JMLR: Workshop and Conference Proceedings vol 35:1–31, 2014

Faster and Sample Near-Optimal Algorithms for Proper Learning Mixtures of Gaussians

Constantinos Daskalakis[*] COSTIS@MIT.EDU
32 Vassar Street, Cambridge MA 02139

Gautam Kamath[†] G@CSAIL.MIT.EDU
32 Vassar Street, Cambridge MA 02139

Abstract

We provide an algorithm for properly learning mixtures of two single-dimensional Gaussians without any separability assumptions. Given $\tilde{O}(1/\varepsilon^2)$ samples from an unknown mixture, our algorithm outputs a mixture that is ε-close in total variation distance, in time $\tilde{O}(1/\varepsilon^5)$. Our sample complexity is optimal up to logarithmic factors, and significantly improves upon both Kalai et al. (2010), whose algorithm has a prohibitive dependence on $1/\varepsilon$, and Feldman et al. (2006), whose algorithm requires bounds on the mixture parameters and depends pseudo-polynomially in these parameters.

One of our main contributions is an improved and generalized algorithm for selecting a good candidate distribution from among competing hypotheses. Namely, given a collection of N hypotheses containing at least one candidate that is ε-close to an unknown distribution, our algorithm outputs a candidate which is $O(\varepsilon)$-close to the distribution. The algorithm requires $O(\log N/\varepsilon^2)$ samples from the unknown distribution and $O(N \log N/\varepsilon^2)$ time, which improves previous such results (such as the Scheffé estimator) from a quadratic dependence of the running time on N to quasilinear. Given the wide use of such results for the purpose of hypothesis selection, our improved algorithm implies immediate improvements to any such use.

Keywords: Gaussian mixture models, proper learning, robust statistics, hypothesis selection

1. Introduction

Learning mixtures of Gaussian distributions is one of the most fundamental problems in Statistics, with a multitude of applications in the natural and social sciences, which has recently received considerable attention in Computer Science literature. Given independent samples from an unknown mixture of Gaussians, the task is to 'learn' the underlying mixture.

In one version of the problem, 'learning' means estimating the *parameters* of the mixture, that is the mixing probabilities as well as the parameters of each constituent Gaussian. The most popular heuristic for doing so is running the EM algorithm on samples from the mixture (Dempster et al., 1977), albeit no rigorous guarantees are known for it in general.

A line of research initiated by Dasgupta (Dasgupta, 1999; Arora and Kannan, 2001; Vempala and Wang, 2002; Achlioptas and McSherry, 2005; Brubaker and Vempala, 2008) provides rigorous guarantees under separability conditions: roughly speaking, it is assumed that the constituent Gaussians have variation distance bounded away from 0 (indeed, in some cases, distance exponentially

[†] Supported by a Sloan Foundation Fellowship, a Microsoft Research Faculty Fellowship, and NSF Award CCF-0953960 (CAREER) and CCF-1101491.

[†] Part of this work was done while the author was supported by an Akamai Presidential Fellowship.

close to 1). This line of work was recently settled by a triplet of breakthrough results (Kalai et al., 2010; Moitra and Valiant, 2010; Belkin and Sinha, 2010), establishing the polynomial solvability of the problem under minimal separability conditions for the parameters to be recoverable in the first place: for any $\varepsilon > 0$, polynomial in n and $1/\varepsilon$ samples from a mixture of n-dimensional Gaussians suffice to recover the parameters of the mixture in $\mathrm{poly}(n, 1/\varepsilon)$ time.

While these results settle the polynomial solvability of the problem, they serve more as a proof of concept in that their dependence on $1/\varepsilon$ is quite expensive.[1] Indeed, even for mixtures of two single-dimensional Gaussians, a practically efficient algorithm for this problem is unknown.

A weaker goal for the learner of Gaussian mixtures is this: given samples from an unknown mixture, find *any* mixture that is close to the unknown one, for some notion of closeness. This PAC-style version of the problem (Kearns et al., 1994) was pursued by Feldman et al. (2006) who obtained efficient learning algorithms for mixtures of n-dimensional, axis-aligned Gaussians. Given $\mathrm{poly}(n, 1/\varepsilon, L)$ samples from such mixture, their algorithm constructs a mixture whose KL divergence to the sampled one is at most ε. Unfortunately, the sample and time complexity of their algorithm depends polynomially on a (priorly known) bound L, determining the range of the means and variances of the constituent Gaussians in every dimension.[2] In particular, the algorithm has pseudo-polynomial dependence on L where there shouldn't be any dependence on L at all (Kalai et al., 2010; Moitra and Valiant, 2010; Belkin and Sinha, 2010).

Finally, yet a weaker goal for the learner would be to construct *any distribution* that is close to the unknown mixture. In this *non-proper* version of the problem the learner is not restricted to output a Gaussian mixture, but can output any (representation of a) distribution that is close to the unknown mixture. For this problem, recent results of Chan et al. (2014) provide algorithms for single-dimensional mixtures, whose sample complexity has near-optimal dependence on $1/\varepsilon$. Namely, given $\tilde{O}(1/\varepsilon^2)$ samples from a single-dimensional mixture, they construct a piecewise polynomial distribution that is ε-close in total variation distance.

Inspired by this recent progress on non-properly learning single-dimensional mixtures, our goal in this paper is to provide sample-optimal algorithms that *properly learn*. We obtain such algorithms for mixtures of two single-dimensional Gaussians. Namely,

Theorem 1 *For all $\varepsilon, \delta > 0$, given $\tilde{O}(\log(1/\delta)/\varepsilon^2)$ independent samples from an arbitrary mixture F of two univariate Gaussians we can compute in time $\tilde{O}(\log^3(1/\delta)/\varepsilon^5)$ a mixture F' such that $d_{\mathrm{TV}}(F, F') \leq \varepsilon$ with probability at least $1 - \delta$. The expected running time of this algorithm is $\tilde{O}(\log^2(1/\delta)/\varepsilon^5)$.*

We note that learning a univariate mixture often lies at the heart of learning multivariate mixtures (Kalai et al., 2010; Moitra and Valiant, 2010), so it is important to understand this fundamental case.

Discussion. Note that our algorithm makes no separability assumptions about the constituent Gaussians of the unknown mixture, nor does it require or depend on a bound on the mixture's parameters. Also, because the mixture is single-dimensional it is not amenable to the techniques of Hsu and Kakade (2013). Moreover, it is easy to see that our sample complexity is optimal up

1. For example, the single-dimensional algorithm in the heart of Kalai et al. (2010) has sample and time complexity of $\Theta(1/\varepsilon^{300})$ and $\Omega(1/\varepsilon^{1377})$ respectively (even though the authors most certainly did not intend to optimize their constants).
2. In particular, it is assumed that every constituent Gaussian in every dimension has mean $\mu \in [-\mu_{\max}, \mu_{\max}]$ and variance $\sigma^2 \in [\sigma^2_{\min}, \sigma^2_{\max}]$ where $\mu_{\max}\sigma_{\max}/\sigma_{\min} \leq L$.

to logarithmic factors. Indeed, a Gaussian mixture can trivially simulate a Bernoulli distribution as follows. Let Z be a Bernoulli random variable that is 0 with probability $1-p$ and 1 with probability p. Clearly, Z can be viewed as a mixture of two Gaussian random variables of 0 variance, which have means 0 and 1 and are mixed with probabilities $1-p$ and p respectively. It is known that $1/\varepsilon^2$ samples are needed to properly learn a Bernoulli distribution, hence this lower bound immediately carries over to Gaussian mixtures.

Approach. Our algorithm is intuitively quite simple, although some care is required to make the ideas work. First, we can guess the mixing weight up to additive error $O(\varepsilon)$, and proceed with our algorithm pretending that our guess is correct. Every guess will result in a collection of candidate distributions, and the final step of our algorithm is a tournament that will select, from among all candidate distributions produced in the course of our algorithm, a distribution that is ε-close to the unknown mixture, if such a distribution exists. To do this we will make use of the following theorem which is our second main contribution in this paper. (See Theorem 21 for a precise statement.)

Informal Theorem 21. *There exists an algorithm* `FastTournament` *that takes as input sample access to an unknown distribution X and a collection of candidate hypothesis distributions $H_1, \ldots, H_N$, as well as an accuracy parameter $\varepsilon > 0$, and has the following behavior: if there exists some distribution among $H_1, \ldots, H_N$ that is ε-close to X, then the distribution output by the algorithm is $O(\varepsilon)$-close to X. Moreover, the number of samples drawn by the algorithm from each of the distributions is $O(\log N/\varepsilon^2)$ and the running time is $O(N \log N/\varepsilon^2)$.*

Devising a hypothesis selection algorithm with the performance guarantees of Theorem 21 requires strengthening the Scheffé-estimate-based approach described in Chapter 6 of Devroye and Lugosi (2001) (see Yatracos (1985); Devroye and Lugosi (1996, 1997), as well as the recent papers of Daskalakis et al. (2012) and Chan et al. (2013)) to continuous distributions whose crossings are difficult to compute exactly as well as to sample-only access to all involved distributions, but most importantly improving the running time to almost linear in the number N of candidate distributions. Further comparison of our new hypothesis selection algorithm to related work is provided in Section 4. It is also worth noting that the tournament based approach of Feldman et al. (2006) cannot be used for our purposes in this paper as it would require a priorly known bound on the mixture's parameters and would depend pseudopolynomially on this bound.

Tuning the number of samples according to the guessed mixing weight, we proceed to draw samples from the unknown mixture, expecting that some of these samples will fall sufficiently close to the means of the constituent Gaussians, where the closeness will depend on the number of samples drawn as well as the unknown variances. We guess which sample falls close to the mean of the constituent Gaussian that has the smaller value of σ/w (standard deviation to mixing weight ratio), which gives us the second parameter of the mixture. To pin down the variance of this Gaussian, we implement a natural idea. Intuitively, if we draw samples from the mixture, we expect that the constituent Gaussian with the smallest σ/w will determine the smallest distance among the samples. Pursuing this idea we produce a collection of variance candidates, one of which truly corresponds to the variance of this Gaussian, giving us a third parameter.

At this point, we have a complete description of one of the component Gaussians. If we could remove this component from the mixture, we would be left with the remaining unknown Gaussian. Our approach is to generate an empirical distribution of the mixture and "subtract out" the component that we already know, giving us an approximation to the unknown Gaussian. For the

purposes of estimating the two parameters of this unknown Gaussian, we observe that the most traditional estimates of location and scale are unreliable, since the error in our approximation may cause probability mass to be shifted to arbitrary locations. Instead, we use robust statistics to obtain approximations to these two parameters.

The empirical distribution of the mixture is generated using the Dvoretzky-Kiefer-Wolfowitz (DKW) inequality (Dvoretzky et al., 1956). With $O(1/\varepsilon^2)$ samples from an *arbitrary* distribution, this algorithm generates an ε-approximation to the distribution (with respect to the Kolmogorov metric). Since this result applies to arbitrary distributions, it generates a hypothesis that is weak, in some senses, including the choice of distance metric. In particular, the hypothesis distribution output by the DKW inequality is discrete, resulting in a total variation distance of 1 from a mixture of Gaussians (or any other continuous distribution), regardless of the accuracy parameter ε. Thus, we consider it to be interesting that such a weak hypothesis can be used as a tool to generate a stronger, proper hypothesis. We note that the Kolmogorov distance metric is not special here - an approximation with respect to other reasonable distance metrics may be substituted in, as long as the description of the hypothesis is efficiently manipulable in the appropriate ways.

We show that, for any target total variation distance ε, the number of samples required to execute the steps outlined above in order to produce a collection of candidate hypotheses one of which is ε-close to the unknown mixture, as well as to run the tournament to select from among the candidate distributions are $\tilde{O}(1/\varepsilon^2)$. The running time is $\tilde{O}(1/\varepsilon^5)$.

Comparison to Prior Work on Learning Gaussian Mixtures. In comparison to the recent breakthrough results (Kalai et al., 2010; Moitra and Valiant, 2010; Belkin and Sinha, 2010), our algorithm has near-optimal sample complexity and much milder running time, where these results have quite expensive dependence of both their sample and time complexity on the accuracy ε, even for single-dimensional mixtures.[3] On the other hand, our algorithm has weaker guarantees in that we properly learn but don't do parameter estimation. In comparison to Feldman et al. (2006), our algorithm requires no bounds on the parameters of the constituent Gaussians and exhibits no pseudo-polynomial dependence of the sample and time complexity on such bounds. On the other hand, we learn with respect to the total variation distance rather than the KL divergence. Finally, compared to Chan et al. (2013, 2014), we properly learn while they non-properly learn and we both have near-optimal sample complexity.

Recently and independently, Acharya et al. (2014a) have also provided algorithms for properly learning spherical Gaussian mixtures. Their primary focus is on the high dimensional case, aiming at a near-linear sample dependence on the dimension. Our focus is instead on optimizing the dependence of the sample and time complexity on ε in the one-dimensional case.

In fact, Acharya et al. (2014a) also study mixtures of k Gaussians in one dimension, providing a proper learning algorithm with near-optimal sample complexity of $\tilde{O}\left(k/\varepsilon^2\right)$ and running time $\tilde{O}_k\left(1/\varepsilon^{3k+1}\right)$. Specializing to two single-dimensional Gaussians ($k = 2$), their algorithm has near-optimal sample complexity, like ours, but is slower by a factor of $O(1/\varepsilon^2)$ than ours. We also remark that, through a combination of techniques from our paper and theirs, a proper learning algorithm for mixtures of k Gaussians can be obtained, with near-optimal sample complexity of $\tilde{O}\left(k/\varepsilon^2\right)$ and running time $\tilde{O}_k\left(1/\varepsilon^{3k-1}\right)$, improving by a factor of $O(1/\varepsilon^2)$ the running time of their k-Gaussian algorithm. Roughly, this algorithm creates candidate distributions in which the parameters of the

3. For example, compared to Kalai et al. (2010) we improve by a factor of at least 150 the exponent of both the sample and time complexity.

first $k-1$ components are generated using methods from Acharya et al. (2014a), and the parameters of the final component are determined using our robust statistical techniques, in which we "subtract out" the first $k-1$ components and robustly estimate the mean and variance of the remainder.

2. Preliminaries

Let $\mathcal{N}(\mu, \sigma^2)$ represent the univariate normal distribution, with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}$, with density function

$$\mathcal{N}(\mu, \sigma^2, x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

The univariate half-normal distribution with parameter σ^2 is the distribution of $|Y|$ where Y is distributed according to $\mathcal{N}(0, \sigma^2)$. The CDF of the half-normal distribution is

$$F(\sigma, x) = \text{erf}\left(\frac{x}{\sigma\sqrt{2}}\right),$$

where $\text{erf}(x)$ is the error function, defined as

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \, \mathrm{d}t.$$

We also make use of the complement of the error function, $\text{erfc}(x)$, defined as $\text{erfc}(x) = 1 - \text{erf}(x)$.

A Gaussian mixture model (GMM) of distributions $\mathcal{N}_1(\mu_1, \sigma_1^2), \ldots, \mathcal{N}_n(\mu_n, \sigma_n^2)$ has PDF

$$f(x) = \sum_{i=1}^{n} w_i \mathcal{N}(\mu_i, \sigma_i^2, x),$$

where $\sum_i w_i = 1$. These w_i are referred to as the mixing weights. Drawing a sample from a GMM can be visualized as the following process: select a single Gaussian, where the probability of selecting a Gaussian is equal to its mixing weight, and draw a sample from that Gaussian. In this paper, we consider mixtures of two Gaussians, so $w_2 = 1 - w_1$. We will interchangeably use w and $1 - w$ in place of w_1 and w_2.

The total variation distance between two probability measures P and Q on a σ-algebra F is defined by

$$d_{\text{TV}}(P, Q) = \sup_{A \in F} |P(A) - Q(A)| = \frac{1}{2}\|P - Q\|_1.$$

For simplicity in the exposition of our algorithm, we make the standard assumption (see, e.g., Feldman et al. (2006); Kalai et al. (2010)) of infinite precision real arithmetic. In particular, the samples we draw from a mixture of Gaussians are real numbers, and we can do exact computations on real numbers, e.g., we can exactly evaluate the PDF of a Gaussian distribution on a real number.

2.1. Bounds on Total Variation Distance for GMMs

We recall a result from Daskalakis et al. (2013):

Proposition 2 (Proposition B.4 of Daskalakis et al. (2013)) *Let $\mu_1, \mu_2 \in \mathbb{R}$ and $0 \leq \sigma_1 \leq \sigma_2$. Then*

$$d_{\text{TV}}(\mathcal{N}(\mu_1, \sigma_1^2), \mathcal{N}(\mu_2, \sigma_2^2)) \leq \frac{1}{2}\left(\frac{|\mu_1 - \mu_2|}{\sigma_1} + \frac{\sigma_2^2 - \sigma_1^2}{\sigma_1^2}\right).$$

The following proposition, whose proof is deferred to the appendix, provides a bound on the total variation distance between two GMMs in terms of the distance between the constituent Gaussians.

Proposition 3 *Suppose we have two GMMs X and Y, with PDFs $w\mathcal{N}_1 + (1-w)\mathcal{N}_2$ and $\hat{w}\hat{\mathcal{N}}_1 + (1-\hat{w})\hat{\mathcal{N}}_2$ respectively. Then $d_{\mathrm{TV}}(X,Y) \le |w-\hat{w}| + wd_{\mathrm{TV}}(\mathcal{N}_1,\hat{\mathcal{N}}_1) + (1-w)d_{\mathrm{TV}}(\mathcal{N}_2,\hat{\mathcal{N}}_2)$.*

Combining these propositions, we obtain the following lemma:

Lemma 4 *Let X and Y by two GMMs with PDFs $w_1\mathcal{N}_1 + w_2\mathcal{N}_2$ and $\hat{w}_1\hat{\mathcal{N}}_1 + \hat{w}_2\hat{\mathcal{N}}_2$ respectively, where $|w_i - \hat{w}_i| \le O(\varepsilon)$, $|\mu_i - \hat{\mu}_i| \le O(\frac{\varepsilon}{w_i})\sigma_i \le O(\varepsilon)\sigma_i$, $|\sigma_i - \hat{\sigma}_i| \le O(\frac{\varepsilon}{w_i})\sigma_i \le O(\varepsilon)\sigma_i$, for all i such that $w_i \ge \frac{\varepsilon}{25}$. Then $d_{\mathrm{TV}}(X,Y) \le \varepsilon$.*

2.2. Kolmogorov Distance

In addition to total variation distance, we will also use the Kolmogorov distance metric.

Definition 5 *The* Kolmogorov distance *between two probability measures with CDFs F_X and F_Y is $d_{\mathrm{K}}(F_X,F_Y) = \sup_{x\in\mathbb{R}} |F_X(x) - F_Y(x)|$.*

We will also use this metric to compare general functions, which may not necessarily be valid CDFs.

We have the following fact, stating that total variation distance upper bounds Kolmogorov distance (Gibbs and Su, 2002).

Fact 6 $d_{\mathrm{K}}(F_X,F_Y) \le d_{\mathrm{TV}}(f_X,f_Y)$

Fortunately, it is fairly easy to learn with respect to the Kolmogorov distance, due to the Dvoretzky-Kiefer-Wolfowitz (DKW) inequality (Dvoretzky et al., 1956).

Theorem 7 (Dvoretzky et al. (1956),Massart (1990)) *Suppose we have n IID samples $X_1,\dots X_n$ from a probability distribution with CDF F. Let $F_n(x) = \frac{1}{n}\sum_{i=1}^n \mathbf{1}_{\{X_i\le x\}}$ be the empirical CDF. Then $\Pr[d_{\mathrm{K}}(F,F_n) \ge \varepsilon] \le 2e^{-2n\varepsilon^2}$. In particular, if $n = \Omega((1/\varepsilon^2)\cdot\log(1/\delta))$, then $\Pr[d_{\mathrm{K}}(F,F_n)\ge\varepsilon] \le \delta$.*

2.3. Representing and Manipulating CDFs

We will need to be able to efficiently represent and query the CDF of probability distributions we construct. This will be done using a data structure we denote the *n-interval partition* representation of a distribution. This allows us to represent a discrete random variable X over a support of size $\le n$. Construction takes $\tilde{O}(n)$ time, and at the cost of $O(\log n)$ time per operation, we can compute $F_X^{-1}(x)$ for $x\in[0,1]$. Full details will be provided in Appendix D.

Using this construction and Theorem 7, we can derive the following proposition:

Proposition 8 *Suppose we have $n = \Theta(\frac{1}{\varepsilon^2}\cdot\log\frac{1}{\delta})$ IID samples from a random variable X. In $\tilde{O}\left(\frac{1}{\varepsilon^2}\cdot\log\frac{1}{\delta}\right)$ time, we can construct a data structure which will allow us to convert independent samples from the uniform distribution over $[0,1]$ to independent samples from a random variable $\hat{X}$, such that $d_{\mathrm{K}}\left(F_X,F_{\hat{X}}\right) \le \varepsilon$ with probability $1-\delta$.*

Over the course of our algorithm, it will be natural to subtract out a component of a distribution.

Lemma 9 *Suppose we have access to the n-interval partition representation of a CDF F, and that there exists a weight w and CDFs G and H such that $d_{\mathrm{K}}\left(H, \frac{F-wG}{1-w}\right) \leq \varepsilon$. Given w and G, we can compute the n-interval partition representation of a distribution $\hat{H}$ such that $d_{\mathrm{K}}(H, \hat{H}) \leq \varepsilon$ in $O(n)$ time.*

A proof and full details are provided in Appendix D.

2.4. Robust Statistics

We use two well known robust statistics, the median and the interquartile range. These are suited to our application for two purposes. First, they are easy to compute with the n-interval partition representation of a distribution. Each requires a constant number of queries of the CDF at particular values, and the cost of each query is $O(\log n)$. Second, they are robust to small modifications with respect to most metrics on probability distributions. In particular, we will demonstrate their robustness on Gaussians when considering distance with respect to the Kolmogorov metric.

Lemma 10 *Let $\hat{F}$ be a distribution such that $d_{\mathrm{K}}(\mathcal{N}(\mu, \sigma^2), \hat{F}) \leq \varepsilon$, where $\varepsilon < \frac{1}{8}$. Then $med(\hat{F}) \triangleq \hat{F}^{-1}(\frac{1}{2}) \in [\mu - 2\sqrt{2}\varepsilon\sigma, \mu + 2\sqrt{2}\varepsilon\sigma]$.*

Lemma 11 *Let $\hat{F}$ be a distribution such that $d_{\mathrm{K}}(\mathcal{N}(\mu, \sigma^2), \hat{F}) \leq \varepsilon$, where $\varepsilon < \frac{1}{8}$. Then $\frac{IQR(\hat{F})}{2\sqrt{2}erf^{-1}(\frac{1}{2})} \triangleq \frac{\hat{F}^{-1}(\frac{3}{4}) - \hat{F}^{-1}(\frac{1}{4})}{2\sqrt{2}erf^{-1}(\frac{1}{2})} \in \left[\sigma - \frac{5}{2erf^{-1}(\frac{1}{2})}\varepsilon\sigma, \sigma + \frac{7}{2erf^{-1}(\frac{1}{2})}\varepsilon\sigma\right]$.*

The proofs are deferred to Appendix B.

2.5. Outline of the Algorithm

We can decompose our algorithm into two components: generating a collection of candidate distributions containing at least one candidate with low statistical distance to the unknown distribution (Theorem 12), and identifying such a candidate from this collection (Theorem 21).

Generation of Candidate Distributions: In Section 3, we deal with generation of candidate distributions. A *candidate distribution* is described by the parameter set $(\hat{w}, \hat{\mu}_1, \hat{\sigma}_1, \hat{\mu}_2, \hat{\sigma}_2)$, which corresponds to the GMM with PDF $f(x) = \hat{w}\mathcal{N}(\hat{\mu}_1, \hat{\sigma}_1^2, x) + (1 - \hat{w})\mathcal{N}(\hat{\mu}_2, \hat{\sigma}_2^2, x)$. As suggested by Lemma 4, if we have a candidate distribution with sufficently accurate parameters, it will have low statistical distance to the unknown distribution. Our first goal will be to generate a collection of candidates that contains at least one such candidate. Since the time complexity of our algorithm depends on the size of this collection, we wish to keep it to a minimum.

At a high level, we sequentially generate candidates for each parameter. In particular, we start by generating candidates for the mixing weight. While most of these will be inaccurate, we will guarantee to produce at least one appropriately accurate candidate $\hat{w}^*$. Then, for each candidate mixing weight, we will generate candidates for the mean of one of the Gaussians. We will guarantee that, out of the candidate means we generated for $\hat{w}^*$, it is likely that at least one candidate $\hat{\mu}_1^*$ will be sufficiently close to the true mean for this component. The candidate means that were generated for other mixing weights have no such guarantee. We use a similar sequential approach to generate candidates for the variance of this component. Once we have a description of the first component, we simulate the process of subtracting it from the mixture, thus giving us a single

Gaussian, whose parameters we can learn. We can not immediately identify which candidates have inaccurate parameters, and they serve only to inflate the size of our collection.

At a lower level, our algorithm starts by generating candidates for the mixing weight followed by generating candidates for the mean of the component with the smaller value of $\frac{\sigma_i}{w_i}$. Note that we do not know which of the two Gaussians this is. The solution is to branch our algorithm, where each branch assumes a correspondence to a different Gaussian. One of the two branches is guaranteed to be correct, and it will only double the number of candidate distributions. We observe that if we take n samples from a single Gaussian, it is likely that there will exist a sample at distance $O(\frac{\sigma}{n})$ from its mean. Thus, if we take $\Theta(\frac{1}{w_i\varepsilon})$ samples from the mixture, one of them will be sufficiently close to the mean of the corresponding Gaussian. Exploiting this observation we obtain candidates for the mixing weight and the first mean as summarized by Lemma 15.

Next, we generate candidates for the variance of this Gaussian. Our specific approach is based on the observation that given n samples from a single Gaussian, the minimum distance of a sample to the mean will likely be $\Theta(\frac{\sigma}{n})$. In the mixture, this property will still hold for the Gaussian with the smaller $\frac{\sigma_i}{w_i}$, so we extract this statistic and use a grid around it to generate sufficiently accurate candidates for σ_i. This is Lemma 17.

At this point, we have a complete description of one of the component Gaussians. Also, we can generate an empirical distribution of the mixture, which gives an adequate approximation to the true distribution. Given these two pieces, we update the empirical distribution by removing probability mass contributed by the known component. When done carefully, we end up with an approximate description of the distribution of the unknown component. At this point, we extract the median and the interquartile range (IQR) of the resulting distribution. These statistics are robust, so they can tolerate error in our approximation. Finally, the median and IQR allow us to derive the last mean and variance of our distribution. This is Lemma 19.

Putting everything together, we obtain the following result whose proof is in Section 3.5.

Theorem 12 *For all $\varepsilon, \delta > 0$, given $\log(1/\delta) \cdot O(1/\varepsilon^2)$ independent samples from an arbitrary mixture F of two univariate Gaussians, we can generate a collection of $\log(1/\delta)\cdot\tilde{O}(1/\varepsilon^3)$ candidate mixtures of two univariate Gaussians, containing at least one candidate F' such that $d_{\mathrm{TV}}(F, F') < \varepsilon$ with probability at least $1 - \delta$.*

Candidate Selection: In view of Theorem 12, to prove our main result it suffices to select from among the candidate mixtures some mixture that is close to the unknown mixture. In Section 4, we describe a tournament-based algorithm (Theorem 21) for identifying a candidate which has low statistical distance to the unknown mixture, concluding the proof of Theorem 1. See our discussion in Section 4 for the challenges arising in obtaining a tournament-based algorithm for continuous distributions whose crossings are difficult to compute exactly, as well as in speeding up the tournament's running time to almost linear in the number of candidates.

3. Generating Candidate Distributions

By Proposition 3, if one of the Gaussians of a mixture has a negligible mixing weight, it has a negligible impact on the mixture's statistical distance to the unknown mixture. Hence, the candidate means and variances of this Gaussian are irrelevant. This is fortunate, since if $\min(w, 1-w) << \varepsilon$ and we only draw $O(1/\varepsilon^2)$ samples from the unknown mixture, as we are planning to do, we have no hope of seeing a sufficient number of samples from the low-weight Gaussian to perform accurate

statistical tests for it. So for this section we will assume that $\min(w, 1-w) \geq \Omega(\varepsilon)$ and we will deal with the other case separately.

3.1. Generating Mixing Weight Candidates

The first step is to generate candidates for the mixing weight. We can obtain a collection of $O(\frac{1}{\varepsilon})$ candidates containing some $\hat{w}^* \in [w-\varepsilon, w+\varepsilon]$ by simply taking the set $\{t\varepsilon \mid t \in [\frac{1}{\varepsilon}]\}$.

3.2. Generating Mean Candidates

The next step is to generate candidates for the mean corresponding to the Gaussian with the smaller value of $\frac{\sigma_i}{w_i}$. Note that, a priori, we do not know whether $i = 1$ or $i = 2$. We try both cases, first generating candidates assuming they correspond to μ_1, and then repeating with μ_2. This will multiply our total number of candidate distributions by a factor of 2. Without loss of essential generality, assume for this section that $i = 1$.

We want a collection of candidates containing $\hat{\mu}_1^*$ such that $\mu_1 - \varepsilon\sigma_1 \leq \hat{\mu}_1^* \leq \mu_1 + \varepsilon\sigma_1$. The following propositions are straightforward and proved in Appendix C.

Proposition 13 *Fix $i \in \{1,2\}$. Given $\frac{20\sqrt{2}}{3w_i\varepsilon}$ samples from a GMM, there will exist a sample $\hat{\mu}_i^* \in \mu_i \pm \varepsilon\sigma_i$ with probability $\geq \frac{99}{100}$.*

Proposition 14 *Fix $i \in \{1,2\}$. Suppose $w_i - \varepsilon \leq \hat{w}_i \leq w_i + \varepsilon$, and $w_i \geq \varepsilon$. Then $\frac{2}{\hat{w}_i} \geq \frac{1}{w_i}$.*

We use these facts to design a simple algorithm: for each candidate $\hat{w}_1$ (from Section 3.1), take $\frac{40\sqrt{2}}{3\hat{w}_1\varepsilon}$ samples from the mixture and use each of them as a candidate for μ_1.

We now examine how many candidate pairs $(\hat{w}, \hat{\mu}_1)$ we generated. Naively, since $\hat{w}_i$ may be as small as $O(\varepsilon)$, the candidates for the mean will multiply the size of our collection by $O\left(\frac{1}{\varepsilon^2}\right)$. However, we note that when $\hat{w}_i = \Omega(1)$, then the number of candidates for μ_i is actually $O\left(\frac{1}{\varepsilon}\right)$. We count the number of candidate triples $(\hat{w}, \hat{\mu}_1)$, combining with previous results in the following:

Lemma 15 *Suppose we have sample access to a GMM with (unknown) parameters $(w, \mu_1, \mu_2, \sigma_1, \sigma_2)$. Then for any $\varepsilon > 0$ and constants $c_w, c_m > 0$, using $O(\frac{1}{\varepsilon^2})$ samples from the GMM, we can generate a collection of $O\left(\frac{\log \varepsilon^{-1}}{\varepsilon^2}\right)$ candidate pairs for (w, μ_1). With probability $\geq \frac{99}{100}$, this will contain a pair $(\hat{w}^*, \hat{\mu}_1^*)$ such that $\hat{w}^* \in w \pm O(\varepsilon)$, $\hat{\mu}_1^* \in \mu_1 \pm O(\varepsilon)\sigma_1$.*

The proof of the lemma is deferred to Appendix C. It implies that we can generate $O\left(\frac{1}{\varepsilon^2}\right)$ candidate triples, such that at least one pair simultaneously describes w and μ_1 to the desired accuracy.

3.3. Generating Candidates for a Single Variance

In this section, we generate candidates for the variance corresponding to the Gaussian with the smaller value of $\frac{\sigma_i}{w_i}$. We continue with our guess of whether $i = 1$ or $i = 2$ from the previous section.

Again, assume for this section that $i = 1$. The basic idea is that we will find the closest point to $\hat{\mu}_1$. We use the following property (whose proof is deferred to Appendix E) to establish a range for this distance, which we can then grid over.

We note that this lemma holds in scenarios more general than we consider here, including $k > 2$ and when samples are drawn from a distribution which is only close to a GMM, rather than exactly a GMM.

Lemma 16 *Let c_1 and c_2 be constants as defined in Proposition 31, and $c_3 = \frac{c_1}{9\sqrt{2}c_2}$. Consider a mixture of k Gaussians f, with components $\mathcal{N}(\mu_1, \sigma_1^2), \ldots, \mathcal{N}(\mu_k, \sigma_k^2)$ and weights $w_1, \ldots, w_k$, and let $j = \arg\min_i \frac{\sigma_i}{w_i}$. Suppose we have estimates for the weights and means for all $i \in [1, k]$:*

- *$\hat{w}_i$, such that $\frac{1}{2}\hat{w}_i \le w_i \le 2\hat{w}_i$*
- *$\hat{\mu}_i$, such that $|\hat{\mu}_i - \mu_i| \le \frac{c_3}{2k}\sigma_j$*

Now suppose we draw $n = \frac{9\sqrt{\pi}c_2}{2\hat{w}_j}$ samples $X_1, \ldots, X_n$ from a distribution $\hat{f}$, where $d_{\mathrm{K}}(f, \hat{f}) \le \delta = \frac{c_1}{2n} = \frac{c_1}{9\sqrt{\pi}c_2}\hat{w}_j$. Then $\min_i |X_i - \hat{\mu}_j| \in [\frac{c_3}{2k}\sigma_j, (\sqrt{2} + \frac{c_3}{2k})\sigma_j]$ with probability $\ge \frac{9}{10}$.

Summarizing what we have so far,

Lemma 17 *Suppose we have sample access to a GMM with parameters $(w, \mu_1, \mu_2, \sigma_1, \sigma_2)$, where $\frac{\sigma_1}{w} \le \frac{\sigma_2}{1-w}$. Furthermore, we have estimates $\hat{w}^* \in w \pm O(\varepsilon)$, $\hat{\mu}_1^* \in \mu_1 \pm O(\varepsilon)\sigma_1$. Then for any $\varepsilon > 0$, using $O(\frac{1}{\varepsilon})$ samples from the GMM, we can generate a collection of $O\left(\frac{1}{\varepsilon}\right)$ candidates for σ_1. With probability $\ge \frac{9}{10}$, this will contain a candidate $\hat{\sigma}_1^*$ such that $\hat{\sigma}_1^* \in (1 \pm O(\varepsilon))\sigma_1$.*

3.4. Learning the Last Component using Robust Statistics

At this point, our collection of candidates must contain a triple $(\hat{w}^*, \hat{\mu}_1^*, \hat{\sigma}_1^*)$ which are sufficiently close to the correct parameters. Intuitively, if we could remove this component from the mixture, we would be left with a distribution corresponding to a single Gaussian, which we could learn trivially. We will formalize the notion of "component subtraction," which will allow us to eliminate the known component and obtain a description of an approximation to the CDF for the remaining component. Using classic robust statistics (the median and the interquartile range), we can then obtain approximations to the unknown mean and variance. This has the advantage of a single additional candidate for these parameters, in comparison to $O(\frac{1}{\varepsilon})$ candidates for the previous mean and variance.

Our first step will be to generate an approximation of the overall distribution. We will do this only once, at the beginning of the entire algorithm. Our approximation is with respect to the Kolmogorov distance. Using the DKW inequality (Theorem 7) and Proposition 8, we obtain a $O(\frac{1}{\varepsilon^2})$-interval partition representation of $\hat{H}$ such that $d_{\mathrm{K}}(\hat{H}, H) \le \varepsilon$, with probability $\ge 1 - \delta$ using $O(\frac{1}{\varepsilon^2} \cdot \log\frac{1}{\delta})$ time and samples (where H is the CDF of the GMM).

Next, for each candidate $(\hat{w}, \hat{\mu}_1, \hat{\sigma}_1)$, we apply Lemma 9 to obtain the $O(\frac{1}{\varepsilon^2})$-interval partition of the distribution with the known component removed, i.e., using the notation of Lemma 9, let H be the CDF of the GMM, F is our DKW-based approximation to H, w is the weight $\hat{w}$, and G is $\mathcal{N}(\hat{\mu}_1, \hat{\sigma}_1^2)$. We note that this costs $O(\frac{1}{\varepsilon^2})$ for each candidate triple, and since there are $\tilde{O}(\frac{1}{\varepsilon^3})$ such triples, the total cost of such operations will be $\tilde{O}(\frac{1}{\varepsilon^5})$. However, since the tournament we will use for selection of a candidate will require $\tilde{\Omega}(\frac{1}{\varepsilon^5})$ anyway, this does not affect the overall runtime of our algorithm.

The following proposition shows that, when our candidate triple is $(w^*, \mu_1^*, \sigma_1^*)$, the distribution that we obtain after subtracting the known component out and rescaling is close to the unknown component.

Proposition 18 *Suppose there exists a mixture of two Gaussians* $F = w\mathcal{N}(\mu_1, \sigma_1^2)+(1-w)\mathcal{N}(\mu_2, \sigma_2^2)$ *where* $O(\varepsilon) \le w \le 1 - O(\varepsilon)$*, and we have* $\hat{F}$*, such that* $d_{\mathrm{K}}(\hat{F}, F) \le O(\varepsilon)$*,* $\hat{w}^*$*, such that* $|\hat{w}^* - w| \le O(\varepsilon)$*,* $\hat{\mu}_1^*$*, such that* $|\mu_1^* - \mu_1| \le O(\varepsilon)\sigma_1$*, and* $\hat{\sigma}_1^*$*, such that* $|\sigma_1^* - \sigma_1| \le O(\varepsilon)\sigma_1$*.*

Then $d_{\mathrm{K}}\left(\mathcal{N}(\mu_2, \sigma_2^2), \frac{\hat{F}-\hat{w}^*\mathcal{N}(\hat{\mu}_1^*, \hat{\sigma}_1^{*2})}{1-\hat{w}^*}\right) \le \frac{O(\varepsilon)}{1-w}$.

Since the resulting distribution is close to the correct one, we can use robust statistics (via Lemmas 10 and 11) to recover the missing parameters. We combine this with previous details into the following Lemma, whose proof is deferred to Appendix C.

Lemma 19 *Suppose we have sample access to a GMM with parameters* $(w, \mu_1, \mu_2, \sigma_1, \sigma_2)$*, where* $\frac{\sigma_1}{w} \le \frac{\sigma_2}{1-w}$*. Furthermore, we have estimates* $\hat{w}^* \in w \pm O(\varepsilon)$*,* $\hat{\mu}_1^* \in \mu_1 \pm O(\varepsilon)\sigma_1$*,* $\hat{\sigma}_1^* \in (1 \pm O(\varepsilon))\sigma_1$*. Then for any* $\varepsilon > 0$*, using* $O(\frac{1}{\varepsilon^2} \cdot \log \frac{1}{\delta})$ *samples from the GMM, with probability* $\ge 1 - \delta$*, we can generate candidates* $\hat{\mu}_2^* \in \mu_2 \pm O\left(\frac{\varepsilon}{1-w}\right)\sigma_2$ *and* $\hat{\sigma}_2^* \in \left(1 \pm O\left(\frac{\varepsilon}{1-w}\right)\right)\sigma_2$*.*

3.5. Putting It Together

Theorem 12 is obtained by combining the previous sections, and its proof is given in the appendix.

4. Quasilinear-Time Hypothesis Selection

The goal of this section is to present a hypothesis selection algorithm, `FastTournament`, which is given sample access to a target distribution X and several hypotheses distributions $H_1, \ldots, H_N$, together with an accuracy parameter $\varepsilon > 0$, and is supposed to select a hypothesis distribution from $\{H_1, \ldots, H_N\}$. The desired behavior is this: if at least one distribution in $\{H_1, \ldots, H_N\}$ is ε-close to X in total variation distance, we want that the hypothesis distribution selected by the algorithm is $O(\varepsilon)$-close to X. We provide such an algorithm whose sample complexity is $O(\frac{1}{\varepsilon^2} \log N)$ and whose running time $O(\frac{1}{\varepsilon^2} N \log N)$, *i.e. quasi-linear in the number of hypotheses*, improving the running time of the state of the art (predominantly the Scheffé-estimate based algorithm in Devroye and Lugosi (2001)) quadratically.

We develop our algorithm in full generality, assuming that we have sample access to the distributions of interest, and without making any assumptions about whether they are continuous or discrete, and whether their support is single- or multi-dimensional. All our algorithm needs is sample access to the distributions at hand, together with a way to compare the probability density/mass functions of the distributions, encapsulated in the following definition. In our definition, $H_i(x)$ is the probability mass at x if H_i is a discrete distribution, and the probability density at x if H_i is a continuous distribution. We assume that H_1 and H_2 are either both discrete or both continuous, and that, if they are continuous, they have a density function.

Definition 20 *Let* H_1 *and* H_2 *be probability distributions over some set* $\mathcal{D}$*. A* PDF comparator for H_1, H_2 *is an oracle that takes as input some* $x \in \mathcal{D}$ *and outputs* 1 *if* $H_1(x) > H_2(x)$*, and* 0 *otherwise.*

Our hypothesis selection algorithm is summarized in the following statement:

Theorem 21 *There is an algorithm* `FastTournament`$(X, \mathcal{H}, \varepsilon, \delta)$*, which is given sample access to some distribution* X *and a collection of distributions* $\mathcal{H} = \{H_1, \ldots, H_N\}$ *over some set*

$\mathcal{D}$, access to a PDF comparator for every pair of distributions $H_i, H_j \in \mathcal{H}$, an accuracy parameter $\varepsilon > 0$, and a confidence parameter $\delta > 0$. The algorithm makes $O\left(\frac{\log 1/\delta}{\varepsilon^2} \cdot \log N\right)$ draws from each of $X, H_1, \ldots, H_N$ and returns some $H \in \mathcal{H}$ or declares "failure." If there is some $H^* \in \mathcal{H}$ such that $d_{\mathrm{TV}}(H^*, X) \leq \varepsilon$ then with probability at least $1 - \delta$ the distribution H that `FastTournament` *returns satisfies $d_{\mathrm{TV}}(H, X) \leq 512\varepsilon$. The total number of operations of the algorithm is $O\left(\frac{\log 1/\delta}{\varepsilon^2}\left(N \log N + \log^2 \frac{1}{\delta}\right)\right)$. Furthermore, the expected number of operations of the algorithm is $O\left(\frac{N \log N/\delta}{\varepsilon^2}\right)$.*

The proof of Theorem 21 is given in Appendix G, while the remainder of this section builds the required machinery for the construction.

Remark 22 *A slight modification of our algorithm provided in Appendix I admits a worst-case running time of $O\left(\frac{\log 1/\delta}{\varepsilon^2}\left(N \log N + \log^{1+\gamma} \frac{1}{\delta}\right)\right)$, for any desired constant $\gamma > 0$, though the approximation guarantee is weakened based on the value of γ. See Corollary 36 and its proof in Appendix I.*

Comparison to Other Hypothesis Selection Methods: The skeleton of the hypothesis selection algorithm of Theorem 21 as well as the improved one of Corollary 36, is having candidate distributions compete against each other in a tournament-like fashion. This approach is quite natural and has been commonly used in the literature; see e.g. Devroye and Lugosi (1996, 1997) and Chapter 6 of Devroye and Lugosi (2001), Yatracos (1985), as well as the recent papers of Daskalakis et al. (2012) and Chan et al. (2013). The hypothesis selection algorithms in these works are significantly slower than ours, as their running times have quadratic dependence on the number N of hypotheses, while our dependence is quasi-linear. Furthermore, our setting is more general than prior work, in that we only require sample access to the hypotheses and a PDF comparator. Previous algorithms required knowledge of (or ability to compute) the probability assigned by every pair of hypotheses to their Scheffé set—this is the subset of the support where one hypothesis has larger PMF/PDF than the other, which is difficult to compute in general, even given explicit descriptions of the hypotheses.

Recent independent work by Acharya et al. (2014a,b) provides a hypothesis selection algorithm, based on the Scheffé estimate in Chapter 6 of Devroye and Lugosi (2001). Their algorithm performs a number of operations that is comparable to ours. In particular, the expected running time of their algorithm is also $O\left(\frac{N \log N/\delta}{\varepsilon^2}\right)$, but our worst-case running time has better dependence on δ. Our algorithm is not based on the Scheffé estimate, using instead a specialized estimator provided in Lemma 23. Their algorithm, described in terms of the Scheffé estimate, is not immediately applicable to sample-only access to the hypotheses, or to settings where the probabilities on Scheffé sets are difficult to compute.

4.1. Choosing Between Two Hypotheses

We start with an algorithm for choosing between two hypothesis distributions. This is an adaptation of a similar algorithm from Daskalakis et al. (2012) to continuous distributions and sample-only access. The proof of the following is in Appendix F.

Lemma 23 *There is an algorithm* `ChooseHypothesis`$(X, H_1, H_2, \varepsilon, \delta)$*, which is given sample access to distributions X, H_1, H_2 over some set $\mathcal{D}$, access to a PDF comparator for H_1, H_2,*

an accuracy parameter $\varepsilon > 0$, and a confidence parameter $\delta > 0$. The algorithm draws $m = O(\log(1/\delta)/\varepsilon^2)$ samples from each of X, H_1 and H_2, and either returns some $H \in \{H_1, H_2\}$ as the winner or declares a "draw." The total number of operations of the algorithm is $O(\log(1/\delta)/\varepsilon^2)$. Additionally, the output satisfies the following properties:

1. *If $d_{\mathrm{TV}}(X, H_1) \leq \varepsilon$ but $d_{\mathrm{TV}}(X, H_2) > 8\varepsilon$, the probability that H_1 is not declared winner is $\leq \delta$;*
2. *If $d_{\mathrm{TV}}(X, H_1) \leq \varepsilon$ but $d_{\mathrm{TV}}(X, H_2) > 4\varepsilon$, the probability that H_2 is declared winner is $\leq \delta$;*
3. *The analogous conclusions hold if we interchange H_1 and H_2 in Properties 1 and 2 above;*
4. *If $d_{\mathrm{TV}}(H_1, H_2) \leq 5\varepsilon$, the algorithm declares a "draw" with probability at least $1 - \delta$.*

4.2. The Slow Tournament

We present a hypothesis selection algorithm, `SlowTournament`, which has the correct behavior, but whose running time is suboptimal. Again we proceed similarly to Daskalakis et al. (2012) making the approach robust to continuous distributions and sample-only access. `SlowTournament` uses the subroutine `ChooseHypothesis` of Lemma 23 to performs pairwise comparisons between all hypotheses in $\mathcal{H}$ and outputs a hypothesis that never lost to (but potentially tied with) other hypotheses. The running time of the algorithm is quadratic in $|\mathcal{H}|$, as all pairs of hypotheses are compared. `FastTournament`, described in Section 4.3, organizes the tournament more efficiently and improves the running time to quasilinear. The proof of Lemma 24 is in Appendix F.

Lemma 24 *There is an algorithm $\texttt{SlowTournament}(X, \mathcal{H}, \varepsilon, \delta)$, which is given sample access to some distribution X and a collection of distributions $\mathcal{H} = \{H_1, \ldots, H_N\}$ over some set $\mathcal{D}$, access to a PDF comparator for every pair of distributions $H_i, H_j \in \mathcal{H}$, an accuracy parameter $\varepsilon > 0$, and a confidence parameter $\delta > 0$. The algorithm makes $m = O(\log(N/\delta)/\varepsilon^2)$ draws from each of $X, H_1, \ldots, H_N$ and returns some $H \in \mathcal{H}$ or declares "failure." If there is some $H^* \in \mathcal{H}$ such that $d_{\mathrm{TV}}(H^*, X) \leq \varepsilon$ then with probability at least $1 - \delta$ the distribution H that `SlowTournament` returns satisfies $d_{\mathrm{TV}}(H, X) \leq 8\varepsilon$. The total number of operations of the algorithm is $O\left(N^2 \log(N/\delta)/\varepsilon^2\right)$.*

4.3. The Fast Tournament

We describe our main result of this section, providing a quasi-linear time algorithm for selecting from a collection of hypothesis distributions $\mathcal{H}$ one that is close to a target distribution X, improving the running time of `SlowTournament` from Lemma 24. Intuitively, there are two cases to consider. Collection $\mathcal{H}$ is either dense or sparse in distributions that are close to X. In the former case, we can sub-sample $\mathcal{H}$ before running `SlowTournament`. In the latter case, we set up a two-phase tournament, whose first phase eliminates all but a sub linear number of hypotheses, and whose second phase runs `SlowTournament` on the surviving hypotheses. Depending on the density of $\mathcal{H}$ in distributions that are close to the target distribution X, we show that one of the aforementioned strategies is guaranteed to output a distribution that is close to X. As we do not know a priori the density of $\mathcal{H}$ in distributions that are close to X, and hence which of our two strategies will succeed in finding a distribution that is close to X, we use both strategies and run a tournament among their outputs, using `SlowTournament` again. Full details are provided in Appendix G. With this final tool in place, the proof of Theorem 1 is straightforward and is presented in Appendix H.

References

Jayadev Acharya, Ashkan Jafarpour, Alon Orlitsky, and Ananda Theerta Suresh. Near-optimal-sample estimators for spherical Gaussian mixtures. *Online manuscript*, 2014a.

Jayadev Acharya, Ashkan Jafarpour, Alon Orlitsky, and Ananda Theertha Suresh. Sorting with adversarial comparators and application to density estimation. In *Proceedings of the 2014 IEEE International Symposium on Information Theory*, ISIT '14, Washington, DC, USA, 2014b. IEEE Computer Society.

Dimitris Achlioptas and Frank McSherry. On spectral learning of mixtures of distributions. In *Proceedings of the 18th Annual Conference on Learning Theory*, COLT '05, pages 458–469. Springer, 2005.

Sanjeev Arora and Ravi Kannan. Learning mixtures of arbitrary Gaussians. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, STOC '01, pages 247–257, New York, NY, USA, 2001. ACM.

Mikhail Belkin and Kaushik Sinha. Polynomial learning of distribution families. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, FOCS '10, pages 103–112, Washington, DC, USA, 2010. IEEE Computer Society.

G. E. P. Box and Mervin E. Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, June 1958.

Spencer Charles Brubaker and Santosh Vempala. Isotropic PCA and affine-invariant clustering. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '08, pages 551–560, Washington, DC, USA, 2008. IEEE Computer Society.

Siu On Chan, Ilias Diakonikolas, Rocco A. Servedio, and Xiaorui Sun. Learning mixtures of structured distributions over discrete domains. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pages 1380–1394, Philadelphia, PA, USA, 2013. SIAM.

Siu On Chan, Ilias Diakonikolas, Rocco A. Servedio, and Xiaorui Sun. Efficient density estimation via piecewise polynomial approximation. In *Proceedings of the 46th Annual ACM Symposium on the Theory of Computing*, STOC '14, New York, NY, USA, 2014. ACM.

Sanjoy Dasgupta. Learning mixtures of Gaussians. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '99, pages 634–644, Washington, DC, USA, 1999. IEEE Computer Society.

Constantinos Daskalakis, Ilias Diakonikolas, and Rocco A. Servedio. Learning Poisson binomial distributions. In *Proceedings of the 44th Annual ACM Symposium on the Theory of Computing*, STOC '12, pages 709–728, New York, NY, USA, 2012. ACM.

Constantinos Daskalakis, Ilias Diakonikolas, Ryan O'Donnell, Rocco A. Servedio, and Li Yang Tan. Learning sums of independent integer random variables. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '13, pages 217–226, Washington, DC, USA, 2013. IEEE Computer Society.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

Luc Devroye and Gábor Lugosi. A universally acceptable smoothing factor for kernel density estimation. *The Annals of Statistics*, 24:2499–2512, 1996.

Luc Devroye and Gábor Lugosi. Nonasymptotic universal smoothing factors, kernel complexity and yatracos classes. *The Annals of Statistics*, 25:2626–2637, 1997.

Luc Devroye and Gábor Lugosi. *Combinatorial methods in density estimation*. Springer, 2001.

A. Dvoretzky, J. Kiefer, and J. Wolfowitz. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *The Annals of Mathematical Statistics*, 27 (3):642–669, 09 1956.

Jon Feldman, Ryan O'Donnell, and Rocco A. Servedio. PAC learning axis-aligned mixtures of Gaussians with no separation assumption. In *Proceedings of the 19th Annual Conference on Learning Theory*, COLT '06, pages 20–34, Berlin, Heidelberg, 2006. Springer-Verlag.

Alison L. Gibbs and Francis E. Su. On choosing and bounding probability metrics. *International Statistical Review*, 70(3):419–435, December 2002.

Daniel Hsu and Sham M. Kakade. Learning mixtures of spherical Gaussians: Moment methods and spectral decompositions. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 11–20, New York, NY, USA, 2013. ACM.

Adam Tauman Kalai, Ankur Moitra, and Gregory Valiant. Efficiently learning mixtures of two Gaussians. In *Proceedings of the 42nd Annual ACM Symposium on the Theory of Computing*, STOC '10, pages 553–562, New York, NY, USA, 2010. ACM.

Michael Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. On the learnability of discrete distributions. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing*, STOC '94, pages 273–282, New York, NY, USA, 1994. ACM.

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

Ankur Moitra and Gregory Valiant. Settling the polynomial learnability of mixtures of Gaussians. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, FOCS '10, pages 93–102, Washington, DC, USA, 2010. IEEE Computer Society.

Santosh Vempala and Grant Wang. A spectral algorithm for learning mixtures of distributions. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, FOCS '02, pages 113–123, Washington, DC, USA, 2002. IEEE Computer Society.

Yannis G. Yatracos. Rates of convergence of minimum distance estimators and Kolmogorov's entropy. *The Annals of Statistics*, 13(2):768–774, 1985.

Appendix A. Gridding

We will encounter settings where we have bounds L and R on an unknown measure X such that $L \le X \le R$, and wish to obtain an estimate $\hat{X}$ such that $(1-\varepsilon)X \le \hat{X} \le (1+\varepsilon)X$. Gridding is a common technique to generate a list of candidates that is guaranteed to contain such an estimate.

Fact 25 *Candidates of the form $L + k\varepsilon L$ define an additive grid with at most $\frac{1}{\varepsilon}\left(\frac{R-L}{L}\right)$ candidates.*

Fact 26 *Candidates of the form $L(1+\varepsilon)^k$ define a multiplicative grid with at most $\frac{1}{\log(1+\varepsilon)}\log\left(\frac{R}{L}\right)$ candidates.*

We also encounter scenarios where we require an additive estimate $X - \varepsilon \le \hat{X} \le X + \varepsilon$.

Fact 27 *Candidates of the form $L + k\varepsilon$ define an absolute additive grid with $\frac{1}{\varepsilon}(R-L)$ candidates.*

Appendix B. Proofs Omitted from Section 2

Proof of Proposition 3: We use $d_{\mathrm{TV}}(P,Q)$ and $\frac{1}{2}\|P-Q\|_1$ interchangeably in the cases where P and Q are not necessarily probability distributions. Let $\mathcal{N}_i = \mathcal{N}(\mu_i, \sigma_i^2)$ and $\hat{\mathcal{N}}_i = \mathcal{N}(\hat{\mu}_i, \hat{\sigma}_i^2)$. By triangle inequality,

$$d_{\mathrm{TV}}(\hat{w}\hat{\mathcal{N}}_1 + (1-\hat{w})\hat{\mathcal{N}}_2, w\mathcal{N}_1 + (1-w)\mathcal{N}_2) \le d_{\mathrm{TV}}(\hat{w}\hat{\mathcal{N}}_1, w\mathcal{N}_1) + d_{\mathrm{TV}}((1-\hat{w})\hat{\mathcal{N}}_2, (1-w)\mathcal{N}_2)$$

Inspecting the first term,

$$\frac{1}{2}\left\|w\mathcal{N}_1 - \hat{w}\hat{\mathcal{N}}_1\right\|_1 = \frac{1}{2}\left\|w\mathcal{N}_1 - w\hat{\mathcal{N}}_1 + w\hat{\mathcal{N}}_1 - \hat{w}\hat{\mathcal{N}}_1\right\|_1 \le w d_{\mathrm{TV}}(\mathcal{N}_1, \hat{\mathcal{N}}_1) + \frac{1}{2}\left|w - \hat{w}\right|,$$

again using the triangle inequality. A symmetric statement holds for the other term, giving us the desired result. ■

Proof of Lemma 10: We will use x to denote the median of our distribution, where $\hat{F}(x) = \frac{1}{2}$. Since $d_{\mathrm{K}}(F,\hat{F}) \le \varepsilon$, $F(x) \le \frac{1}{2} + \varepsilon$. Using the CDF of the normal distribution, we obtain $\frac{1}{2} + \frac{1}{2}\mathrm{erf}\left(\frac{x-\mu}{\sqrt{2\sigma^2}}\right) \le \frac{1}{2} + \varepsilon$. Rearranging, we get $x \le \mu + \sqrt{2}\mathrm{erf}^{-1}(2\varepsilon)\sigma \le \mu + 2\sqrt{2}\varepsilon\sigma$, where the inequality uses the Taylor series of erf^{-1} around 0 and Taylor's Theorem. By symmetry of the Gaussian distribution, we can obtain the corresponding lower bound for x. ■

Proof of Lemma 11: First, we show that

$$F^{-1}\left(\frac{3}{4}\right) \in \left[\mu + \sqrt{2}\mathrm{erf}^{-1}\left(\frac{1}{2}\right)\sigma - \frac{5\sqrt{2}}{2}\sigma\varepsilon, \mu + \sqrt{2}\mathrm{erf}^{-1}\left(\frac{1}{2}\right)\sigma + \frac{7\sqrt{2}}{2}\sigma\varepsilon\right].$$

Let $x = F^{-1}\left(\frac{3}{4}\right)$. Since $d_{\mathrm{K}}(F,\hat{F}) \le \varepsilon$, $F(x) \le \frac{3}{4} + \varepsilon$. Using the CDF of the normal distribution, we obtain $\frac{1}{2} + \frac{1}{2}\mathrm{erf}\left(\frac{x-\mu}{\sqrt{2\sigma^2}}\right) \le \frac{3}{4} + \varepsilon$. Rearranging, we get $x \le \mu + \sqrt{2}\mathrm{erf}^{-1}\left(\frac{1}{2} + 2\varepsilon\right)\sigma \le \mu + \sqrt{2}\mathrm{erf}^{-1}\left(\frac{1}{2}\right)\sigma + \frac{7\sqrt{2}}{2}\varepsilon\sigma$, where the inequality uses the Taylor series of erf^{-1} around $\frac{1}{2}$ and Taylor's Theorem. A similar approach gives the desired lower bound.

By symmetry, we can obtain the bounds

$$F^{-1}\left(\frac{1}{4}\right) \in \left[\mu - \sqrt{2}\mathrm{erf}^{-1}\left(\frac{1}{2}\right)\sigma - \frac{7\sqrt{2}}{2}\sigma\varepsilon, \mu - \sqrt{2}\mathrm{erf}^{-1}\left(\frac{1}{2}\right)\sigma + \frac{5\sqrt{2}}{2}\sigma\varepsilon\right].$$

Combining this with the previous bounds and rescaling, we obtain the lemma statement. ■

Appendix C. Proofs Omitted from Section 3

Proof of Proposition 13: The probability that a sample is from $\mathcal{N}_i$ is w_i. Using the CDF of the half-normal distribution, given that a sample is from $\mathcal{N}_i$, the probability that it is at a distance $\leq \varepsilon\sigma_i$ from μ_i is $\mathrm{erf}\left(\frac{\varepsilon}{\sqrt{2}}\right)$.

If we take a single sample from the mixture, it will satisfy the desired conditions with probability at least $w_i\mathrm{erf}\left(\frac{\varepsilon}{\sqrt{2}}\right)$. If we take $\frac{20\sqrt{2}}{3w_i\varepsilon}$ samples from the mixture, the probability that some sample satisfies the conditions is at least

$$1-\left(1-w_i\mathrm{erf}\left(\frac{\varepsilon}{\sqrt{2}}\right)\right)^{\frac{20\sqrt{2}}{3w_i\varepsilon}} \geq 1-\left(1-w_i\cdot\frac{3}{4}\frac{\varepsilon}{\sqrt{2}}\right)^{\frac{20\sqrt{2}}{3w_i\varepsilon}} \geq 1-e^{-5} \geq \frac{99}{100}$$

where the first inequality is by noting that $\mathrm{erf}(x)\geq\frac{3}{4}x$ for $x\in[0,1]$. ■

Proof of Proposition 14: $w_i\geq\varepsilon$ implies $w_i\geq\frac{w_i+\varepsilon}{2}$, and thus $\frac{2}{\hat{w}_i}\geq\frac{2}{w_i+\varepsilon}\geq\frac{1}{w_i}$. ■

Proof of Lemma 15: Aside from the size of the collection, the rest of the conclusions follow from Propositions 13 and 14.

For a given $\hat{w}$, the number of candidates $\hat{\mu}_1$ we consider is $\frac{40\sqrt{2}}{3\hat{w}\varepsilon}$. We sum this over all candidates for $\hat{w}$, namely, $\varepsilon, 2\varepsilon, \ldots, 1-\varepsilon$, giving us

$$\sum_{t=1}^{\frac{1}{\varepsilon}-1}\frac{40\sqrt{2}}{3k\varepsilon^2}=\frac{40\sqrt{2}}{3\varepsilon^2}H_{\frac{1}{\varepsilon}-1}=O\left(\frac{\log\varepsilon^{-1}}{\varepsilon^2}\right)$$

where H_n is the nth harmonic number. ■

Proof of Lemma 17: Let Y be the nearest sample to $\hat{\mu}_1$. From Lemma 16, with probability $\geq\frac{9}{10}$, $|Y-\hat{\mu}_1|\in[\frac{c_3}{4}\sigma_1,(\sqrt{2}+\frac{c_3}{4})\sigma_1]$.

We can generate candidates by rearranging the bounds to obtain

$$\frac{Y}{\sqrt{2}+\frac{c_3}{4}}\leq\sigma_1\leq\frac{Y}{\frac{c_3}{4}}$$

Applying Fact 26 and noting that $\frac{R}{L}=O(1)$, we conclude that we can grid over this range with $O(\frac{1}{\varepsilon})$ candidates.■

Proof of Lemma 19: The proof follows the sketch outlined in Section 3.4. We first use Proposition 8 to construct an approximation $\hat{F}$ of the GMM F. Using Proposition 18, we see that $d_{\mathrm{K}}\left(\mathcal{N}(\mu_2,\sigma_2^2),\frac{\hat{F}-\hat{w}^*\mathcal{N}(\hat{\mu}_1^*,\hat{\sigma}_1^{*2})}{1-\hat{w}^*}\right)\leq\frac{O(\varepsilon)}{1-w}$. By Lemma 9, we can compute a distribution $\hat{H}$ such that $d_{\mathrm{K}}(\mathcal{N}(\mu_2,\sigma_2^2),\hat{H})\leq\frac{O(\varepsilon)}{1-w}$. Finally, using the median and interquartile range and the guaranteed provided by Lemmas 10 and 11, we can compute candidates $\hat{\mu}_2^*\in\mu_2\pm O\left(\frac{\varepsilon}{1-w}\right)\sigma_2$ and $\hat{\sigma}_2^*\in\left(1\pm O\left(\frac{\varepsilon}{1-w}\right)\right)\sigma_2$ from $\hat{H}$, as desired. ■

Proof of Proposition 18:

$$
\begin{aligned}
& d_{\mathrm{K}}\left(\mathcal{N}(\mu_2,\sigma_2^2), \frac{\hat{F}-\hat{w}^*\mathcal{N}(\hat{\mu}_1^*,\hat{\sigma}_1^{*2})}{1-\hat{w}^*}\right) \\
&= \frac{1}{1-\hat{w}} d_{\mathrm{K}}(\hat{w}^*\mathcal{N}(\hat{\mu}_1^*,\hat{\sigma}_1^{*2}) + (1-\hat{w}^*)\mathcal{N}(\mu_2,\sigma_2^2), \hat{F}) \\
&\leq \frac{1}{1-\hat{w}} (d_{\mathrm{K}}(\hat{w}^*\mathcal{N}(\hat{\mu}_1^*,\hat{\sigma}_1^{*2}) + (1-\hat{w}^*)\mathcal{N}(\mu_2,\sigma_2^2), F) + d_{\mathrm{K}}(F,\hat{F})) \\
&\leq \frac{1}{1-\hat{w}} (d_{\mathrm{TV}}(\hat{w}^*\mathcal{N}(\hat{\mu}_1^*,\hat{\sigma}_1^{*2}) + (1-\hat{w}^*)\mathcal{N}(\mu_2,\sigma_2^2), F) + O(\varepsilon)) \\
&\leq \frac{1}{1-\hat{w}} (|w-\hat{w}| + d_{\mathrm{TV}}(\mathcal{N}(\mu_1,\sigma_1), \mathcal{N}(\hat{\mu}_1^*,\hat{\sigma}_1^{*2}) + O(\varepsilon)) \\
&\leq \frac{O(\varepsilon)}{1-\hat{w}} \\
&\leq \frac{O(\varepsilon)}{1-w}
\end{aligned}
$$

The equality is a rearrangement of terms, the first inequality is the triangle inequality, the second inequality uses Fact 6, and the third and fourth inequalities use Propositions 2 and 3 respectively.■

C.1. Proof of Theorem 12

Proof of Theorem 12: We produce two lists of candidates corresponding to whether $\min(w, 1-w) = \Omega(\varepsilon)$ or not:

- In the first case, combining Lemmas 15, 17, and 19 and taking the Cartesian product of the resulting candidates for the mixture's parameters, we see that we can obtain a collection of $O\left(\frac{\log \varepsilon^{-1}}{\varepsilon^3}\right)$ candidate mixtures. With probability $\geq \frac{4}{5}$, this will contain a candidate $(\hat{w}^*, \hat{\mu}_1^*, \hat{\mu}_2^*, \hat{\sigma}_1^*, \hat{\sigma}_2^*)$ such that $\hat{w} \in w \pm O(\varepsilon), \hat{\mu}_i \in \mu_i \pm O(\varepsilon)\sigma_i$ for $i = 1, 2$, and $\hat{\sigma}_i \in (1 \pm O(\varepsilon))\sigma_i$ for $i = 1, 2$. Note that we can choose the hidden constants to be as small as necessary for Lemma 4, and thus we can obtain the desired total variation distance.

  Finally, note that the number of samples that we need for the above to hold is $O(1/\varepsilon^2)$. For this, it is crucial that we *first* draw a sufficient $O(1/\varepsilon^2)$ samples from the mixture (specified by the worse requirement among Lemmas 15, 17, and 19), and *then* execute the candidate generation algorithm outlined in Lemmas 15, 17, and 19. In particular, we do not want to redraw samples for every branching of this algorithm.

  Finally, to boost the success probability, we repeat the entire process $\log_5 \delta^{-1}$ times and let our collection of candidate mixutres be the union of the collections from these repetitions. The probability that none of these collections contains a suitable candidate distribution is $\leq \left(\frac{1}{5}\right)^{\log_5 \delta^{-1}} \leq \delta$.

- In the second case, i.e. when one of the weights, w.l.o.g. w_2, is $O(\varepsilon)$, we set $\hat{w}_1 = 1$ and we only produce candidates for (μ_1, σ_1^2). Note that this scenario fits into the framework of Lemmas 10 and 11. Our mixture F is such that $d_{\mathrm{K}}(F, \mathcal{N}(\mu_1, \sigma_1^2)) \leq d_{\mathrm{TV}}(F, \mathcal{N}(\mu_1, \sigma_1^2)) \leq O(\varepsilon)$. By the DKW inequality (Theorem 7), we can use $O(\frac{1}{\varepsilon^2} \cdot \log \frac{1}{\delta})$ samples to generate the

empirical distribution, which gives us a distribution $\hat{F}$ such that $d_{\mathrm{TV}}(\hat{F}, \mathcal{N}(\mu_1, \sigma_1^2)) \leq O(\varepsilon)$ (by triangle inequality), with probability $\geq 1 - \delta$. From this distribution, using the median and interquartile range and the guarantees of Lemmas 10 and 11, we can extract $\hat{\mu}_1^*$ and $\hat{\sigma}_1^*$ such that $|\hat{\mu}_1^* - \mu_1| \leq O(\varepsilon)\sigma_1$ and $|\hat{\sigma}_1^* - \sigma_1| \leq O(\varepsilon)\sigma_1$. Thus, by Lemma 4, we can achieve the desired total variation distance.

■

Appendix D. Details about Representing and Manipulating CDFs

First, we remark that, given a discrete random variable X over a support of size n, in $\tilde{O}(n)$ time, we can construct a data structure which allows us to compute $F_X^{-1}(x)$ for $x \in [0, 1]$ at the cost of $O(\log n)$ time per operation. This data structures will be a set of disjoint intervals which form a partition of $[0, 1]$, each associated with a value. A value $x \in [0, 1]$ will be mapped to the value associated to the interval which contains x. This data structure can be constructed by mapping each value to an interval of width equal to the probability of that value. This data structure can be queried in $O(\log n)$ time by performing binary search on the left endpoints of the intervals. We name this the *n-interval partition* representation of a distribution. To avoid confusion with intervals that represent elements of the σ-algebra of the distribution, we refer to the intervals that are stored in the data structure as *probability intervals*.

We note that, if we are only concerned with sampling, the order of the elements of the support is irrelevant. However, we will sort the elements of the support in order to perform efficient modifications later.

At one point in our learning algorithm, we will have a candidate which correctly describes one of the two components in our mixture of Gaussians. If we could "subtract out" this component from the mixture, we would be left with a single Gaussian - in this setting, we can efficiently perform parameter estimation to learn the other component. However, if we naively subtract the probability densities, we will obtain negative probability densities, or equivalently, non-monotonically increasing CDFs. To deal with this issue, we define a process we call monotonization. Intuitively, this will shift negative probability density to locations with positive probability density. We show that this preserves Kolmogorov distance and that it can be implemented efficiently.

Definition 28 *Given a bounded function $f : \mathbb{R} \to \mathbb{R}$, the* monotonization *of f is $\hat{f}$, where $\hat{f}(x) = \sup_{y \leq x} f(x)$.*

We argue that if a function is close in Kolmogorov distance to a monotone function, then so is its monotonization.

Proposition 29 *Suppose we have two bounded functions F and G such that $d_{\mathrm{K}}(F, G) \leq \varepsilon$, where F is monotone non-decreasing. Then $\hat{G}$, the monotonization of G, is such that $d_{\mathrm{K}}(F, \hat{G}) \leq \varepsilon$.*

Proof We show that $|F(x) - \hat{G}(x)| \leq \varepsilon$ holds for an arbitrary point x, implying that $d_{\mathrm{K}}(F, \hat{G}) \leq \varepsilon$. There are two cases: $F(x) \geq \hat{G}(x)$ and $F(x) < \hat{G}(x)$.

If $F(x) \geq \hat{G}(x)$, using the fact that $\hat{G}(x) \geq G(x)$ (due to monotonization), we can deduce $|F(x) - \hat{G}(x)| \leq |F(x) - G(x)| \leq \varepsilon$.

If $F(x) < \hat{G}(x)$, consider an infinite sequence of points $\{y_i\}$ such that $G(y_i)$ becomes arbitrarily close to $\sup_{y \leq x} G(x)$. By monotonicity of F, we have that $|F(x) - \hat{G}(x)| \leq |F(y_i) -$

$G(y_i)| + \delta_i \leq \varepsilon + \delta_i$, where $\delta_i = |\hat{G}(x) - G(y_i)|$. Since δ_i can be taken arbitrarily small, we have $|F(x) - \hat{G}(x)| \leq \varepsilon$. ■

We will need to efficiently compute the monotonization in certain settings, when subtracting one monotone function from another.

Proposition 30 *Suppose we have access to the n-interval partition representation of a CDF F. Given a monotone non-decreasing function G, we can compute the n-interval partition representation of the monotonization of $F - G$ in $O(n)$ time.*

Proof Consider the values in the n-interval partition of F. Between any two consecutive values v_1 and v_2, F will be flat, and since G is monotone non-decreasing, $F - G$ will be monotone non-increasing. Therefore, the monotonization of $F - G$ at $x \in [v_1, v_2)$ will be the maximum of $F - G$ on $(-\infty, v_1]$. The resulting monotonization will also be flat on the same intervals as F, so we will only need to update the probability intervals to reflect this monotonization.

We will iterate over probability intervals in increasing order of their values, and describe how to update each interval. We will need to keep track of the maximum value of $F - G$ seen so far. Let m be the maximum of $F - G$ for all $x \leq v$, where v is the value associated with the last probability interval we have processed. Initially, we have the value $m = 0$. Suppose we are inspecting a probability interval with endpoints $[l, r]$ and value v. The left endpoint of this probability interval will become $\hat{l} = m$, and the right endpoint will become $\hat{r} = r - G(v)$. If $\hat{r} \leq \hat{l}$, the interval is degenerate, meaning that the monotonization will flatten out the discontinuity at v - therefore, we simply delete the interval. Otherwise, we have a proper probability interval, and we update $m = \hat{r}$.

This update takes constant time per interval, so the overall time required is $O(n)$. ■

To conclude, we prove Lemma 9.

Proof of Lemma 9: First, by assumption, we know that $\frac{1}{1-w} d_{\mathrm{K}}((1-w)H, F - wG) \leq \varepsilon$. By Proposition 30, we can efficiently compute the monotonization of $F - wG$ - name this $(1-w)\hat{H}$. By Proposition 29, we have that $\frac{1}{1-w} d_{\mathrm{K}}((1-w)H, (1-w)\hat{H}) \leq \varepsilon$. Renormalizing the distributions gives the desired approximation guarantee.

To justify the running time of this procedure, we must also argue that the normalization can be done efficiently. To normalize the distribution $(1-w)\hat{H}$, we make another $O(n)$ pass over the probability intervals and multiply all the endpoints by $\frac{1}{r^*}$, where r^* is the right endpoint of the rightmost probability interval. We note that r^* will be exactly $1 - w$ because the value of $F - wG$ at ∞ is $1 - w$, so this process results in the distribution $\hat{H}$. ■

Appendix E. Robust Estimation of Scale from a Mixture of Gaussians

In this section, we examine the following statistic:

Given some point $x \in \mathbb{R}$ and n IID random variables $X_1, \ldots, X_n$, what is the minimum distance between x and any X_i?

We give an interval in which this statistic is likely to fall (Proposition 31), and examine its robustness when sampling from distributions which are statistically close to the distribution under consideration (Proposition 33). We then apply these results to mixtures of Gaussians (Proposition 34 and Lemma 16).

Proposition 31 *Suppose we have n IID random variables $X_1, \ldots, X_n \sim X$, some $x \in \mathbb{R}$, and $y = F_X(x)$. Let I_N be the interval $[F_X^{-1}(y - \frac{c_1}{n}), F_X^{-1}(y + \frac{c_1}{n})]$ and I_F be the interval $[F_X^{-1}(y - \frac{c_2}{n}), F_X^{-1}(y + \frac{c_2}{n})]$ for some constants $0 < c_1 < c_2 \leq n$, and $I = I_F \backslash I_N$. Let $j = \arg\min_i |X_i - x|$. Then $\Pr[X_j \in I] \geq \frac{9}{10}$ for all $n > 0$.*

Proof We show that $\Pr[X_j \notin I] \leq \frac{1}{10}$ by showing that the following two bad events are unlikely:

1. We have a sample which is too close to x
2. All our samples are too far from x

Showing these events occur with low probability and combining with the union bound gives the desired result.

Let Y be the number of samples at distance $< \frac{c_1}{n}$ in distance in the CDF, i.e., $Y = |\{i \mid |F_X^{-1}(X_i) - y| < \frac{c_1}{n}\}|$. By linearity of expectation, $E[Y] = 2c_1$. By Markov's inequality, $\Pr(Y > 0) < 2c_1$. This allows us to upper bound the probability that one of our samples is too close to x.

Let Z be the number of samples at distance $< \frac{c_2}{n}$ in distance in the CDF, i.e., $Z = |\{i \mid |F_X^{-1}(X_i) - y| < \frac{c_2}{n}\}|$, and let Z_i be an indicator random variable which indicates this property for X_i. We use the second moment principle,

$$\Pr(Z > 0) \geq \frac{E[Z]^2}{E[Z^2]}$$

By linearity of expectation, $E[Z]^2 = 4c_2^2$.

$$\begin{aligned} E[Z^2] &= \sum_i E[Z_i^2] + \sum_i \sum_{j \neq i} E[Z_i Z_j] \\ &= 2c_2 + n(n-1)\left(\frac{4c_2^2}{n^2}\right) \\ &\geq 2c_2 + 4c_2^2 \end{aligned}$$

And thus, $\Pr(Z = 0) \leq \frac{1}{2c_2+1}$. This allows us to upper bound the probability that all of our samples are too far from x.

Setting $c_1 = \frac{1}{40}$ and $c_2 = \frac{19}{2}$ gives probability $< \frac{1}{20}$ for each of the bad events, resulting in a probability $< \frac{1}{10}$ of either bad event by the union bound, and thus the desired result. ■

We will need the following property of Kolmogorov distance, which states that probability mass within every interval is approximately preserved:

Proposition 32 *If $d_{\mathrm{K}}(f_X, f_Y) \leq \varepsilon$, then for all intervals $I \subseteq \mathbb{R}$, $|f_X(I) - f_Y(I)| \leq 2\varepsilon$.*

Proof For an interval $I = [a, b]$, we can rewrite the property as

$$\begin{aligned} |f_X(I) - f_Y(I)| &= |(F_X(b) - F_X(a)) - (F_Y(b) - F_Y(a))| \\ &\leq |F_X(b) - F_Y(b)| + |F_X(a) - F_Y(a)| \\ &\leq 2\varepsilon \end{aligned}$$

as desired, where the first inequality is the triangle inequality and the second inequality is due to the bound on Kolmogorov distance. ■

The next proposition says that if we instead draw samples from a distribution which is close in total variation distance, the same property approximately holds with respect to the original distribution.

Proposition 33 *Suppose we have n IID random variables $\hat{X}_1, \ldots, \hat{X}_n \sim \hat{X}$ where $d_{\mathrm{K}}(f_X, f_{\hat{X}}) \leq \delta$, some $x \in \mathbb{R}$, and $y = F_X(x)$. Let I_N be the interval $[F_X^{-1}(y - \frac{c_1}{n} + \delta), F_X^{-1}(y + \frac{c_1}{n} - \delta)]$ and I_F be the interval $[F_X^{-1}(y - \frac{c_2}{n} - \delta), F_X^{-1}(y + \frac{c_2}{n} + \delta)]$ for some constants $0 < c_1 < c_2 \leq n$, and $I = I_F \backslash I_N$. Let $j = \arg\min_i |X_i - a|$. Then $\Pr[X_j \in I] \geq \frac{9}{10}$ for all $n > 0$.*

Proof First, examine interval I_N. This interval contains $\frac{2c_1}{n} - 2\delta$ probability measure of the distribution F_X. By Proposition 32, $|F_X(I_N) - F_{\hat{X}}(I_N)| \leq 2\delta$, so $F_{\hat{X}}(I_N) \leq \frac{2c_1}{n}$. One can repeat this argument to show that the amount of measure contained by $F_{\hat{X}}$ in $[F_X^{-1}(y - \frac{c_2}{n} - \delta), F_X^{-1}(y + \frac{c_2}{n} + \delta)]$ is $\geq \frac{2c_2}{n}$.

As established through the proof of Proposition 31, with probability $\geq \frac{9}{10}$, there will be no samples in a window containing probability measure $\frac{2c_1}{n}$, but there will be at least one sample in a window containing probability measure $\frac{2c_2}{n}$. Applying the same argument to these intervals, we can arrive at the desired result. ■

We examine this statistic for some mixture of k Gaussians with PDF f around the point corresponding to the mean of the component with the minimum value of $\frac{\sigma_i}{w_i}$. Initially, we assume that we know this location exactly and that we are taking samples according to f exactly.

Proposition 34 *Consider a mixture of k Gaussians with PDF f, components $\mathcal{N}(\mu_1, \sigma_1^2), \ldots, \mathcal{N}(\mu_k, \sigma_k^2)$ and weights $w_1, \ldots, w_k$. Let $j = \arg\min_i \frac{\sigma_i}{w_i}$. If we take n samples $X_1, \ldots, X_n$ from the mixture (where $n > \frac{3\sqrt{\pi}c_2}{2w_j}$), then $\min_i |X_i - \mu_j| \in \left[\frac{\sqrt{2\pi}c_1\sigma_j}{kw_jn}, \frac{3\sqrt{2\pi}c_2\sigma_j}{2w_jn}\right]$ with probability $\geq \frac{9}{10}$, where c_1 and c_2 are as defined in Proposition 31.*

Proof We examine the CDF of the mixture around μ_i. Using Proposition 1 (and symmetry of a Gaussian about its mean), it is sufficient to show that

$$\left[\mu_i + \frac{\sqrt{2\pi}c_1\sigma_i}{kw_in}, \mu_i + \frac{3\sqrt{2\pi}c_2\sigma_i}{2w_in}\right] \supseteq \left[F^{-1}(F(\mu_i) + \frac{c_1}{n}), F^{-1}(F(\mu_i) + \frac{c_2}{n})\right],$$

where F is the CDF of the mixture. We show that each endpoint of the latter interval bounds the corresponding endpoint of the former interval.

First, we show $\frac{c_1}{n} \geq F\left(\mu_i + \frac{\sqrt{2\pi}c_1\sigma_i}{kw_in}\right) - F(\mu_i)$. Let $I = \left[\mu_i, \mu_i + \frac{\sqrt{2\pi}c_1\sigma_i}{kw_in}\right]$, f be the PDF of the mixture, and f_i be the PDF of component i of the mixture. The right-hand side of the inequality

we wish to prove is equal to

$$\begin{aligned}\int_I f(x)\,\mathrm{d}x &= \int_I \sum_{j=1}^{k} w_j f_j(x)\,\mathrm{d}x \\ &\leq \int_I \sum_{j=1}^{k} w_j \frac{1}{\sigma_j\sqrt{2\pi}}\,\mathrm{d}x \\ &\leq \int_I \frac{kw_i}{\sigma_i\sqrt{2\pi}}\,\mathrm{d}x \\ &= \frac{c_1}{n}\end{aligned}$$

where the first inequality is since the maximum of the PDF of a Gaussian is $\frac{1}{\sigma\sqrt{2\pi}}$, and the second is since $\frac{\sigma_j}{w_j} \leq \frac{\sigma_i}{w_i}$ for all j.

Next, we show $\frac{c_2}{n} \leq F\left(\mu_i + \frac{3\sqrt{2\pi}c_2\sigma_i}{2w_i n}\right) - F(\mu_i)$. We note that the right-hand side is the probability mass contained in the interval - a lower bound for this quantity is the probability mass contributed by the particular Guassian we are examining, which is $\frac{w_i}{2}\mathrm{erf}\left(\frac{3\sqrt{\pi}c_2}{2w_i n}\right)$. Taking the Taylor expansion of the error function gives

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\left(x - \frac{x^3}{3} + O(x^5)\right) \geq \frac{2}{\sqrt{\pi}}\left(\frac{2}{3}x\right)$$

if $x < 1$. Applying this here, we can lower bound the contributed probability mass by $\frac{w_i}{2}\frac{2}{\sqrt{\pi}}\frac{2}{3}\frac{3\sqrt{\pi}c_2}{2w_i n} = \frac{c_2}{n}$, as desired. ■

Finally, we deal with uncertainties in parameters and apply the robustness properties of Proposition 33 in the following lemma:

Proof of Lemma 16: We analyze the effect of each uncertainty:

- First, we consider the effect of sampling from $\hat{f}$, which is δ-close to f. By using Proposition 33, we know that the nearest sample to μ_j will be at CDF distance between $\frac{c_1}{n} - \delta \geq \frac{c_1}{2n}$ and $\frac{c_2}{n} + \delta \leq \frac{3c_2}{2n}$. We can then repeat the proof of Proposition 34 with c_1 replaced by $\frac{c_1}{2}$ and c_2 replaced by $\frac{3c_2}{2}$. This gives us that $\min_i |X_i - \mu_j| \in \left[\frac{\sqrt{\pi}c_1}{\sqrt{2}kw_j n}\sigma_j, \frac{9\sqrt{\pi}c_2}{2\sqrt{2}w_j n}\sigma_j\right]$ (where $n \geq \frac{9\sqrt{\pi}c_2}{4w_j}$) with probability $\geq \frac{9}{10}$.
- Next, substituting in the bounds $\frac{1}{2}\hat{w}_j \leq w_j \leq 2\hat{w}_j$, we get $\min_i |X_i - \mu_j| \in \left[\frac{\sqrt{\pi}c_1}{2\sqrt{2}k\hat{w}_j n}\sigma_j, \frac{9\sqrt{\pi}c_2}{\sqrt{2}\hat{w}_j n}\sigma_j\right]$ (where $n \geq \frac{9\sqrt{\pi}c_2}{2\hat{w}_j}$) with probability $\geq \frac{9}{10}$.
- We use $n = \frac{9\sqrt{\pi}c_2}{2\hat{w}_j}$ samples to obtain: $\min_i |X_i - \mu_j| \in \left[\frac{c_3}{k}\sigma_j, \sqrt{2}\sigma_j\right]$ with probability $\geq \frac{9}{10}$.
- Finally, applying $|\hat{\mu}_j - \mu_j| \leq \frac{c_3}{2k}\sigma_j$ gives the lemma statement.

■

Appendix F. Omitted Proofs from Section 4

Proof of Lemma 23: We set up a competition between H_1 and H_2, in terms of the following subset of $\mathcal{D}$:

$$\mathcal{W}_1 \equiv \mathcal{W}_1(H_1, H_2) := \{w \in \mathcal{D} \,|\, H_1(w) > H_2(w)\}\,.$$

In terms of $\mathcal{W}_1$ we define $p_1 = H_1(\mathcal{W}_1)$ and $p_2 = H_2(\mathcal{W}_1)$. Clearly, $p_1 > p_2$ and $d_{\mathrm{TV}}(H_1, H_2) = p_1 - p_2$. The competition between H_1 and H_2 is carried out as follows:

1a. Draw $m = O\left(\frac{\log(1/\delta)}{\varepsilon^2}\right)$ samples $s_1, \ldots, s_m$ from X, and let $\hat{\tau} = \frac{1}{m}|\{i \mid s_i \in \mathcal{W}_1\}|$ be the fraction of them that fall inside $\mathcal{W}_1$.

1b. Similarly, draw m samples from H_1, and let $\hat{p}_1$ be the fraction of them that fall inside $\mathcal{W}_1$.

1c. Finally, draw m samples from H_2, and let $\hat{p}_2$ be the fraction of them that fall inside $\mathcal{W}_1$.

2. If $\hat{p}_1 - \hat{p}_2 \leq 6\varepsilon$, declare a draw. Otherwise:

3. If $\hat{\tau} > \hat{p}_1 - 2\varepsilon$, declare H_1 as winner and return H_1; otherwise,

4. if $\hat{\tau} < \hat{p}_2 + 2\varepsilon$, declare H_2 as winner and return H_2; otherwise,

5. Declare a draw.

Notice that, in Steps 1a, 1b and 1c, the algorithm utilizes the PDF comparator for distributions H_1 and H_2. The correctness of the algorithm is a consequence of the following claim.

Claim 1 *Suppose that $d_{\mathrm{TV}}(X, H_1) \leq \varepsilon$. Then:*

1. *If $d_{\mathrm{TV}}(X, H_2) > 8\varepsilon$, then the probability that the competition between H_1 and H_2 does not declare H_1 as the winner is at most $6e^{-m\varepsilon^2/2}$;*

2. *If $d_{\mathrm{TV}}(X, H_2) > 4\varepsilon$, then the probability that the competition between H_1 and H_2 returns H_2 as the winner is at most $6e^{-m\varepsilon^2/2}$.*

The analogous conclusions hold if we interchange H_1 and H_2 in the above claims. Finally, if $d_{\mathrm{TV}}(H_1, H_2) \leq 5\varepsilon$, the algorithm will declare a draw with probability at least $1 - 6e^{-m\varepsilon^2/2}$.

Proof of Claim 1: Let $\tau = X(\mathcal{W}_1)$. The Chernoff bound (together with a union bound) imply that, with probability at least $1 - 6e^{-m\varepsilon^2/2}$, the following are simultaneously true: $|p_1 - \hat{p}_1| < \varepsilon/2$, $|p_2 - \hat{p}_2| < \varepsilon/2$, and $|\tau - \hat{\tau}| < \varepsilon/2$. Conditioning on these:

- If $d_{\mathrm{TV}}(X, H_1) \leq \varepsilon$ and $d_{\mathrm{TV}}(X, H_2) > 8\varepsilon$, then from the triangle inequality we get that $p_1 - p_2 = d_{\mathrm{TV}}(H_1, H_2) > 7\varepsilon$, hence $\hat{p}_1 - \hat{p}_2 > p_1 - p_2 - \varepsilon > 6\varepsilon$. Hence, the algorithm will go beyond Step 2. Moreover, $d_{\mathrm{TV}}(X, H_1) \leq \varepsilon$ implies that $|\tau - p_1| \leq \varepsilon$, hence $|\hat{\tau} - \hat{p}_1| < 2\varepsilon$. So the algorithm will stop at Step 3, declaring H_1 as the winner of the competition between H_1 and H_2.

- If $d_{\mathrm{TV}}(X, H_2) \leq \varepsilon$ and $d_{\mathrm{TV}}(X, H_1) > 8\varepsilon$, then as in the previous case we get from the triangle inequality that $p_1 - p_2 = d_{\mathrm{TV}}(H_1, H_2) > 7\varepsilon$, hence $\hat{p}_1 - \hat{p}_2 > p_1 - p_2 - \varepsilon > 6\varepsilon$. Hence, the algorithm will go beyond Step 2. Moreover, $d_{\mathrm{TV}}(X, H_2) \leq \varepsilon$ implies that $|\tau - p_2| \leq \varepsilon$, hence $|\hat{\tau} - \hat{p}_2| < 2\varepsilon$. So $\hat{p}_1 > \hat{\tau} + 4\varepsilon$. Hence, the algorithm will not stop at Step 3, and it will stop at Step 4 declaring H_2 as the winner of the competition between H_1 and H_2.

- If $d_{\mathrm{TV}}(X, H_1) \leq \varepsilon$ and $d_{\mathrm{TV}}(X, H_2) > 4\varepsilon$, we distinguish two subcases. If $\hat{p}_1 - \hat{p}_2 \leq 6\varepsilon$, then the algorithm will stop at Step 2 declaring a draw. If $\hat{p}_1 - \hat{p}_2 > 6\varepsilon$, the algorithm proceeds to Step 3. Notice that $d_{\mathrm{TV}}(X, H_1) \leq \varepsilon$ implies that $|\tau - p_1| \leq \varepsilon$, hence $|\hat{\tau} - \hat{p}_1| < 2\varepsilon$. So the algorithm will stop at Step 3, declaring H_1 as the winner of the competition between H_1 and H_2.

- If $d_{\mathrm{TV}}(X, H_2) \leq \varepsilon$ and $d_{\mathrm{TV}}(X, H_1) > 4\varepsilon$, we distinguish two subcases. If $\hat{p}_1 - \hat{p}_2 \leq 6\varepsilon$, then the algorithm will stop at Step 2 declaring a draw. If $\hat{p}_1 - \hat{p}_2 > 6\varepsilon$, the algorithm proceeds to Step 3. Notice that $d_{\mathrm{TV}}(X, H_2) \leq \varepsilon$ implies that $|\tau - p_2| \leq \varepsilon$, hence $|\hat{\tau} - \hat{p}_2| < 2\varepsilon$. Hence, $\hat{p}_1 > \hat{p}_2 + 6\varepsilon \geq \hat{\tau} + 4\varepsilon$, so the algorithm will not stop at Step 3 and will proceed to Step 4. Given that $|\hat{\tau} - \hat{p}_2| < 2\varepsilon$, the algorithm will stop at Step 4, declaring H_2 as the winner of the competition between H_1 and H_2.

- If $d_{\mathrm{TV}}(H_1, H_2) \leq 5\varepsilon$, then $p_1 - p_2 \leq 5\varepsilon$, hence $\hat{p}_1 - \hat{p}_2 \leq 6\varepsilon$. So the algorithm will stop at Step 2 declaring a draw.

■

■

Proof of Lemma 24: Draw $m = O(\log(2N/\delta)/\varepsilon^2)$ samples from each of $X, H_1, \ldots, H_N$ and, using the same samples, run

$$\texttt{ChooseHypothesis}\left(X, H_i, H_j, \varepsilon, \frac{\delta}{2N}\right),$$

for every pair of distributions $H_i, H_j \in \mathcal{H}$. If there is a distribution $H \in \mathcal{H}$ that was never a loser (but potentially tied with some distributions), output any such distribution. Otherwise, output "failure."

We analyze the correctness of our proposed algorithm in two steps. First, suppose there exists $H^* \in \mathcal{H}$ such that $d_{\mathrm{TV}}(H^*, X) \leq \varepsilon$. We argue that, with probability at least $1 - \frac{\delta}{2}$, H^* never loses a competition against any other $H' \in \mathcal{H}$ (so the tournament does not output "failure"). Consider any $H' \in \mathcal{H}$. If $d_{\mathrm{TV}}(X, H') > 4\varepsilon$, by Lemma 23 the probability that H^* is not declared a winner or tie against H' is at most $\frac{\delta}{2N}$. On the other hand, if $d_{\mathrm{TV}}(X, H') \leq 4\varepsilon$, the triangle inequality gives that $d_{\mathrm{TV}}(H^*, H') \leq 5\varepsilon$ and, by Lemma 23, the probability that H^* does not draw against H' is at most $\frac{\delta}{2N}$. A union bound over all N distributions in $\mathcal{H}$ shows that with probability at least $1 - \frac{\delta}{2}$, the distribution H^* never loses a competition.

We next argue that with probability at least $1 - \frac{\delta}{2}$, every distribution $H \in \mathcal{H}$ that never loses must be 8ε-close to X. Fix a distribution H such that $d_{\mathrm{TV}}(X, H) > 8\varepsilon$. Lemma 23 implies that H loses to H^* with probability at least $1 - \delta/2N$. A union bound gives that with probability at least $1 - \frac{\delta}{2}$, every distribution H such that $d_{\mathrm{TV}}(X, H) > 8\varepsilon$ loses some competition.

Thus, with overall probability at least $1-\delta$, the tournament does not output "failure" and outputs some distribution H such that $d_{\mathrm{TV}}(H, X) \leq 8\varepsilon$. ■

Appendix G. Details and Analysis of `FastTournament`

Proof of Theorem 21: Let p be the fraction of the elements of $\mathcal{H}$ that are 8ε-close to X. The value of p is unknown to our algorithm. Regardless, we propose two strategies for selecting a distribution from $\mathcal{H}$, one of which is guaranteed to succeed whatever the value of p is. We assume throughout this proof that N is larger than a sufficiently large constant, otherwise our claim follows directly from Lemma 24.

S1: Pick a random subset $\mathcal{H}' \subseteq \mathcal{H}$ of size $\lceil 3\sqrt{N} \rceil$, and run $\texttt{SlowTournament}(X, \mathcal{H}', 8\varepsilon, e^{-3})$ to select some distribution $\tilde{H} \in \mathcal{H}'$.

Claim 2 *The number of samples drawn by* S1 *from each of the distributions in* $\mathcal{H} \cup \{X\}$ *is* $O(\frac{1}{\varepsilon^2} \log N)$*, and the total number of operations is* $O(\frac{1}{\varepsilon^2} N \log N)$*. Moreover, if* $p \in [\frac{1}{\sqrt{N}}, 1]$ *and there is some distribution in* $\mathcal{H}$ *that is* ε*-close to* X*, then the distribution* $\tilde{H}$ *output by* S1 *is* 64ε*-close to* X*, with probability at least* $9/10$*.*

Proof The probability that $\mathcal{H}'$ contains no distribution that is 8ε-close to X is at most

$$(1-p)^{\lceil 3\sqrt{N} \rceil} \leq e^{-3}.$$

If $\mathcal{H}'$ contains at least one distribution that is 8ε-close to X, then by Lemma 24 the distribution output by $\texttt{SlowTournament}(X, \mathcal{H}', 8\varepsilon, e^{-3})$ is 64ε-close to X with probability at least $1 - e^{-3}$. From a union bound, it follows that the distribution output by S1 is 64ε-close to X, with probability at least $1 - 2e^{-3} \geq 9/10$. The bounds on the number of samples and operations follow from Lemma 24. ■

S2: There are two phases in this strategy:

- **Phase 1:** This phase proceeds in $T = \lfloor \log_2 \frac{\sqrt{N}}{2} \rfloor$ iterations, $i_1, \ldots, i_T$. Iteration i_ℓ takes as input a subset $\mathcal{H}_{i_{\ell-1}} \subseteq \mathcal{H}$ (where $\mathcal{H}_{i_0} \equiv \mathcal{H}$), and produces some $\mathcal{H}_{i_\ell} \subset \mathcal{H}_{i_{\ell-1}}$, such that $|\mathcal{H}_{i_\ell}| = \left\lceil \frac{|\mathcal{H}_{i_{\ell-1}}|}{2} \right\rceil$, as follows: randomly pair up the elements of $\mathcal{H}_{i_{\ell-1}}$ (possibly one element is left unpaired), and for every pair (H_i, H_j) run $\texttt{ChooseHypothesis}(X, H_i, H_j, \varepsilon, 1/3N)$. We do this with a small caveat: instead of drawing $O(\log(3N)/\varepsilon^2)$ fresh samples (as required by Lemma 23) in every execution of `ChooseHypothesis` (from whichever distributions are involved in that execution), we draw $O(\log(3N)/\varepsilon^2)$ samples from each of $X, H_1, \ldots, H_N$ once and for all, and reuse the same samples in all executions of `ChooseHypothesis`.
- **Phase 2:** Given the collection $\mathcal{H}_{i_T}$ output by Phase 1, we run $\texttt{SlowTournament}(X, \mathcal{H}_{i_T}, \varepsilon, 1/4)$ to select some distribution $\hat{H} \in \mathcal{H}_{i_T}$. (We use fresh samples for the execution of `SlowTournament`.)

Claim 3 *The number of samples drawn by* S2 *from each of the distributions in* $\mathcal{H} \cup \{X\}$ *is* $O(\frac{1}{\varepsilon^2}\log N)$*, and the total number of operations is* $O(\frac{1}{\varepsilon^2} N \log N)$*. Moreover, if* $p \in (0, \frac{1}{\sqrt{N}}]$ *and there is some distribution in* $\mathcal{H}$ *that is* ε*-close to* X*, then the distribution* $\hat{H}$ *output by* S2 *is* 8ε*-close to* X*, with probability at least* $1/4$*.*

Proof Suppose that there is some distribution $H^* \in \mathcal{H}$ that is ε-close to X. We first argue that with probability at least $\frac{1}{3}$, $H^* \in \mathcal{H}_{i_T}$. We show this in two steps:

(a) Recall that we draw samples from $X, H_1, \ldots, H_N$ before Phase 1 begins, and reuse the same samples whenever required by some execution of `ChooseHypothesis` during Phase 1. Fix a realization of these samples. We can ask the question of what would happen if we executed `ChooseHypothesis`$(X, H^*, H_j, \varepsilon, 1/3N)$, for some $H_j \in \mathcal{H}\backslash\{H^*\}$ using these samples. From Lemma 23, it follows that, if H_j is farther than 8ε-away from X, then H^* would be declared the winner by `ChooseHypothesis`$(X, H^*, H_j, \varepsilon, 1/3N)$, with probability at least $1 - 1/3N$. By a union bound, our samples satisfy this property simultaneously for all $H_j \in \mathcal{H} \setminus \{H^*\}$ that are farther than 8ε-away from X, with probability at least $1 - 1/3$. Henceforth, we condition that our samples have this property.

(b) Conditioning on our samples having the property discussed in (a), we argue that $H^* \in \mathcal{H}_{i_T}$ with probability at least $1/2$ (so that, with overall probability at least $1/3$, it holds that $H^* \in \mathcal{H}_{i_T}$). It suffices to argue that, with probability at least $1/2$, in all iterations of Phase 1, H^* is not matched with a distribution that is 8ε-close to X. This happens with probability at least:

$$(1-p)(1-2p)\cdots(1-2^{T-1}p) \geq 2^{-2p\sum_{i=0}^{T-1} 2^i} = 2^{-2p(2^T-1)} \geq 1/2.$$

Indeed, given the definition of p, the probability that H^* is not matched to a distribution that is 8ε-close to X is at least $1-p$ in the first iteration. If this happens, then (because of our conditioning from (a)), H^* will survive this iteration. In the next iteration, the fraction of surviving distributions that are 8ε-close to X and are different than H^* itself is at most $2p$. Hence, the probability that H^* is not matched to a distribution that is 8ε-close to X is at least $1-2p$ in the second iteration, etc.

Now, conditioning on $H^* \in \mathcal{H}_{i_T}$, it follows from Lemma 24 that the distribution $\hat{H}$ output by `SlowTournament`$(X, \mathcal{H}_{i_T}, \varepsilon, 1/4)$ is 8ε-close to X with probability at least $3/4$.

Hence, with overall probability at least $1/4$, the distribution output by S2 is 8ε-close to X.

The number of samples drawn from each distribution in $\mathcal{H} \cup \{X\}$ is clearly $O(\frac{1}{\varepsilon^2}\log N)$, as Phase 1 draws $O(\frac{1}{\varepsilon^2}\log N)$ samples from each distribution and, by Lemma 24, Phase 2 also draws $O(\frac{1}{\varepsilon^2}\log N)$ samples from each distribution.

The total number of operations is bounded by $O(\frac{1}{\varepsilon^2} N \log N)$. Indeed, Phase 1 runs `ChooseHypothesis` $O(N)$ times, and by Lemma 23 and our choice of $1/3N$ for the confidence parameter of each execution, each execution takes $O(\log N/\varepsilon^2)$ operations. So the total number of operations of Phase 1 is $O(\frac{1}{\varepsilon^2} N \log N)$. On the other hand, the size of $\mathcal{H}_{i_T}$ is at most $\frac{2^{\lceil \log_2 N \rceil}}{2^T} = \frac{2^{\lceil \log_2 N \rceil}}{2^{\lfloor \log_2 \frac{\sqrt{N}}{2} \rfloor}} \leq 8\sqrt{N}$. So by Lemma 24, Phase 2 takes $O(\frac{1}{\varepsilon^2} N \log N)$ operations. ■

Given strategies S1 and S2, we first design an algorithm which has the stated worst-case number of operations. The algorithm `FastTournament`$_A$ works as follows:

1. Execute strategy S1 $k_1 = \log_2 \frac{2}{\delta}$ times, with fresh samples each time. Let $\tilde{H}_1, \ldots, \tilde{H}_{k_1}$ be the distributions output by these executions.
2. Execute strategy S2 $k_2 = \log_4 \frac{2}{\delta}$ times, with fresh samples each time. Let $\hat{H}_1, \ldots, \hat{H}_{k_2}$ be the distributions output by these executions.
3. Set $\mathcal{G} \equiv \{\tilde{H}_1, \ldots, \tilde{H}_{k_1}, \hat{H}_1, \ldots, \hat{H}_{k_2}\}$. Execute $\texttt{SlowTournament}(X, \mathcal{G}, 64\varepsilon, \delta/2)$.

Claim 4 $\texttt{FastTournament}_A$ *satisfies the properties described in the statement of Theorem 21, except for the bound on the expected number of operations.*

Proof of Claim 4: The bounds on the number of samples and operations follow immediately from our choice of k_1, k_2, Claims 2 and 3, and Lemma 24. Let us justify the correctness of the algorithm. Suppose that there is some distribution in $\mathcal{H}$ that is ε-close to X. We distinguish two cases, depending on the fraction p of distributions in $\mathcal{H}$ that are ε-close to X:

- $p \in [\frac{1}{\sqrt{N}}, 1]$: In this case, each execution of S1 has probability at least $9/10$ of outputting a distribution that is 64ε-close to X. So the probability that none of $\tilde{H}_1, \ldots, \tilde{H}_{k_1}$ is 64ε-close to X is at most $(\frac{1}{10})^{k_1} \leq \delta/2$. Hence, with probability at least $1 - \delta/2$, $\mathcal{G}$ contains a distribution that is 64ε-close to X. Conditioning on this, $\texttt{SlowTournament}(X, \mathcal{G}, 64\varepsilon, \delta/2)$ will output a distribution that is 512ε-close to X with probability at least $1 - \delta/2$, by Lemma 24. Hence, with overall probability at least $1 - \delta$, the distribution output by $\texttt{FastTournament}$ is 512ε-close to X.
- $p \in (0, \frac{1}{\sqrt{N}}]$: This case is analyzed analogously. With probability at least $1 - \delta/2$, at least one of $\hat{H}_1, \ldots, \hat{H}_{k_2}$ is 8ε-close to X (by Claim 3). Conditioning on this, $\texttt{SlowTournament}(X, \mathcal{G}, 64\varepsilon, \delta/2)$ outputs a distribution that is 512ε-close to X, with probability at least $1 - \delta/2$ (by Lemma 24). So, with overall probability at least $1 - \delta$, the distribution output by $\texttt{FastTournament}$ is 512ε-close to X.

■

We now describe an algorithm which has the stated expected number of operations. The algorithm $\texttt{FastTournament}_B$ works as follows:

1. Execute strategy S1, let $\tilde{H}_1$ be the distribution output by this execution.
2. Execute strategy S2, let $\tilde{H}_2$ be the distribution output by this execution.
3. Execute $\texttt{ChooseHypothesis}(X, \tilde{H}_i, H, 64\varepsilon, \delta/N^3)$ for $i \in \{1, 2\}$ and all $H \in \mathcal{H}$. If either $\tilde{H}_1$ or $\tilde{H}_2$ never loses, output that hypothesis. Otherwise, remove $\tilde{H}_1$ and $\tilde{H}_2$ from $\mathcal{H}$, and repeat the algorithm starting from step 1, unless $\mathcal{H}$ is empty.

Claim 5 $\texttt{FastTournament}_B$ *satisfies the properties described in the statement of Theorem 21, except for the worst-case bound on the number of operations.*

Proof of Claim 5: We note that we will first draw $O(\log(N^3/\delta)/\varepsilon^2)$ from each of $X, H_1, \ldots, H_N$ and use the same samples for every execution of `ChooseHypothesis` to avoid blowing up the sample complexity. Using this fact, the sample complexity is as claimed.

We now justify the correctness of the algorithm. Since we run `ChooseHypothesis` on a given pair of hypotheses at most once, there are at most N^2 executions of this algorithm. Because each fails with probability at most $\frac{\delta}{N^3}$, by the union bound, the probability that any execution of `ChooseHypothesis` ever fails is at most δ, so all executions succeed with probability at least $1 - \frac{\delta}{N}$. Condition on this happening for the remainder of the proof of correctness. In Step 3 of our algorithm, we compare some $\tilde{H}$ with every other hypothesis. We analyze two cases:

- Suppose that $d_{\mathrm{TV}}(X, \tilde{H}) \leq 64\varepsilon$. By Lemma 23, $\tilde{H}$ will never lose, and will be output by $\texttt{FastTournament}_B$.
- Suppose that $d_{\mathrm{TV}}(X, \tilde{H}) > 512\varepsilon$. Then by Lemma 23, $\tilde{H}$ will lose to any candidate H' with $d_{\mathrm{TV}}(X, H') \leq 64\varepsilon$. We assumed there exists at least one hypothesis with this property in the beginning of the algorithm. Furthermore, by the previous case, if this hypothesis were selected by S1 or S2 at some prior step, the algorithm would have terminated; so in particular, if the algorithm is still running, this hypothesis could not have been removed from $\mathcal{H}$. Therefore, $\tilde{H}$ will lose at least once and will not be output by $\texttt{FastTournament}_B$.

The correctness of our algorithm follows from the second case above. Indeed, if the algorithm outputs a distribution $\tilde{H}$, it must be the case that $d_{\mathrm{TV}}(X, \tilde{H}) \leq 512\varepsilon$. Moreover, we will not run out of hypotheses before we output a distribution. Indeed, we only discard a hypothesis if it was selected by S1 or S2 and then lost at least once in Step 3. Furthermore, in the beginning of our algorithm there exists a distribution H such that $d_{\mathrm{TV}}(X, H) \leq 64\varepsilon$. If ever selected by S1 or S2, H will not lose to any distribution in Step 3, and the algorithm will output a distribution. If it is not selected by S1 or S2, H won't be removed from $\mathcal{H}$.

We now reason about the expected running time of our algorithm. First, consider the case when all executions of `ChooseHypothesis` are successful, which happens with probability $\geq 1 - \frac{\delta}{N}$. If either S1 or S2 outputs a distribution such that $d_{\mathrm{TV}}(X, \tilde{H}) \leq 64\varepsilon$, then by the first case above it will be output by $\texttt{FastTournament}_B$. If this happened with probability at least p independently in every iteration of our algorithm, then the number of iterations would be stochastically dominated by a geometric random variable with parameter p, so the expected number of rounds would be upper bounded by $\frac{1}{p}$. By Claims 2 and 3, $p \geq \frac{1}{4}$, so, when `ChooseHypothesis` never fails, the expected number of rounds is at most 4. Next, consider when at least one execution of `ChooseHypothesis` fails, which happens with probability $\leq \frac{\delta}{N}$. Since $\texttt{FastTournament}_B$ removes at least one hypothesis in every round, there are at most N rounds. Combining these two cases, the expected number of rounds is at most $(1 - \frac{\delta}{N})4 + \frac{\delta}{N}N \leq 5$.

By Claims 2 and 3 and Lemma 23, each round requires $O(N \log N + N \log N/\delta)$ operations. Since the expected number of rounds is $O(1)$, we obtain the desired bound on the expected number of operations. ■

In order to obtain all the guarantees of the theorem simultaneously, our algorithm `FastTournament` will alternate between steps of $\texttt{FastTournament}_A$ and $\texttt{FastTournament}_B$, where both algorithms are given an error parameter equal to $\frac{\delta}{2}$. If either algorithm outputs a hypothesis, `FastTournament` outputs it. By union bound and Claims 4 and 5, both $\texttt{FastTournament}_A$

and $\texttt{FastTournament}_B$ will be correct with probability at least $1-\delta$. The worst-case running time is as desired by Claim 4 and since interleaving between steps of the two tournaments will multiply the number of steps by a factor of at most 2. We have the expected running time similarly, by Claim 5. ■

Appendix H. Proof of Theorem 1

Theorem 1 is an immediate consequence of Theorems 12 and 21. Namely, we run the algorithm of Theorem 12 to produce a collection of Gaussian mixtures, one of which is within ε of the unknown mixture F. Then we use `FastTournament` of Theorem 21 to select from among the candidates a mixture that is $O(\varepsilon)$-close to F. For the execution of `FastTournament`, we need a PDF comparator for all pairs of candidate mixtures in our collection. Given that these are described with their parameters, our PDF comparators evaluate the densities of two given mixtures at a challenge point x and decide which one is largest. We also need sample access to our candidate mixtures. Given a parametric description $(w, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2)$ of a mixture, we can draw a sample from it as follows: first draw a uniform $[0,1]$ variable whose value compared to w determines whether to sample from $\mathcal{N}(\mu_1, \sigma_1^2)$ or $\mathcal{N}(\mu_2, \sigma_2^2)$ in the second step; for the second step, use the Box-Muller transform (Box and Muller, 1958) to obtain a sample from either $\mathcal{N}(\mu_1, \sigma_1^2)$ or $\mathcal{N}(\mu_2, \sigma_2^2)$ as decided in the first step.

Appendix I. Faster Slow and Fast Tournaments

In this section, we describe another hypothesis selection algorithm. This algorithm is faster than `SlowTournament`, though at the cost of a larger constant in the approximation factor. In most reasonable parameter regimes, this algorithm is slower than `FastTournament`, and still has a larger constant in the approximation factor. Regardless, we go on to show how it can be used to improve upon the worst-case running time of `FastTournament`.

Theorem 35 *For any constant $\gamma > 0$, there is an algorithm $\texttt{RecursiveSlowTournament}_\gamma(X, \mathcal{H}, \varepsilon, \delta)$, which is given sample access to some distribution X and a collection of distributions $\mathcal{H} = \{H_1, \ldots, H_N\}$ over some set $\mathcal{D}$, access to a PDF comparator for every pair of distributions $H_i, H_j \in \mathcal{H}$, an accuracy parameter $\varepsilon > 0$, and a confidence parameter $\delta > 0$. The algorithm makes $m = O(\log(N/\delta)/\varepsilon^2)$ draws from each of $X, H_1, \ldots, H_N$ and returns some $H \in \mathcal{H}$ or declares "failure." If there is some $H^* \in \mathcal{H}$ such that $d_{\mathrm{TV}}(H^*, X) \le \varepsilon$ then with probability at least $1-\delta$ the distribution H that `RecursiveSlowTournament` returns satisfies $d_{\mathrm{TV}}(H, X) \le O(\varepsilon)$. The total number of operations of the algorithm is $O\left(N^{1+\gamma}\log(N/\delta)/\varepsilon^2\right)$.*

Proof For simplicity, assume that $\sqrt{N}$ is integer. (If not, introduce into $\mathcal{H}$ multiple copies of an arbitrary $H \in \mathcal{H}$ so that $\sqrt{N}$ becomes an integer.) Partition $\mathcal{H}$ into $\sqrt{N}$ subsets, $\mathcal{H} = \mathcal{H}_1 \sqcup \mathcal{H}_2 \sqcup \ldots \sqcup \mathcal{H}_{\sqrt{N}}$ and do the following:

1. Set $\delta' = \delta/2$, draw $O(\log(\sqrt{N}/\delta')/\varepsilon^2)$ samples from X and, using the same samples, run $\texttt{SlowTournament}(X, \mathcal{H}_i, \varepsilon, \delta')$ from Lemma 24 for each i;
2. Run $\texttt{SlowTournament}(X, \mathcal{W}, 8\varepsilon, \delta')$, where $\mathcal{W}$ are the distributions output by `SlowTournament` in the previous step. If $\mathcal{W} = \emptyset$ output "failure".

Let us call the above algorithm $\texttt{SlowTournament}^{\otimes 1}(X, \mathcal{H}, \varepsilon, \delta)$, before proceeding to analyze its correctness, sample and time complexity. Suppose there exists a distribution $H \in \mathcal{H}$ such that $d_{\mathrm{TV}}(H, X) \leq \varepsilon$. Without loss of generality, assume that $H \in \mathcal{H}_1$. Then, from Lemma 24, with probability at least $1 - \delta'$, $\texttt{SlowTournament}(X, \mathcal{H}_1, \varepsilon, \delta')$ will output a distribution H' such that $d_{\mathrm{TV}}(H', X) \leq 8\varepsilon$. Conditioning on this and applying Lemma 24 again, with conditional probability at least $1-\delta'$ $\texttt{SlowTournament}(X, \mathcal{W}, 8\varepsilon, \delta')$ will output a distribution H'' such that $d_{\mathrm{TV}}(H'', X) \leq 64\varepsilon$. So with overall probability at least $1-\delta$, $\texttt{SlowTournament}^{\otimes 1}(X, \mathcal{H}, \varepsilon, \delta)$ will output a distribution that is 64ε-close to X. The number of samples that the algorithm draws from X is $O(\log(N/\delta)/\varepsilon^2)$, and the running time is

$$\sqrt{N} \times O\left(N \log(N/\delta')/\varepsilon^2\right) + O\left(N \log(N/\delta')/(8\varepsilon)^2\right) = O\left(N^{3/2} \log(N/\delta)/\varepsilon^2\right).$$

So, compared to $\texttt{SlowTournament}$, $\texttt{SlowTournament}^{\otimes 1}$ has the same sample complexity asymptotics and the same asymptotic guarantee for the distance from X of the output distribution, but the exponent of N in the running time improved from 2 to $3/2$.

For $t = 2, 3, \ldots$, define $\texttt{SlowTournament}^{\otimes t}$ by replacing $\texttt{SlowTournament}$ by $\texttt{SlowTournament}^{\otimes t-1}$ in the code of $\texttt{SlowTournament}^{\otimes 1}$. It follows from the same analysis as above that as t increases the exponent of N in the running time gets arbitrarily close to 1. In particular, in one step an exponent of $1 + \alpha$ becomes an exponent of $1 + \alpha/2$. So for some constant t, $\texttt{SlowTournament}^{\otimes t}$ will satisfy the requirements of the theorem. ■

As a corollary, we can immediately improve the running time of $\texttt{FastTournament}$ at the cost of the constant in the approximation factor. The construction and analysis is nearly identical to that of $\texttt{FastTournament}$. The sole difference is in step 3 of $\texttt{FastTournament}_A$ - we replace $\texttt{SlowTournament}$ with $\texttt{RecursiveSlowTournament}_\gamma$.

Corollary 36 *For any constant $\gamma > 0$, there is an algorithm $\texttt{FastTournament}_\gamma(X, \mathcal{H}, \varepsilon, \delta)$, which is given sample access to some distribution X and a collection of distributions $\mathcal{H} = \{H_1, \ldots, H_N\}$ over some set $\mathcal{D}$, access to a PDF comparator for every pair of distributions $H_i, H_j \in \mathcal{H}$, an accuracy parameter $\varepsilon > 0$, and a confidence parameter $\delta > 0$. The algorithm makes $O\left(\frac{\log 1/\delta}{\varepsilon^2} \cdot \log N\right)$ draws from each of $X, H_1, \ldots, H_N$ and returns some $H \in \mathcal{H}$ or declares "failure." If there is some $H^* \in \mathcal{H}$ such that $d_{\mathrm{TV}}(H^*, X) \leq \varepsilon$ then with probability at least $1 - \delta$ the distribution H that $\texttt{SlowTournament}$ returns satisfies $d_{\mathrm{TV}}(H, X) \leq O(\varepsilon)$. The total number of operations of the algorithm is $O\left(\frac{\log 1/\delta}{\varepsilon^2}(N \log N + \log^{1+\gamma} \frac{1}{\delta})\right)$. Furthermore, the expected number of operations of the algorithm is $O\left(\frac{N \log N/\delta}{\varepsilon^2}\right)$.*