Sparse Solutions to Nonnegative Linear Systems and Applications

Aditya Bhaskara
Google Research NYC

Ananda Theertha Suresh
University of California, San Diego

Morteza Zadimoghaddam
Google Research NYC

Abstract

We give an efficient algorithm for finding sparse approximate solutions to linear systems of equations with nonnegative coefficients. Unlike most known results for sparse recovery, we do not require *any* assumption on the matrix other than non-negativity. Our algorithm is combinatorial in nature, inspired by techniques for the *set cover* problem, as well as the multiplicative weight update method.

We then present a natural application to learning mixture models in the PAC framework. For learning a mixture of k axis-aligned Gaussians in d dimensions, we give an algorithm that outputs a mixture of $O(k/\epsilon^3)$ Gaussians that is ϵ-close in statistical distance to the true distribution, without any separation assumptions. The time and sample complexity is roughly $O(kd/\epsilon^3)^d$. This is polynomial when d is constant – precisely the regime in which known methods fail to identify the components efficiently.

Given that non-negativity is a natural assumption, we believe that our result may find use in other settings in which we wish to approximately *explain* data using a small number of a (large) candidate set of components.

1 Introduction

Sparse recovery, or the problem of finding sparse solutions (i.e., solutions with a few non-zero entries) to linear systems of equations, is a fundamental problem in signal processing, machine learning and theoretical computer science. In its simplest form, the goal is to find a solution to a given system of equations $Ax = b$ that minimizes $\|x\|_0$ (which we call the *sparsity* of x).

It is known that sparse recovery is NP hard in general. It is related to the question of finding if a set of points in d-dimensional space are in *general position* – i.e., they do not lie in any $(d-1)$ dimensional subspace [Khachiyan, 1995]. A strong negative result in the same vein is due to Arora et al. [1993] and (independently) Amaldi and Kann [1998], who prove that it is not possible to approximate the quantity $\min\{\|x\|_0 : Ax = b\}$ to a factor better than $2^{(\log n)^{1/2}}$ unless NP has quasi polynomial time algorithms.

While these negative results seem forbidding, there are some instances in which sparse recovery is possible. Sparse recovery is a basic problem in the field of compressed sensing, and in a beautiful line of work, Candes et al. [2006], Donoho [2006] and others show that convex relaxations can be used for sparse recovery when the matrix A has certain structural properties, such as *incoherence*, or the so-called restricted isometry property (RIP). However the focus in compressed sensing is to *design* matrices A (with as few rows or 'measurements' as possible) that allow the recovery of sparse vectors x given Ax. Our focus is instead on solving the sparse recovery problem for a *given* A, b, similar to that of [Natarajan, 1995, Donoho and Elad, 2003]. In general, checking if a given A possesses the RIP is a hard problem [Bandeira et al., 2012].

Motivated by the problem of PAC learning mixture models (see below), we consider the sparse recovery problem when the matrix A, the vector b, and the solution we seek all have *non-negative entries*. In this case, we prove that *approximate* sparse recovery is always possible, with some loss in the sparsity. We obtain the following trade-off:

Theorem 1.1. *(Informal) Suppose the matrix A and vector b have non-negative entries, and suppose there exists a k-sparse[1] non-negative x^* such that $Ax^* = b$. Then for any $\epsilon > 0$, there is an efficient algorithm that produces an x_{alg} that is $O(k/\epsilon^3)$ sparse, and satisfies $\|Ax_{alg} - b\|_1 \leq \epsilon \|b\|_1$.*

[1] I.e., has at most k nonzero entries.

Appearing in Proceedings of the 18^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2015, San Diego, CA, USA. JMLR: W&CP volume 38.

The key point is that our upper bound on the error is in the ℓ_1 norm (which is the largest among all ℓ_p norms). Indeed the trade-off between the sparsity of the obtained solution and the error is much better understood if the error is measured in the ℓ_2 norm. In this case, the natural greedy 'coordinate ascent', as well as the algorithm based on sampling from a "dense" solution give non-trivial guarantees (see Natarajan [1995], Shalev-Shwartz et al. [2010]). If the columns of A are normalized to be of unit length, and we seek a solution x with $\|x\|_1 = 1$, one can find an x' such that $\|Ax' - b\|_2 < \epsilon$ and x' has only $O(\frac{\log(1/\epsilon)}{\epsilon^2})$ non-zero coordinates. A similar bound can be obtained for general convex optimization problems, under strong convexity assumptions on the loss function [Shalev-Shwartz et al., 2010].

While these methods are powerful, they do not apply (to the best of our knowledge) when the error is measured in the ℓ_1 norm, as in our applications. More importantly, they do not take advantage of the fact that there *exists* a k-sparse solution (without losing a factor that depends on the largest eigenvalue of $A^\dagger$ as in Natarajan [1995], or without additional RIP style assumptions as in Shalev-Shwartz et al. [2010]).

The second property of our result is that we do not rely on the *uniqueness* of the solution (as is the case with approaches based on convex optimization). Our algorithm is more combinatorial in nature, and is inspired by multiplicative weight update based algorithms for the *set cover* problem, as described in Section 2. Finally, we remark that we do not need to assume that there is an "exact" sparse solution (i.e., $Ax^* = b$), and a weaker condition suffices. See Theorem 2.1 for the formal statement.

Are there natural settings for the sparse recovery problem with non-negative A, b? One application we now describe is that of learning mixture models in the PAC framework [Valiant, 1984, Kearns et al., 1994].

Learning mixture models

A common way to model data in learning applications is to view it as arising from a "mixture model" with a small number of parameters. Finding the parameters often leads to a better understanding of the data. The paradigm has been applied with a lot of success to data in speech, document classification, and so on [Reynolds and Rose, 1995, Titterington et al., 1985, Lindsay, 1995]. Learning algorithms for Gaussian mixtures, hidden Markov models, topic models for documents, etc. have received wide attention both in theory and practice.

In this paper, we consider the problem of learning a mixture of Gaussians. Formally, given samples from a mixture of k Gaussians in d dimensions, the goal is to recover the components with high probability. The problem is extremely well studied, starting with the early heuristic methods such as expectation-maximization (EM). The celebrated result of Dasgupta [1999] gave the first rigorous algorithm to recover mixture components, albeit under a *separation* assumption. This was then improved in several subsequent works (c.f. Arora and Kannan [2001], Vempala and Wang [2002], Dasgupta and Schulman [2007]).

More recently, by a novel use of the classical method of moments, Kalai et al. [2010] and Belkin and Sinha [2010] showed that any d-dimensional Gaussian mixture with a constant number of components k can be recovered in polynomial time (without any strong separation). However the dependence on k in these works is exponential. Moitra and Valiant [2010] showed that this is *necessary* if we wish to recover the *true* components, even in one dimension.

In a rather surprising direction, Hsu and Kakade [2013], and later Bhaskara et al. [2014] and Anderson et al. [2014] showed that if the dimension d is large (at least k^c for a constant $c > 0$), then tensor methods yield polynomial time algorithms for parameter recovery, under mild non-degeneracy assumptions. Thus the case of small d and much larger k seems to be the most challenging for current techniques, if we do not have separation assumptions. Due to the lower bound mentioned above, we cannot hope to recover the true parameters used to generate the samples.

Our parameter setting. We consider the case of constant d, and arbitrary k. As mentioned earlier, this case has sample complexity exponential in k if we wish to recover the *true* components of the mixture (Moitra and Valiant [2010]). We thus consider the corresponding PAC learning question (Valiant [1984]): given parameters $\epsilon, \delta > 0$ and samples from a mixture of Gaussians as above, can we find a mixture of k Gaussians such that the statistical distance to the original mixture is $< \epsilon$ with success probability (over samples) $\geq (1 - \delta)$?

Proper vs improper learning. The question stated above is usually referred to as *proper* learning: given samples from a distribution f in a certain class (in this case a mixture of k Gaussians), we are required to output a distribution $\hat{f}$ in the same class, such that $\left\|f - \hat{f}\right\|_1 \leq \epsilon$. A weaker notion that is often studied is *improper* learning, in which $\hat{f}$ is allowed to be arbitrary (it some contexts, it is referred to as *density estimation*).

Proper learning is often much harder than improper learning. To wit, the best known algorithms for proper

learning of Gaussian mixtures run in time exponential in k. It was first studied by Feldman et al. [2006], who gave an algorithm with sample complexity polynomial in k, d, but run time exponential in k. Later works improved the sample complexity, culminating in the works by Daskalakis and Kamath [2014], Acharya et al. [2014], who gave algorithms with *optimal* sample complexity, for the case of spherical Gaussians. We note that even here, the run times are $\text{poly}(d, 1/\epsilon)^k$.

Meanwhile for improper learning, there are efficient algorithms known for learning mixtures of very general one dimensional distributions (monotone, unimodal, log-concave, and so on). A sequence of works by Chan et al. [2013, 2014] give algorithms that have near-optimal sample complexity (of $\tilde{\mathcal{O}}(k/\epsilon^2)$), and run in polynomial time. However it is not known how well these methods extend to higher dimensions.

In this paper we consider something in between proper and improper learning. We wish to return a mixture of Gaussians, but with one relaxation: we allow the algorithm to output a mixture with slightly more than k components. Specifically, we obtain a tradeoff between the number of components in the output mixture, and the distance to the original mixture. Our theorem here is as follows

Theorem 1.2. *(Informal) Suppose we are given samples from a mixture of k axis-aligned Gaussians in d dimensions. There is an algorithm with running time and sample complexity $O\left(\frac{1}{\epsilon^3} \cdot \left(\frac{kd}{\epsilon^3}\right)^d\right)$, and outputs a mixture of $O(k/\epsilon^3)$ axis-aligned Gaussians which is ϵ-close in statistical distance to the original mixture, with high probability.*

The algorithm is an application of our result on solving linear systems. Intuitively, we consider a matrix whose columns are the probability density functions (p.d.f.) of all possible Gaussians in $\mathbb{R}^d$, and try to write the p.d.f. of the given mixture as a sparse linear combination of these. To obtain finite bounds, we require careful discretization, which is described in Section 3.

Is the trade-off optimal? It is natural to ask if our tradeoff in Theorem 1.1 is the best possible (from the point of view of efficient algorithms). We conjecture that the optimal tradeoff is k/ϵ^2, up to factors of $O(\log(1/\epsilon))$ in general. We can prove a weaker result, that for obtaining an ϵ approximation in the ℓ_1 norm to the general sparse recovery problem using polynomial time algorithms, we cannot always get a sparsity better than $k \log(1/\epsilon)$ unless $\mathcal{P} = \mathcal{NP}$.

While this says that *some* dependence on ϵ is necessary, it is quite far from our algorithmic bound of $O(k/\epsilon^3)$. In Section 4, we will connect this to similar disparities that exist in our understanding of the set cover problem. We present a *random planted* version of the set cover problem, which is beyond all known algorithmic techniques, but for which there are no known complexity lower bounds. We show that unless this planted set cover problem can be solved efficiently, we cannot hope to obtain an ϵ-approximate solution with sparsity $o(k/\epsilon^2)$. This suggests that doing better than k/ϵ^2 requires significantly new algorithmic techniques.

1.1 Basic notation

We will write $\mathbb{R}_+$ for the set of non-negative reals. For a vector x, its ith co-ordinate will be denoted by x_i, and for a matrix A, A_i denotes the ith column of A. For vectors x, y, we write $x \leq y$ to mean entry-wise inequality. We use $[n]$ to denote the set of integers $\{1, 2, \ldots, n\}$. For two distributions p and q, we use $\|p - q\|_1$ to denote the ℓ_1 distance between them.

2 Approximate sparse solutions

2.1 Outline

Our algorithm is inspired by techniques for the well-known *set cover* problem: given a collection of n sets $S_1, S_2, \ldots, S_n \subseteq [m]$, find the sub-collection of the smallest size that covers all the elements of $[m]$. In our problem, if we set A_i to be the indicator vector of the set S_i, and b to be the vector with all entries equal to one, a sparse solution to $Ax = b$ essentially covers all the elements of $[m]$ using only a *few* sets, which is precisely the set cover problem. The difference between the two problems is that in linear equations, we are required to 'cover' all the elements *precisely once* (in order to have equality), and additionally, we are allowed to use sets fractionally.

Motivated by this connection, we define a potential function which captures the notion of covering all the elements "equally". For a vector $x \in \mathbb{R}^n$, we define

$$\Phi(x) := \sum_j b_j (1+\delta)^{(Ax)_j / b_j} \tag{1}$$

This is a mild modification of the potential function used in the multiplicative weight update method (Freund and Schapire [1997], Arora et al. [2012]). Suppose for a moment that $\|b\|_1 = 1$. Now, consider some x with $\|x\|_1 = 1$. If $(Ax)_j = b_j$ for all j, the potential $\Phi(x)$ would be precisely $(1 + \delta)$. On the other hand, if we had $(Ax)_j / b_j$ varying significantly for different j, the potential would (intuitively) be significantly larger; this suggests an algorithm that tries to increment x coordinate-wise, while keeping the potential small. Since we change x coordinate-wise, having a small number of iterations implies sparsity. The key

to the analysis is to prove that at any point in the algorithm, there is a "good" choice of coordinate that we can increment so as to make progress. We now make this intuition formal, and prove the following

Theorem 2.1. *Let A be an $m \times n$ non-negative matrix, and $b \in \mathbb{R}^m$ be a non-negative vector. Suppose there exists a k-sparse non-negative vector x^* such that $\|Ax^*\|_1 = \|b\|_1$ and $Ax^* \leq (1+\epsilon_0)b$, for some $0 < \epsilon_0 < 1/16$. Then for any $\epsilon \geq 16\epsilon_0$, there is an efficient algorithm that produces an x_{alg} that is $O(k/\epsilon^3)$ sparse, and satisfies $\|Ax_{alg} - b\|_1 \leq \epsilon \|b\|_1$.*

Normalization

For the rest of the section, m, n will denote the dimensions of A, as in the statement of Theorem 2.1. Next, note that by scaling all the entries of A, b appropriately, we can assume without loss of generality that $\|b\|_1 = 1$. Furthermore, since for any i, multiplying x_i by c while scaling A_i by $(1/c)$ maintains a solution, we may assume that for all i, we have $\|A_i\|_1 = 1$ (if $A_i = 0$ to start with, we can simply drop that column). Once we make this normalization, since A, b are non-negative, any non-negative solution to $Ax = b$ must also satisfy $\|x\|_1 = 1$.

2.2 Algorithm

We follow the outline above, having a total of $O(k/\epsilon^3)$ iterations. At iteration t, we maintain a solution $x^{(t)}$, obtained by incrementing precisely one co-ordinate of $x^{(t-1)}$. We start with $x^{(0)} = 0$; thus the final solution is $O(k/\epsilon^3)$-sparse.

We will denote $y^{(t)} := Ax^{(t)}$. Apart from the potential Φ introduced above (Eq.(1)), we keep track of another quantity:

$$\psi(x) := \sum_j (Ax)_j.$$

Note that since the entries of A, x are non-negative, this is simply $\|Ax\|_1$.

Running time. Each iteration of the algorithm can be easily implemented in time $O(mn \log(mn)/\delta)$ by going through all the indices, and for each index, checking for a θ in multiples of $(1+\delta)$.

Note that the algorithm increases $\psi(x^{(t)})$ by at least $1/Ck$ in every iteration (because the increase is precisely θ, which is $\geq 1/Ck$), while increasing Φ as slowly as possible. Our next lemma says that once ψ is large enough (while having a good bound on Φ), we can get a "good" solution. I.e., it connects the quantities Φ and ψ to the ℓ_1 approximation we want to obtain.

Lemma 2.2. *Let $x \in \mathbb{R}^n$ satisfy the condition $\Phi(x) \leq (1+\delta)^{(1+\eta)\psi(x)}$, for some $\eta > 0$. Then we have*

$$\left\| \frac{Ax}{\psi(x)} - b \right\|_1 \leq 2\left(\eta + \frac{1}{\delta\psi(x)}\right). \tag{2}$$

Proof. For convenience, let us write $y = Ax$, and $\widetilde{y} = \frac{y}{\|y\|_1}$ (i.e., the normalized version). Note that since each column of A has unit ℓ_1 norm, we have $\psi(x) = \|Ax\|_1 = \|y\|_1$. Since $\widetilde{y}$ and b are both normalized, we have

$$\|\widetilde{y} - b\|_1 = 2 \cdot \sum_{j \,:\, \widetilde{y}_j > b_j} (\widetilde{y}_j - b_j).$$

From now on, we will denote $S := \{j \;:\; \widetilde{y}_j > b_j\}$, and write $p := \sum_{j \in S} b_j$. Thus to prove the lemma, it suffices to show that

$$\sum_{j \in S} (\widetilde{y}_j - b_j) \leq \left(\eta + \frac{1}{\delta\psi(x)}\right). \tag{3}$$

Now, note that the LHS above can be written as $\sum_{j \in S} b_j \left(\frac{\widetilde{y}_j}{b_j} - 1\right)$. We then have

$$\begin{aligned}
(1+\delta)^{\frac{\psi(x)}{p} \cdot \sum_{j \in S} b_j \left(\frac{\widetilde{y}_j}{b_j} - 1\right)} &\leq (1+\delta)^{\sum_{j \in S} \frac{b_j}{p} \cdot \left(\frac{y_j}{b_j} - \psi(x)\right)} \\
&\leq \sum_{j \in S} \frac{b_j}{p} \cdot (1+\delta)^{\left(\frac{y_j}{b_j} - \psi(x)\right)} \quad \text{(convexity)} \\
&\leq \frac{1}{p} \cdot \sum_j b_j (1+\delta)^{\left(\frac{y_j}{b_j} - \psi(x)\right)} \quad \text{(sum over } \textit{all } j\text{)} \\
&\leq \frac{1}{p} \cdot \Phi(x) \cdot (1+\delta)^{-\psi(x)} \\
&\leq \frac{1}{p} \cdot (1+\delta)^{\eta\psi(x)} \quad \text{(hypothesis on } \Phi\text{)}.
\end{aligned}$$

Thus taking logarithms (to base $(1+\delta)$), we can bound the LHS of Eq.(3) by

$$\frac{p}{\psi(x)} \log_{(1+\delta)}(1/p) + p\eta \leq \frac{1}{\delta\psi(x)} + \eta.$$

```
procedure solve({A, b, k, ε})
// find an ε-approximate solution.
begin
    Initialize x^(0) = 0; set parameters T = Ck/δ^2;
    C = 16/ε; δ = ε/16.
    for t = 0, ..., T − 1 do
        Find a coordinate i and a scaling θ > 0 such
        that θ ≥ 1/Ck, and the ratio
        Φ(x^(t) + θe_i)/Φ(x^(t)) is minimized.
        Set x^(t+1) = x^(t) + θe_i.
    end
    Output x_alg = x^(t) / ||x^(t)||_1.
end
```

The last inequalities use the standard facts that $\ln(1+\delta) \geq \delta/2$ (for $\delta < 1$), and $p \leq 1$, and $p\ln(1/p) \leq (1/e)$ for any p. This shows Eq. (3), hence the lemma. □

The next lemma shows that we can always find an index i and a θ as we seek in the algorithm.

Lemma 2.3. *Suppose there exists a k-sparse vector x^* such that $\Phi(x^*) \leq (1+\delta)^{(1+\epsilon_0)}$. Then for any $C > 1$, and any $x^{(t)} \in \mathbb{R}^n$, there exists an index i, and a scalar $\theta \geq 1/(Ck)$, such that*

$$\Phi(x^{(t)} + \theta e_i) \leq (1+\delta)^{\theta(1+\epsilon_0)(1+\delta)/(1-(1/C))}\Phi(x^{(t)}).$$

Proof. x^* is k-sparse, so we may assume w.l.o.g., that $x^* = \theta_1 e_1 + \theta_2 e_2 + \cdots + \theta_k e_k$. Let us define

$$\Delta_i = \Phi(x^{(t)} + \theta_i e_i) - \Phi(x^{(t)}).$$

First, we will show that

$$\sum_{i=1}^{k} \Delta_i \leq \Phi(x^{(t)})\big[(1+\delta)^{(1+\epsilon_0)} - 1\big]. \tag{4}$$

To see this, note that the LHS of Eq.(4) equals

$$\sum_j b_j \left(\sum_{i=1}^{k} (1+\delta)^{(A(x^{(t)}+\theta_i e_i))_j/b_j} - (1+\delta)^{(Ax^{(t)})_j/b_j} \right)$$
$$= \sum_j b_j (1+\delta)^{(Ax^{(t)})_j/b_j} \left(\sum_{i=1}^{k} (1+\delta)^{(A(\theta_i e_i))_j/b_j} - 1 \right)$$
$$\leq \sum_j b_j (1+\delta)^{(Ax^{(t)})_j/b_j} \left((1+\delta)^{(Ax^*)_j/b_j} - 1 \right).$$

In the last step, we used the fact that the function $f(t) := (1+\delta)^t - 1$ is sub-additive (Lemma A.1), and the fact that $x^* = \sum_{i=1}^{k} \theta_i e_i$. Now, using the bound we have on $\Phi(x^*)$, we obtain Eq. (4). Furthermore, since $\|x^*\|_1 = 1$, $\sum_i \theta_i = 1$.

Now we can apply the averaging lemma A.2 with the numbers $\{\Delta_i, \theta_i\}_{i=1}^k$, to conclude that for any $C > 1$, there exists an $i \in [k]$ such that $\theta_i \geq 1/(Ck)$, and

$$\Delta_i \leq \Phi(x^{(t)}) \cdot \frac{(1+\delta)^{(1+\epsilon_0)} - 1}{1 - (1/C)}.$$

Thus we have that for this choice of i, and $\theta = \theta_i$,

$$\Phi(x^{(t)} + \theta e_i) \leq \Phi(x^{(t)}) \left(1 + \theta \cdot \frac{(1+\delta)^{(1+\epsilon_0)} - 1}{1 - (1/C)} \right).$$

Now we can simplify the term in the parenthesis using Lemma A.3 (twice) to obtain

$$1 + \theta \cdot \frac{(1+\delta)^{(1+\epsilon_0)} - 1}{1 - (1/C)} \leq 1 + \frac{\theta \cdot \delta(1+\epsilon_0)}{1 - (1/C)}$$
$$\leq (1+\delta)^{\theta(1+\epsilon_0)(1+\delta)/(1-(1/C))}.$$

This completes the proof of the lemma. □

Proof of Theorem 2.1. By hypothesis, we know that there exists an x^* such that $\|Ax^*\|_1 = 1$ (or equivalently $\|x^*\|_1 = 1$) and $Ax^* \leq (1+x^*)b$. Thus for this x^*, we have $\Phi(x^*) \leq (1+\delta)^{(1+\epsilon_0)}$, so Lemma 2.3 shows that in each iteration, the algorithm succeeds in finding an index i and $\theta > 1/Ck$ satisfying the conclusion of the lemma. Thus after T steps, we end up with $\psi(x^{(T)}) \geq 1/\delta^2$, thus we can appeal to Lemma 2.2. Setting $\eta := 2(\epsilon_0 + \delta + 1/C)$, and observing that

$$(1+\epsilon_0)(1+\delta)/(1-1/C) < 1 + \eta,$$

the lemma implies that the ℓ_1 error is $\leq 2(\eta + \delta) < \epsilon$, from our choice of η, δ. This completes the proof. □

Remark 2.4. *The algorithm above finds a column to add by going through indices $i \in [m]$, and checking if there is a scaling of A_i that can be added. But in fact, any procedure that allows us to find a column with a small value of $\Phi(x^{(t+1)})/\Phi(x^{(t)})$ would suffice for the algorithm. For example, the columns could be parametrized by a continuous variable, and we may have a procedure that only searches over a discretization.*[2] *We could also have an optimization algorithm that outputs the column to add.*

3 Learning Gaussian mixtures

3.1 Notation

Let $N(\mu, \sigma^2)$ denote the density of a d-dimensional axis-aligned Gaussian distribution with mean μ and diagonal covariance matrix σ^2 respectively. Thus a k-component Gaussian mixture has the density $\sum_{r=1}^{k} w_r N(\mu_r, \sigma_r^2)$. We use f to denote the underlying mixture and p_r to denote component r. We use $(\hat{\ })$ to denote empirical or other estimates; the usage becomes clear in context. For an interval I, let $|I|$ denote its length. For a set S, let $n(S)$ be the number of samples in that set.

3.2 Algorithm

The problem for finding components of a k-component Gaussian mixture f can be viewed as finding a sparse solution for system of equations

$$Aw = f, \tag{5}$$

where columns of A are the possible mixture components and w is the weight vector and f is the density of the underlying mixture. If f is known exactly, and A is known explicitly, (5) can be solved using $\mathsf{solve}(\{A, f, k, \epsilon\})$.

[2] This is a fact we will use in our result on learning mixtures of Gaussians.

However, a direct application of solve has two main issues. Firstly, f takes values over $\mathbb{R}^d$ and thus is an infinite dimensional vector. Thus a direct application of solve is not computationally feasible. Secondly f is unknown and has to be estimated using samples. Also, for algorithm solve's performance guarantees to hold, we need an estimate $\hat{f}$ such that $\hat{f}(x) \geq f(x)(1-\epsilon)$, for all x. This kind of a global multiplicative condition is difficult to satisfy for continuous distributions. To avoid these issues, we carefully discretize the mixture of Gaussians. More specifically, we partition $\mathbb{R}^d$ into rectangular regions $\mathcal{S} = \{S_1, S_2, \ldots\}$ such that $S_i \cap S_j = \emptyset$ and $\cup_{S\in\mathcal{S}} S = \mathbb{R}^d$. Furthermore we flatten the Gaussian within each region to induce a new distribution over $\mathbb{R}^d$ as follows:

Definition 3.1. *For a distribution p and a partition $\mathcal{S}$, the new distribution $p^{\mathcal{S}}$ is defined as follows.*[3] *If $x, y \in S$ for some $S \in \mathcal{S}$, then $p^{\mathcal{S}}(x) = p^{\mathcal{S}}(y)\ \forall S \in \mathcal{S}$, $p(S) = p^{\mathcal{S}}(S)$.*

Note that we use the standard notation that $p(S)$ denotes the total probability mass of the distribution p over the region S. Now, let $A^{\mathcal{S}}$ be a matrix with rows indexed by $S \in \mathcal{S}$ and columns indexed by distributions p such that $A^{\mathcal{S}}(S, p) = p(S)$. $A^{\mathcal{S}}$ is a matrix with potentially infinitely many columns, but finitely many rows (number of regions in our partition).

Using samples, we generate a partition of $\mathbb{R}^d$ such that the following properties hold. (a) $f^{\mathcal{S}}(S)$ can be estimated to sufficient multiplicative accuracy for each set $S \in \mathcal{S}$. (b) If we output a mixture of $\mathcal{O}(k/\epsilon^3)$ Gaussians $A^{\mathcal{S}}w'$ such that $\sum_{S\in\mathcal{S}} |(A^{\mathcal{S}}w')(S) - f^{\mathcal{S}}(S)|$ is small, then $\|Aw' - f\|_1$ is also small.

For the first one to hold, we require the sets to have large probabilities and hence requires $\mathcal{S}$ to be a coarse partition of $\mathbb{R}^d$. The second condition requires the partition to be 'fine enough', that a solution after partitioning can be used to produce a solution for the corresponding continuous distributions. How do we construct such a partition?

If all the Gaussian components have similar variances and the means are not too far apart, then a rectangular grid with carefully chosen width would suffice for this purpose. However, since we make no assumptions on the variances, we use a sample-dependent partition (i.e., use some samples from the mixture to get a rough estimate for the 'location' of the probability mass). To formalize this, we need a few more definitions.

Definition 3.2. *A partition of a real line is given by $\mathcal{I} = \{I_1, I_2, \ldots\}$ where I_ts are continuous intervals, $I_t \cap I_{t'} = \emptyset\ \forall t, t'$, and $\cup_{I\in\mathcal{I}} I = \mathbb{R}$.*

[3]We are slightly abusing notation, with $p^{\mathcal{S}}$ denoting both the p.d.f. and the distribution itself.

Since we have d dimensions, we have d such partitions. We denote by $\mathcal{I}_i$ the partition of axis i. t^{th} of coordinate i is denoted by $I_{i,t}$.

For ease of notation, we use subscript r to denote components (of the mixture), i to denote coordinates ($1 \leq i \leq d$), and t to denote the interval indices corresponding to coordinates. We now define induced partition based on intervals and a notion of "good" distributions.

Definition 3.3. *Given partitions $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \ldots \mathcal{I}_d$ for coordinates 1 to d, define $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \ldots \mathcal{I}_d$-induced partition $\mathcal{S} = \{S_v\}$ as follows: for every d-tuple v, $x \in S_v$ iff $x_i \in I_{i,v_i}\ \forall v$.*

Definition 3.4. *A product distribution $p = p_1 \times p_2 \times \ldots \times p_d$ is $(\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \ldots \mathcal{I}_d), \epsilon$-good if for every coordinate i and every interval $I_{i,t}$, $p_i(I_{i,t}) \leq \epsilon$.*

Intuitively, ϵ-good distributions have small mass in every interval and hence binning it would not change the distribution by much. Specifically in Lemma B.1, we show that for such distributions $\left\|p - p^{\mathcal{S}}\right\|_1$ is bounded.

We now have all the tools to describe the algorithm. Let $\epsilon_1 = \epsilon^3/kd^2$. The algorithm first divides $\mathbb{R}^d$ into a rectangular gridded fine partition $\mathcal{S}$ with $\approx \epsilon_1^{-d}$ bins such that most of them have probability $\geq \epsilon_1^{d+1}$. We then group the bins with probability $< \epsilon_1^{d+1}$ to create a slightly coarser partition $\mathcal{S}'$. The resulting $\mathcal{S}'$ is coarse enough that $f^{\mathcal{S}'}$ can be estimated efficiently, and is also fine enough to ensure that we do not lose much of the Gaussian structure by binning.

We then limit the columns of $A^{\mathcal{S}'}$ to contain only Gaussians that are $(\mathcal{I}_1, \mathcal{I}_2, \ldots \mathcal{I}_d), 2\epsilon^2/d^2$-good. In Lemma B.1, we show that for all of these we do not lose much of the Gaussian structure by binning. Thus $\mathsf{solve}(A^{\mathcal{S}'}, b, k, \epsilon)$ yields us the required solution. With these definitions in mind, the algorithm is given in $\mathsf{Learn}(\{(x_1, \ldots x_{2n}), k, \epsilon\})$. Note that the number of rows in $A^{\mathcal{S}'}$ is $|\mathcal{S}'| \leq |\mathcal{S}| = \epsilon_1^{-d}$.

We need to bound the time complexity of finding a Gaussian in each iteration of the algorithm (to apply Remark 2.4). To this end we need to find a finite set of candidate Gaussians (columns of $A^{\mathcal{S}'}$) such that running solve using a matrix restricted to these columns (call it $A^{\mathcal{S}'}_{\text{finite}}$) finds the desired mixture up to error ϵ. Note that for this, we need to ensure that there is at least one candidate (column of $A^{\mathcal{S}'}_{\text{finite}}$) that is close to each of the true mixture components.

We ensure this as follows. Obtain a set of n' samples from the Gaussian mixture and for each pair of samples x, y consider the Gaussian whose mean is x and the variance along coordinate i is $(x_i - y_i)^2$. Similar to the proof of the one-dimensional version in Acharya et al. [2014], it follows that for any ϵ' choosing $n' \geq$

$\Omega((\epsilon')^{-d})$, this set contains Gaussians that are ϵ' close to each of the underlying mixture components. For clarity of exposition, we ignore this additional error which can be made arbitrarily small and treat ϵ' as 0.

procedure Learn($\{(x_1, \dots x_{2n}), k, \epsilon\}$)
begin
1 Set parameter $\epsilon_1 = \epsilon^3/(kd^2)$.
2 Use first n samples to find $\mathcal{I}_1, \mathcal{I}_2 \dots \mathcal{I}_d$ such that number of samples x such that $x_i \in I_{i,t}$ is $n\epsilon_1$. Let $\mathcal{S}$ be the corresponding induced partition.
3 Use the remaining n samples to do:
4 Let $U = \cup S_v : n(S_v) \le n\epsilon_1^d \epsilon$.
5 Let $\mathcal{S}' = \{U\} \cup \{S \in \mathcal{S} : n(S_v) > n\epsilon_1^d \epsilon\}$.
6 Set $b(U) = 2\epsilon$ and

$$\forall S \in \mathcal{S}' \setminus \{U\}, b(S) = \frac{(1-2\epsilon)n(S)}{\sum_{S \in \mathcal{S}' \setminus \{U\}} n(S)}$$

7 Let $A^{\mathcal{S}'}$ be the matrix with columns corresponding to distributions p that are $(\mathcal{I}_1, \mathcal{I}_2, \dots \mathcal{I}_d), 2\epsilon^2/d^2$-good axis-aligned Gaussians, and $A^{\mathcal{S}'}_{\text{finite}}$ be the candidates obtained as above, using $\epsilon' = \epsilon_1/10$.
8 solve($A^{\mathcal{S}'}_{\text{finite}}, b, k, 64\epsilon$) using Remark 2.4.
9 Output the w.
end

3.3 Proof of correctness

We first show that b satisfies the necessary conditions for solve that are given in Theorem 2.1. The proof follows from Chernoff bound and the fact that empirical mass in most sets $S \in \mathcal{S}'$ is $\ge \epsilon_1^d \epsilon$.

Lemma 3.5 (Appendix B.1)**.** *If $n \ge \frac{8}{\epsilon_1^d \epsilon^3} \log \frac{2}{\delta \epsilon_1^d}$, then with probability $\ge 1 - \delta$*

$$\forall S \in \mathcal{S}', b(S) \ge f^{\mathcal{S}'}(S)(1 - 3\epsilon),$$

and $\sum_{S \in \mathcal{S}'} |f^{\mathcal{S}'}(S) - b(S)| \le 6\epsilon$.

Using the above lemma, we now prove that Learn returns a good solution such that $\left\| f^{\mathcal{S}} - \hat{f}^{\mathcal{S}} \right\|_1 \le \mathcal{O}(\epsilon)$.

Lemma 3.6. *Let $n \ge \max\left(\frac{2}{\epsilon_1^2} \log \frac{2d}{\delta}, \frac{8}{\epsilon_1^d \epsilon^3} \log \frac{2}{\delta \epsilon_1^d}\right)$. With probability $\ge 1 - 2\delta$, Learn returns a solution $\hat{f}$ such that the resulting mixture satisfies*

$$\left\| f^{\mathcal{S}} - \hat{f}^{\mathcal{S}} \right\|_1 \le 74\epsilon.$$

Proof. We first show that $A^{\mathcal{S}'}$ has columns corresponding to all the components r, such that $w_r \ge \epsilon/k$. For a mixture f let f_i be the projection of f on coordinate i. Note that $\hat{f}_i(I_{i,t}) = \epsilon_1 \, \forall i, t$. Therefore by Dvoretzky-Kiefer-Wolfowitz theorem (see, Massart [1990]) and the union bound if $n \ge \frac{2}{\epsilon_1^2} \log \frac{2d}{\delta}$, with probability $\ge 1 - \delta$,

$$f_i(I_{i,t}) \le \epsilon_1 + \epsilon_1 \le 2\epsilon_1 \, \forall i, t.$$

Since $f_i = \sum_{r=1}^k w_r p_{r,i}$, with probability $\ge 1 - \delta$,

$$p_{r,i}(I_{i,t}) \le \frac{2\epsilon_1}{w_r} \, \forall i, r.$$

If $w_r \ge \epsilon/k$, then $p_{r,i}(I_{i,t}) \le 2\epsilon^2/d^2$ and thus $A^{\mathcal{S}'}$ contains all the underlying components r such that $w_r \ge \epsilon/k$. Let w^* be the weights corresponding to components such that $w_r \ge \epsilon/k$. Therefore $\|w^*\|_1 \ge 1 - \epsilon$. Furthermore by Lemma 3.5, $b(S) \ge f^{\mathcal{S}'}(1 - 3\epsilon) \ge (1 - 3\epsilon)(A^{\mathcal{S}'} w^*)(S)$. Therefore, we have $\|b\| = \left\| A^{\mathcal{S}'} w^* / \|w^*\| \right\|$ and

$$\begin{aligned} b(S) &\ge (1 - 3\epsilon)(A^{\mathcal{S}'} w^*)(S) \, \|w^*\| / \|w^*\| \\ &\ge (1 - 4\epsilon)(A^{\mathcal{S}'} w^* / \|w^*\|)(S). \end{aligned}$$

Hence, By Theorem 2.1, algorithm returns a solution $A^{\mathcal{S}'} w'$ such that $\left\| A^{\mathcal{S}'} w' - b \right\|_1 \le 64\epsilon$. Thus by Lemma 3.5, $\sum_{S \in \mathcal{S}'} |(A^{\mathcal{S}'} w')(S) - f^{\mathcal{S}'}(S)| \le 70\epsilon$. Let $\hat{f}$ be the estimate corresponding to solution w'. Since $f^{\mathcal{S}'}$ and $\hat{f}^{\mathcal{S}'}$ are flat within sets S, we have $\left\| \hat{f}^{\mathcal{S}'} - f^{\mathcal{S}'} \right\|_1 \le 70\epsilon$.

Since $\mathcal{S}'$ and $\mathcal{S}$ differ only in the set U and by Lemma 3.5, $f^{\mathcal{S}}(U) = f^{\mathcal{S}'}(U) \le \epsilon/(1 - 3\epsilon)$, we have

$$\left\| \hat{f}^{\mathcal{S}} - f^{\mathcal{S}} \right\|_1 \le \left\| \hat{f}^{\mathcal{S}'} - f^{\mathcal{S}'} \right\|_1 + 2f^{\mathcal{S}}(U) \le 74\epsilon.$$

Note that the total error probability is $\le 2\delta$. □

We show that flattened Gaussians in one dimension are close to the corresponding unflattened Gaussian.

Lemma 3.7 (Appendix B.2)**.** *Let p be a one dimensional Gaussian distribution and $\mathcal{I} = (I_1, I_2, \dots)$ be a partition of the real line such that $\forall I \in \mathcal{I}$, I is a continuous interval and $p(I) \le \epsilon$. Then $\left\| p - p^{\mathcal{I}} \right\|_1 \le 30\sqrt{\epsilon}$.*

Lemma 3.6 shows that $\hat{f}^{\mathcal{S}}$ is close to $f^{\mathcal{S}}$. Lemma 3.7 shows that in one dimension flattened Gaussians are close to the unflatttend one. We extend Lemma 3.7 to d dimensions and prove

Theorem 3.8 (Appendix B.3)**.** *Let $\epsilon_1 = \epsilon^3/kd^2$ and $n \ge \max\left(\frac{2}{\epsilon_1^2} \log \frac{2d}{\delta}, \frac{8}{\epsilon_1^d \epsilon^3} \log \frac{2}{\delta \epsilon_1^d}\right)$. Then given $2n$ samples from an axis-aligned Gaussian mixture f, with probability $\ge 1 - 2\delta$, Learn returns an estimate mixture $\hat{f}$ with at most $\mathcal{O}(k/\epsilon^3)$ components such that*

$$\left\| \hat{f} - f \right\|_1 \le 170\epsilon.$$

The run time of the algorithm is $\mathcal{O}\left(1/\epsilon_1\right)^d$.

If we consider the leading term in sample complexity, for $d = 1$ our bound is $\tilde{\mathcal{O}}(k^2/\epsilon^6)$, and for $d > 1$, our bound is $\tilde{\mathcal{O}}((kd^2)^d/\epsilon^{3d+3})$. While this is not the optimal sample complexity (see Acharya et al. [2014]), we gain significantly in the running time.

4 Lower bounds

We now investigate lower bounds towards obtaining sparse approximate solutions to nonnegative systems. Our first result is that unless $\mathcal{P} = \mathcal{NP}$, we need to lose a factor at least $\log(1/\epsilon)$ in the sparsity to be ϵ-close in the ℓ_1 norm. Formally,

Theorem 4.1. *For any $\epsilon > 0$, given an instance of the sparse recovery problem A, b that is promised to have a k-sparse nonnegative solution, it is $\mathcal{NP}$-hard to obtain an $o\left(k \ln\left(\frac{1}{\epsilon}\right)\right)$-sparse solution x_{alg} with $\|Ax_{alg} - b\|_1 < \epsilon \|b\|_1$.*

Theorem 4.1 is inspired by the hard instances of Max k-Cover problem [Feige, 1998, Feige et al., 2004, Feige and Vondrák, 2010].

Hard Instances of Max k-Cover. For any $c > 0$, and $\delta > 0$, given a collection of n sets $S_1, S_2, \ldots, S_n \subseteq [m]$, it is $\mathcal{NP}$-Hard to distinguish between the following two cases:

YES case: There are k disjoint sets in this collection whose union is $[m]$.

NO case: The union of any $\ell \leq ck$ sets of this collection has size at most $(1 - (1 - \frac{1}{k})^\ell + \delta)n$.

Proof outline, Theorem 4.1. We reduce hard instance of the Max k-cover problem to our problem as follows. For each set S_i, we set A_i (column i in A) to be the indicator vector of set S_i. We also let b to be the vector with all entries equal to one. In the YES case, it is easy to construct a k-sparse x s.t. $Ax = b$, while in the NO case, finding a solution of sparsity $o(k \ln \frac{1}{\epsilon})$ contradicts the hardness result stated above. The details are deferred to the supplement. □

Second, we show that unless a certain variant of set cover can be solved efficiently, we cannot hope to obtain an ϵ-approximate solution with sparsity $o(k/\epsilon^2)$. We will call this the *planted set cover* problem:

Definition 4.2. *(Planted set cover (k, m) problem) Given parameters m and $k > m^{3/4}$, find an algorithm that distinguishes with probability $> 2/3$ between the following distributions over set systems over m elements and $n = O(m/\log m)$ sets:*

NO *case: The set system is random, with element i in set j with probability $1/k$ (independently).*

YES *case: We take a random set system with $n - k$ sets as above, and add a random k-partition of the elements as the remaining k sets. (Thus there is a perfect cover using k sets.)*

To the best of our knowledge, solving this distinguishing problem is beyond our algorithmic techniques. The situation is similar in spirit to the planted clique and planted dense subgraph problems on random graphs, as well as random 3-SAT [Alon et al., 1998, Bhaskara et al., 2010, Feige, 2002]. This shows that obtaining sparse approximate solutions with sparsity $o(k/\epsilon^2)$ requires significantly new techniques. Formally, we show the following (proof deferred to the supplement).

Theorem 4.3. *Let $m^{3/4} < k < m/\log^2 m$. Any algorithm that finds an $o(k/\epsilon^2)$ sparse ϵ-approximate solution to non-negative linear systems can solve the planted set cover (k, m) problem.*

5 Experiments

To better understand the analysis, we implemented our algorithm with some natural settings for matrices A. We now give a brief summary of the results (tables are deferred to the supplement).

Random A: The most natural choice for A are $m \times n$ matrices, with each entry distributed independently. We picked each entry uniformly in $(0, 1)$, and then normalized the columns.[4] The vector b is obtained by taking a random combination of the first k columns of A. Here we observed the following: as n grows, keeping m fixed, the number of non-zero entries in the solution (found by the algorithm) slowly grows, until it stabilizes. There is also a disparity between the number of iterations of the algorithm (which we used to bound the sparsity), and the actual sparsity. In order to allow for a large n relative to m, we fix a small value of m $(= 15)$, and $k = 5$. We also show the behavior with varying ϵ.

Gaussian mixture in one dimension The next choice for A comes from our result on learning mixtures of Gaussians. Here we consider the one-dimensional version, in which we are given a mixture of k Gaussians on a line, whose components we wish to find. We picked $k = 4$, and unit variance for the Gaussians. The means were picked at random in the interval $(0, 20)$, which was discretized into 200 sub-intervals. We then considered 200 candidate Gaussians, with means at the centers of each sub-interval. The goal is to approximate the given mixture by a mixture of these candidate Gaussians (up to an error ϵ) using as few components as possible. In the supplement, we show the results for varying ϵ. We also give the true means, and the means of the Gaussians used in the approximation found by the algorithm.

[4] Note that unlike the case of random Gaussian entries, the columns here are not incoherent, and nor does A possess the restricted isometry property.

References

J. Acharya, A. Jafarpour, A. Orlitsky, and A. T. Suresh. Near-optimal-sample estimators for spherical gaussian mixtures. *CoRR*, abs/1402.4746, 2014.

N. Alon, M. Krivelevich, and B. Sudakov. Finding a large hidden clique in a random graph. *Random Structures and Algorithms*, 13(3-4):457–466, 1998. ISSN 1098-2418.

E. Amaldi and V. Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(12):237 – 260, 1998. ISSN 0304-3975.

J. Anderson, M. Belkin, N. Goyal, L. Rademacher, and J. R. Voss. The more, the merrier: the blessing of dimensionality for learning large gaussian mixtures. In *Proceedings of The 27th Conference on Learning Theory, COLT 2014, Barcelona, Spain, June 13-15, 2014*, pages 1135–1164, 2014.

S. Arora and R. Kannan. Learning mixtures of arbitrary gaussians. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 247–257, 2001.

S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. In *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*, pages 724–733, Nov 1993.

S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

A. S. Bandeira, E. Dobriban, D. G. Mixon, and W. F. Sawin. Certifying the restricted isometry property is hard, 2012.

M. Belkin and K. Sinha. Polynomial learning of distribution families. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 103–112, 2010.

A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan. Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 201–210, 2010.

A. Bhaskara, M. Charikar, A. Moitra, and A. Vijayaraghavan. Smoothed analysis of tensor decompositions. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 594–603, 2014.

E. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, Feb 2006. ISSN 0018-9448.

S. Chan, I. Diakonikolas, R. A. Servedio, and X. Sun. Learning mixtures of structured distributions over discrete domains. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1380–1394, 2013.

S. Chan, I. Diakonikolas, R. A. Servedio, and X. Sun. Efficient density estimation via piecewise polynomial approximation. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 604–613, 2014.

S. Dasgupta. Learning mixtures of gaussians. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 634–644, 1999.

S. Dasgupta and L. J. Schulman. A probabilistic analysis of EM for mixtures of separated, spherical gaussians. *Journal of Machine Learning Research*, 8:203–226, 2007.

C. Daskalakis and G. Kamath. Faster and sample near-optimal algorithms for proper learning mixtures of gaussians. In *Proceedings of The 27th Conference on Learning Theory, COLT 2014, Barcelona, Spain, June 13-15, 2014*, pages 1183–1213, 2014.

D. L. Donoho. Compressed sensing. *IEEE Trans. Inf. Theor.*, 52(4):1289–1306, Apr. 2006. ISSN 0018-9448.

D. L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

U. Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, July 1998. ISSN 0004-5411.

U. Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity, Montréal, Québec, Canada, May 21-24, 2002*, page 5, 2002.

U. Feige and J. Vondrák. The submodular welfare problem with demand queries. *Theory of Computing*, 6(1):247–290, 2010.

U. Feige, L. Lovász, and P. Tetali. Approximating min sum set cover. *Algorithmica*, 40(4):219–234, 2004.

J. Feldman, R. A. Servedio, and R. O'Donnell. PAC learning axis-aligned mixtures of gaussians with no separation assumption. In *Learning Theory,*

19th Annual Conference on Learning Theory, COLT 2006, Pittsburgh, PA, USA, June 22-25, 2006, Proceedings, pages 20–34, 2006.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

D. Hsu and S. M. Kakade. Learning mixtures of spherical gaussians: moment methods and spectral decompositions. In *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 11–20, 2013.

A. T. Kalai, A. Moitra, and G. Valiant. Efficiently learning mixtures of two gaussians. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 553–562, 2010.

M. J. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire, and L. Sellie. On the learnability of discrete distributions. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 273–282, 1994.

L. Khachiyan. On the complexity of approximating extremal determinants in matrices. *Journal of Complexity*, 11(1):138 – 153, 1995. ISSN 0885-064X.

B. G. Lindsay. *Mixture Models: Theory, Geometry and Applications*. NSF-CBMS Conference series in Probability and Statistics, Penn. State University, 1995.

P. Massart. The tight constant in the dvoretzky-kiefer-wolfowitz inequality. *The Annals of Probability*, 18 (3):1269–1283, 07 1990.

A. Moitra and G. Valiant. Settling the polynomial learnability of mixtures of gaussians. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 93–102, 2010.

B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM J. Comput.*, 24(2):227–234, Apr. 1995. ISSN 0097-5397.

D. A. Reynolds and R. C. Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, 3(1):72–83, 1995.

M. Rudelson and R. Vershynin. Non-asymptotic theory of random matrices: extreme singular values, 2010.

S. Shalev-Shwartz, N. Srebro, and T. Zhang. Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM J. on Optimization*, 20 (6):2807–2832, Aug. 2010. ISSN 1052-6234.

D. M. Titterington, A. F. Smith, and U. E. Makov. *Statistical analysis of finite mixture distributions*. Wiley New York, 1985.

L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, Nov. 1984. ISSN 0001-0782.

S. Vempala and G. Wang. A spectral algorithm for learning mixtures of distributions. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, page 113, 2002.