Provably Learning Mixtures of Gaussians and More

Jonathan Huggins
jhh2143@columbia.edu

1 Introduction

Given a random sample, how can one accurately estimate the parameters of the probabilistic model that generated the data? This is a fundamental question in statistical inference and, more recently, in machine learning. One of the most widely studied instances of this problem is estimating the parameters of a mixture of Gaussians, since doing so is of fundamental importance in a wide range of subjects, from physics to social science. The classic, and most popular approach, for learning Gaussian mixture models (GMMs) is the EM algorithm [Dempster et al., 1977]. The EM algorithm is a search heuristic over the parameters that finds a local maximum of the likelihood function, and therefore makes no guarantees that it will converge to an estimate that is close to the true parameters. Furthermore, in practice it has been found to converge very slowly.

In light of the lack for performance guarantees for learning GMMs, Dasgupta [1999] instigated over a decade of work by the theoretical computer science community on provably recovering, with high probability, good estimates of GMM parameters in polynomial time and sample size. A range of special cases—such as requiring spherical Gaussians, separation conditions on the parameters, or a shared covariance matrix—were tackled first. But recently, Belkin and Sinha [2010b] and Moitra and Valiant [2010] showed it was possible to learn *any* GMM, with no separation or covariance conditions.

Most of the techniques that have been developed, and all those that we will discuss in detail, rely on projecting the data onto some subspace with dimension polynomial in the number of mixture components. The mixture components are learned in that subspace, then the GMM parameters in the original high-dimensional space is reconstructed. At a high level, the goal with this approach is to project the data in such a way that data points generated from different components are separated in some formal sense. What is meant by separation depends on the details of the algorithm. There are a number of approaches to projecting: for example, Dasgupta [1999] and others project the data onto random subspaces while Vempala and Wang [2002] introduced spectral techniques — that is, using singular value decomposition (SVD) to project deterministically onto the principal components. Methods that project onto many subspaces attempt to ensure either that some "best" subspace can be chosen or that most of the subspaces will be "good" in some sense.

Before delving into the details of provably learning mixture models, we briefly outline what is to come. In Section 2, we present some important basic definitions and formally define the problem of learning a GMM. Important conceptual considerations when trying to learn high-

dimensional Gaussian distributions are also discussed. Next, a generic projection approach to learning GMM parameters is sketched and a few of the results which will not be discussed in detail are reviewed in Section 3. In Section 4, we give a more detailed analysis of spectral techniques, including their limitations and benefits, while attempting to highlight the more generally applicable theorems and strategies. In Section 5, we discuss the recent state-of-the-art projection-based techniques [Belkin and Sinha, 2010b, Moitra and Valiant, 2010], which are capable of learning arbitrary mixtures. Finally, in Section 6, we conclude with some remarks and possible directions for further research.

2 Preliminaries

We are interested in recovering GMM parameters when given access to a sequence of i.i.d. draws generated by an oracle. Formally, a Gaussian mixture model is a convex combination of k different n-dimensional Gaussians with weights $w_i \in [0,1]$, means $\mu_i \in \mathbb{R}^n$ and covariance matrices $\Sigma_i \in \mathbb{R}^{n\times n}$ (so the weights must sum to 1: $\sum_{i=1}^k w_i = 1$). Let $F_i = \mathcal{N}(\mu_i, \Sigma_i)$ denote the distribution of the i-th component of the mixture

$$F_i(\vec{x}) = \frac{1}{(2\pi)^{n/2}\sqrt{\det(\Sigma_i)}} \exp\left(-\frac{1}{2}(\vec{x}-\mu_i)^T\Sigma_i^{-1}(\vec{x}-\mu_i)\right).$$

The density of the GMM is $F = \sum_{i=1}^k w_i F_i$. We will use θ to refer to the set of weight, mean, and covariance parameters

$$\theta = \{(w_1, \mu_1, \Sigma_1), (w_2, \mu_2, \Sigma_2), \ldots, (w_k, \mu_k, \Sigma_k)\}.$$

and use $\hat{F}, \hat{F}_i$, $\hat{\theta}$, etc. along with their common variations (e.g. $\hat{\theta}'$) to indicate estimates. Throughout, $w_{\min} = \min_i\{w_i\}$ denotes the minimum mixture weight and σ_i denotes the largest singular value of Σ_i, i.e. the maximum standard deviation along any direction in $\mathbb{R}^n$. S will denote the set of sample points and $S_i \subset S$ the points drawn from F_i. Unless stated otherwise it is assumed that the exact number of mixture components k is known.

In order to appreciate some of the considerations involved in learning GMMs, it is important to understand some of the surprising, even strange, properties of high-dimensional Gaussians. When n is large, most of the probability mass lies far away from a Gaussian's mean. For the simple case of a spherical Gaussian, if $\vec{x} \sim \mathcal{N}(\mu, \sigma^2 I_n)$, then the expected square of the Euclidean norm $\mathbb{E}(||\vec{x}-\mu||^2) = n\sigma^2$. By the law of large numbers, for large n the distribution of the squared length must be tightly concentrated about its expected value. Thus, most of the mass lies within a thin shell a distance $\sqrt{n}\sigma$ from the mean of the Gaussian [Dasgupta, 1999]. In the more general case of an arbitrary Gaussian $\mathcal{N}(\mu, \Sigma)$, for large n the distribution is concentrated at a Mahalanobis distance $||\vec{x}||_{\mu,\Sigma} = \sqrt{(\vec{x}-\mu)^T\Sigma^{-1}(\vec{x}-\mu)} = \sqrt{n}$ from the Gaussian's center. This result suggests that even if two Gaussians have the same mean, there may still be some hope of determining which one a point was drawn from! This "concentration of distance" result and many variations on it are very useful when trying to learn mixtures of Gaussians.

2.1 Defining the Problem

In order to properly define the learning problem and determine an appropriate notion of efficient learning, we must detail a few considerations. First, we must define what is means for the components of a GMM to be different. If two components of the mixture have the same distribution, after all, there is no hope of distinguishing them. There are a variety of ways of defining such a separability condition, but one natural notion is based on the distance between the means, while accounting for the variance of the Gaussians.

Definition 2.1. A pair of Gaussians $\mathcal{N}(\mu_1, \Sigma_1)$ and $\mathcal{N}(\mu_2, \Sigma_2)$ have **separation** Δ if

$$||\mu_1 - \mu_2|| \geq \Delta(\sigma_1 + \sigma_2).$$

A mixture has **separation** Δ if each pair of components have separation Δ.

Some authors replace $\sigma_1 + \sigma_2$ with $\max\{\sigma_1, \sigma_2\}$, but this will alter results by at most a factor of 2, and thus does not affect big-O results. Alternatively, separability can be thought of in statistical terms by considering the density functions of the mixture components directly.

Definition 2.2. [Moitra and Valiant, 2010] A GMM $F = \sum_i w_i F_i$ is **Δ-statistically learnable** if $w_{\min} \geq \Delta$ and $\min_{i \neq j} D(F_i, F_j) \geq \Delta$. Here, if $f(x)$ and $g(x)$ are probability distributions on $\mathbb{R}^n$, then the **statistical distance** between the distributions is defined as

$$D(f(x), g(x)) = \frac{1}{2} \int_{\mathbb{R}^n} |f(x) - g(x)| dx.$$

A general notion of the difficulty of learnability for any family of probability distributions (which reduces to a separation condition in the GMM case) arises from treating θ as a vector within a parameter space $\Theta \subset \mathbb{R}^m$ and considering the open ball around θ such that all the probability distributions arising from parameter choices within that ball are distinct.

Definition 2.3. [Belkin and Sinha, 2010b] Let p_θ be a family of probability distributions parameterized by $\theta \in \Theta$. For each θ, define the **radius of identifiability** to be

$$\mathcal{R}(\theta) = \sup\{r > 0 | \forall \theta_1 \neq \theta_2, (||\theta_1 - \theta|| < r \wedge ||\theta_2 - \theta|| < r) \implies (p_{\theta_1} \neq p_{\theta_2})\}$$

If the condition cannot be satisfied, then $\mathcal{R}(\theta) = 0$.

In the case of a GMM, $\mathcal{R}(\theta)$ has a simple form that measures the minimal pairwise parameter distances, while also taking into account the minimum mixing weight.

Proposition 2.4. *[Belkin and Sinha, 2010b] The radius of identifiability of a GMM with parameter set θ is given by*

$$\mathcal{R}(\theta)^2 = \min\left(\frac{1}{4} \min_{i \neq j} \left\{||\mu_i - \mu_j||^2 + ||\Sigma_i - \Sigma_j||_F^2\right\}, w_{\min}^2\right),$$

where $|| \cdot ||_F$ denotes the Frobenius norm.

In addition to accounting for the separation of GMM components, we must also decide how precisely we wish to learn the distribution and formalize what we mean by "learn precisely." Some algorithms solve the problem of *clustering*—identifying which component generated each data point. Once this is known, there are many algorithms available that provably estimate the GMM parameters, so clustering methods need not worry about this parameter precision issue explicitly. Many of the notions of closeness that have been used in the context of GMM learning are captured by the following definition, generalized from that found in Moitra and Valiant [2010].

Definition 2.5. A parameter set for a mixture of k Gaussians $\hat{\theta} = \{(\hat{w}_1, \hat{\mu}_1, \hat{\Sigma}_1), \ldots, (\hat{w}_k, \hat{\mu}_k, \hat{\Sigma}_k)\}$ is an **ϵ-close estimate for** θ if there is a permutation $\pi \in S_k$ such that for all $i \in [k]$,

1. $|w_i - \hat{w}_{\pi(i)}| \leq \epsilon$
2. $d(\mathcal{N}(\mu_i, \Sigma_i), \mathcal{N}(\hat{\mu}_{\pi(i)}, \hat{\Sigma}_{\pi(i)})) \leq \epsilon$

where $d(\cdot, \cdot)$ denotes some measure of distance between the Gaussians.

There are many possibilities for distance measure d. For example, we could use the statistical distance $D(\cdot, \cdot)$ defined above or a measure of *parameter distance* such as

$$D_p\left(\mathcal{N}(\mu, \Sigma), \mathcal{N}(\mu', \Sigma')\right) = ||\mu - \mu'|| + ||\Sigma - \Sigma'||_F.$$

Typically, when defining an "efficient learning algoirthm," there should be at worst a polynomial dependence on each parameter (or on some natural function of that parameter) given to the algorithm. However, Moitra and Valiant [2010] showed that there is no hope for such a dependence with respect to k, the number of mixture components. In fact, any general algorithm for learning GMMs must have an exponential dependence on k. This does not preclude learning special classes of GMMs—such as those consisting spherical Gaussians or imposing a separation requirement—in time polynomial in k. Finally, any learning algorithm for GMMs will require $\Omega(1/w_{\min})$ samples in order to ensure we have seen a sample from each component with high probability. With all those considerations in mind, we are now prepared to give a PAC learning-style definition for (efficiently) learning general GMMs. The definition is purposefully vague since, as we have seen, there is no standard meaning for an estimate to be within ϵ of the true parameters and no single measure for component separation.

Definition 2.6. An algorithm A **efficiently learns** the class of k-component, n-dimensional GMMs if, given a precision parameter $1 > \epsilon > 0$ and confidence parameter $1 > \delta > 0$, A outputs an estimate $\hat{\theta}$ that is within ϵ of the true parameters θ with probability at least $1 - \delta$, using $\text{poly}(n, 1/\epsilon, 1/\delta, 1/w_{\min}, 1/\Delta)$ time and samples. The samples are drawn i.i.d. from the GMM F parameterized by θ while Δ characterizes the separation of the true mixture.

With appropriate modifications, this definition is applicable to mixtures of other distributions as well. In order to concentrate on the techniques of general interest for learning GMMs, in the proceeding discussion of specific algorithms and techniques, we will sometimes elide mention of the particular measures used by an algorithm to calculate Δ and whether an estimate is ϵ-close. It is worth mentioning the slightly different (and easier) PAC learning framework used by Feldman et al. [2006], which only requires the density to be estimated, not the parameters. They show how to do accomplish this estimation task for mixtures of axis aligned Gaussian, with no separation conditions, by using the method of moments.

3 Learning GMMs

A prototypical learning algorithm for GMMs involves three steps. First, the n-dimensional data is projected onto one or more subspaces whose dimension(s) are polynomial in k. Since in most applications (and in the asymptotic limit) $k \ll n$, learning with these subspaces is (usually) efficient. The projection procedure maintains the Gaussianity of the data, so the problem is, roughly speaking, reduced to one of learning a poly(k)-dimensional GMM. Hence, the next step is either to identify each sample with the component of the mixture that generated it or use some other method to estimate components. Many of the procedures that take the first approach rely on "picking the low-hanging fruit." That is, they find the most separated component first, then recurse on the remaining data. This approach takes advantage of properties like the concentration of distance discussed in the previous section. Once components have been identified, the final step is to reconstruct the Gaussian components in the original n-dimensional space. Reconstruction is often quite technical and tends not to provide much high-level insight into the problem, so we will primarily focus on solutions to the first two steps.

In addition to the pioneering work of Dasgupta [1999] (whose algorithm required separation of $\Omega(n^{\frac{1}{2}})$) and the density estimation results of Feldman et al. [2006], there are many other pieces of research which we briefly review now for completeness. Other early work includes that of Dasgupta and Schulman [2000], who improved the original separation requirement to $\Omega(n^{\frac{1}{4}})$ for spherical Gaussians using an two-round variant of the EM algorithm. Soon after, Arora and Kennan [2001] showed how to learn general GMMs under the same separation condition. They did not use projection, relying solely on concentration of distance results. More recently, [Belkin and Sinha, 2010a] describe how to learn identical spherical Gaussians with no minimum separation required. This is done using projection onto k components and the Fourier transform. See Table 1 for a comparison of methods.

4 Spectral Learning

Spectral methods rely on calculating the singular value decomposition of the samples (or a subset of them), then projecting to the subspace spanned by the top right singular vectors, also known as the principal components. The original spectral algorithm for learning mixtures of spherical Gaussians was proposed by Vempala and Wang [2002]. In fact, their result applies to mixtures of *weakly isotropic* distributions. A weakly isotropic distribution is one for which the variance of any 1-dimensional projection is a constant. Although only applicablele to a small subset of GMMs, the techniques and results provide good motivation for, and are indicative of, what is to come. The algorithm repeatedly projects the data onto the top k principle components. The reason this works is that with high probability, the space spanned by these vectors differs from that spanned by the mixture component means μ_i by no more than a small factor. Thus, the structure of the distribution is approximately preserved under projection. After projecting, at least one set of points S_i generated by a single component is found using distance concentration, which guarantees the points from a single component will be clustered close together with high probability. The identified points are removed from consideration and the whole procedure, including projection, is repeated until all points are

clustered. At this stage estimating the means and covariance matrices is easy. The condition on the means of the Gaussians is fairly weak, only requiring separation $\Omega(k^{\frac{1}{4}} \log^{\frac{1}{4}}(n/w_{\min}))$. Recall that S is the set of sample points.

Algorithm 1 Spectral algorithm for learning spherical GMMs [Vempala and Wang, 2002]

$M \leftarrow |S|$
while $S \neq \emptyset$ **do**
  Compute the k-dimensional SVD subspace W of S
  Project S onto W
  $R \leftarrow \max_{x \in S} \min_{y \in S} ||x - y||$
  $S' \leftarrow \{x \in S : \min_{y \in S} ||x - y|| \leq 3\hat{\epsilon} R^2\}$
  $G \leftarrow \emptyset$
  while $S' \neq \emptyset$ **do**
    Let x, y be the two closest points in S'
    $\ell \leftarrow ||x - y||^2 \left(1 + 8\sqrt{\frac{6 \ln \frac{M}{\delta}}{k}}\right)$
    $H \leftarrow \{w \in S' : ||x - w||^2 \leq \ell\}$
    $S' \leftarrow S' \setminus H$
    $G \leftarrow G \cup \{H\}$
  end while
  Report each $H \in G$ with variance greater than $3\epsilon R^2/k$ as the set of points generated by one component of the mixture and remove those points from S
end while

This procedure nicely illustrates some common elements of GMM learning. In addition to using concentration of distance and projection, it also relies on getting the "low-hanging fruit" and then recursing. The benefits of doing this with an eye toward obtaining provable guarantees is clear. For such a recursive procedure to work, one need "only" show that on each iteration the "easiest" component(s) can be identified. After that, the problem is reduced to learning a k'-component mixture, for some $k' < k$. So by induction the procedure will terminate, solving the original k-component problem.

Kannan et al. [2005] refined and extended these spectral projection techniques, providing a vast generalization applicable to any mixture of log-concave distributions (which includes GMMs). In the log-concave setting, it is not longer true that the SVD subspace (approximately) contains the means. However, it turns out that for *any* mixture, the components means will on average remain separated after projection.

Theorem 4.1. *[Kannan et al., 2005] Let W be the k-dimensional SVD subspace of sample set S, where the samples are generated from some k-component (not necessarily log-concave) mixture. For each $i \in [k]$, let $\mu_{i,S}$ be the mean of S_i and $\sigma^2_{i,S}(W)$ be the maximum variance of S_i along any direction in W. Then,*

$$\sum_{i=1}^{k} |S_i| d(\mu_{i,S}, W)^2 \leq k \sum_{i=1}^{k} |S_i| \sigma^2_{i,S}(W),$$

where $d(x, W)$ denotes the distance between the point x and the subspace W.

The other good news that follows from this result is that, since the distance of a point from the mean of the distribution it was drawn from can only decrease after projection, the ratio of the inter- to intra-component distances is magnified. In this general setting, however, there is no guarantee that the samples from different components do not overlap. This issue is overcome by requiring the mixture components to be log-concave distributions. Such distributions have two key properties. First, as in the special case of Gaussians, they are well-behaved in the sense that the projection of a log-concave distribution remains log-concave. Second, the distribution is concentrated about the mean. Specifically, the distance of a random point from the mean has an exponential tail.

Lemma 4.2. *Let F be a any log-concave distribution over $\mathbb{R}^n$ with mean μ and second moment $R^2 = \mathbb{E}_F[(X-\mu)^2]$. Then there exists an absolute constant c such that $\forall t > 1$,*

$$\Pr(|X - \mu| > tR) < e^{-ct}.$$

With these results in mind, the algorithm proceeds very similarly to the previous one by taking an iterative, distance-based approach. Each iteration identifies exactly one component.

Algorithm 2 Spectral algorithm for learning log-concave mixtures [Kannan et al., 2005]

$m \leftarrow |S|$
while $S \neq \emptyset$ **do**
  Compute the k-dimensional SVD subspace W using a subset T of S of size m_0
  $S \leftarrow S \setminus T$
  Project S onto W
  for all $x \in S$ **do**
    - Calculate the set $S(x)$ consisting of the closest $\frac{1}{2}w_{\min}m$ points to x
    - Find the mean $\mu(x)$ of $S(x)$
    - Form the matrix $A(x)$ whose rows are $y - \mu(x)$ for each $y \in S(x)$
    - Calculate $\sigma(x)$, the largest singular value of $A(p)$, i.e. the maximum standard deviation of $S(x)$ in W
  end for
  $x_0 \leftarrow \arg\max_{x \in S} \sigma(x)$
  $T_0 \leftarrow \left\{x \in T : ||W(x_0) - W(x)|| \leq \frac{\sqrt{k}\log N}{w_{\min}}\sigma(x)\right\}$, where $W(x)$ denotes the projection of x to W
  Report T_0 as the set of points generated by one component of the mixture and remove those points from T
end while

There are two downsides to this algorithm when compared to that for spherical Gaussians. One is that the mixture must have separation $\Omega^*(k^{\frac{3}{2}}/w_{\min}^2)$, which is a bit stronger than what Vempala and Wang require (the Ω^* notation suppresses logarithmic terms in the other parameters). In addition, kN_0 samples are used for in the k SVD computations, where $N_0 = \text{poly}(n, 1/\epsilon, \log(1/\delta), 1/w_{\min})$. For technical reasons, these points must be discarded and cannot be classified. While our definition of learning only requires estimating the parameters, in practice a classification of the samples is often of interest as well, so it is unfortunate that many points must be discarded. Of course, other methods, such as assigning each discarded point to the Gaussian with the closest mean could be used. Although there is no guarantee

of correctness, in an applied setting such a guarantee for these extra points, while nice, is probably not critical.

In Achlioptas and McSherry [2005], the separation requirement for learning mixtures of log-concave distributions using spectral methods was reduced to $\Omega(k + \sqrt{k \log n})$. Conceptually, their approach is very similar to the two already described.

5 Projection Approaches

The most general algorithms for GMM learning [Moitra and Valiant, 2010, Belkin and Sinha, 2010b] have not been based on spectral methods, but instead rely on random or brute force projection. Neither requires any separation assumptions.

5.1 Random Projection

Moitra and Valiant [2010] present an algorithm with a statistical flavor: it returns an ϵ-close estimate for θ, with statistical distance $D(\cdot, \cdot)$ used to measure distance between the Gaussians. Within this subsection ϵ-close estimates will always be based on statistical distance. Their analysis makes use of parameter distance as well, however. While in general the parameter and statistical distance between two Gaussian can be unrelated, this only happens when the variances are allowed to be arbitrarily large or small. So, as long as there are reasonable upper and lower bounds on the variances, we can convert from one to the other. The results of Moitra and Valiant [2010] are generalizations of, and rely on insights from, Kalai et al. [2010]. The core algorithms developed in both papers assume that that the mixture distribution is in *isotropic position*. A distribution is in isotropic position if it is in weakly isotropic position, has mean zero, and has variance one. In each case the results are completely general, however, since a sample can be put into isotropic position. If the sample is sufficiently large, then the underlying distribution for the transformed data will be in (nearly) isotropic position, which is sufficient to ensure correctness of the core algorithms.

Kalai et al. [2010] rely on three key lemmas to learn mixtures of two Gaussians: the 1D Learnability Lemma, the Random Projection Lemma, and the Parameter Recovery Lemma. The 1D Learnability Lemma states that the parameters of a two component univariate Gaussian mixture can be efficiently recovered as long as the Gaussians have a non-negligible statistical distance. The Random Projection Lemma guarantees that, for an isotropically positioned n-dimensional mixture of two Gaussians with non-negligible statistical distance, with high probability the projection of the mixture onto a random unit vector u will yield a univariate mixture whose Gaussians still have non-negligible statistical distance. Finally, the Parameter Recovery Lemma states that, if we have accurate estimates for an n-dimensional Gaussian in n^2 sufficiently different directions, we can accurately recover its parameters. These three lemmas suggest the procedure outlined in Algorithm 3 for learning mixtures of two Gaussians. A "robust" version serves as a component in the algorithm for learning arbitrary mixtures.

When learning many Gaussians, the Parameter Recovery Lemma remains applicable and the 1D Learnability Lemma can be generalized, although doing so is quite technical. The algorithm for learning the univariate parameters relies on "deconvolving" the mixture by a

Algorithm 3 Learning mixtures of two Gaussians [Kalai et al., 2010]

Pick a random unit vector u
Choose n^2 vectors $u_1, \ldots, u_{n^2}$ that are fairly close to u
for all $i \in [n^2]$ **do**
  learn very accurate univariate parameters for the projection of the mixture onto u_i
end for
Recover the true n-dimensional parameters for the mixture with high probability

suitable Gaussian in order to increase the separation of the components. Then the method of moments is used to find the parameters whose lower order moments closely match the empirical moments of the data. Such a match is found by performing a simple grid search over the parameter space. We will refer to the procedure for learning k-component univariate mixtures as the Basic Univariate Algorithm. It requires that the pairwise parameter distances are greater than a known ϵ, with $w_{\min} \geq \epsilon$.

The problem is that for more than two Gaussians, the Random Projection Lemma ceases to hold. In fact, it is possible to construct mixtures of three Gaussians that, while separated in n dimensions, will with high probability appear extremely close to being a mixture of two Gaussians when projected onto a random vector u. More generally, if there are k components, projections onto u may yield different numbers of components $k', k'' < k$ depending on the direction. Worse, this may happen even when comparing, say, n^2 projections that are only slightly different, as was done in Algorithm 3. Fortunately, this problem can be easily overcome. Say the first perturbed projection u_1 produces a k'-component univariate mixture. For one thing, we can ignore any projections that produce fewer than k' components. And, if a later projection u_i produces $k'' > k'$ components, then we should restart the procedure with $u = u_i$. In addition to finding n^2 univariate parameter estimates with the same number of components, we would also like these parameter estimates to be much more precise than the distances between the projections and have the components of each estimate be reasonably far apart. These two requirements ensure that components are identifiable across the n^2 estimates so that all the 1D estimates can be combined to reconstruct an estimate $\hat{\theta}$ of the parameters in n dimensions. The procedure for producing the n^2 estimates and reconstructing the n-dimensional estimate is called the Partition Pursuit Algorithm. Since the number of components k' in $\hat{\theta}$ may be less than k, we need to define in what sense $\hat{\theta}$ is a good estimate of the true parameters θ.

Definition 5.1. [Moitra and Valiant, 2010] Given a GMM F of k Gaussians parameterized by θ, a GMM $\hat{F}$ of $k' \leq k$ Gaussians parameterized by $\hat{\theta}$ is an **ϵ-correct subdivision of** F if there is a surjective map $\pi : [k] \to [k']$ such that

1. $\forall j \in [k'], |\sum_{i:\pi(i)=j} w_i - \hat{w}_j| \leq \epsilon$
2. $\forall i \in [k], D_p\big(\mathcal{N}(\mu_i, \Sigma_i), \mathcal{N}(\hat{\mu}_{\pi(i)}, \hat{\Sigma}_{\pi(i)})\big) \leq \epsilon$

Note that unlike in the definition of ϵ-close estimate, which relied on statistical distance, this definition is based on parameter distance.

The Partition Pursuit Algorithm cannot employ the Basic Univariate Algorithm directly, however, since the former can make no guarantees to the latter about parameter distances.

The Basic Univariate Algorithm expects a k-component univariate mixture and returns a k-component estimate. Since the components may in fact be arbitrarily close together, we must employ the General Univariate Algorithm, which returns an ϵ-correct subdivision of the univariate mixture as the Partition Pursuit Algorithm requires. To accomplish this, the Basic Univariate Algorithm is run repeatedly with different precisions. As long as all the pairwise parameter distances are much larger or smaller than the target precision, the Basic Univariate Algorithm will function as if it were given a mixture with possibly fewer than k components. Thus, if the precision values are chosen far enough apart, then each pairwise parameter distance can only corrupt one run. Therefore, the the majority of the parameters will agree with each other on each run, so the General Univariate Algorithm will be able to find a consensus amongst the runs and construct an ϵ-correct subdivision of the mixture.

The final Anisotropic Algorithm only requires an upper bound on the number of components in the mixture. To achieve this flexibility, once an ϵ-correct subdivision $\hat{F}$ of F is obtained, it must be determined whether 1) there is a components in $\hat{F}$ that corresponds to more than one component in F and 2) if so, which component in $\hat{F}$ this is. These problems are solved by the Hierarchical Clustering Algorithm. It turns out that with high probability, a component $\hat{F}_j$ will contain more than one true component if and only if its covariance matrix has a small eigenvalue. This eigenvalue can be used to partition $\mathbb{R}^n$ into two sets (A, B), such that one of the components in $\hat{F}_j$ is in A and another is in B. Moreover, with high probability, samples drawn from a component F_i of the true mixture will either be in A or in B. Depending on whether such an offending component is detected, the Hierarchical Clustering Algorithm either returns $\hat{F}$ or the partition. Thus, we obtain the following high-level algorithm for learning arbitrary GMMs.

Algorithm 4 Anisotropic Algorithm for learning GMMs [Moitra and Valiant, 2010]

Place the samples in isotropic position
Run the Hierarchical Clustering Algorithm (HCA)
if HCA returns $\hat{F}$ **then**
  return $\hat{F}$, which is an ϵ-close estimate of F with high probability
else {HCA returns partition a (A, B)}
  Draw an additional sample set S from sample oracle for F
  Run the Anisotropic Algorithm on the samples $S_A \subset S$ that are in A to get $\hat{F}_A$
  Run the Anisotropic Algorithm on the samples $S_B \subset S$ that are in B to get $\hat{F}_B$
  return $\hat{F} = \frac{|S_A|}{|S|}\hat{F}_A + \frac{|S_B|}{|S|}\hat{F}_B$
end if

The Anisotropic Algorithm and its subroutines use a number of interesting strategies. A familiar strategy also used by the spectral algorithms is to break the samples apart recursively. The spectral algorithms, however, try to find the samples that were generated by the "easy" components. The Anisotropic Algorithm, on the other hand, splits apart components that are close together and then analyzes them separately, which has the advantage of allowing it to handle components that are arbitrarily close together. Another technique which is used repeatedly is isolating limited failures. One instance is in the Projection Pursuit Algorithm, which restarts after a univariate estimate returns more components than a previous one and ignores estimates with too few components. In a sense the Hierarchical Clustering Algorithm also isolates (and exploits) failures by finding the estimated components which consist of more

than one true component, then splitting the data from merged components so that the failure cannot happen again.

5.2 Deterministic Projection

A different projection-based approach to learning arbitrary GMMs is taken in Belkin and Sinha [2010b]. They first develop some general results for polynomial families of distributions. Within this section, families of distributions will be parameterized by vector θ which belongs to a parameter set $\Theta \subset \mathbb{R}^m$. The parameter set is assumed to be compact and semi-algebraic (so closed and bounded). A set is semi-algebraic in $\mathbb{R}^m$ if it is the finite union of sets defined by a system of algebraic equations and inequalities. Although this way of thinking about θ is slightly different from the convention otherwise used, in practice it makes little difference. This is because the Frobenius norm of a matrix (used in most of the other results that analyze covariance matrices) is equivalent to the ℓ^2-norm of a flattened matrix (used in this subsection). The semi-algebraic requirement is not very limiting, as spheres, polytopes, and the set of semi-definite matrices are all examples of semi-algebraic sets.

The first set of key results apply to polynomial family distributions. Many distribution families of interest are polynomial, including the Gaussian, gamma, chi-square, and poisson distributions. Conveniently, the convex combinations of polynomial families form a polynomial family, so mixtures of polynomial distributions are also polynomial.

Definition 5.2. [Belkin and Sinha, 2010b] A family of probability density functions p_θ parameterized by θ is a **polynomial family** if each raw l-dimensional moment $M_{i_1,\ldots,i_l}(\theta) = \int x_1^{i_1} \cdots x_l^{i_l} dp_\theta$ of the distribution exists and can be represented as a polynomial of the parameters $(\theta^1, \ldots, \theta^m)$ and if p_θ is uniquely defined by its moments.

It is convenient to order the moments lexicographically and denote them by $M_1(\theta), \ldots, M_n(\theta), \ldots$, which corresponds to the standard ordering in the one-dimensional case. Polynomial families have a number of useful properties that make them amenable to analysis. Of particular interest is the following result, which will allow the method of moments to be used (cf. Feldman et al. [2006], Kalai et al. [2010], and Moitra and Valiant [2010]).

Theorem 5.3. *[Belkin and Sinha, 2010b] Let p_θ be a polynomial family of distributions. Then there exists some $N \in \mathbb{N}$ such that $p_{\theta_1} = p_{\theta_2}$ if and only if $M_i(\theta_1) = M_i(\theta_2)$ for all $i \in [N]$.*

The theorem hints that we should only need to compare a finite number of moments of a polynomial distribution in order to determine whether it closely matches the true distribution, whose moments are estimated from the data. In order to formalize this intuition, it is useful to define a generalized notion of an ϵ-ball in the parameter space that takes into account the case in which different parameters may have identical probability distributions. This occurs in the case of Gaussian mixtures since the components may be ordered arbitrarily. Let $\mathcal{E}(\theta) = \{\omega \in \Theta : p_\omega = p_\theta\}$ be the set of parameters which have identical distributions to p_θ. With these, we can construct an ϵ-"neighborhood" of θ, which is, roughly speaking, the union of neighborhoods around each point in $\mathcal{E}(\theta)$

$$\mathcal{N}(\theta, \epsilon) = \left\{\omega \in \Theta : \exists \omega', \theta' \in \Theta, 0 < \epsilon' < \epsilon \text{ s.t. } \omega' \in \mathcal{E}(\theta') \wedge \left|\left|\omega - \omega'\right|\right| < \epsilon' \wedge \left|\left|\theta - \theta'\right|\right| < \epsilon - \epsilon'\right\}.$$

We will be interested in finding a parameter estimate within the ϵ-neighborhood of the true parameters. Note that inclusion in $\mathcal{N}$ is symmetric: if $\theta_1 \in \mathcal{N}(\theta_2, \epsilon)$, then $\theta_2 \in \mathcal{N}(\theta_1, \epsilon)$.

The facts that $\mathcal{N}(\theta, \epsilon)$ and the parameter space Θ are semi-algebraic sets, that Θ is bounded, and that p_θ has polynomial moments are important because they allow for results from abstract algebra and real algebraic geometry to be utilized to understand the structure of these sets and the distributions' moments. In particular, these features can be used to derive a lower bound that guarantees that if $M_i(\theta_1)$ and $M_i(\theta_2)$ are sufficiently different for at least one $i \in [N]$, then θ_1 and θ_2 are not in each other's ϵ-neighborhoods. There is also an upper bound that is polynomial in the standard parameters on how different $M_i(\theta_1)$ and $M_i(\theta_2)$ can be. These bounds allow an estimate within the ϵ-neighborhood of the true parameters to be found with probability $1 - \delta$ by computing the first N empirical moments using $\text{poly}(\frac{1}{\epsilon}, \frac{1}{\delta})$ samples. After that a simple search over a rectangular grid of size $O(\frac{\epsilon^c}{N\sqrt{m}})$ will find parameters for a distribution with moments that closely match the empirical ones. Here c is a constant that depends on the polynomial family. A corollary to this result is based on the radius of identifiability $\mathcal{R}(\theta)$, which was introduced in Section 2.

Theorem 5.4. *[Belkin and Sinha, 2010b] For parameter vector $\theta \in \Theta$, if $\mathcal{E}(\theta) = \{\theta_1, \ldots, \theta_k\}$ is a finite set, then there exists an algorithm such that, given $\epsilon > 0$, outputs $\hat{\theta}$ within $\epsilon' = \min(\epsilon, \min_j \mathcal{R}(\theta_j))$ of θ_i for some $i \in [k]$ with probability $1 - \delta$ using a number of samples polynomial in $\frac{1}{\epsilon'}$ and $\frac{1}{\delta}$.*

Further work is required to apply this general result, since the number of parameters in a GMM grows with the dimension. Thus, the algorithm just outlined is not efficient when applied to n-dimensional data. In the case of a Gaussian mixture, however, it can be shown that there is guaranteed to be some $(2k^2)$-dimensional coordinate subspace W such that projection onto W shrinks $\mathcal{R}(\theta)$ by at most a factor of $1/n$. This subspace can be found by trying all $\binom{n}{2k^2}$ coordinate projections and estimating the parameters for each one using Theorem 5.4. It is possible to estimate the radius of identifiability for GMMs in polynomial time using these parameters, so we do this for each projection and choose the one with the largest $\mathcal{R}$. Theorem 5.4 allows us to estimate the parameters for the $2k^2$ coordinates in W with high accuracy. Let e_i denote the standard basis vector for the i-th coordinate. The rest of the means can be estimated by projecting onto $W_i = \text{span}(W, e_i)$ for coordinates $e_i \notin W$. The remaining entries in the covariance matrices can be estimated by projecting onto $W_{ij} = \text{span}(W, e_i)$ where either $e_i \notin W$ or $e_j \notin W$. This procedure is summarized in Algorithm 5.

There is an appealing simplicity and potentially wide applicability to this deterministic projection approach. But, unfortunately, Belkin and Sinha [2010b] do not quite provide a complete algorithm since they do not actually give values for the constants c and N, which are needed to implement the algorithm. Their theorems only prove the *existence* of such constants, but provide no direction as to how one might calculate them for a polynomial family of interest.

6 Conclusion and Future Directions

With the positive results of Moitra and Valiant [2010] and Belkin and Sinha [2010b], along with the negative result of Moitra and Valiant [2010] on the exponential dependence of learning

Algorithm 5 Learning polynomial families [Belkin and Sinha, 2010b]

Find the $2k^2$ dimensional coordinate subspace W with maximum empirical $\mathcal{R}(\theta)$
Project the samples S onto W and estimate the means and covariance entries for these $2k^2$ coordinates (by Theorem 5.4)
for all $e_i \notin W$ **do**
  $W_i \leftarrow \text{span}(W, e_i)$
  project S onto W_i
  estimate the component variances and means along e_i (by Theorem 5.4)
  for all $e_j \notin W$ **do**
    $W_{ij} \leftarrow \text{span}(W, e_i, e_j)$
    project S onto W_{ij}
    estimate the component covariances between e_i and e_j (by Theorem 5.4)
  end for
end for

on the number of components, the problem of probably learning Gaussian mixture models has arguably been solved. Certainly both these general methods would benefit from efficiency gains to make them more attractive for practical implementations, so research is this area will undoubtably continue. Given this fairly happy state of affairs, however, one natural question to ask is how one might learn GMMs if the number of components is unknown. If there is a reasonable upper bound on the k, then the Anisotropic Algorithm can be used. It also seems plausible that running any of the algorithms presented here with $k = 1, 2, \ldots$ and analyzing the likelihood of each parameter estimate, one could choose the right k with high probability. This procedure could be very slow given the exponential dependence on k and the need (most likely) to run the algorithm for values of k larger than the true one.

Another logical direction for further research is finding ways to provably learn other distributions and mixtures. GMMs serve as good starting point because they are so widely used, but there are many other distributions of interest. Some work has already been done in this area, such as learning mixtures of heavy tailed distributions [Dasgupta et al., 2005, Chaudhuri and Rao, 2008a] and mixtures of product distributions [Chaudhuri and Rao, 2008b, Feldman et al., 2008]. In addition, we have already discussed spectral results for mixtures of log-concave distributions. Among the works discussed in detail, that of Belkin and Sinha [2010b] seems to have the greatest potential for generalization. In particular, a promising approach would be to combine the positive results related to projecting log-concave distributions, such as those in Kannan et al. [2005], with the results for polynomial families. One possibility is to project onto the top principal components and develop maximal ϵ-neighborhood shrinkage results for this projection. Then the method of moments could be used via Theorem 5.4, although a more sophisticated parameter reconstruction algorithm than the one that was used in Algorithm 5 would be required. It worth noting that the classes of log-concave and polynomial distributions have a non-trivial intersection—the Gaussian, gamma (with shape parameter ≥ 1), chi-square, and Laplace distributions all fall in both categories—so such an algorithm could have wide applicability.

Author	Min. Separation	Mixture Class	Method	Comments
Dasgupta [1999]	$\sqrt{n}$	Gaussian with shared covariance matrix	Random projection	
Dasgupta and Schulman [2000]	$n^{\frac{1}{4}}$	Spherical Gaussian	EM	
Arora and Kennan [2001]	$n^{\frac{1}{4}}$	Gaussian	Distance-based	
Vempala and Wang [2002]	$k^{\frac{1}{4}}$	Spherical Gaussian	Spectral, distance-based	
Kannan et al. [2005]	$k^{\frac{3}{2}}/w_{min}^2$	Log-concave	Spectral, distance-based	need to know w_i
Achlioptas and McSherry [2005]	$k + \sqrt{k \log n}$	Gaussian	Spectral	
Feldman et al. [2006]	> 0	Axis aligned Gaussians	Method of moments (MoM)	no parameter estimation
Belkin and Sinha [2010a]	> 0	Identical spherical Gaussian	Spectral	
Kalai et al. [2010]	≥ 0	Gaussian with two components	Random projections, MoM	
Moitra and Valiant [2010]	≥ 0	Gaussian	Random projections, MoM	
Belkin and Sinha [2010b]	≥ 0	Gaussian	Deterministic projections, MoM	

Table 1: Comparison of some methods for learning GMMs. The final three methods allow means to be the same as long as the associated covariances are different.

References

D. Achlioptas and F. McSherry. On spectral learning of mixtures of distributions. In *COLT*, 2005.

S. Arora and R. Kennan. Learning mixtures of arbitrary Gaussians. In *STOC*, 2001.

M. Belkin and K. Sinha. Toward learning Gaussian mixtures with arbitrary separation. In *COLT*, 2010a.

M. Belkin and K. Sinha. Polynomial learning of distribution families. In *FOCS*, 2010b.

K. Chaudhuri and S. Rao. Beyond Gaussians: Spectral methods for learning mixtures of heavy tailed distributions. In *COLT*, 2008a.

K. Chaudhuri and S. Rao. Learning mixtures of product distributions using correlations and independence. In *COLT*, 2008b.

S. Dasgupta. Learning mixtures of Gaussians. In *FOCS*, 1999.

S. Dasgupta and L. Schulman. A two round variant of em for gaussian mixtures. In *Conference on Uncertainty in Artificial Intelligence*, 2000.

S. Dasgupta, J. E. Hopcroft, J. Kleinberg, and M. Sandler. On learning mixture of heavy tailed distributions. In *FOCS*, 2005.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. Ser. B*, 39:1–38, 1977.

J. Feldman, R. A. Servedio, and R. O'Donnell. Pac learning axis aligned mixtures of Gaussians with no separation assumption. In *COLT*, 2006.

J. Feldman, R. O'Donnell, and R. A. Servedio. Learning mixture of product distributions over discrete domains. *SIAM Journal of Computing*, 37(5):1536–1564, 2008.

A. T. Kalai, A. Moitra, and G. Valiant. Efficiently learning mixtures of two gaussians. In *STOC*, 2010.

R. Kannan, H. Salamasian, and S. Vempala. The spectral method for general mixture models. In *COLT*, 2005.

A. Moitra and G. Valiant. Settling the polynomial learnability of mixtures of gaussians. In *FOCS*, 2010.

S. Vempala and G. Wang. A spectral algorithm for learning mixture models. In *FOCS*, 2002.