonic analysis is finding *sparse* representations for various types of data, e.g. signals, images, natural language, etc [20], [35]. But of course finding such a representation is crucially dependent on having the right dictionary $A \in \mathbb{R}^{n \times m}$, so that each data point can be written as a sparse linear combination of the columns of A. There are many popular choices for this dictionary, such as sinusoids, wavelets, ridgelets, curvelets, etc [35]. There is no universally optimum choice, since the best basis usually depends on the dataset: wavelets for impulsive events, ridgelets for discontinuities in edges, curvelets for smooth curves, etc.. One standard approach is to combine known bases into a single dictionary, which is often called "redundant" or "overcomplete" because m can be much larger than n. Such redundant dictionaries have proven especially effective for example at allowing a sparse representation even if an image contains many different "types" of features such as impulsive events, discontinuities and smooth curves.

There is also evidence that such hand-designed dictionaries do not do as well as dictionaries that are fit to the dataset using automated methods. The problem of discovering the best dictionary to a dataset is called *dictionary learning* and also referred to as *sparse coding* in machine learning. In image processing such discovered dictionaries are used to perform denoising [21], edge-detection [37], super-resolution [49] and compression. In machine learning they are used for feature selection [42] and for building classifiers atop sparse coding routines [31]. Dictionary learning is also a basic building block in design of deep learning systems [43]. See [1], [20] for further applications. In fact, the dictionary learning problem was identified by Olshausen and Field [40] as part of study on internal image representations in the visual cortex. Their work suggested that basis vectors in learnt dictionaries often correspond to well-known image filters such as Gabor filters. Our goal is to design an algorithm for this problem with provable guarantees in the same spirit as recent work on nonnegative matrix factorization [5], topic models [6], [4] and mixtures models [39], [10]. (We will later discuss why current algorithms in [34], [22], [2], [32], [33] do not come with such guarantees.)

Designing provably correct algorithms for dictionary learning has proved challenging. Even if the dictionary is completely known, it can be NP-hard to represent a vector u as a sparse linear combination of the columns of A (even if such a representation exists) [15]. However for many natural types of dictionaries, the problem of finding a sparse representation is computationally easy. The pioneering work of Donoho and Huo [18] (building on the uncertainty principle of Donoho and Stark [19]) presented a number of important examples of dictionaries that are *incoherent* and gave an algorithm to find a sparse representation in a known, incoherent dictionary if one exists. More precisely:

Definition 1.1 (μ-incoherent)**.** An $n \times m$ matrix A whose columns are unit vectors is said to be *μ-incoherent* if for all $i \neq j$ we have

$$\langle A_i, A_j \rangle \leq \mu/\sqrt{n}.$$

We sometimes say just *incoherent* if μ is small, like $\log n$.

A randomly chosen dictionary is incoherent with high probability (even if $m = n^C$). Donoho and Huo [18] gave many other important examples of incoherent dictionaries, such as one constructed from *spikes* and *sines*, as well as those built up from wavelets and sines, or even wavelets and ridgelets. There is a rich body of literature devoted to incoherent dictionaries (see additional references in [24]). Donoho and Huo [18] proved that given $u = Av$ where v has k nonzero entries, where $k \leq \sqrt{n}/2\mu$, *basis pursuit* (solvable by a linear program) recovers v exactly and it is unique. Gilbert, Muthukrishnan and Strauss [24] (and subsequently [47]) gave algorithms for recovering v

even in the presence of additive noise. Tropp [46] gave a more general *exact recovery condition* (ERC) under which the sparse recovery problem for incoherent dictionaries can be algorithmically solved. All of these require $n > k^2\mu^2$. In a foundational work, Candes, Romberg and Tao [11] showed that basis pursuit solves the sparse recovery problem even for $n = O(k \log(m/k))$ if A satisfies the weaker *restricted isometry property* [12]. Also if A is a full-rank square matrix, then we can compute v from $A^{-1}u$, trivially. But our focus here will be on incoherent and overcomplete dictionaries.

The main result in this paper is an algorithm that *provably* learns an unknown, incoherent dictionary from random samples $Y = AX$ where X is a random vector that has at most

$$k \leq c \min(\sqrt{n}/\mu \log n, m^{1/2-\epsilon})$$

non-zero entries (for any $\epsilon > 0$). Hence we can handle almost as many non-zeros as we could tolerate *even if we knew the dictionary* A. The precise conditions on X are described in Section 1.3. We can also relax some of these conditions at the cost of increased running time or requiring X to be more sparse; see Section 6.1.

1.2 Previous Work

There has also been significant prior work on dictionary learning and sparse coding. Lewicki and Sejnowski [34] provided the first approach, and subsequently Engan, Aase and Husoy [22] introduce the *method of optimal directions* (MOD) and Aharon, Elad and Bruckstein [2] introduced *K-SVD*, both of which have had considerable success in practice. For completeness, we will describe these latter two approaches. Suppose the algorithm is given a matrix Y, generated as AX where A is the unknown dictionary and X is an unknown matrix with iid columns.

Method of Optimal Direction [22] : Start with an initial guess A, and then alternately update either A or X:

- Given A, compute a sparse X so that $AX \approx Y$ (using e.g. matching pursuit [36] or basis pursuit [13])
- Given X, compute the A that minimizes $\|AX - Y\|_F$

This algorithm converges to a local optimum, because in each step the error $\|AX - Y\|_F$ will only decrease.

K-SVD [2] Start with an initial guess for A. Then repeat the following procedure:

- Given A, compute a sparse X so that $AX \approx Y$ (again, using a pursuit method)
- Group all data points $Y^{(1)}$ where the corresponding X vector has a non-zero at index i. Subtract off components in the other directions

$$Y^{(1)} - \sum_{j \neq i} A_j X_j^{(1)}$$

- Compute the first singular vector v_1 of the residual matrix, and update the column A_i to v_1

However, these algorithms do not come with provable guarantees. After all, since the first step in the algorithm uses some arbitrary guess for A there may not be *any* sparse representation X so that $AX \approx Y$. Furthermore, even if the algorithm learns some dictionary there is no guarantee that it is incoherent, and consequently no guarantee that one can use it to solve sparse recovery.

Instead, our goal here is to give an algorithm for dictionary learning that has provable guarantees. Assuming the data is generated by an incoherent dictionary, we recover it to within arbitrary accuracy. The most closely related work is an elegant paper of Spielman, Wang and Wright [45], giving an algorithm that provably recovers A to arbitrary accuracy if it is a full column rank matrix, and X satisfies reasonable assumptions. However, requiring A to be full column rank precludes most interesting settings where the dictionary is redundant and hence cannot have full column rank (see e.g. [18], [20], [35]).

1.3 Our Results

First we define the class of distributions that the k-sparse vectors must be drawn from.

Definition 1.2 (Distribution class $\Gamma_{k,C}$)**.** A distribution is in $\Gamma_{k,C}$ if it is suported on k-sparse vectors X such that (i) each component X_i has a $\Theta(k/m)$ probability of being non-zero (ii) the probability that $X_i \neq 0$ is non-positively correlated with other coordinates being non-zero (iii) if X_i is non-zero, its magnitude is in the range $[1, C]$ and its expectation is zero. We use just Γ if k, C are clear from context.

Remark 1.3. The requirement that nonzero X_i's be bounded away from zero in magnitude is similar in spirit to the *Spike-and-Slab Sparse Coding* (S3C) model of Goodfellow et al. [26], which also encourages nonzero latent variables to be bounded away from zero.

Theorem 1.4. *There is a polynomial time algorithm to learn a μ-incoherent dictionary A from random examples of the form $Y = AX$, where X is chosen according to $\Gamma_{k,C}$.*

- *If $k \leq c\min(m^{2/5}, \frac{\sqrt{n}}{\mu \log n})$ and we are given $p \geq \Omega(\max(m^2/k^2 \log m, \frac{m \log m}{\epsilon^2}))$ samples, then the algorithm succeeds with high probability and returns a dictionary $\hat{A}$ so that for each i, $\|A_i - \hat{A}_i\| \leq \epsilon$. The algorithm runs in time $\tilde{O}(p^2 + m^2 p)$.*
- *If (for some integer $\ell > 0$) $k \leq c\min(m^{(\ell-1)/(2\ell-1)}, \frac{\sqrt{n}}{\mu \log n})$, then there is a polynomial time algorithm with the same guarantees and sample complexity that runs in time $\tilde{O}(p^2 + k^{\ell-1}m + m^2 p)$*

Some important remarks about this theorem:

Remark 1.5. The sparsity that this algorithm can tolerate – roughly $\frac{\sqrt{n}}{\mu \log n}$ or $m^{1/2-\epsilon}$ – approaches the sparsity that the best known algorithms require *even if A is known.*

Furthermore since this procedure learns a dictionary that is close to the true dictionary, what it learns must also be incoherent. Hence:

Remark 1.6. This algorithm learns a dictionary for which we can actually solve sparse recovery problems.

Our algorithm is also noise tolerant - e.g. if we are given samples of the form $Y = AX + \eta$ where η has bounded correlation with our samples, then our algorithm will still succeed in recovering A (see Remark 3.6).

We also know of extensions to our algorithm that can weaken various assumptions about Γ with some consequence on the running time or the quality of result, see Section 6.1.

Our approach The main idea is to compute an overlapping clustering. If given many samples, we can find all the samples for which $X_i \neq 0$ then we can recover the column A_i from a singular value decomposition. (Alternatively, we can instead find all the samples for which $X_i > 0$ and then the samples will be biased in the direction of A_i). This is exactly the approach taken in K-SVD [22]. The difference is that in K-SVD, the samples are clustered after running matching pursuit on an intermediate guess for the dictionary. If this guess is not accurate, then the clustering could be far from correct. Our main observation is that for incoherent dictionaries, we can recover the clustering without knowing the dictionary! We build a graph – called the *connection graph*. We give a simple combinatorial algorithm for recovering a provably correct overlapping clustering from this graph, and from this we can immediately recover the true dictionary. In order to prove correctness of our combinatorial algorithm, we rely on tools from discrete geometry, namely the *piercing number* [38], [3]. There is also a connection between our problems and those studied in community detection literature, where recent papers [8], [9] give provable algorithms for finding all of the overlapping communities in a graph. However these algorithms applied to our problem would run in time quasi-polynomial in the sparsity k (see Section 7). Instead, the algorithm we present runs in polynomial time even for large values of k.

We remark that algorithms for independent component analysis [14] can also provably learn an unknown dictionary A from random examples. Suppose we are given random samples of the form AX where the components of X are *exactly* independent. Frieze, Jerrum and Kannan [23] gave an algorithm with provable guarantees that recovers the dictionary A up to arbitrary accuracy, provided that the random variables in the vector X are non-Gaussian (since in that case A is only determined up to rotations anyways). Subsequent work on this problem considers the over-determined case and gives provable algorithms even when A is $n \times m$ with m larger than n [16], [27]. However, these algorithms are very brittle to the assumption that the entries in X be independent; if this condition is even slightly violated, the algorithms can fail. The weakest conditions that we are aware of require that X be 4-wise independent, and typically even higher orders of independence are required for the overcomplete case [16], [27]. Hence these algorithms are not well-suited for dictionary learning where this condition seems too restrictive.

2 Notation

Throughout this paper, we will use $Y^{(i)}$ to denote the i^{th} sample and $X^{(i)}$ as the vector that generated it – i.e. $Y^{(i)} = AX^{(i)}$. Let X_i be the i^{th} coordinate in a vector X and let $\mathrm{supp}(X)$ denote the support of X. Also we will use $\|M\|_F$ to denote the Frobenius norm and $\|M\|_2$ to denote the spectral norm. Moreover we will use Γ to denote the distribution on k-sparse vectors X that is used to generate our samples, and Γ_i will denote the restriction of this distribution to vectors X where $X_i \neq 0$. Lastly, when we are working with a graph G we will use $\Gamma_G(u)$ to denote the set of neighbors of u in G.

3 The Connection Graph

Let A be an unknown $n \times m$ dictionary whose columns are incoherent – i.e. each column is a unit vector and for all $i \neq j$ we have

$$\langle A_i, A_j \rangle \leq \mu/\sqrt{n}$$

Our goal is to recover the dictionary A given sparse linear combinations of its columns. In particular, suppose that there is a distribution Γ on k-sparse columns X of length m. We will assume the following generative model:

- each X has at most k non-zeroes and the probability that $X_i \neq 0$ is $\Theta(k/m)$ and furthermore $X_i \neq 0$ is non-positively correlated with other coordinates being non-zero
- if a coordinate is non-zero, its magnitude is in the range $[1, C]$ and its expectation is zero.

In fact, our analysis will immediately extend to even more general models (see Section 6.1) but for simplicity we encourage the reader to restrict the distribution Γ so that the support of X is uniformly random over all k-tuples and that if a coordinate is non-zero it is a Rademacher random variable (equally likely to be $+1$ or -1) since this will allow us to not have to keep track of as many extraneous constants through the various steps of our analysis.

Our strategy is to form an overlapping clustering where we gather together all the samples Y for which $X_i \neq 0$. We suggest two approaches for how to use this overlapping clustering to determine the dictionary in Section 5.1 and Section 5.2 respectively. The first approach is to further determine which set of samples have $X_i > 0$. This set of samples will be biased in exactly the direction of A_i and hence we can recover this column by taking an average. Alternatively, we can compute the direction of maximum variance. This latter approach recovers (roughly) the direction A_i and yields insight into why approaches like K-SVD [22] are effective in practice.

But how can we compute the correct overlapping clustering? As a starting point: if there are two samples $Y^{(1)} = AX^{(1)}$ and $Y^{(2)} = AX^{(2)}$ then we can determine if the support of $X^{(1)}$ and $X^{(2)}$ intersect, but with false negatives:

Lemma 3.1. *If $k^2\mu < \sqrt{n}/2$ then $|\langle Y^{(1)}, Y^{(2)}\rangle| > 1/2$ implies that he support of $X^{(1)}$ and $X^{(2)}$ intersect*

Proof: Suppose that the support of $X^{(1)}$ and $X^{(2)}$ are disjoint. Then the following upper bound holds:

$$|\langle Y^{(1)}, Y^{(2)}\rangle| \leq \sum_{i \neq j} |\langle A_i, A_j\rangle X_i^{(1)} X_j^{(2)}| \leq k^2\mu/\sqrt{n} < 1/2$$

and this implies the lemma. ■

This would be a weak bound in our setting, since we would require that $k = O(n^{1/4}/\sqrt{\mu})$. Whereas if the dictionary is known, then classic results in compressed sensing show (algorithmically) how to recover a vector that is a linear combination of at most $\sqrt{n}/2\mu$ columns of A [18]. In fact we can impose weaker conditions on k, if we use the randomness of the values in X. We will appeal to the classic Hanson-Wright inequality:

Theorem 3.2 (Hanson-Wright)**.** *[29] Let X be a vector of independent, sub-Gaussian random variables with mean zero and variance one. Let M be a symmetric matrix. Then*

$$Pr[|X^T M X - tr(M)| > t] \leq 2exp\{-c\min(t^2/\|M\|_F^2, t/\|M\|_2)\}$$

Lemma 3.3. *Suppose $k\mu < \frac{\sqrt{n}}{C' \log n}$ for large enough constant C'. Then if the support of $X^{(1)}$ and $X^{(2)}$ are disjoint, with high probability $|\langle Y^{(1)}, Y^{(2)}\rangle| < 1/2$*

Proof: Let N be the $k \times k$ submatrix resulting from restricting $A^T A$ to the locations where $X^{(1)}$ and $X^{(2)}$ are non-zero. Set M to be a $2k \times 2k$ matrix where the $k \times k$ submatrices in the top-left and bottom-right are zero, and the $k \times k$ submatrices in the bottom-left and top-right are $(1/2)N$ and $(1/2)N^T$ respectively. Here we think of the vector X as being a length $2k$ vector whose first k

entries are the non-zero entries in $X^{(1)}$ and whose last k entries are the non-zero entries in $X^{(2)}$. And by construction, we have that

$$\langle Y^{(1)}, Y^{(2)} \rangle = X^T M X$$

We can now appeal to the Hanson-Wright inequality (above). Note that since the support of $X^{(1)}$ and $X^{(2)}$ do not intersect, the entries in M are each at most $\mu/\sqrt{n}$ and so the Frobenius norm of M is at most $\frac{\mu k}{\sqrt{2n}}$. This is also an upper-bound on the spectral norm of M. We can set $t = 1/2$, and for $k\mu < \sqrt{n}/C' \log n$ both terms in the minimum are $\Omega(\log n)$ and this implies the lemma. ■

We can build a graph of which pairs intersect, by connecting a pair of samples $Y^{(1)}$ and $Y^{(2)}$ if $|\langle Y^{(1)}, Y^{(2)} \rangle| > 1/2$.

Definition 3.4. Given p samples $Y^{(1)}, Y^{(2)}, ..., Y^{(p)}$, build a *connection graph* on p nodes where i and j are connected by an edge if and only if $|\langle Y^{(1)}, Y^{(2)} \rangle| > 1/2$.

This graph will "miss" some edges, since if a pair $X^{(1)}$ and $X^{(2)}$ have intersecting support we do not necessarily meet the above condition, but (with high probability) this graph will not have any false positives:

Corollary 3.5. *With high probability, each edge (i, j) that is present in the connection graph corresponds to a pair where the support of $X^{(i)}$ and $X^{(j)}$ intersect.*

It is important to note that a sample $Y^{(1)}$ will intersect many other samples (say, $Y^{(2)}$ and $Y^{(3)}$), each of which corresponds to a coordinate i (reps. i') where the support of $X^{(2)}$ (resp. $X^{(3)}$) intersect with the support of $X^{(1)}$. But the challenge is that we do not know if they intersect $X^{(1)}$ in the same coordinate or not. We will use the other samples as a way to "cross-check" and determine if $X^{(1)}, X^{(2)}$ and $X^{(3)}$ have a common intersection.

Remark 3.6. Even if we are given samples $Y = AX + \eta$ where η is additive noise, under natural conditions on the correlation of η with our samples (e.g. if it is spherical Gaussian noise whose expected norm is at most $o(\sqrt{n})$) this will not affect the construction of the connection graph, and hence will not affect the proof of correctness of our algorithms.

4 Overlapping Clustering

Our goal in this section is to determine which samples Y have $X_i \neq 0$ just from the connection graph. To do this, we will identify a combinatorial condition that allows us to decide whether or not a triple $Y^{(1)}, Y^{(2)}$ and $Y^{(3)}$ has the property that the support of the vectors that generated them – $X^{(1)}, X^{(2)}$ and $X^{(3)}$ respectively – have a common intersection. From this condition, it is straightforward to give an algorithm that correctly groups together all of the samples Y that have $X_i \neq 0$. We will need the following elementary lemma:

Lemma 4.1. *Suppose the support of X intersects the support of each of $X^{(1)}, X^{(2)}, ..., X^{(\ell)}$. Then*

$$Pr[\text{ for all } j = 1, 2, ..., \ell, |\langle Y, Y^{(j)} \rangle| > 1/2] \geq 2^{-\ell}$$

Proof: We have that with high probability that $\text{round}(\langle Y, Y^{(j)} \rangle) = \langle X, X^{(j)} \rangle$ for all j using Lemma 3.3, and hence we need to prove that with probability at least $2^{-\ell}$, each pair $\langle X, X^{(j)} \rangle$ is non-zero. We can prove this by induction, by considering the $X^{(j)}$'s in an order such that no set

$S_j = \text{supp}(X) \cap \text{supp}(X^{(j)})$ is entirely contained in a earlier set S_i (for $i < j$). Then conditioned on all previous inner-products $\langle X, X^{(i)} \rangle$ being non-zero, the probability that $\langle X^{(1)}, X^{(j)} \rangle$ is non-zero is at least 1/2. ■

Now given two samples $Y^{(1)}$ and $Y^{(2)}$ so that the support of $X^{(1)}$ and $X^{(2)}$ intersect uniquely at index i, we will prove that all the samples Y for which $X_i \neq 0$ can be recovered because the expected number of common neighbors between $Y^{(1)}, Y^{(2)}$ and Y will be much larger than in any other case (in particular if $X_i = 0$ instead).

Claim 4.2. *Suppose* $\text{supp}(X^{(1)}) \cap \text{supp}(X^{(2)}) \cap \text{supp}(X^{(3)}) \neq \emptyset$, *then*

$$Pr_Y[\textit{ for all } j = 1, 2, 3, |\langle Y, Y^{(j)} \rangle| > 1/2] \geq \frac{k}{8m}$$

Proof: Let $i \in \textit{supp}(X^{(1)}) \cap \textit{supp}(X^{(2)}) \cap \textit{supp}(X^{(3)})$. Then the probability that $\textit{supp}(X) \ni i$ is exactly k/m, and hence using the previous lemma this implies the claim. ■

Intuitively, if $\text{supp}(X^{(1)}) \cap \text{supp}(X^{(2)}) \cap \text{supp}(X^{(3)}) = \emptyset$ then the expected number of common neighbors of $Y^{(1)}, Y^{(2)}$ and $Y^{(3)}$ should be smaller. But in principle we should be concerned that the support of X could still intersect the support of each of $X^{(1)}, X^{(2)}$ and $X^{(3)}$. Let $a = |\text{supp}(X^{(1)}) \cap \text{supp}(X^{(2)})|$, $b = |\text{supp}(X^{(1)}) \cap \text{supp}(X^{(3)})|$ and $c = |\text{supp}(X^{(2)}) \cap \text{supp}(X^{(3)})|$. Then:

Lemma 4.3. *Suppose that* $\text{supp}(X^{(1)}) \cap \text{supp}(X^{(2)}) \cap \text{supp}(X^{(3)}) = \emptyset$. *Then the probability that the support of* X *intersects the support of each of* $X^{(1)}, X^{(2)}$ *and* $X^{(3)}$ *is at most*

$$\frac{k^6}{m^3} + \frac{3k^3(a+b+c)}{m^2}$$

Proof: We can break up the event whose probability we would like to bound into two (not necessarily disjoint) events: (1) the probability that X intersects each of $X^{(1)}, X^{(2)}$ and $X^{(3)}$ disjointly (i.e. it contains a point $i \in \text{supp}(X^{(1)})$ but $i \notin \text{supp}(X^{(2)}), \text{supp}(X^{(3)})$, and similarly for the other sets). (2) the probability that X contains a point in the common intersection of two of the sets, and one point from the remaining set. Clearly if the support of X intersects the support of each of $X^{(1)}, X^{(2)}$ and $X^{(3)}$ then at least one of these two events must occur.

The probability of the first event is at most the probability that the support of X contains at least one element from each of three disjoint sets of size at most k. The probability that the support of X contains an element of just one such set is at most the expected intersection which is $\frac{k^2}{m}$, and since the expected intersection of X with each of these sets are non-positively correlated (because they are disjoint) we have that the probability of the first event can be bounded by $\frac{k^6}{m^3}$.

Similarly, for the second event: consider the probability that the support of X contains an element in $\text{supp}(X^{(1)}) \cap \text{supp}(X^{(2)})$. Since $\text{supp}(X^{(1)}) \cap \text{supp}(X^{(2)}) \cap \text{supp}(X^{(3)}) = \emptyset$, the support of X must also contain an element in $\text{supp}(X^{(3)})$ too. The expected intersection of X and $\text{supp}(X^{(1)}) \cap \text{supp}(X^{(2)})$ is $\frac{ka}{m}$ and the expected intersection of X and $X^{(3)}$ is $\frac{k^2}{m}$, and again the expectations are non-positively correlated since the two sets $\text{supp}(X^{(1)}) \cap \text{supp}(X^{(2)})$ and $\text{supp}(X^{(3)})$ are disjoint by assumption. Repeating this argument for the other pairs completes the proof of the lemma. ■

Note that the probability that two sets of size k intersect in at least Q elements is at most $(\frac{k}{m})^Q$ by the same non-positive correlation argument. Hence we can assume that with high probability there is *no* pair of sets that intersect in more than Q locations.

Algorithm 1 OverlappingCluster, **Input:** p samples $Y^{(1)}, Y^{(2)}, ..., Y^{(p)}$

1. Compute a graph G on p nodes where there is an edge between i and j iff $|\langle Y^{(1)}, Y^{(2)}\rangle| > 1/2$
2. Initialize list of triples to be empty
3. Set $T = \frac{pk}{10m}$
4. Repeat $\Omega(mk \log^2 m)$ times:
5. Choose a random edge (u, v) in G and a random neighbor w of u
6. If $|\Gamma_G(u) \cap \Gamma_G(v) \cap \Gamma_G(w)| \geq T$
7. Add (u, v, w) to the list of triples
8. For each node x, add x to each set $S_{u,v,w}$ where (u, v, w) is a triple and $|\Gamma_G(u) \cap \Gamma_G(v) \cap \Gamma_G(x)| \geq T$
9. Delete any set $S_{u,v,w}$ that contains another set $S_{a,b,c}$
10. Output the remaining sets $S_{u,v,w} \cup \{u, v, w\}$

And comparing the lemma and the claim above, we find that if $k \leq cm^{2/5}$ then the expected number of common neighbors is much larger if $\text{supp}(X^{(1)}) \cap \text{supp}(X^{(2)}) \cap \text{supp}(X^{(3)}) \neq \emptyset$ than if the intersection is empty. Under this condition, if we take $p = O(m^2/k^2 \log n)$ samples each triple with a common intersection will have at least T common neighbors, and each triple whose intersection is empty will have less than $T/2$ common neighbors.

Hence we can search for a triple with a common intersection as follows: We can find a pair that intersects, since (with high probability) for any pair $|\langle Y^{(1)}, Y^{(2)}\rangle| > 1/2$ the support of $X^{(1)}$ and $X^{(2)}$ intersect. We can try neighbors of $Y^{(1)}$ at random, and we can use the number of common neighbors as a test to verify whether all three sets have a common intersection.

Definition 4.4. We will call a triple of samples $Y^{(1)}, Y^{(2)}$ and $Y^{(3)}$ an *identifying triple* for coordinate i if the support of $X^{(1)}, X^{(2)}$ and $X^{(3)}$ intersect and furthermore the support of $X^{(1)}$ and $X^{(2)}$ is exactly $\{i\}$.

Theorem 4.5. *The algorithm* OverlappingCluster *finds an overlapping clustering where each set corresponds to some i and contains all $Y^{(j)}$ for which $i \in \text{supp}(X^{(j)})$. The algorithm runs in time $\widetilde{O}(p^2 n)$ and succeeds with high probability if $k \leq c \min(m^{2/5}, \frac{\sqrt{n}}{\mu \log n})$ and if $p = \Omega(m^2/k^2 \log m)$*

Proof: We can use Lemma 3.3 to conclude that each edge in G corresponds to a pair whose support intersects. We can appeal to Lemma 4.3 and Claim 4.2 to conclude that for $p = \Omega(m^2/k^2 \log m)$, with high probability each triple with a common intersection has at least T common neighbors, and each triple without a common intersection has at most $T/2$ common neighbors.

In fact, for a random edge (u, v), the probability that the common intersection of u and v (of the supports of their X's) is exactly $\{i\}$ is $\Omega(1/m)$ because we know that their X's do intersect, and that intersection has a constant probability of being size one and it is uniformly distributed over m possible locations. Furthermore the probability that a randomly chosen neighbor w of u contains i (in the support of its X) is at least k/m, and hence appealing to a coupon collector argument we conclude that if the inner loop is run at least $\Omega(km \log^2 m)$ times then the algorithm finds an identifying triple (u, v, w) for each column A_i with high probability.

Note that we may have triples that are not an identifying triple for some coordinate i. However, any other triple (u, v, w) found by the algorithm must have a common intersection. Consider for example a triple (u, v, w) where u and v have a common intersection $\{i, j\}$. Then we know that

Algorithm 2 OverlappingAverage, **Input:** p samples $Y^{(1)}, Y^{(2)}, ...Y^{(p)}$

1. Run OverlappingCluster (or OverlappingCluster2) on the p samples
2. Let $\mathcal{C}_1, \mathcal{C}_2, ...\mathcal{C}_m$ be the m returned overlapping clusters
3. For each $\mathcal{C}_i$
4. For each pair $u, v \in \mathcal{C}_i$ such that $X^{(u)}$ and $X^{(v)}$ have a unique intersection
5. Label the pair $+1$ if $\langle Y^{(u)}, Y^{(v)} \rangle > 0$ and otherwise label it -1.
6. Choose an arbitrary $u_i \in \mathcal{C}_i$, and set $S_i = \{u_i\}$
7. For each $v \in \mathcal{C}_i$
8. If the pair u_i, v is labeled $+1$ add v to S_i
9. Else if there is $w \in \mathcal{C}_i$ where the pairs u_i, w and v, w have the same label, add v to S_i.
10. If $|S_i| \leq |\mathcal{C}_i|/2$ set $S_i = \mathcal{C}_i \backslash S_i$.
11. Let $\hat{A}_i = \sum_{v \in S_i} X^{(v)} / \| \sum_{v \in S_i} X^{(v)} \|_2$
12. Output $\hat{A}$, where each column is $\hat{A}_i$ for some i

there is some other triple (a, b, c) which is an identifying triple for i and hence $S_{a,b,c} \subset S_{u,v,w}$. (In fact this containment is strict, since $S_{u,v,w}$ will also contain a set corresponding to an identifying triple for j too). Then the second-to-last step in the algorithm will necessarily delete all such triples $S_{u,v,w}$. What is the running time of this algorithm? We need $O(p^2 n)$ time to build the connection graph, and the loop takes $\widetilde{O}(mpk)$ time. Finally, the deletion step requires time $\widetilde{O}(m^2 p)$ since there will be $\widetilde{O}(m^2)$ triples found in the previous step. This concludes the proof of correctness of the algorithm, and its running time analysis. ■

5 Recovering the Dictionary

5.1 Finding the Relative Signs

Here we show how to recover the columns of A once we have learned the correct overlapping clustering. The key observation is that if the support of $X_i^{(1)}$ and $X_i^{(2)}$ uniquely intersect in index i then the sign of $\langle Y^{(1)}, Y^{(2)} \rangle$ is equal to the sign of $X_i^{(1)} X_i^{(2)}$. And if there are enough such pairs $X_i^{(1)}$ and $X_i^{(2)}$, then we can correctly determine the relative sign of every pair $X_i^{(1)}$ and $X_i^{(2)}$ in the same cluster $\mathcal{C}_i$. We formalize this idea in the following lemma:

Lemma 5.1. *In Algorithm 2, S_i is either $\{u : X_i^{(u)} > 0\}$ or $\{u : X_i^{(u)} < 0\}$.*

Proof: It suffices to prove the lemma at the start of Step 10, since this step only takes the complement of S_i with respect to $\mathcal{C}_i$. Appealing to Lemma 3.3 we conclude that if the support of $X^{(u)}$ and $X^{(v)}$ uniquely intersect in coordinate i then the sign of $\langle Y^{(u)}, Y^{(v)} \rangle$ is equal to the sign of $X_i^{(u)} X_i^{(v)}$. Hence when Algorithm 2 adds an element to S_i it must have the same sign as the i^{th} component of $X^{(u_i)}$. What remains is to prove that each node $v \in \mathcal{C}_i$ is correctly labeled. We will do this by showing that for any such vertex, there is a length two path of labeled pairs that connects u_i to v, and this is true because the number of labeled pairs is large. We need the following simple claim:

Claim 5.2. *If $p > m^2 \log m/k^2$ then with high probability any two clusters share at most $2pk^2/m^2$ nodes in common.*

This follows since the probability that a node is contained in any fixed pair of clusters is at most k^2/m^2. Then for any node $u \in \mathcal{C}_i$, we would like to lower bound the number of labeled pairs it has in $\mathcal{C}_i$. Since u is in at most $k-1$ other clusters $\mathcal{C}_{i_1}, ..., \mathcal{C}_{i_{k-1}}$, the number of pairs u, v where $v \in \mathcal{C}_i$ that are not labeled for $\mathcal{C}_i$ is at most

$$\sum_{t=1}^{k-1} |\mathcal{C}_{i_t} \cap \mathcal{C}_i| \leq k \cdot 2pk^2/m^2 \ll pk/3m = |\mathcal{C}_i|/3$$

Therefore for a fixed node u for at least a $2/3$ fraction of the other nodes $w \in \mathcal{C}_i$ the pair u, w is labeled. Hence we conclude that for each pair of nodes $u_i, v \in \mathcal{C}_i$ the number of w for which both u_i, w and w, v are labeled is at least $|\mathcal{C}_i|/3 > 0$ and so for every v, there is a labeled path of length two connecting u_i to v. ■

Using this lemma, we are ready to prove Algorithm 2 correctly learns all columns of A.

Theorem 5.3. *The algorithm* OverlappingAverage *outputs a dictionary $\hat{A}$ so that for each i, $\|A_i - \hat{A}_i\| \leq \epsilon$ with high probability if $k \leq c \min(m^{2/5}, \frac{\sqrt{n}}{\mu \log n})$ and if*

$$p = \Omega \max(m^2/k^2 \log m, m \log m/\epsilon^2)$$

Furthermore the algorithm runs in time $O(p^2k^2/m)$.

Proof: We can invoke Lemma 5.1 and conclude that S_i is either $\{u : X_i^{(u)} > 0\}$ or $\{u : X_i^{(u)} < 0\}$, whichever set is larger. Let us suppose that it is the former. Then each $Y^{(u)}$ in S_i is an independent sample from the distribution conditioned on $X_i > 0$, which we call Γ_i^+. We have that $\mathbf{E}_{\Gamma_i^+}[AX] = cA_i$ where c is a constant in $[1, C]$ because $\mathbf{E}_{\Gamma_i^+}[X_j] = 0$ for all $j \neq i$.

Let us compute the variance:

$$\mathop{\mathbf{E}}_{\Gamma_i^+}[\|AX - \mathop{\mathbf{E}}_{\Gamma_i^+} AX\|^2] \leq \mathop{\mathbf{E}}_{\Gamma_i^+} X_i^2 + \sum_{j \neq i} \mathop{\mathbf{E}}_{\Gamma_i^+}[X_j^2] \leq C^2 + \sum_{j \neq i} C^2 k/m \leq C^2(k+1),$$

Note that there are no cross-terms because the signs of each X_j are independent. Furthermore we can bound the norm of each vector $Y^{(u)}$ via incoherence. We can conclude that if $|S_i| > C^2 k \log m/\epsilon^2$, then with high probability $\|\hat{A}_i - A_i\|_2 \leq \epsilon$ using vector Bernstein's inequality ([28], Theorem 12). This latter condition holds because we set S_i to itself or its complement based on which one is larger. ■

5.2 An Approach via SVD

Here we give an alternative algorithm for recovering the dictionary based instead on SVD. The advantage is that approaches such as K-SVD which are quite popular in practice also rely on finding directions of maximum variance, so the analysis we provide here yields insights into why these approaches work. However, we note that the crucial difference is that we rely on finding the correct overlapping clustering in the first step of our dictionary learning algorithms, whereas K-SVD and approaches like it attempt to alternate between updating the dictionary and fixing the overlapping clustering.

Let us fix some notation: Let Γ_i be the distribution conditioned on $X_i \neq 0$. Then once we have found the overlapping clustering, each cluster is a set of random samples from Γ_i. Also let $\alpha = |\langle u, A_i \rangle|$.

Definition 5.4. Let $R_i^2 = 1 + \sum_{j \neq i} \langle A_i, A_j \rangle^2 E_{\Gamma_i}[X_j^2]$.

Note that R_i^2 is the projected variance of Γ_i on the direction $u = A_i$. Our goal is to show that for any $u \neq A_i$ (i.e. $\alpha \neq 1$), the variance is strictly smaller.

Lemma 5.5. *The projected variance of Γ_i on u is at most*

$$\alpha^2 R_i^2 + \alpha\sqrt{(1-\alpha^2)}\frac{2\mu k}{\sqrt{n}} + (1-\alpha^2)(\frac{k}{m} + \frac{\mu k}{\sqrt{n}})$$

Proof: Let $u^{||}$ and $u^{\perp}$ be the components of u in the direction of A_i and perpendicular to A_i. Then we want bound $E_{\Gamma_i}[\langle u, Y \rangle^2]$ where Y is sampled from Γ_i. Since the signs of each X_j are independent, we can write

$$E_{\Gamma_i}[\langle u, Y \rangle^2] = \sum_j E_{\Gamma_i}[\langle u, A_j X_j \rangle^2] = \sum_j E_{\Gamma_i}[\langle u^{||} + u^{\perp}, A_j X_j \rangle^2]$$

Since $\alpha = \|u^{||}\|$ we have:

$$E_{\Gamma_i}[\langle u, Y \rangle^2] = \alpha^2 R_i^2 + E_{\Gamma_i}[\sum_{j \neq i} (2\langle u^{||}, A_j \rangle \langle u^{\perp}, A_j \rangle + \langle u^{\perp}, A_j \rangle^2) X_j^2]$$

Also $E_{\Gamma_i}[X_j^2] = (k-1)/(m-1)$. Let v be the unit vector in the direction $u^{\perp}$. We can write

$$E_{\Gamma_i}[\sum_{j \neq i} \langle u^{\perp}, A_j \rangle^2 X_j^2] = (1-\alpha^2)(\frac{k-1}{m-1}) v^T A_{-i} A_{-i}^T v$$

where A_{-i} denotes the dictionary A with the i^{th} column removed. The maximum over v of $v^T A_{-i} A_{-i}^T v$ is just the largest singular value of $A_{-i} A_{-i}^T$ which is the same as the largest singular value of $A_{-i}^T A_{-i}$ which by the Greshgorin Disk Theorem (see e.g. [30]) is at most $1 + \frac{\mu}{\sqrt{n}} m$. And hence we can bound

$$E_{\Gamma_i}[\sum_{j \neq i} \langle u^{\perp}, A_j \rangle^2 X_j^2] \leq (1-\alpha^2)(\frac{k}{m} + \frac{\mu k}{\sqrt{n}})$$

Also since $|\langle u^{||}, A_j \rangle| = \alpha |\langle A_i, A_j \rangle| \leq \alpha\mu/\sqrt{n}$ we obtain:

$$E[\sum_{j \neq i} 2\langle u^{||}, A_j \rangle \langle u^{\perp}, A_j \rangle X_j^2] \leq \alpha\sqrt{(1-\alpha^2)}\frac{2\mu k}{\sqrt{n}}$$

and this concludes the proof of the lemma. ■

Definition 5.6. Let $\zeta = \max\{\frac{\mu k}{\sqrt{n}}, \sqrt{\frac{k}{m}}\}$, so the expression in Lemma 5.5 can be be an upper bounded by $\alpha^2 R_i^2 + 2\alpha\sqrt{1-\alpha^2} \cdot \zeta + (1-\alpha^2)\zeta^2$.

We will show that an approach based on SVD recovers the true dictionary up to additive accuracy $\pm\zeta$. Note that here ζ is a parameter that converges to zero as the size of the problem increases, but is not a function of the number of samples. So unlike the algorithm in the previous subsection, we cannot make the error in our algorithm arbitrarily small by increasing the number of samples, but this algorithm has the advantage that it succeeds even when $\mathbf{E}[X_i] \neq 0$.

Algorithm 3 OverlappingSVD, **Input:** p samples $Y^{(1)}, Y^{(2)}, ...Y^{(p)}$

1. Run OverlappingCluster (or OverlappingCluster2) on the p samples
2. Let $\mathcal{C}_1, \mathcal{C}_2, ...\mathcal{C}_m$ be the m returned overlapping clusters
3. Compute $\hat{\Sigma}_i = \frac{1}{|\mathcal{C}_i|} \sum_{Y \in \mathcal{C}_i} YY^T$
4. Compute the first singular value $\hat{A}_i$ of $\hat{\Sigma}_i$
5. Output $\hat{A}$, where each column is $\hat{A}_i$ for some i

Corollary 5.7. *The maximum singular value of Γ_i is at least R_i and the direction u satisfies $\|u - A_i\|_2 \leq O(\zeta)$. Furthermore the second largest singular value is bounded by $O(R_i^2 \zeta^2)$.*

Proof: The bound in Lemma 5.5 is only an upper bound, however the direction $\alpha = 1$ has variance $R_i^2 > 1$ and hence the direction of maximum variance must correspond to $\alpha \in [1 - O(\zeta^2), 1]$. Then we can appeal to the variational characterization of singular values (see [30]) that

$$\sigma_2(\Sigma_i) = \max_{u \perp A_i} \frac{u^T \Sigma_i u}{u^T u}$$

Then condition that $\alpha \in [-O(\zeta), O(\zeta)]$ for the second singular value implies the second part of the corollary. ■

Since we have a lower bound on the separation between the first and second singular values of Σ_i, we can apply Wedin's Theorem and show that we can recover A_i approximately even in the presence of noise.

Theorem 5.8 (Wedin)**.** *[48] Let $\delta = \sigma_1(M) - \sigma_2(M)$ and let $M' = M + E$ and furthermore let v_1 and v_1' be the first singular vectors of M and M' respectively. Then*

$$\sin \Theta(v_1, v_1') \leq C \frac{\|E\|}{\delta}$$

Hence even if we do not have access to Σ_i but rather an approximation to it $\hat{\Sigma}_i$ (e.g. an empirical covariance matrix computed from our samples), we can use the above perturbation bound to show that we can still recover a direction that is close to A_i – and in fact converges to A_i as we take more and more samples.

Theorem 5.9. *The algorithm* OverlappingSVD *outputs a dictionary $\hat{A}$ so that for each i, $\|A_i - \hat{A}_i\| \leq \zeta$ with high probability if $k \leq c \min(m^{2/5}, \frac{\sqrt{n}}{\mu \log n})$ and if*

$$p \geq \max(m^2/k^2 \log m, \frac{mn \log m \log n}{\zeta^2})$$

Proof: Appealing to Theorem 4.5, we have that with high probability the call to OverlappingCluster returns the correct overlapping clustering. Then given $\frac{n \log n}{\zeta^2}$ samples from the distribution Γ_i the classic result of Rudelson implies that the computed empirical covariance matrix $\hat{\Sigma}_i$ is close in spectral norm to the true co-variance matrix [44]. This, combined with the separation of the first and second singular values established in Corollary 5.7 and Wedin's Theorem 5.8 imply that we recover each column of A up to an additive accuracy of ϵ and this implies the theorem. Note that since we only need to compute the first singular vector, this can be done via power iteration [25] and hence the bottleneck in the running time is the call to OverlappingCluster. ■

Algorithm 4 OverlappingCluster2, **Input:** p samples $Y^{(1)}, Y^{(2)}, ..., Y^{(p)}$, integer ℓ

1. Compute a graph G on p nodes where there is an edge between i and j iff $|\langle Y^{(1)}, Y^{(2)}\rangle| > 1/2$
2. Initialize list of ℓ-tuples to be empty
3. Set $T = \frac{pk}{Cm2^\ell}$
4. Repeat $\Omega(k^{\ell-2} m \log^2 m)$ times:
5. Choose a random node u in G, and $\ell - 1$ neighbors $u_1, u_2, ...u_{\ell-1}$
6. If $|\Gamma_G(u) \cap \Gamma_G(u_1) \cap ... \cap \Gamma_G(u_{\ell-1})| \geq T$
7. Add $(u, u_1, u_2, ...u_{\ell-1})$ to the list of ℓ-tuples
8. For each node x, add x to each set $S_{u,u_1,u_2,...u_{\ell-1}}$ where $(u_1, u_2, ...u_{\ell-1})$ is an ℓ-tuple and $|\Gamma_G(u) \cap \Gamma_G(u_1) \cap ... \cap \Gamma_G(x)| \geq T$
9. Delete any set $S_{u,u_1,u_2,...u_{\ell-1}}$ that contains another set $S_{v,v_1,v_2,...v_{\ell-1}}$
10. Output the remaining sets $S_{u,u_1,u_2,...u_{\ell-1}} \cup \{u, u_1, u_2, ...u_{\ell-1}\}$

6 A Higher-Order Algorithm

Here we extend the algorithm OverlappingCluster presented in Section 4 to succeed even when $k \leq c \min(m^{1/2-\epsilon}, \sqrt{n}/\mu \log n)$. We can then use the algorithm OverlappingSVD in conjunction with this new clustering algorithm and obtain an analogue of Theorem 5.9 that works even for

$$k \leq c \min(m^{1/2-\epsilon}, \sqrt{n}/\mu \log n)$$

The premise of OverlappingCluster is that we can distinguish whether or not a triple of sets $X^{(1)}, X^{(2)}, X^{(3)}$ has a common intersection based on their number of common neighbors in the connection graph. However for $k = \omega(m^{2/5})$ this is no longer true! But we will instead consider higher-order groups of sets. In particular, for any $\epsilon > 0$ there is an ℓ so that we can distinguish whether an ℓ-tuple of sets $X^{(1)}, X^{(2)}, ..., X^{(\ell)}$ has a common intersection or not based on their number of common neighbors even for $k = \Omega(m^{1/2-\epsilon})$.

The main technical challenge is in showing that if the sets $X^{(1)}, X^{(2)}, ..., X^{(\ell)}$ do not have a common intersection, that we can upper bound the probability that a random set X intersects each of them. To accomplish this, we will need to bound the number of ways of piercing ℓ sets $X^{(1)}, X^{(2)}, ..., X^{(\ell)}$ that have bounded pairwise intersections by at most s points (see Lemma 6.5), and from this an analysis of OverlappingCluster2 will be immediate.

What we need is an analogue of Claim 4.2 and Lemma 4.3. First the easy part:

Claim 6.1. *Suppose* $\text{supp}(X^{(1)}) \cap \text{supp}(X^{(2)}) \cap ... \cap \text{supp}(X^{(\ell)}) \neq \emptyset$, *then*

$$Pr_Y[\text{ for all } j = 1, 2, ..., \ell, |\langle Y, Y^{(j)}\rangle| > 1/2] \geq 2^{-\ell} \frac{k}{m}$$

The proof of this claim is identical to the proof of Claim 4.2. But what about an analogue of Lemma 4.3? To analyze the probability that a set X intersects each of the sets $X^{(1)}, X^{(2)}, ..., X^{(\ell)}$ we will rely on the following standard definition:

Definition 6.2. Given a collection of sets $supp(X^{(1)}), supp(X^{(2)}), ..., supp(X^{(\ell)})$, the *piercing number* is the minimum number of points $p_1, p_2, ..., p_r$ so that each set contains at least one point p_i.

The notion of piercing number is well-studied in combinatorics (see e.g. [38]). However, one is usually interested in upper-bounding the piercing number. For example, a classic result of Alon and Kleitman concerns the (p, q)-problem [3]: Suppose we are given a collection of sets that has the property that each choice of p of them has a subset of q which intersect. Then how large can the piercing number be? Alon and Kleitman proved that the piercing number is at most a fixed constant $c(p, q)$ independent of the number of sets [3].

However, here our interest in piercing number is not in bounding the minimum number of points needed but rather in analyzing how many ways there are of piercing a collection of sets with at most s points, since this will directly yield bounds on the probability that X intersects each of $X^{(1)}, X^{(2)}, ..., X^{(\ell)}$. We will need as a condition that each pair of sets has bounded intersection, and this holds in our model with high-probability.

Claim 6.3. *With high probability, the intersection of any pair* $\text{supp}(X^{(1)}), \text{supp}(X^{(2)})$ *has size at most* Q

Definition 6.4. We will call a set of ℓ sets a (k, Q) family if each set has size at most k and the intersection of each pair of sets has size at most Q.

Lemma 6.5. *The number of ways of piercing* (k, Q) *family (of* ℓ *sets) with* s *points is at most* $(\ell k)^s$. *And crucially if* $\ell \geq s + 1$, *then the number of ways of piercing it with* s *points is at most* $Qs(s + 1) + (\ell k)^{s-1}$.

Proof: The first part of the lemma is the obvious upper bound. Now let us assume $\ell \geq s + 1$: Then given a set of s points that pierce the sets, we can partition the ℓ sets into s sets based on which of the s points is hits the set. (In general, a set may be hit by more than one point, but we can break ties arbitrarily). Let us fix any $s + 1$ of the ℓ sets, and let U be the the union of the pairwise intersections of each of these sets. Then U has size at most $Qs(s + 1)$. Furthermore by the Pidgeon Hole Principle, there must be a pair of these sets that is hit by the same point. Hence one of the s points must belong to the set U, and we can remove this point and appeal to the first part of the lemma (removing any sets that are hit by this point). This concludes the proof of the second part of the lemma, too. ■

Corollary 6.6. *The probability that the support of* X *hits each set in a* (k, Q) *family (of* ℓ *sets) is at most*

$$\sum_{2 \leq s \leq \ell - 1} (C_s + (\ell k)^{s-1}) \Big(\frac{k}{m}\Big)^s + \sum_{s \geq \ell} \Big(\frac{\ell k^2}{m}\Big)^s$$

where C_s *is a constant depending polynomially on* s.

Proof: We can break up the probability of the event that the support of X hits each set in a (k, Q) family into another family of events. Let us consider the probability that X pierces the family with $s \leq \ell - 1$ points or $s \geq \ell$ points. In the former case, we can invoke the second part of Lemma 6.5 and the probability that X hits any particular set of s points is at most $(k/m)^s$. In the latter case, we can invoke the first part of Lemma 6.5. ■

Note that if $k \leq m^{1/2}$ then k/m is always greater than or equal to $k^{s-1}(k/m)^s$. And so asymptotically the largest term in the above sum is $(k^2/m)^\ell$ which we want to be asymptotically smaller than k/m which is the probability in Claim 6.1. So if $k \leq cm^{(\ell-1)/(2\ell-1)}$ then above bound is $o(k/m)$ which is asymptotically smaller than the probability that a given set of ℓ nodes that have a common intersection are each connected to a random (new) node in the connection graph. So

again, we can distinguish between whether or not an ℓ-tuple has a common intersection or not and this immediately yields a new overlapping clustering algorithm that works for k almost as large as $\sqrt{m}$, although the running time depends on how close k is to this bound.

Theorem 6.7. *The algorithm* OverlappingCluster2(ℓ) *finds an overlapping clustering where each set corresponds to some i and contains all $Y^{(j)}$ for which $i \in \mathrm{supp}(X^{(j)})$. The algorithm runs in time $\widetilde{O}(k^{\ell-2}mp + p^2n)$ and succeeds with high probability if $k \leq c\min(m^{(\ell-1)/(2\ell-1)}, \frac{\sqrt{n}}{\mu \log n})$ and if $p = \Omega(m^2/k^2 \log m)$*

Hence this yields a polynomial time algorithm for finding an unknown incoherent dictionary whenever $k \leq c\min(\frac{\sqrt{n}}{\mu \log n}, m^{1/2-\epsilon})$ using Theorem 5.3. The proof of correctness of this algorithm is identical to one of Theorem 5.3 except that the definition of an *identifying ℓ-tuple* for coordinate i is a set of ℓ samples that have a common intersection and for which the first $\ell - 1$ have a common intersection that is exactly $\{i\}$. And note that the probability of finding an identifying ℓ-tuple for coordinate i is at least $\Omega(1/(mk^{\ell-1}))$.

The threshold of $k \leq \sqrt{n}/2\mu$ is a natural barrier: Suppose we are given a vector u of the form $u = Av$ where v has at most k non-zeros. Then if $k \leq \sqrt{n}/2\mu$, the vector v is uniquely defined and is the sparsest solution to the system $u = Ax$ (where x is the variable). However when $k > \sqrt{n}/2\mu$ this is no-longer true and there are incoherent dictionaries where a vector u admits more than one representation as a sparse linear combination of the columns of A. In fact, there are many known algorithms for recovering v from u up to the uniqueness threshold *when the dictionary A is known.* The above algorithm gives a method to recover the dictionary at almost the same threshold – i.e. if $k \leq c\min(\frac{\sqrt{n}}{\mu \log n}, m^{1/2-\epsilon})$.

6.1 Extensions

Here we have studied the dictionary learning problem under what we believe to be quite flexible and general conditions. To underscore this point, let us further examine the conditions we have assumed and we note how these conditions can actually be further relaxed. Our algorithm is based on three steps: constructing the connection graph, finding the overlapping clustering, and recovering the dictionary. However if we invoke Lemma 3.1 (as opposed to Lemma 3.3) then the properties we need of the connection graph follow from each X being at most k sparse for $k \leq n^{1/4}/\sqrt{\mu}$ without any distributional assumptions.

Furthermore, the crucial steps in finding the overlapping clustering are bounds on the probability that a sample X intersects a triple with a common intersection, and the probability that it does so when there is no common intersection (Claim 4.2 and Lemma 4.3). Indeed, these bounds hold whenever the probability of two sets intersecting in two or more locations is smaller (by, say, a factor of k) than the probability of the sets intersecting once. This can be true even if elements in the sets have significant positive correlation (but for the ease of exposition, we have emphasized the simplest models at the expense of generality). Lastly, Algorithm 2 we can instead consider the difference between the averages for S_i and $\mathcal{C}_i \backslash S_i$ and this succeeds even if $\mathbf{E}[X_i]$ is non-zero. This last step does use the condition that the variables X_i are independent, but if we instead use Algorithm 3 we can circumvent this assumption and still recover a dictionary that is close to the true one. Finally, the "bounded away from zero" assumption in Definition 1.2 can be relaxed: the resulting algorithm recovers a dictionary that is close enough to the true one and still allows sparse recovery.

7 Discussion: Overlapping Communities

There is a connection between the approach used here, and the recent work on algorithms for finding overlapping communities (see in particular [8], [9]). We can think of the set of samples Y for which $X_i \neq 0$ as a "community". Then each sample is in more than one community, and indeed for our setting of parameters each sample is contained in k communities. We can think of the main approach of this paper as:

Observation 1. *If we can find all of the overlapping communities, then we can learn an unknown dictionary.*

So how can we find these overlapping communities? The recent papers [8], [9] pose deterministic conditions on what constitutes a community (e.g. each node outside of the community has fewer edges into the community than do other members of the community). These papers provide algorithms for finding all of the communities, provided these conditions are met. However for our setting of parameters, both of these algorithms would run in quasi-polynomial time. For example, the parameter "d" in the paper [8] is an upper-bound on how many communities a node can belong to, and the running time of the algorithms in [8] are quasi-polynomial in this parameter. But in our setting, each sample Y belongs to k communities – one for each non-zero value in X – and the most interesting setting here is when k is polynomially large. Similarly, the parameter "θ" in [9] can be thought of as: If node u is in community c, what is the ratio of the edges incident to u that leave the community c compared to the number that stay inside c? Again, for our purposes this parameter "θ" is roughly k and the algorithms in [9] depend quasi-polynomially on this parameter.

Hence these algorithms would not suffice for our purposes because when applied to learning an unknown dictionary, their running time would depend quasi-polynomially on the sparsity k. In contrast, our algorithms run in polynomial time in all of the parameters, albeit for a more restricted notion of what constitutes a community (but one that seems quite natural from the perspective of dictionary learning). Our algorithm OverlappingCluster finds all of the overlapping "communities" provided that whenever a triple of nodes shares a common community they have many more common neighbors than if they do not all share a single community. The correctness of the algorithm is quite easy to prove, once this condition is met; but here the main work was in showing that our generative model meets these neighborhood conditions.

References

[1] M. Aharon. *Overcomplete Dictionaries for Sparse Representation of Signals.* PhD Thesis, 2006. 1.1

[2] M. Aharon, M. Elad and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. on Signal Processing*, 54(11):4311–4322, 2006. (document), 1.1, 1.2, 1.2

[3] N. Alon and D. Kleitman. Piercing convex sets and the Hadwigder Debrunner (p, q)-problem. *Advances in Mathematics*, 96:103–112, 1992. 1.3, 6

[4] A. Anandkumar, D. Foster, D. Hsu, S. Kakade, Y. Liu. A spectral algorithm for latent dirichlet allocation. In *NIPS*, pp. 926–934, 2012. 1.1

[5] S. Arora, R. Ge, R. Kannan and A. Moitra. Computing a nonnegative matrix factorization – provably In *STOC*, pp. 145–162, 2012. 1.1

[6] S. Arora, R. Ge and A. Moitra. Learning topic models - going beyond SVD. In *FOCS*, pdp. 1–10, 2012. 1.1

[7] S. Arora, R. Ge, A. Moitra and S. Sachdeva. Provable ICA with unknown gaussian noise, and implications for gaussian mixtures and autoencoders. In *NIPS*, pp. 2384–2392, 2012.

[8] S. Arora, R. Ge, S. Sachdeva and G. Schoenebeck. Finding overlapping communities in social networks: Towards a rigorous approach. *EC*, 2012. 1.3, 7, 7

[9] M. Balcan, C. Borgs, M. Braverman, J. Chayes and S-H Teng. Finding endogenously formed communities. *SODA*, 2013. 1.3, 7, 7

[10] M. Belkin and K. Sinha. Polynomial learning of distribution families. *FOCS*, pp. 103–112, 2010. 1.1

[11] E. Candes, J. Romberg and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications of Pure and Applied Math.*, pp. 1207–1223, 2006. 1.1

[12] E. Candes and T. Tao. Decoding by linear programming. *IEEE Trans. on Information Theory*, 51(12):4203–4215, 2005. 1.1

[13] S. Chen, D. Donoho and M. Saunders. Atomic decomposition by basis pursuit. *SIAM J. on Scientific Computing*, 20(1):33–61, 1998. 1.2

[14] P. Comon. Independent component analysis: A new concept? *Signal Processing*, pp. 287–314, 1994. 1.3

[15] G. Davis, S. Mallat and M. Avellaneda. Greedy adaptive approximations. *J. of Constructive Approximation*, 13:57–98, 1997. 1.1

[16] L. De Lathauwer, J Castaing and J. Cardoso. Fourth-order cumulant-based blind identification of underdetermined mixtures. *IEEE Trans. on Signal Processing*, 55(6):2965–2973, 2007. 1.3

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

[18] D. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Trans. on IT*, 47(7):2845–2862, 1999. (document), 1.1, 1.1, 1.2, 3

[19] D. Donoho and P. Stark. Uncertainty principles and signal recovery. *SIAM J. on Appl. Math.*, 49(3):906–931, 1989. 1.1

[20] M. Elad. *Sparse and Redundant Representations*. Springer, 2010. 1.1, 1.2

[21] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. on Signal Processing*, 15(12):3736–3745, 2006. 1.1

[22] K. Engan, S. Aase and J. Hakon-Husoy. Method of optimal directions for frame design. *ICASSP*, 5:2443–2446, 1999. (document), 1.1, 1.2, 1.2, 1.3, 3

[23] A. Frieze, M. Jerrum, R. Kannan. Learning linear transformations. *FOCS*, pp. 359–368, 1996. 1.3

[24] A. Gilbert, S. Muthukrishnan and M. Strauss. Approximation of functions over redundant dictionaries using coherence. *SODA*, 2003. 1.1

[25] G. Golub, C. van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996. 5.2

[26] I. J. Goodfellow, A. Courville, and Y. Bengio. Large-scale feature learning with Spike-and-Slab Sparse Coding. *Proc ICML*, pp 718–726, 2012. 1.3

[27] N. Goyal, S. Vempala and Y. Xiao. Fourier PCA. *arxiv:1306.5825*, 2013. 1.3

[28] D. Gross. Recovering low-rank matrices from few coefficients in any basis. *arxiv:0910.1879*, 2009. 5.1

[29] D. Hanson and F. Wright. A bound on tail probabilities for quadratic forms in independent random variables. *Annals of Math. Stat.*, 42(3):1079–1083, 1971. 3.2

[30] R. Horn and C. Johnson. *Matrix Analysis*. Cambridge University Press, 1990. 5.2, 5.2

[31] K. Kavukcuoglu, M. Ranzato and Y. LeCun. Fast inference in sparse coding algorithms with applications to object recognition. NYU Tech Report, 2008. 1.1

[32] K. Kreutz-Delgado, J. Murray, B. Rao, K. Engan, T. Lee and T. Sejnowski. Dictionary learning algorithms for sparse representation. *Neural Computation*, 2003. 1.1

[33] H. Lee, A. Battle, R. Raina and A. Ng. Efficient sparse coding algorithms. *NIPS*, 2006. 1.1

[34] M. Lewicki and T. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12:337–365, 2000. 1.1, 1.2

[35] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic-Press, 1998. 1.1, 1.2

[36] S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Trans. on Signal Processing*, 41(12):3397–3415, 1993. 1.2

[37] J. Mairal, M. Leordeanu, F. Bach, M. Herbert and J. Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. *ECCV*, 2008. 1.1

[38] J. Matousek. *Lectures on Discrete Geometry*. Springer, 2002. 1.3, 6

[39] A. Moitra and G. Valiant. Setting the polynomial learnability of mixtures of gaussians. *FOCS*, pp. 93–102, 2010. 1.1

[40] B. Olshausen and B. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):331–3325, 1997. 1.1

[41] C. Papadimitriou, P. Raghavan, H. Tamaki and S. Vempala. Latent Semantic Indexing: A Probabilistic Analysis. *JCSS*, pp. 217–235, 2000.

[42] M. Pontil, A. Argyriou and T. Evgeniou. Multi-task feature learning. *NIPS*, 2007. 1.1

[43] M. Ranzato, Y. Boureau and Y. LeCun. Sparse feature learning for deep belief networks *NIPS*, 2007 1.1

[44] M. Rudelson. Random vectors in the isotropic position. *J. of Functional Analysis*, 164:60–72, 1999. 5.2

[45] D. Spielman, H. Wang and J. Wright. Exact recovery of sparsely-used dictionaries. *Journal of Machine Learning Research*, 2012. (document), 1.2

[46] J. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. on IT*, 50(10):2231–2242, 2004. 1.1

[47] J. Tropp, A. Gilbert, S. Muthukrishnan and M. Strauss. Improved sparse approximation over quasi-incoherent dictionaries. *IEEE International Conf. on Image Processing*, 2003. 1.1

[48] P. Wedin. Perturbation bounds in connection with singular value decompositions. *BIT*, 12:99–111, 1972. 5.8

[49] J. Yang, J. Wright, T. Huong and Y. Ma. Image super-resolution as sparse representation of raw image patches. *CVPR*, 2008. 1.1