Sample-Optimal Density Estimation in Nearly-Linear Time

Jayadev Acharya∗
EECS, MIT
jayadev@csail.mit.edu

Ilias Diakonikolas†
Informatics, U. of Edinburgh
ilias.d@ed.ac.uk

Jerry Li‡
EECS, MIT
jerryzli@csail.mit.edu

Ludwig Schmidt§
EECS, MIT
ludwigs@mit.edu

Abstract

We design a new, fast algorithm for agnostically learning univariate probability distributions whose densities are well approximated by piecewise polynomial functions. Let $f$ be the density function of an arbitrary univariate distribution, and suppose that $f$ is OPT close in $L_1$-distance to an unknown piecewise polynomial function with $t$ interval pieces and degree $d$. Our algorithm draws $n = O(t(d + 1)/\epsilon^2)$ samples from $f$, runs in time $\tilde{O}(n \cdot \text{poly}(d))$, and with probability at least $9/10$ outputs an $O(t)$-piecewise degree-$d$ hypothesis $h$ that is $4 \cdot \text{OPT} + \epsilon$ close to $f$.

Our general algorithm yields (nearly) sample-optimal and nearly-linear time estimators for a wide range of structured distribution families over both continuous and discrete domains in a unified way. For most of our applications, these are the first sample-optimal and nearly-linear time estimators in the literature. As a consequence, our work resolves the sample and computational complexities of a broad class of inference tasks via a single “meta-algorithm”. Moreover, we experimentally demonstrate that our algorithm performs very well in practice.

Our algorithm consists of three “levels”: (i) At the top level, we employ an iterative greedy algorithm for finding a good partition of the real line into the pieces of a piecewise polynomial. (ii) For each piece, we show that the sub-problem of finding a good polynomial fit on the current interval can be solved efficiently with a separation oracle method. (iii) We reduce the task of finding a separating hyperplane to a combinatorial problem and give an efficient algorithm for this problem. Combining these three procedures gives a density estimation algorithm with the claimed guarantees.

∗Supported by a grant from the MIT-Shell Energy Initiative.
†Supported by a Marie Curie CIG, EPSRC grant EP/L021749/1 and a SICSA grant.
‡Supported by NSF grant CCF-1217921 and DOE grant DE-SC0008923.
§Supported by a grant from the MIT-Shell Energy Initiative.
# Contents

1 Introduction  
1.1 Our Results and Techniques  
1.2 Related Work  

2 Preliminaries  

3 Paper Outline  

4 Iterative Merging Algorithm  
4.1 The Histogram Merging Algorithm  
4.2 The General Merging algorithm  
4.3 Putting everything together  

5 A fast $A_k$-projection oracle for polynomials  
5.1 The set of feasible polynomials  
5.2 Separation oracles and approximately feasible polynomials  
5.3 Bounds on the radii of enclosing and enclosed balls  
5.4 Finding the best polynomial  

6 The separation oracle and the $A_k$-computation oracle  
6.1 Overview of $\text{APPROXSEPORACLE}$  
6.2 Testing Non-negativity and Boundedness  
6.3 An $A_k$-computation oracle  
6.3.1 A Reduction from Continuous to Discrete  
6.3.2 Description of $\text{COMPUTEDISCRETEAK}$  
6.3.3 Description of $\text{COMPUTEAK}$  

7 Applications  
7.1 Mixture of log-concave distributions  
7.2 Mixture of Gaussians  
7.3 Densities in Besov spaces  
7.4 Mixtures of $t$-monotone distributions  
7.5 Mixtures of discrete distributions  

8 Experimental Evaluation  
8.1 Histogram hypotheses  
8.2 Piecewise linear hypotheses  

A Analysis of the General Merging Algorithm: Proof of Theorem 17  

B Additional Omitted Proofs  
B.1 Proof of Fact 26  
B.2 Proof of Lemma 34
C Learning discrete piecewise polynomials

C.1 Problem statement in the discrete setting

C.2 The algorithm in the discrete setting

C.2.1 The $A_k$-norm and general merging in the discrete setting

C.2.2 Efficient $A_k$-projection and computation oracles for polynomials
1 Introduction

Estimating an unknown probability density function based on observed data is a classical problem in statistics that has been studied since the late nineteenth century, starting with the pioneering work of Karl Pearson [Pea95]. Distribution estimation has become a paradigmatic and fundamental unsupervised learning problem with a rich history and extensive literature (see e.g., [BBBB72, DG85, Sil86, Sco92, DL01]). A number of general methods for estimating distributions have been proposed in the mathematical statistics literature, including histograms, kernels, nearest neighbor estimators, orthogonal series estimators, maximum likelihood, and more. We refer the reader to [Ize91] for a survey of these techniques. During the past few decades, there has been a large body of work in computer science on this topic with a focus on computational efficiency [KMR+94, FM99, FOS05, BS10, KMV10, MV10, KSV08, VW02, DDS12a, DDS12b, DDO+13, CDSS14a].

Suppose that we are given a number of samples from an unknown distribution that belongs to (or is well-approximated by) a given family of distributions \( \mathcal{C} \), e.g., it is a mixture of a small number of Gaussians. Our goal is to estimate the unknown distribution in a precise, well-defined way. In this work, we focus on the problem of density estimation (non-proper learning), where the goal is to output an approximation of the unknown density without any constraints on its representation. That is, the output hypothesis is not necessarily a member of the family \( \mathcal{C} \). The “gold standard” in this setting is to design learning algorithms that are both statistically and computationally efficient. More specifically, the ultimate goal is to obtain estimators whose sample size is information-theoretically optimal, and whose running time is (nearly) linear in their sample size. An important additional requirement is that our learning algorithms are agnostic or robust under model misspecification, i.e., they succeed even if the target distribution does not belong to the given family \( \mathcal{C} \) but is merely well-approximated by a distribution in \( \mathcal{C} \).

We study the problem of density estimation for univariate distributions, i.e., distributions with a density \( f : \Omega \rightarrow \mathbb{R}_+ \), where the sample space \( \Omega \) is a subset of the real line. While density estimation for families of univariate distributions has been studied for several decades, both the sample and time complexity were not yet well understood before this work, even for surprisingly simple classes of distributions, such as mixtures of Binomials, and mixtures of Gaussians. Our main result is a general learning algorithm that can be used to estimate a wide variety of structured distribution families. For each such family, our general algorithm simultaneously satisfies all three of the aforementioned criteria, i.e., it is agnostic, (nearly) sample optimal, and runs in nearly-linear time.

Our algorithm is based on learning a piecewise polynomial function that approximates the unknown density. This approach of using piecewise polynomial approximation has been used in this context before — our main contribution is to improve the computational complexity of this method and to make it nearly-optimal for a wide range of distribution families. The key idea of using piecewise polynomials for learning is that the existence of good piecewise polynomial approximations for a family \( \mathcal{C} \) of distributions can be leveraged for the design of efficient learning algorithms for \( \mathcal{C} \). The main algorithmic ingredient that makes this method possible is an efficient algorithm for agnostically learning piecewise polynomial density functions. In prior work, Chan, Diakonikolas, Servedio, and Sun [CDSS14a] obtained a nearly-sample optimal and polynomial time algorithm for this learning problem. Unfortunately, however, the polynomial exponent in their running time is quite high, which makes the algorithm prohibitively slow for most applications.

In this paper, we design a new, fast algorithm for agnostically learning piecewise polynomial distributions, which in turn yields sample-optimal and nearly-linear time estimators for a wide range
of structured distribution families over both continuous and discrete domains. For most of our applications, these are the first sample-optimal and nearly-linear time estimators in the literature. As a consequence, our work resolves the sample and computational complexity of a broad class of inference tasks via a single “meta-algorithm”. Moreover, we experimentally demonstrate that our algorithm performs very well in practice. We stress that a significant number of new algorithmic and technical ideas are needed for our main result, as we explain next.

1.1 Our Results and Techniques

In this section, we describe our results in detail, compare them to prior work, and give an overview of our new algorithmic ideas.

Preliminaries. We consider univariate probability density functions (pdf’s) defined over a known finite interval \( I \subseteq \mathbb{R} \). (We remark that this assumption is without loss of generality and our results easily apply to densities defined over the entire real line.)

We focus on a standard notion of learning an unknown probability distribution from samples [KMR+94], which is a natural analogue of Valiant’s well-known PAC model for learning Boolean functions [Val84] to the unsupervised setting of learning an unknown probability distribution. (We remark that our definition is essentially equivalent to the notion of \( L_1 \) minimax rate of convergence in statistics [DL01].) A distribution learning problem is defined by a class \( \mathcal{C} \) of probability distributions over a domain \( \Omega \). Given \( \epsilon > 0 \) and sample access to a target distribution with density \( f \), the goal of an agnostic learning algorithm for \( \mathcal{C} \) is to compute a hypothesis \( h \) such that, with probability at least 9/10, it holds \( \|h - f\|_1 \leq C \cdot \text{OPT}_C(f) + \epsilon \), where \( \text{OPT}_C(f) := \inf_{q \in \mathcal{C}} \|q - f\|_1 \), i.e., \( \text{OPT}_C(f) \) is the \( L_1 \)-distance between \( f \) and the closest distribution to it in \( \mathcal{C} \), and \( C \geq 1 \) is a universal constant.

We say that a function \( f \) over an interval \( I \) is a \( t \)-piecewise degree-\( d \) polynomial if there is a partition of \( I \) into \( t \) disjoint intervals \( I_1, \ldots, I_t \) such that \( f(x) = f_j(x) \) for all \( x \in I_j \), where each of \( f_1, \ldots, f_t \) is a polynomial of degree at most \( d \). Let \( \mathcal{P}_{t,d}(I) \) denote the class of all \( t \)-piecewise degree-\( d \) polynomials over the interval \( I \).

Our Results. Our main algorithmic result is the following:

**Theorem 1 (Main).** Let \( f : I \to \mathbb{R}_+ \) be an unknown target distribution over \( I \), where \( I \) is either an interval or the discrete set \([N]\). There is an algorithm with the following performance guarantee: Given parameters \( t, d \in \mathbb{Z}_+ \), an error tolerance \( \epsilon > 0 \), and any \( \gamma > 0 \), the algorithm draws \( n = O_\gamma(t(d+1)/\epsilon^2) \) samples from \( f \), runs in time \( \tilde{O}(n \cdot \text{poly}(d+1)) \) and with probability at least 9/10 outputs an \( O(t) \)-piecewise degree-\( d \) hypothesis \( h \) such that \( \|f - h\|_1 \leq (3 + \gamma)\text{OPT}_{t,d}(f) + \epsilon \), where \( \text{OPT}_{t,d}(f) := \inf_{r \in \mathcal{P}_{t,d}(I)} \|f - r\|_1 \) is the error of the best \( t \)-piecewise degree-\( d \) approximation to \( f \).

In prior work, [CDSS14a] gave a learning algorithm for this problem that uses \( \tilde{O}(t(d+1)/\epsilon^2) \) samples and runs in \( \text{poly}(t, d+1, 1/\epsilon) \) time. We stress that the algorithm of [CDSS14a] is prohibitively slow. In particular, the running time of their approach is \( \Omega(t^3 \cdot (d^{3.5}/\epsilon^{3.5} + d^{6.5}/\epsilon^{2.5})) \), which renders their result more of a “proof of principle” than a computationally efficient algorithm.

This prompts the following question: Is such a high running time necessary to achieve this level of sample efficiency? Ideally, one would like a sample-optimal algorithm with a low-order polynomial running time (ideally, linear).

Our main result shows this is indeed possible in a very strong sense. The running time of our algorithm is linear in \( t/\epsilon^2 \) (up to a \( \log(1/\epsilon) \) factor), which is essentially the best possible; the polynomial
dependence on $m$ is $\tilde{O}(m^{3+\omega})$, where $\omega$ is the matrix multiplication exponent. This substantially improved running time is of critical importance for the applications of Theorem 1. Moreover, the sample complexity of our algorithm removes the extraneous logarithmic factors present in the sample complexity of [CDSS14a], and matches the information-theoretic lower bound up to a constant factor. As we explain below, Theorem 1 leads to (nearly) sample-optimal and nearly-linear time estimators for a wide range of natural and well-studied families. For most of these applications, ours is the first estimator with simultaneously nearly optimal sample and time complexity.

Our new algorithm is clean and modular. As a result, Theorem 1 also applies to discrete distributions over an ordered domain (e.g., $[N]$). The approach of [CDSS14a] does not extend to polynomial approximation over discrete domains, and designing such an algorithm was left as an open problem in their work. As a consequence, we obtain several new applications to learning mixtures of discrete distributions. In particular, we obtain the first nearly sample optimal and nearly-linear time estimators for mixtures of Binomial and Poisson distributions. To the best of our knowledge, no polynomial time algorithm with nearly optimal sample complexity was known for these basic learning problems prior to this work.

Applications

We now explain how to use Theorem 1 in order to agnostically learn structured distribution families. Given a class $C$ that we want to learn, we proceed as follows: (i) Prove that any distribution in $C$ is $\epsilon/2$-close in $L_1$-distance to a $t$-piecewise degree-$d$ polynomial, for appropriate values of $t$ and $d$. (ii) Use Theorem 1 for these values of $t$ and $d$ to agnostically learn the target distribution up to error $\epsilon/2$. Note that $t$ and $d$ will depend on the desired error $\epsilon$ and the underlying class $C$. We emphasize that there are many combinations of $t$ and $d$ that guarantee an $\epsilon/2$-approximation of $C$ in Step (i). To minimize the sample complexity of our learning algorithm in Step (ii), we would like to determine the values of $t$ and $d$ that minimize the product $t(d + 1)$. This is, of course, an approximation theory problem that depends on the structure of the family $C$.

For example, if $C$ is the family of log-concave distributions, the optimal $t$-histogram approximation with accuracy $\epsilon$ requires $\Theta(1/\epsilon^3)$ intervals. This leads to an algorithm with sample complexity $\Theta(1/\epsilon^3)$. On the other hand, it can be shown that any log-concave distribution has a piecewise linear $\epsilon$-approximation with $\Theta(1/\epsilon^{1/2})$ intervals [CDSS14a, DK15], which yields an algorithm with sample complexity $\Theta(1/\epsilon^{5/2})$. Perhaps surprisingly, this sample bound cannot be improved using higher degree piecewise polynomials; one can show an information-theoretic lower bound of $\Omega(1/\epsilon^{5/2})$ for learning log-concave densities [DL01]. Hence, Theorem 1 gives a sample-optimal and nearly-linear time agnostic learning algorithm for this fundamental problem. We remark that piecewise polynomial approximations are “closed” under taking mixtures. As a corollary, Theorem 1 also yields an $O(k/\epsilon^{5/2})$ sample and nearly-linear time algorithm for learning an arbitrary mixture of $k$ log-concave distributions. Again, there exists a matching information-theoretic lower bound of $\Omega(k/\epsilon^{5/2})$.

As a second example, let $C$ be the class of mixtures of $k$ Gaussians in one dimension. It is not difficult to show that learning such a mixture of Gaussians up to total variation distance $\epsilon$ requires $\Omega(k/\epsilon^2)$ samples. By approximating the corresponding probability density functions by piecewise polynomials of degree $O(\log(1/\epsilon))$, we obtain an agnostic learning algorithm for this class that uses $n = O(k/\epsilon^2)$ samples and runs in time $\tilde{O}(n)$. Similar bounds can be obtained for several other natural parametric mixture families.

Note that for a wide range of structured families, the optimal choice of the degree $d$ (i.e.,

---

1This includes all structured families considered in [CDSS14a] and several previously-studied distributions not covered by [CDSS14a].
the choice minimizing $t(d + 1)$ among all $\epsilon/2$-approximations) will be at most poly-logarithmic in $1/\epsilon$. For several classes (such as unimodal, monotone hazard rate, and log-concave distributions), the degree $d$ is even a constant. As a consequence, Theorem 1 yields (nearly) sample optimal and nearly-linear time estimators for all these families in a unified way. In particular, we obtain sample optimal (or nearly sample optimal) and nearly-linear time estimators for a wide range of structured distribution families, including arbitrary mixtures of natural distributions such as multimodal, concave, convex, log-concave, monotone hazard rate, Gaussian, Poisson, Binomial, function in Besov spaces, and others.

See Table 1 for a summary of these applications. For each distribution family in the table, we provide a comparison to the best previous result. Note that we do not aim to exhaustively cover all possible applications of Theorem 1, but rather to give some selected applications that are indicative of the generality and power of our method.

<table>
<thead>
<tr>
<th>Class of Distributions</th>
<th>Sample complexity</th>
<th>Time complexity</th>
<th>Reference</th>
<th>Optimality</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$-histograms</td>
<td>$\tilde{O}(\frac{1}{\epsilon^2})$</td>
<td>$\tilde{O}(\frac{1}{\epsilon^2})$</td>
<td>[CDSS14b] Theorem 10</td>
<td>SO, TOS</td>
</tr>
<tr>
<td>$t$-piecewise degree-$d$ polynomials</td>
<td>$\tilde{O}(\frac{td}{\epsilon^2})$</td>
<td>$\tilde{O}(\frac{td}{\epsilon^2})$</td>
<td>[CDSS14a] Theorem 1</td>
<td>NTO</td>
</tr>
<tr>
<td>$k$-mixture of log-concave</td>
<td>$\tilde{O}(\frac{k}{\epsilon^2})$</td>
<td>$\tilde{O}(\frac{k}{\epsilon^2})$</td>
<td>[CDSS14a] Theorem 42</td>
<td>SO, TOS</td>
</tr>
<tr>
<td>$k$-mixture of Gaussians</td>
<td>$\tilde{O}(\frac{k}{\epsilon^2})$</td>
<td>$\tilde{O}(\frac{k}{\epsilon^2})$</td>
<td>[CDSS14a] Theorem 43</td>
<td>NSO, NTO</td>
</tr>
<tr>
<td>Besov space $B^\alpha_q(L_p([0, 1]))$</td>
<td>$O_\alpha\left(\frac{\log^{\omega}(\frac{1}{\epsilon})}{\epsilon^{2+1/\omega}}\right)$</td>
<td>$\tilde{O}_\alpha\left(\frac{1}{\epsilon^{2+1/\omega}}\right)$</td>
<td>[WN07] Theorem 44</td>
<td>SO, TOS</td>
</tr>
<tr>
<td>$k$-mixture of $t$-monotone</td>
<td>$\tilde{O}(\frac{kt}{\epsilon^{2+1/\omega}})$</td>
<td>$\tilde{O}(\frac{kt}{\epsilon^{2+1/\omega}})$</td>
<td>[CDSS14a] Theorem 45</td>
<td>SO, TOS</td>
</tr>
<tr>
<td>$k$-mixture of $t$-modal</td>
<td>$\tilde{O}(\frac{kt}{\epsilon^{2+1/\omega}})$</td>
<td>$\tilde{O}(\frac{kt}{\epsilon^{2+1/\omega}})$</td>
<td>[CDSS14a] Theorem 46</td>
<td>SO, TOS</td>
</tr>
<tr>
<td>$k$-mixture of MHR</td>
<td>$\tilde{O}(\frac{k\log(N/\epsilon)}{\epsilon^2})$</td>
<td>$\tilde{O}(\frac{k\log(N/\epsilon)}{\epsilon^2})$</td>
<td>[CDSS14a] Theorem 47</td>
<td>SO, TOS</td>
</tr>
<tr>
<td>$k$-mixture of Binomial, Poisson</td>
<td>$\tilde{O}(\frac{k\log(N/\epsilon)}{\epsilon^2})$</td>
<td>$\tilde{O}(\frac{k\log(N/\epsilon)}{\epsilon^2})$</td>
<td>[CDSS14a] Theorem 48</td>
<td>NSO, NTO</td>
</tr>
</tbody>
</table>

SO : Sample complexity is optimal up to a constant factor.
NSO : Sample complexity is optimal up to a poly-logarithmic factor.
TOS : Time complexity is optimal (up to sorting the samples).
NTO : Time complexity is optimal up to a poly-logarithmic factor.

Table 1: A list of applications to agnostically learning specific families of distributions. Note that for all examples, our algorithm runs in time that is nearly-linear in the information-theoretically optimal sample complexity.
Moreover, our non-proper learning algorithm is also useful for proper learning. Indeed, Theorem 1 has recently been used [LS15] as a crucial component to obtain the fastest known agnostic algorithm for properly learning a mixture of univariate Gaussian distributions. Note that non-proper learning and proper learning for a family $\mathcal{C}$ are equivalent in terms of sample size: given any (non-proper) hypothesis we can do a brute-force search to find its closest density in $\mathcal{C}$. The challenging part is to perform this computation efficiently. Roughly speaking, given a piecewise polynomial hypothesis, [LS15] design an efficient algorithm to find the closest mixture of $k$ Gaussians.

Our Techniques. We now provide a brief overview of our new algorithm and techniques in parallel with a comparison to the previous algorithm of [CDSS14a]. We require the following definition. For any $k \geq 0$ and an interval $I \subseteq \mathbb{R}$, define the $A_k$-norm of a function $f : I \rightarrow \mathbb{R}$ to be

$$\|f\|_{A_k} \overset{\text{def}}{=} \sup_{I_1, \ldots, I_k} \sum_{i=1}^k |f(I_i)|,$$

where the supremum is over all disjoint intervals $I_1, \ldots, I_k$ in $I$, and $f(J) \overset{\text{def}}{=} \int_J f(x) \, dx$ for any measurable set $J \subseteq I$. Our main probabilistic tool is the following well-known version of the VC inequality:

**Theorem 2 (VC Inequality [VC71, DL01]).** Let $f : I \rightarrow \mathbb{R}$ be an arbitrary pdf over $I$, and let $\hat{f}$ be the empirical pdf obtained after taking $n$ i.i.d. samples from $f$. Then

$$\mathbb{E}[\|f - \hat{f}\|_{A_k}] \leq O\left(\sqrt{\frac{k}{n}}\right).$$

Given this theorem, it is not difficult to show that the following two step procedure is an agnostic learning algorithm for $\mathcal{P}_{t,d}$:

(i) Draw a set of $n = \Theta(t(d+1)/\epsilon^2)$ samples from $f$;

(ii) Output the piecewise-polynomial hypothesis $h \in \mathcal{P}_{t,d}$ that minimizes the quantity $\|h - \hat{f}\|_{A_k}$ up to an additive error $\gamma = O(\epsilon)$, where $k = O(t(d+1))$.

We remark that the optimization problem in Step (ii) is non-convex. However, it has sufficient structure so that it can be solved in polynomial time. Intuitively, an algorithm for Step (ii) involves two main ingredients: (1) An efficient procedure to agnostically learn a degree-$d$ polynomial in a given sub-interval of the domain. (2) An efficient procedure to “discover” the unknown “optimal” $t$ intervals. We remark that the procedure for (2) will use the procedure for (1) multiple times as a subroutine.

[CDSS14a] solve an appropriately relaxed version of Step (ii) by a combination of linear programming and dynamic programming. Roughly speaking, they formulate a polynomial size linear program to agnostically learn a degree-$d$ polynomial in a given interval, and use a dynamic program in order to discover the correct $t$ intervals. It should be emphasized that the algorithm of [CDSS14a] is theoretically efficient (polynomial time), but prohibitively slow for real applications with large data sets. In particular, the linear program of [CDSS14a] has $\Omega(t/d)$ variables and $\Omega(d^2/\epsilon^2 + d^6/\epsilon)$ constraints. Hence, the running time of their algorithm using the fastest known LP solver for their instance [LS14] is at least $\tilde{\Omega}(d^{3.5}/\epsilon^{3.5} + d^{6.5}/\epsilon^{2.5})$. Moreover, the dynamic program to implement (2) has running time at least $\Omega(t^3)$. This leads to an overall running time of...
\( \tilde{\Omega} \left( t^3 \cdot \left( d^{3.5}/\epsilon^{3.5} + d^{6.5}/\epsilon^{2.5} \right) \right) \), which quickly becomes unrealistic even for modest values of \( \epsilon, t, \) and \( d \).

We now provide a sketch of our new algorithm. At a high-level, we implement procedure (2) above using an iterative greedy algorithm. Our algorithm circumvents the need for a dynamic programming approach as follows: The main idea is to iteratively merge pairs of intervals by calling an oracle for procedure (1) in every step, until the number of intervals becomes \( O(t) \). Our iterative algorithm and its subtle analysis are directly inspired by the VC inequality. Roughly speaking, in each iteration the algorithm estimates the contribution to an appropriate notion of error when two consecutive intervals are merged, and it merges pairs of intervals with small error. This procedure ensures that the number of intervals in our partition decreases geometrically with the number of iterations.

Our algorithm for procedure (1) is based on convex programming and runs in time \( \text{poly}(d+1)/\epsilon^2 \) (note that the dependence on \( \epsilon \) is optimal). To achieve this significant running time improvement, we use a novel approach that is quite different from that of [CDSS14a]. Roughly speaking, we are able to exploit the problem structure exhibited by the \( A_d \) optimization problem in order to separate the problem dimension \( d \) from the problem dimension \( 1/\epsilon \), and only solve a convex program of dimension \( d \) (as opposed to dimension \( \text{poly}(d/\epsilon) \) in [CDSS14a]). More specifically, we consider the convex set of non-negative polynomials with \( A_{d+1} \) distance at most \( \tau \) from the empirical distribution. While this set is defined through a large number of constraints, we show that it is possible to design a separation oracle that runs in time nearly linear in both the number of samples and the degree \( d \). Combined with tools from convex optimization such as the Ellipsoid method or Vaidya’s algorithm, this gives an efficient algorithm for procedure (1).

1.2 Related Work

There has been a long history in statistics on estimating structured families of distributions. For distributions over continuous domains, a very natural type of structure to consider is some sort of “shape constraint” on the probability density function (pdf) defining the distribution. Statistical research in this area started in the 1950’s, and the reader is referred to the book [BBBB72] for a summary of the early work. Most of the literature has focused on one-dimensional distributions, with a few exceptions during the past decade. Various structural restrictions have been studied over the years, starting from monotonicity, unimodality, convexity, and concavity [Gre56, Bru58, Rao69, Weg70, HP76, Gro85, Bir87a, Bir87b, Fou97, CT04, JW09], and more recently focusing on structural restrictions such as log-concavity and \( k \)-monotonicity [BW07, DR09, BRW09, GW09, BW10, KM10, Wal09]. The reader is referred to [GJ14] for a recent book on the subject. Mixtures of structured distributions have received much attention in statistics [Lin95, RW84, TSM85] and, more recently, in theoretical computer science [Das99, DS00, AK01, VW02, FOS05, AM05, MV10].

The most common method used in statistics to address shape constrained inference problems is the Maximum Likelihood Estimator (MLE) and its variants. While the MLE is very popular and quite natural, we note that it is not agnostic, and it may in general require to solve an intractable optimization problem (e.g., for the case of mixture models.)

Piecewise polynomials (splines) have been extensively used as tools for inference tasks, including density estimation, see, e.g., [WW83, WN07, Sto94, SHKT97]. We remark that splines in the statistics literature have been used in the context of the MLE, which is very different than our approach. Moreover, the degree of the splines used is typically bounded by a small constant and the underlying algorithms are heuristic in most cases. A related line of work in mathematical
statistics [KP92, DJKP95, KPT96, DJKP96, DJ98] uses non-linear estimators based on wavelet
techniques to learn continuous distributions whose densities satisfy various smoothness constraints,
such as Triebel and Besov-type smoothness. We remark that the focus of these works is on the
statistical efficiency of the proposed estimators.

For the problem of learning piecewise constant distributions with \( t \) unknown interval pieces,
[CDSS14b] recently gave an \( n = \tilde{O}(t/\epsilon^2) \) sample and \( \tilde{O}(n) \) time algorithm. However, their approach
does not seem to generalize to higher degrees. Recall that Theorem 1 removes all unnecessary
logarithmic factors from the sample complexity, and furthermore, our algorithm runs in time pro-
portional to the time required to sort the samples.

Our iterative merging idea is quite robust: together with Hegde, the authors of the current
paper have shown that an analogous approach yields sample optimal and efficient algorithms for
agnostically learning discrete distributions with piecewise constant functions under the
\( \ell_2 \)-distance metric [ADH+15]. We emphasize that learning under the \( \ell_2 \)-distance is easier than under the \( L_1 \)-
distance, and that the analysis of [ADH+15] is significantly simpler than the analysis in the current
paper. Moreover, the algorithmic subroutine of finding a good polynomial fit on a fixed interval
required by the \( \ell_2 \) algorithm is substantially simpler than the subroutine we require here. Indeed,
for us, the associated optimization problem has exponentially many linear constraints, and thus
cannot be fully described, even in polynomial time.

Paper Structure. After some preliminaries in Section 2, we give an outline of our algorithm
in Section 3. Sections 4 – 6 contain the various components of our algorithm. Section 7 gives a
detailed description of our applications to learning structured distribution families, and we conclude
in Section 8 with our experimental evaluation.

2 Preliminaries

We consider univariate probability density functions (pdf’s) defined over a known finite interval
\( I \subseteq \mathbb{R} \). For an interval \( J \subseteq I \) and a positive integer \( k \), we will denote by \( \mathcal{I}_J^k \) the family of
all sets of at most \( k \) disjoint intervals \( I_1, \ldots, I_k \) where each \( I_i \subseteq J \). For a measurable function
\( g : I \rightarrow \mathbb{R} \), and a measurable set \( S \), let \( g(S) \overset{\text{def}}{=} \int_S g \). The \( L_1 \) norm of \( g \) over a subinterval \( J \subseteq I \) is
defined as \( \|g\|_{1,J} = \int_J |g(x)| \, dx \). More generally, for any set of disjoint intervals \( J \in \mathcal{I}_J^k \), we define
\( \|g\|_{1,J} = \sum_{J_i \in J} \|g\|_{1,J_i} \).

We now define a norm which induces a corresponding distance metric that will be crucial for
this paper:

Definition 3 (\( A_k \)-norm). Let \( k \) be a positive integer and let \( g : I \rightarrow \mathbb{R} \) be measurable. For any
subinterval \( J \subseteq I \), the \( A_k \)-norm of \( g \) on \( J \) is defined as

\[
\|g\|_{A_k,J} \overset{\text{def}}{=} \sup_{J_i \in \mathcal{I}_J^k \atop \text{such that for all } J_i \subseteq J} \sum_{M \in I} |g(M)|.
\]

When \( J = I \), we omit the second subscript and simply write \( \|g\|_{A_k} \).

More generally, for any set of disjoint intervals \( J = \{J_1, \ldots, J_t\} \) where each \( J_i \subseteq I \), we define
\( \|g\|_{A_k,J} = \sup_{I} \sum_{J \in I \atop \text{such that for all } J \subseteq J_i} |g(J)| \)

where the supremum is taken over all \( I \in \mathcal{I}_J^k \) such that for all \( J \in I \) there is a \( J_i \in J \) with \( J \subseteq J_i \).
We note that the definition of the $A_k$-norm in this work is slightly different than that in [DL01, CDSS14a] but is easily seen to be essentially equivalent. The VC inequality (Theorem 2) along with uniform convergence bounds (see, e.g., Theorem 2.2. in [CDSS13] or p. 17 in [DL01]), yields the following:

**Corollary 4.** Fix $\epsilon > 0$ and $0 < \delta < 1$. Let $f : I \to \mathbb{R}$ be an arbitrary pdf over $I$, and let $\hat{f}$ be the empirical pdf obtained after taking $n = \Theta((k + \log 1/\delta)/\epsilon^2)$ i.i.d. samples from $f$. Then with probability at least $1 - \delta$,

$$\|f - \hat{f}\|_{A_k} \leq \epsilon.$$ 

**Definition 5.** Let $g : I \to \mathbb{R}$. We say that $g$ has at most $k$ sign changes if there exists a partition of $I$ into intervals $I_1, \ldots, I_{k+1}$ such that for all $i \in [k+1]$ either $g(x) \geq 0$ for all $x \in I_i$ or $g(x) \leq 0$ for all $x \in I_i$.

We will need the following elementary facts about the $\|\cdot\|_{A_k}$-norm.

**Fact 6.** Let $J \subseteq I$ be an arbitrary interval or finite set of intervals. Let $g : I \to \mathbb{R}$ be a measurable function.

(a) If $g$ has at most $k - 1$ sign changes in $J$, then $\|g\|_{1,J} = \|g\|_{A_k,J}$.

(b) For all $k \geq 1$, we have $\|g\|_{A_k,J} \leq \|g\|_{1,J}$.

(c) Let $\alpha$ be a positive integer. Then, $\|g\|_{A_{\alpha \cdot k},J} \leq \alpha \cdot \|g\|_{A_k,J}$.

(d) Let $f$ be a pdf over $I$, and let $J_1, \ldots, J_\ell$ be finite sets of disjoint subintervals of $I$, such that for all $i, i'$ and for all $I \in J_i$ and $I' \in J_{i'}$, $I$ and $I'$ are disjoint. Then, for all positive integers $m_1, \ldots, m_\ell$, $\sum_{i=1}^\ell \|f\|_{A_{m_i},J_i} \leq \|f\|_{A_M}$, where $M = \sum_{i=1}^\ell m_i$.

## 3 Paper Outline

In this section we give a high-level description of our algorithm for learning $t$-piecewise degree-$d$ polynomials. Our algorithm can be divided into three layers.

**Level 1: General merging** At the top level, we design an iterative merging algorithm to find the closest piecewise polynomial approximation to the unknown target density. Our merging algorithm applies more generally to broad classes of piecewise hypotheses. Let $D$ be a class of hypotheses satisfying the following: (i) The number of intersections between any two hypotheses in $D$ is bounded. (ii) Given an interval $J$ and an empirical distribution $\tilde{f}$, we can efficiently find the best fit to $\tilde{f}$ from functions in $D$ with respect to the $A_k$-distance. (iii) We can efficiently compute the $A_k$-distance between the empirical distribution and any hypothesis in $D$. Under these assumptions, our merging algorithm agnostically learns piecewise hypotheses where each piece is in the class $D$.

We describe our merging algorithm in Section 4. As a special case, in Section 4.1 we first develop a merging algorithm for piecewise flat approximation. Then, in Section 4.2 we present the full merging algorithm for more general classes of distributions, assuming that they satisfy (i)-(iii).

When we adapt the general merging algorithm to a new class of piecewise hypotheses, the main algorithmic challenge is constructing a procedure for property (ii). More formally, we require a procedure with the following guarantee.
Definition 7. Fix $\eta > 0$. An algorithm $O_p(\hat{f}, J, \eta)$ is an $\eta$-approximate $A_k$-projection oracle for $D$ if it takes as input an interval $J$ and $\hat{f}$, and returns a hypothesis $h \in D$ such that

$$\|h - \hat{f}\|_{A_k} \leq \inf_{h' \in D} \|h' - \hat{f}\|_{A_k} + \eta.$$ 

One of our main contributions is an efficient $A_k$-projection oracle for the class of degree-$d$ polynomials, which we describe next.

**Level 2: $A_k$-projection for polynomials** Our $A_k$-projection oracle computes the coefficients $c \in \mathbb{R}^{d+1}$ of a degree-$d$ polynomial $p_c$ that approximately minimizes the $A_k$-distance to the empirical distribution $\hat{f}$ in the given interval $J$. Moreover, our oracle ensures that $p_c$ is non-negative on $J$.

At a high-level, we formulate the $A_k$-projection as a convex optimization problem. A key insight is that we can construct an efficient, approximate separation oracle for the set of polynomials that have an $A_k$-distance of at most $\tau$ to the empirical distribution $\hat{f}$. Combining this separation oracle with existing convex optimization algorithms allows us to solve the feasibility problem of checking whether we can achieve a given $A_k$-distance $\tau$. We then convert the feasibility problem to the optimization variant via a binary search over $\tau$.

Note that the set non-negative polynomials is a spectrahedron (the feasible set of a semidefinite program). After restricting the set of coefficients to non-negative polynomials, we can simplify the definition of the $A_k$-distance: it suffices to consider sets of intervals with endpoints at the locations of samples. Hence, we can replace the supremum in the definition of the $A_k$-distance by a maximum over a finite set, which shows that the set of polynomials that are both non-negative and $\tau$-close to $\hat{f}$ in $A_k$-distance is also a spectrahedron. This suggests that the $A_k$-projection problem could be solved by a black-box application of an SDP solver. However, this would lead to a running time that is exponential in $k$ because there are more than $\binom{2s}{k}$ possible sets of intervals, where $s$ is the number of sample points in the current interval $J$. While the authors of [CDSS14a] introduce an encoding of the $A_k$-constraint with fewer linear inequalities, their approach increases the number of variables in the optimization problem to depend polynomially on $\frac{1}{\epsilon}$, which leads to a super-linear running time.

We develop this algorithm in Section 5. Instead of using black-box SDP or LP solvers, we must construct an algorithm that exploits additional structure in the $A_k$-projection problem. Most importantly, our algorithm separates the dimension of the desired degree-$d$ polynomial from the number of samples (or equivalently, the error parameter $\epsilon$). This allows us to achieve a running time that is nearly-linear for a wide range of distributions. Interestingly, we can solve our SDP significantly faster than the LP which has been proposed in [CDSS14a] for the same problem. We achieve this by combining Vaidya’s cutting plane method [Vai96] with an efficient separation oracle that leverages the structure of the $A_k$-distance. This separation oracle is the third level of our algorithm.

**Level 3: $A_k$-separation oracle for polynomials** Our separation oracle must test two properties for a given polynomial $p_c$ with coefficients $c \in \mathbb{R}^{d+1}$: (i) Is the polynomial $p_c$ non-negative on the given interval $J$? (ii) Is the $A_k$-distance between $p_c$ and the empirical distribution $\hat{f}$ at most $\tau$?

We implement Test (i) by using known algorithms for finding roots of real polynomials efficiently [Pan01]. Note however, that root-finding algorithms cannot be exact for degrees larger than 4. Hence, we can only construct an approximate separation oracle. Nevertheless, we show that such an approximate oracle is still sufficient for solving the convex program outlined above.
We develop an efficient approximate separation oracle in Section 6. At a high level, our algorithm proceeds as follows. We first verify that our current candidate polynomial \( p_c \) is “almost” non-negative at every point. Assuming that \( p_c \) passes this test, we then compute the \( A_k \)-distance between \( p_c \) and \( \hat{f} \). We reduce this problem to a discrete variant by showing that the endpoints of intervals jointly maximizing the \( A_k \)-distance are guaranteed to coincide with sample points of the empirical distribution (assuming \( p_c \) is non-negative on the current interval). Using this insight, we give a combinatorial algorithm for this discrete variant that runs in time nearly-linear in the number of sample points in \( J \) and the degree \( d \). This algorithm is related to the problem of finding maximum-scoring DNA segment sets in computational biology [Csu04]. Once we have found a set of intervals maximizing the \( A_k \)-distance, we can convert it to a separating hyperplane for the polynomial coefficients \( c \) and the set of non-negative polynomials with \( A_k \)-distance at most \( \tau \) to \( \hat{f} \).

Combining these algorithmic components yields our algorithm with guarantees stated in Theorem 1.

4 Iterative Merging Algorithm

In this section we describe and analyze our iterative merging algorithm. We start with the case of histograms and then provide the generalization to piecewise polynomials.

4.1 The Histogram Merging Algorithm

Recall that a \( t \)-histogram is a function \( h : I \to \mathbb{R} \) that is piecewise constant with at most \( t \) interval pieces, i.e., there is a partition of \( I \) into intervals \( I_1, \ldots, I_t \) with \( t' \leq t \) such that \( h \) is constant on each \( I_i \). Given sample access to an arbitrary pdf \( f \) over \( I \) and a positive integer \( t \), we would like to efficiently compute a good \( t \)-histogram approximation to \( f \). Namely, if \( H_t = H_t(I) \) denotes the set of \( t \)-histogram probability density functions over \( I \) and \( \text{OPT}_t = \inf_{h \in H_t} \| h - f \|_1 \), our goal is to output an \( O(t) \)-histogram \( h : I \to \mathbb{R} \) that satisfies \( \| h - f \|_1 \leq C \cdot \text{OPT}_t + O(\epsilon) \) with high probability over the samples, where \( C \) is a universal constant.

The following notion of flattening a function over an interval will be crucial for our algorithm:

**Definition 8.** For a function \( g : I \to \mathbb{R} \) and an interval \( J = [u, v] \subseteq I \), we define the flattening of \( g \) over \( J \), denoted \( \overline{g}_J \), to be the constant function defined on \( J \) as

\[
\overline{g}_J(x) \overset{\text{def}}{=} \frac{g(J)}{v - u} \quad \text{for all } x \in J.
\]

For a set \( \mathcal{I} \) of disjoint intervals in \( I \) we define the flattening of \( g \) over \( \mathcal{I} \) to be the function \( \overline{g}_\mathcal{I} \) on \( \bigcup_{J \in \mathcal{I}} J \) which for each \( J \in \mathcal{I} \) satisfies \( \overline{g}_\mathcal{I}(x) = \overline{g}_J(x) \) for all \( x \in J \).

We start by providing an intuitive explanation of our algorithm followed by a proof of correctness. The algorithm draws \( n = \Theta((t + \log 1/\delta)/\epsilon^2) \) samples \( x_1 \leq x_2 \leq \ldots \leq x_n \) from \( f \). We start with the following partition of \( I = [a, b] \):

\[
\mathcal{I}_0 = \{(a, x_1), [x_1, x_1], (x_1, x_2), [x_2, x_2], \ldots, (x_{n-1}, x_n), [x_n, x_n], (x_n, b)\}.
\]

This is the partition where each interval is either a single sample point or the interval between two consecutive samples. Starting from this partition, our algorithm greedily merges intervals in a sequence of iterations. When deciding which intervals to merge, the following notion of approximation error will be crucial:
Definition 9. For a function $g : I \to \mathbb{R}$ and an interval $J \subseteq I$, define $e(g, J) = \|g - \overline{g}_J\|_{A_1, J}$. We call this quantity the $A_1$-error of $g$ on $J$.

In the $j$-th iteration, given the current partition $I_j$, we greedily merge pairs of consecutive intervals to form the new partition $I_{j+1}$. Let $s_j$ be the number of intervals in $I_j$. In particular, given $I_j = \{I_{1,j}, \ldots, I_{s_j,j}\}$, we consider the intervals

$$I'_{\ell,j} = I_{2\ell-1,j} \cup I_{2\ell,j}$$

for all $1 \leq \ell \leq s_j/2$. We first iterate through $1 \leq \ell \leq s_j/2$, and calculate the quantities

$$e_{\ell,j} = e(\hat{f}, I'_{\ell,j+1})$$

i.e., the $A_1$-errors of the empirical distribution on the candidate intervals.

To construct $I_{j+1}$, the algorithm keeps track of the largest $O(t)$ errors $e_{\ell,j}$. For each $\ell$ with $e_{\ell,j}$ being one of the $O(t)$ largest errors, we do not merge the corresponding intervals $I_{2\ell-1,j}, I_{2\ell,j}$. That is, we include $I_{2\ell-1,j}$ and $I_{2\ell,j}$ in the new partition $I_{j+1}$. Otherwise, we include their union $I'_{\ell,j+1}$ in $I_{j+1}$. We perform this procedure $O(\log \frac{1}{\epsilon})$ times and arrive at some final partition $I$. Our output hypothesis is the flattening of $\hat{f}$ with respect to $I$.

For a formal description of our algorithm, see the pseudocode given in Algorithm 1 below. In addition to the parameter $t$, our algorithm has a parameter $\alpha \geq 1$ which controls the trade-off between the constant $C$ achieved by the algorithm and the number of pieces in the output histogram.

We prove the following theorem:

Theorem 10. Algorithm ConstructHistogram($f, t, \alpha, \epsilon, \delta$) uses $n = O((\alpha t + \log 1/\delta)/\epsilon^2)$ samples from $f$, runs in time $O(n \log(1/\epsilon))$, and outputs a hypothesis $h$ and a corresponding partition $I$ with $|I| \leq 2\alpha \cdot t$ such that with probability at least $1 - \delta$ we have

$$\|h - f\|_1 \leq 2 \cdot \text{OPT}_t + \frac{4 \cdot \text{OPT}_t + 4\epsilon}{\alpha - 1} + \epsilon .$$

Proof. We start by analyzing the running time. To this end, we show that the number of intervals decreases exponentially with the number of iterations. At the $j$th iteration we merge all but $\alpha t$ intervals. Therefore,

$$s_{j+1} = \alpha t + \frac{s_j - \alpha t}{2} = \frac{3s_j}{4} + \frac{2\alpha t - s_j}{4} .$$

Note that the algorithm enters the while loop when $s_j > 2\alpha t$, implying that

$$s_{j+1} < \frac{3s_j}{4} .$$

The number of intervals is at least $\alpha t$, when the algorithm exits the while loop. Therefore, the number of iterations of the while loop is at most

$$O \left( \log \frac{n}{\alpha t} \right) = O \left( \log \frac{1}{\epsilon} \right) ,$$

which follows by plugging the value of $n$ from the theorem statement. We now show that each iteration takes time $O(n)$. Without loss of generality assume that we compute the $A_1$ distance

---

2We assume $s_j$ is even for simplicity.
Algorithm 1 Approximating with histograms by merging.

1: function ConstructHistogram(f, t, α, ε, δ)
2: Take n = Θ((αt + log 1/δ) / ε²) samples x₁ ≤ x₂ ≤ ... ≤ xₙ. Form the empirical f̂ from these samples.
3: Let I₀ ← {[a, x₁], [x₁, x₂], [x₂, x₃], ..., (xₙ₋₁, xₙ), [xₙ, xₙ], (xₙ, b]} be the initial partition.
4: j ← 0
5: while |I_j| > 2α · t do
6: Let I_j = {I₁,j, I₂,j, ..., I_s_j−1,j, I_s_j,j}
7: for ℓ ∈ {1, 2, ..., s_j} do
8: I′_{ℓ,j+1} ← I_{2ℓ−1,j} ∪ I_{2ℓ,j}
9: e_{ℓ,j} ← e(f, I′_{ℓ,j+1})
10: end for
11: Let L be the set of ℓ ∈ {1, 2, ..., s_j} with the αt largest errors e_{ℓ,j} and let M be its complement.
12: I_{j+1} ← \bigcup_{ℓ \in L} I_{2ℓ−1,j} ∪ I_{2ℓ,j}
13: I_{j+1} ← I_{j+1} ∪ \{I′_{ℓ,j+1} \mid ℓ \in M\}
14: j ← j + 1
15: end while
16: return I = I_j and the flattening f̂_I
17: end function

only over intervals ending at a data sample. For an interval J = [c, d] containing m sample points, x₁, ..., xₙ, let C_j = (x_j−1) - (d-c) / n. The A₁ distance of J is given by max C_j - min C_j, and can therefore be computed in time proportional to the number of sample points in the interval. Therefore the total time of the algorithm is O(n log(1/ε)), as claimed.

We now proceed to bound the error. Let I = {I₁, ..., Iₙ} be the partition of I returned by ConstructHistogram. The desired bound on |I| follows immediately because the algorithm terminates only when |I| ≤ 2αt. The rest of the proof is dedicated to the proof of (2).

Fix h* ∈ H₁ such that ∥h* - f∥₁ = OPT₁. Let I* = {I₁*, ..., Iₙ*} be the partition induced by the discontinuities of h*. Call a point at a boundary of any I_j* a jump of h*. For any interval J ⊆ I we define Γ(J) to be the number of jumps of h* in the interior of J. Since we draw n = Ω((αt + log 1/δ) / ε²) samples, Corollary 4 implies that with probability at least 1 − δ, we have

∥f̂ - f∥_{A₁(2αt)} ≤ ε.

We condition on this event throughout the analysis.

We split the total error into three terms based on the final partition I:

Case 1: Let F be the set of intervals in I with zero jumps in h*, i.e., F = {J ∈ I \ Γ(J) = 0}.

Case 2a: Let J₀ be the set of intervals in I that were created in the initial partitioning step of the algorithm and which contain a jump of h*, i.e., J₀ = {J ∈ I \ Γ(J) > 0 and J ∈ I₀}.

Case 2b: Let J₁ be the set of intervals in I that contain at least one jump, and were created by merging two other intervals, i.e., J₁ = {J ∈ I \ Γ(J) > 0 and J \notin I₀}.
Notice that $\mathcal{F}, \mathcal{J}_0, \mathcal{J}_1$ form a partition of $\mathcal{I}$, and thus

$$\|h - f\|_1 = \|h - f\|_{1, \mathcal{F}} + \|h - f\|_{1, \mathcal{J}_0} + \|h - f\|_{1, \mathcal{J}_1}.$$  

We will bound the error from above in the three cases separately. In particular, we will show:

\begin{align*}
\|h - f\|_{1, \mathcal{F}} &\leq 2 \cdot \|f - h^*\|_{1, \mathcal{F}} + \|\hat{f} - f\|_{A_{\mathcal{F}}}, \quad (3) \\
\|h - f\|_{1, \mathcal{J}_0} &\leq \|\hat{f} - f\|_{A_{\mathcal{J}_0}} , \quad (4) \\
\|h - f\|_{1, \mathcal{J}_1} &\leq 4 \cdot \|\mathbf{OPT}\|_1 + 4\epsilon + 2 \cdot \|f - h^*\|_{1, \mathcal{J}_1} + \|\hat{f} - f\|_{A_{\mathcal{J}_1} + \epsilon}. \quad (5)
\end{align*}

Using these results along with the fact that $\|f - h^*\|_{1, \mathcal{F}} + \|f - h^*\|_{1, \mathcal{J}_1} \leq \mathbf{OPT}$, we have

\begin{align*}
\|h - f\|_1 &\leq 2 \cdot \mathbf{OPT} + \frac{4 \cdot \mathbf{OPT}}{\alpha - 1} + \|\hat{f} - f\|_{A_{\mathcal{F}}} + \|\hat{f} - f\|_{A_{\mathcal{J}_0}} + \|\hat{f} - f\|_{A_{\mathcal{J}_1} + \epsilon} \\
&\leq 2 \cdot \mathbf{OPT} + \frac{4 \cdot \mathbf{OPT}}{\alpha - 1} + \|\hat{f} - f\|_{A_{(2^a + 1)t}} \\
&\leq 2 \cdot \mathbf{OPT} + \frac{4 \cdot \mathbf{OPT}}{\alpha - 1} + \epsilon ,
\end{align*}

where inequality (a) follows from Fact 6(d) and inequality (b) follows from the VC inequality. Thus, it suffices to prove Equations (3)–(5).

**Case 1.** We first consider the interval $\mathcal{F}$. By the triangle inequality we have

$$\|h - f\|_{1, \mathcal{F}} \leq \|f - h^*\|_{1, \mathcal{F}} + \|h - h^*\|_{1, \mathcal{F}}.$$  

Thus to show (3), it suffices to show that

$$\|h - h^*\|_{1, \mathcal{F}} \leq \|f - h^*\|_{1, \mathcal{F}} + \|\hat{f} - f\|_{A_{\mathcal{F}}}.$$  

We prove a slightly more general version of (6) that holds over all finite sets of intervals not containing any jump of $h^*$. We will use this general version also later in our proof.

**Lemma 11.** Let $\mathcal{J} \in \mathcal{I}_1^2$ so that $\Gamma(J) = 0$ for all $J \in \mathcal{J}$. Let $\overline{h} = \overline{f_\mathcal{J}}$ denote the flattening of $\hat{f}$ on $\mathcal{J}$. Then

$$\|\overline{h} - h^*\|_{1, \mathcal{J}} \leq \|f - h^*\|_{1, \mathcal{J}} + \|\hat{f} - f\|_{A_{\mathcal{J}}}.$$  

Note that this is indeed a generalization of (6) since for any point $x$ in any interval of $\mathcal{F}$, we have $h(x) = \overline{f_\mathcal{F}}(x)$.

**Proof of Lemma 11.** In any interval $J \in \mathcal{J}$ with $\Gamma(J) = 0$, we have

$$\|\overline{h} - h^*\|_{1, J} \overset{(a)}{=} |\overline{h}(J) - h^*(J)| \overset{(b)}{=} |\hat{f}(J) - h^*(J)|,$$  

13
where (a) follows from the fact that \( \overline{h} \) and \( h^* \) are constant in \( J \), and (b) follows from the definition of \( \overline{h} \). Thus, we get

\[
\| \overline{h} - h^* \|_{1,J} = \sum_{J \in F} \| \overline{h} - h^* \|_{1,J} \\
= \sum_{J \in J} | \hat{f}(J) - h^*(J) | \\
\leq \sum_{J \in J} | \hat{f}(J) - f(J) | + \sum_{J \in J} | f(J) - h^*(J) | \\
\leq \| \hat{f} - f \|_{A(J),J} + \| f - h^* \|_{1,J},
\]

where (c) uses the triangle inequality and (d) follows from the definition of \( A \)-distance.

**Case 2a.** Next, we analyze the error for the intervals \( J_0 \). The set \( I_0 \) contains only singletons and intervals with no sample points. By definition, only the intervals in \( I_0 \) that contain no samples may contain a jump of \( h^* \). The singleton intervals containing the sample points do not include jumps, and are hence covered by Case 1. Since \( J_0 \) does not contain any samples, our algorithm assigns \( h(J) = \hat{f}(J) = 0 \) for any \( J \in J_0 \). Hence,

\[
\| h - f \|_{1,J_0} = \| f \|_{1,J_0}.
\]

We thus have the following sequence of (in)equalities

\[
\| h - f \|_{1,J_0} = \| f \|_{1,J_0} \\
= \sum_{J \in J_0} | f(J) | \\
= \sum_{J \in J_0} | f(J) - \hat{f}(J) | \\
\leq \| \hat{f} - f \|_{A(J_0),J_0},
\]

where the last step uses the definition of the \( A \)-norm.

**Case 2b.** Finally, we bound the error for intervals in \( J_1 \), i.e., intervals that were created by merging in some iteration of our algorithm and also contain jumps. As before, our first step is the following triangle inequality:

\[
\| h - f \|_{1,J_1} \leq \| h - h^* \|_{1,J_1} + \| h^* - f \|_{1,J_1}.
\]

Consider an interval \( J \in J_1 \). Since \( h \) is a constant in \( J \) and \( h^* \) has \( \Gamma(J) \) jumps in \( J \), \( h - h^* \) has at most \( \Gamma(J) \) sign changes in \( J \). Therefore,

\[
\| h - h^* \|_{1,J} \overset{(a)}{=} \| h - h^* \|_{A(\Gamma(J)) + 1, J} \\
\overset{(b)}{\leq} \| h - \hat{f} \|_{A(\Gamma(J)) + 1, J} + \| \hat{f} - f \|_{A(\Gamma(J)) + 1, J} + \| f - h^* \|_{A(\Gamma(J)) + 1, J} \\
\overset{(c)}{\leq} (\Gamma(J) + 1) \| h - \hat{f} \|_{A(J),J} + \| \hat{f} - f \|_{A(\Gamma(J)) + 1, J} + \| f - h^* \|_{1,J},
\]

where (c) uses the triangle inequality and (d) follows from the definition of \( A \)-distance.
where equality (a) follows from Fact 6(a), inequality (b) is the triangle inequality, and inequality (c) uses Fact 6(c). Finally, we bound the $A_1$ distance in the first term above.

**Lemma 12.** For any $J \in \mathcal{J}_t$, we have

$$\|h - \hat{f}\|_{A_1,J} \leq \frac{2\text{OPT}_t + 2\epsilon}{(\alpha - 1)t}.$$  (8)

Before proving the lemma, we show how to use it to complete Case 2b. Since $h$ is the flattening of $\hat{f}$ over $J$, we have that $\|h - \hat{f}\|_{A_1,J} = e(\hat{f}, J)$. Applying (7) gives:

$$\|h - h^*\|_{1,\mathcal{J}_t} = \sum_{J \in \mathcal{J}_t} \|h - h^*\|_{1,J}$$

$$\leq \sum_{J \in \mathcal{J}_t} \left( (\Gamma(J) + 1)\|h - \hat{f}\|_{A_1,J} + \|\hat{f} - f\|_{A(\Gamma(J)+1),J} + \|f - h^*\|_{1,J} \right)$$

$$\leq \frac{2\cdot\text{OPT}_t + 2\epsilon}{(\alpha - 1)t} \cdot \left( \sum_{J \in \mathcal{J}_t} (\Gamma(J) + 1) \right) + \sum_{J \in \mathcal{J}_t} \|\hat{f} - f\|_{A(\Gamma(J)+1),J} + \|f - h^*\|_{1,\mathcal{J}_t}$$

$$\leq \frac{4\cdot\text{OPT}_t + 4\epsilon}{(\alpha - 1)} + \|\hat{f} - f\|_{A(\Gamma(J)+1),\mathcal{J}_t} + \|f - h^*\|_{1,\mathcal{J}_t}$$

where inequality (a) uses the fact that $\Gamma(J) \geq 1$ for these intervals and hence

$$\sum_{J \in \mathcal{J}_t} (\Gamma(J) + 1) \leq 2 \sum_{J \in \mathcal{J}_t} \Gamma(J) \leq 2t.$$  

We now complete the final step by proving Lemma 12.

**Proof of Lemma 12.** Recall that in each iteration of our algorithm, we merge pairs of intervals except those with the $\alpha t$ largest errors. Therefore, if two intervals were merged, there were at least $\alpha t$ other interval pairs with larger error. We will use this fact to bound the error on the intervals in $\mathcal{J}_t$.

Consider any interval $J \in \mathcal{J}_t$, and suppose it was created in the $j$th iteration of the while loop of our algorithm, i.e., $J = I'_{ij+1} = I_{2i-1,j} \cup I_{2i,j}$ for some $i \in \{1, \ldots, s_j/2\}$. Note that this interval remains intact in the remainder of the algorithm. Recall that the intervals $I'_{ij+1}$, for $i \in \{1, \ldots, s_j/2\}$, are the possible candidates for merging at iteration $j$. Let $h' = \hat{f}_{I'_{ij+1}}$ be the distribution obtained by flattening the empirical distribution over these candidate intervals $I'_{ij+1} = \{I'_{1,ij+1}, \ldots, I'_{s_j/2,ij+1}\}$. Note that $h'(x) = h(x)$ for $x \in J$ because $J$ was created in this iteration.

Let $\mathcal{L}$ be the set of candidate intervals $I'_{ij+1}$ in the set $I'_{ij+1}$ with the largest $\alpha t$ errors $e(\hat{f}, I'_{ij+1})$. Let $\mathcal{L}_0$ be the intervals in $\mathcal{L}$ that do not contain any jumps of $h^*$. Since $h^*$ has at most $t$ jumps, $|\mathcal{L}_0| \geq (\alpha - 1)t$. Moreover, for any $I' \in \mathcal{L}_0$, by the triangle inequality

$$e(\hat{f}, I') = \|h' - \hat{f}\|_{A_1,I'}$$

$$\leq \|h' - h^*\|_{A_1,I'} + \|f - h^*\|_{A_1,I'} + \|f - \hat{f}\|_{A_1,I'}$$

$$\leq \|h' - h^*\|_{A_1,I'} + \|f - h^*\|_{1,I'} + \|f - \hat{f}\|_{A_1,I'}.$$
Summing over the intervals in $\mathcal{L}_0$,

$$\sum_{I' \in \mathcal{L}_0} e(\hat{f}, I') \leq \sum_{I' \in \mathcal{L}_0} \left( \|h' - h^*\|_{A_1,I'} + \|f - h^*\|_{1,I'} + \|f - \hat{f}\|_{A_1,I'} \right)$$

$$\leq \left( \sum_{I' \in \mathcal{L}_0} \|h' - h^*\|_{A_1,I'} \right) + \|f - h^*\|_{1,\mathcal{L}_0} + \|f - \hat{f}\|_{A_2,\mathcal{L}_0}$$

$$\leq \left( \sum_{I' \in \mathcal{L}_0} \|h' - h^*\|_{A_1,I'} \right) + \text{OPT}_t + \epsilon \ , \quad (9)$$

where recall that we had conditioned on the last term being at most $\epsilon$ throughout the analysis.

Since both $h$ and $h^*$ are flat on each interval $I' \in \mathcal{L}_0$, by Lemma 11, we have

$$\sum_{I' \in \mathcal{L}_0} \|h' - h^*\|_{A_1,I'} \leq \|f - h^*\|_{1,\mathcal{L}_0} + \|\hat{f} - f\|_{A_1,\mathcal{L}_0} \leq \text{OPT}_t + \epsilon .$$

Plugging this into (9) gives

$$\sum_{I' \in \mathcal{L}_0} e(\hat{f}, I') \leq 2 \cdot \text{OPT}_t + 2 \epsilon .$$

Since $J$ was created by merging in this iteration, this implies that $e(\hat{f}, J)$ is no larger than $e(\hat{f}, I')$ on any of the intervals $I' \in \mathcal{L}_0$ (see lines 11 - 13 of Algorithm 1), and therefore is not larger than the average. Recalling that $|\mathcal{L}_0| \geq (\alpha - 1)t$, we obtain

$$e(\hat{f}, J) = \|h' - \hat{f}\|_{A_1,J} = \|h - \hat{f}\|_{A_1,J} \leq \frac{\sum_{I' \in \mathcal{L}_0} e(\hat{f}, I')}{(\alpha - 1)t} \leq \frac{2 \text{OPT}_t + 2 \epsilon}{(\alpha - 1)t} ,$$

completing the proof of the lemma.

### 4.2 The General Merging algorithm

We are now ready to present our general merging algorithm, which is a generalized version of the histogram merging algorithm introduced in Section 4.1. The histogram algorithm only uses three main properties of histogram hypotheses: (i) The number of intersections of two $t$-histogram hypotheses is bounded by $O(t)$. (ii) Given an interval $J$ and an empirical distribution $\hat{f}$, it is easy to find a good histogram fit to $\hat{f}$ on this interval. (iii) We can compute the $A_1$-errors of candidate intervals efficiently.

Note that property (i) limits the complexity of the hypothesis class and leads to a tight sampling bound while properties (ii) and (iii) are algorithmic ingredients. We can generalize these three notions to arbitrary classes of piecewise hypotheses. Let $\mathcal{D}$ be a class of hypotheses. Then the generalized variants of (i) to (iii) are: (i) The number of intersections between any two hypotheses in $\mathcal{D}$ is bounded. (ii) Given an interval $J$ and an empirical distribution $\hat{f}$, we can efficiently find the best fit to $\hat{f}$ from functions in $\mathcal{D}$ in terms of the $A_k$-distance. (iii) We can also efficiently compute the $A_k$-distance between the empirical distribution and any hypothesis in $\mathcal{D}$. Using these generalized notions, the histogram merging algorithm naturally extends to agnostically learning piecewise hypotheses where each piece is in the class $\mathcal{D}$.

To formally state our results, we will need a few definitions.
Definition 13. Let $\mathcal{D}$ be a family of measurable functions defined over subsets of $I$. $\mathcal{D}$ is said to be full if for each $J \subseteq I$, there exists a function $g$ in $\mathcal{D}$ whose domain is $J$. Let $\mathcal{D}_J$ be the elements of $\mathcal{D}$ whose domain is $J$.

Definition 14. A function $h : I \to \mathbb{R}$ is a $t$-piece $\mathcal{D}$-function if there exists a partition of $I$ into intervals $I_1, \ldots, I_t$ with $t' \leq t$, such that for every $i$ there exists $h_i \in \mathcal{D}_{I_i}$ satisfying that $h = h_i$ on $I_i$. Let $\mathcal{D}_t$ denote the set of all $t$-piece $\mathcal{D}$-functions.

Example 1. Let $\mathcal{H}_J$ be the set of constant functions defined on $J$. Then if $\mathcal{H} = \bigcup_{J \subseteq I} \mathcal{H}_J$, the set $\mathcal{H}_t$ of $t$-piece $\mathcal{H}$-functions is the set piecewise constant functions on $I$ with at most $t$ interval pieces. We will refer to these functions as $t$-histograms.

Example 2. For $J \subseteq I$, we define $\mathcal{P}_{J,d}$ to be set of degree-$d$ nonnegative polynomials on $J$, and $\mathcal{P}_d \overset{\text{def}}{=} \bigcup_{J} \mathcal{P}_{J,d}$. Since the degree $d$ will be fixed throughout paper, we sometimes simply denote this set by $\mathcal{P}$. The set $\mathcal{P}_{t,d}$ of $t$-piece $\mathcal{P}$-functions is the set of $t$-piece degree-$d$ non-negative polynomials. It is easy to see that this class is full over the real line.

Fix a positive integer $t$ and a class of functions $\mathcal{D}$. Given sample access to any pdf $f : I \to \mathbb{R}_+$, we want to compute a good $\mathcal{D}_t$ approximation to $f$. We define $\text{OPT}_{\mathcal{D},t} \overset{\text{def}}{=} \inf_{g \in \mathcal{D}_t} \|g - f\|_1$. Our goal is to find an $O(t)$-piece $\mathcal{D}$-function $h : I \to \mathbb{R}$ such that $\|h - f\|_1 \leq C \cdot \text{OPT}_t + O(\epsilon)$, with high probability over the samples, where $C$ is a universal constant.

We consider function classes $\mathcal{D}$ such that any two functions in $\mathcal{D}$ intersect a bounded number of times.

Definition 15. Suppose $h \in \mathcal{D}_J$ and $h' \in \mathcal{D}_k$ for some $k \geq t$. A full family $\mathcal{D}$ is $d$-sign restricted if the function $h - h'$ has at most $(s + 1)d$ sign changes on $J$ whenever there is a partition $I_1, \ldots, I_k$ of $I$ so that $h' = h$ for some $h \in \mathcal{D}_{I_i}$ on each $I_i$ and $J$ contains at most $s$ points on the boundaries of the $I_i$.

Since any two polynomials of degree $d$ intersect at most $d$ times, they form a $d$-sign restricted class.

Our iterative merging algorithm takes as input samples from an arbitrary distribution, and an integer $t$, and outputs an $O(t)$-piecewise $\mathcal{D}$ distribution with agnostic guarantees, assuming access to objects we call $\mathcal{A}_k$-projection and $\mathcal{A}_k$-computation oracles. $\mathcal{A}_k$-projection oracle was defined in Definition 7 and is restated below along with the definition of $\mathcal{A}_k$-computation oracle (Definition 16).

Definition 7. Fix $\eta > 0$. An algorithm $\mathcal{O}_p(\hat{f}, J, \eta)$ is an $\eta$-approximate $\mathcal{A}_k$-projection oracle for $\mathcal{D}$ if it takes as input an interval $J$ and $\hat{f}$, and returns a hypothesis $h \in \mathcal{D}$ such that

$$\|h - \hat{f}\|_{\mathcal{A}_k} \leq \inf_{h' \in \mathcal{D}} \|h' - \hat{f}\|_{\mathcal{A}_k} + \eta.$$ 

Definition 16. Fix $\eta > 0$. An algorithm $\mathcal{O}_c(\hat{f}, h, J, \eta)$ is an $\eta$-approximate $\mathcal{A}_k$-computation oracle for $\mathcal{D}$ if it takes as input $\hat{f}$, a subinterval $J \subseteq I$, and a function $h, J \in \mathcal{D}_J$, and returns a value $\xi$ such that

$$\|h, J - \hat{f}\|_{\mathcal{A}_k, J - \xi} \leq \eta.$$
We consider a $d$-sign restricted full family $\mathcal{D}$, and a fixed $\eta > 0$. Let $R_p(I) = R_p(I, \hat{f}, \mathcal{O}_p)$ and $R_c(I) = R_c(I, \hat{f}, \mathcal{O}_c)$ be the time used by $\mathcal{O}_p$ and $\mathcal{O}_c$ respectively. With a slight abuse of notation, for a collection of $\leq 2n$ intervals containing $n$ points in the support of the empirical, we also define $R_p(n)$ and $R_c(n)$ to be the maximum time taken by $\mathcal{O}_p$ and $\mathcal{O}_c$ respectively.

We are now ready to state the main theorem of this section:

**Theorem 17.** Let $\mathcal{O}_p$ and $\mathcal{O}_c$ be $\eta$-approximate $A_k$-projection and $A_k$-computation oracles for $\mathcal{D}$. Algorithm **General-Merging**$(f, t, \alpha, \epsilon, \delta)$ uses $n = O((\alpha dt + \log 1/\delta)/\epsilon^2)$ samples, runs in time $O((R_p(n) + R_c(n)) \log (\eta n / \alpha t))$, and outputs a hypothesis $h$ and an interval partition $\mathcal{I}$ such that $|\mathcal{I}| \leq 2\alpha \cdot t$ and with probability at least $1 - \delta$ we have

$$
\|h - f\|_1 \leq 3 \cdot \text{OPT}_{\mathcal{D}, t} + \frac{\text{OPT}_{\mathcal{D}, t} + \epsilon}{\alpha - 1} + 2\epsilon + \eta.
$$

(10)

In the remaining of this section we provide an intuitive explanation of the algorithm of Theorem 17 followed by a detailed pseudocode.

The algorithm **General-Merging** and its analysis is a generalization of the **ConstructHistogram**. More formally, the algorithm proceeds greedily, as before. We take $n = O((\alpha dt + \log 1/\delta)/\epsilon^2)$ samples $x_1 \leq \ldots \leq x_n$. We construct $\mathcal{I}_0$ as in (1). In the $j$-th iteration, given the current partition $\mathcal{I}_j = \{I_{1,j}, \ldots, I_{s_j,j}\}$ with $s_j$ intervals, consider the intervals

$$
I'_{\ell,j+1} = I_{2\ell-1,j} \cup I_{2\ell,j}
$$

for $\ell \leq s_j/2$. As with histograms, we want to compute the errors in each of the new intervals created. To do this, we first call the $A_k$-projection oracle with $k = d + 1$ on this interval to find the (nearly) best function in $\mathcal{D}$ for $\hat{f}$ over these new intervals, namely:

$$
h'_{\ell,j} = \mathcal{O}_p\left(\hat{f}, I'_{\ell,j+1}, \frac{\eta}{O(t)}\right).
$$

To compute the (approximate) error, we now call the $A_k$-computation oracle with $k = d + 1$, namely we compute:

$$
e_{\ell,j} = \mathcal{O}_c\left(\hat{f}, h'_{\ell,j}, I'_{\ell,j+1}, \frac{\eta}{O(t)}\right).
$$

As in **ConstructHistogram**, we keep the intervals with the largest $O(t)$ errors intact and merge the remaining pairwise intervals. We perform this procedure $O(\log \frac{n}{t})$ times and arrive at some final partition $\mathcal{I}$ with $O(t)$ pieces. Our output hypothesis is the output of $\mathcal{O}_p(\hat{f}, I)$ over each of the final intervals $I$.

The formal pseudocode for our algorithm is given in Algorithm 2. We assume $\mathcal{D}$ and $d$ is known and fixed and is not mentioned explicitly as an input to the algorithm. Note that we run the algorithm with $\eta = \epsilon$ so that Theorem 17 has an additional $O(\epsilon)$ error. The proof of Theorem 17 is very similar to that of the histogram merging and is deferred to Appendix A.

**4.3 Putting everything together**

In sections 5 and 6.3 we develop an efficient approximate $A_k$–projection oracle and an $A_k$–computation oracle for $\mathcal{P}_d$, respectively. We show that:
Algorithm 2 Approximating with general hypotheses by merging.

1: function \textsc{General-Merging}(f, d, t, \alpha, \epsilon, \delta)
2: \hspace{1em} Take \( n = \Theta((\alpha d t + \log 1/\delta)/\epsilon^2) \) samples \( x_1 \leq x_2 \leq \ldots \leq x_n \). Form the empirical \( \hat{f} \) from these samples.
3: \hspace{1em} Let \( \mathcal{I}_0 \leftarrow \{[a, x_1), [x_1, x_1), (x_1, x_2), [x_2, x_2], \ldots, (x_{n-1}, x_n), [x_n, x_n), (x_n, b]\} \) be the initial partition.
4: \hspace{1em} \( j \leftarrow 0 \)
5: \hspace{1em} while \( |\mathcal{I}_j| > 2\alpha \cdot t \) do
6: \hspace{2em} Let \( \mathcal{I}_j = \{I_{1,j}, I_{2,j}, \ldots, I_{s_j-1,j}, I_{s_j,j}\} \)
7: \hspace{2em} for \( \ell \in \{1, 2, \ldots, s_j\} \) do
8: \hspace{3em} \( I'_{\ell,j+1} \leftarrow I_{2\ell-1,j} \cup I_{2\ell,j} \)
9: \hspace{3em} \( h'_{\ell,j} \leftarrow \mathcal{O}_p(\hat{f}, I'_{\ell,j+1}, \frac{s_j}{2\alpha t}) \)
10: \hspace{3em} \( e_{\ell,j} \leftarrow \mathcal{O}_c(\hat{f}, h'_{\ell,j}, I'_{\ell,j+1}, \frac{s_j}{2\alpha t}) \)
11: \hspace{2em} end for
12: \hspace{1em} Let \( L \) be the set of \( \ell \in \{1, 2, \ldots, s_j\} \) with the \( \alpha t \) largest errors \( e_{\ell,j} \) and let \( M \) be its complement.
13: \hspace{1em} \( \mathcal{I}_{j+1} \leftarrow \bigcup_{\ell \in L} \{I_{2\ell-1,j}, I_{2\ell,j}\} \)
14: \hspace{1em} \( \mathcal{I}_{j+1} \leftarrow \mathcal{I}_{j+1} \cup \{I'_{\ell,j+1} | \ell \in M\} \)
15: \hspace{1em} \( j \leftarrow j + 1 \)
16: end while
17: return \( \mathcal{I} = \mathcal{I}_j \) and the functions \( \mathcal{O}_p(\hat{f}, J, \frac{s_j}{2\alpha t}) \) for \( J \in \mathcal{I} \)
18: end function

**Theorem 18.** Fix \( J \subset [-1, 1] \) and \( \eta > 0 \). For all \( k \leq d \), there is an \( \eta \)-approximate \( \mathcal{A}_k \)-projection oracle for \( \mathcal{P}_d \) which runs in time

\[
    O\left(\left(d^4 \log \log 1/\eta + sd^2 + d^{\omega+2}\log \frac{1}{\eta}\right)\right).
\]

where \( s \) is the size of the support of \( \hat{f} \) on \( J \).

**Theorem 19.** There is an \( \eta \)-approximate \( \mathcal{A}_k \)-computation oracle for \( \mathcal{P} \), which runs in time \( O(s (\log s + \log d + \log \log d)) \) where \( s \) is the size of the support of \( \hat{f} \) on \( J \).

\textsc{General-Merging}, when used in conjunction with the oracles \( \mathcal{O}_p \) and \( \mathcal{O}_c \) promised in Theorems 18 and 19 respectively, yields Theorem 1. In particular, set \( \eta = \epsilon \). Then, we have that for this choice of oracles, \( R_p(n) + R_c(n) = O(n d^{\omega+2} \log 1/\epsilon) \). This completes the proof.

5 A fast \( \mathcal{A}_k \)-projection oracle for polynomials

We now turn our attention to the \( \mathcal{A}_k \)-projection problem, which appears as the main subroutine in the general merging algorithm (see Section 4.2). In this section we let \( E \subset J \) be the set of samples drawn. To emphasize the dependence of the empirical distribution on \( E \), in this section we denote the empirical distribution by \( \hat{f}_E \). Given an interval \( J = [a, b] \) and a set of samples \( E \subset J \), the goal of the \( \mathcal{A}_k \)-projection oracle is to find a hypothesis \( h \in \mathcal{H} \) such that the \( \mathcal{A}_k \)-distance between the
empirical distribution $\hat{f}_E$ and the hypothesis $h$ is minimized. In contrast to the merging algorithm, the $A_k$-projection oracle depends on the underlying hypothesis class $H$, and here we present an efficient oracle for non-negative polynomials with fixed degree $d$. In particular, our $A_k$-projection oracle computes the coefficients $c \in \mathbb{R}^{d+1}$ of a degree-$d$ polynomial $p_c$ that approximately minimizes the $A_k$-distance to the given empirical distribution $\hat{f}_E$ in the interval $J$. Moreover, our oracle ensures that $p_c$ is non-negative for all $x \in J$.

At a high-level, we formulate the $A_k$-projection as a convex optimization problem. A key insight is that we can construct an efficient, approximate separation oracle for the set of polynomials that have an $A_k$-distance of at most $\tau$ to the empirical distribution $\hat{f}_E$. Combining this separation oracle with existing convex optimization algorithms allows us to solve the feasibility problem of checking whether we can achieve a given $A_k$-distance $\tau$. We then convert the feasibility problem to the optimization variant via a binary search over $\tau$.

In order to simplify notation, we assume that the interval $J$ is $[-1, 1]$ and that the mass of the empirical distribution $\hat{f}_E$ is 1. Note that the general $A_k$-projection problem can easily be converted to this special case by shifting and scaling the sample locations and weights before passing them to the $A_k$-projection subroutine. Similarly, the resulting polynomial can be transformed to the original interval and weights.\(^3\)

### 5.1 The set of feasible polynomials

For the feasibility problem, we are interested in the set of degree-$d$ polynomials that have an $A_k$-distance of at most $\tau$ to the empirical distribution $\hat{f}_E$ on the interval $J = [-1, 1]$ and are also non-negative on $J$. More formally, we study the following set.

**Definition 20** (Feasible polynomials). Let $E \subset J$ be the samples of an empirical distribution with $\hat{f}_E(J) = 1$. Then the set of $(\tau, d, k, E)$-feasible polynomials is

$$C_{\tau, d, k, E} := \left\{ c \in \mathbb{R}^{d+1} \mid \| p_c - \hat{f}_E \|_{A_k, J} \leq \tau \text{ and } p_c(x) \geq 0 \text{ for all } x \in J \right\}.$$  

When $d$, $k$, and $E$ are clear from the context, we write only $C_{\tau}$ for the set of $\tau$-feasible polynomials.

Considering the original $A_k$-projection problem, we want to find an element $c^* \in C_{\tau^*}$, where $\tau^*$ is the smallest value for which $C_{\tau^*}$ is non-empty. We solve a slightly relaxed version of this problem, i.e., find an element $c$ for which the $A_k$-constraint and the non-negativity constraint are satisfied up to small additive constants. We then post-process the polynomial $p_c$ to make it truly non-negative while only increasing the $A_k$-distance by a small amount.

Note that we can “unwrap” the definition of the $A_k$-distance and write $C$ as an intersection of sets in which each set enforces the constraint $\sum_{i=1}^k |p_c(I_i) - \hat{f}_E(I_i)| \leq \tau$ for one collection of $k$ disjoint intervals $\{I_1, \ldots, I_k\}$. For a fixed collection of intervals, we can then write each $A_k$-constraint as the intersection of linear constraints in the space of polynomials. Similarly, we can

\(^3\)Technically, this step is actually necessary in order to avoid a running time that depends on the shape of the unknown pdf $f$. Since the pdf $f$ could be supported on a very small interval only, the corresponding polynomial approximation could require arbitrarily large coefficients (the empirical distribution would have all samples in a very small interval). In that case, operations such as root-finding with good precision could take an arbitrary amount of time. In order to circumvent this issue, we make use of the real-RAM model to rescale our samples to $[-1, 1]$ before processing them further. Combined with the assumption of unit probability mass, this allows us to bound the coefficients of candidate polynomials in the current interval.
write the non-negativity constraint as an intersection of pointwise non-negativity constraints, which are again linear constraints in the space of polynomials. This leads us to the following key lemma. Note that convexity of $C_\tau$ could be established more directly, but considering $C_\tau$ as an intersection of halfspaces illustrates the further development of our algorithm (see also the comments after the lemma).

**Lemma 21** (Convexity). The set of $\tau$-feasible polynomials is convex.

**Proof.** From the definitions of $C_\tau$ and the $A_k$-distance, we have

$$C_\tau = \{ c \in \mathbb{R}^{d+1} | \| p_c - \hat{f}_E \|_{A_k,J} \leq \tau \text{ and } p_c(x) \geq 0 \text{ for all } x \in J \}$$

$$= \{ c \in \mathbb{R}^{d+1} | \sup_{I \in J} \sum_{I \in I} |p_c(I) - \hat{f}_E(I)| \leq \tau \} \cap \{ c \in \mathbb{R}^{d+1} | p_c(x) \geq 0 \text{ for all } x \in J \}$$

$$= \bigcap_{I \in J} \{ c \in \mathbb{R}^{d+1} | \sum_{I \in I} |p_c(I) - \hat{f}_E(I)| \leq \tau \} \cap \bigcap_{x \in J} \{ c \in \mathbb{R}^{d+1} | p_c(x) \geq 0 \} .$$

In the last line, we used the notation $I = \{I_1, \ldots, I_k\}$. Since the intersection of a family of convex sets is convex, it remains to show that the individual $A_k$-distance sets and non-negativity sets are convex. Let

$$M = \bigcap_{I \in J} \bigcap_{\xi \in \{-1,1\}^k} \{ c \in \mathbb{R}^{d+1} | \sum_{i=1}^k \xi_i (p_c(I_i) - \hat{f}_E(I_i)) \leq \tau \}$$

$$\mathcal{N} = \bigcap_{x \in J} \{ c \in \mathbb{R}^{d+1} | p_c(x) \geq 0 \} .$$

We start with the non-negativity constraints encoding the set $\mathcal{N}$. For a fixed $x \in J$, we can expand the constraint $p_c(x) \geq 0$ as

$$\sum_{i=0}^d c_i \cdot x^i \geq 0 ,$$

which is clearly a linear constraint on the $c_i$. Hence, the set $\{ c \in \mathbb{R}^{d+1} | p_c(x) \geq 0 \}$ is a halfspace for a fixed $x$ and thus also convex.

Next, we consider the $A_k$-constraints $\sum_{i=1}^k \xi_i (p_c(I_i) - \hat{f}_E(I_i)) \leq \tau$ for the set $M$. Since the intervals $I_1, \ldots, I_k$ are now fixed, so is $\hat{f}_E(I_i)$. Let $\alpha_i$ and $\beta_i$ be the endpoints of the interval $I_i$, i.e., $I_i = [\alpha_i, \beta_i]$. Then we have

$$p_c(I_i) = \int_{\alpha_i}^{\beta_i} p_c(x) \, dx = P_c(\beta_i) - P_c(\alpha_i) ,$$

where $P_c(x)$ is the indefinite integral of $P_c(x)$, i.e.,

$$P_c(x) = \sum_{i=0}^d c_i \cdot \frac{x^{i+1}}{i+1} .$$

---

4 Norms give rise to convex sets and the set of non-negative polynomials is also convex.
So for a fixed \( x \), \( P_c(x) \) is a linear combination of the \( c_i \). Consequently \( \sum_{i=1}^{k} \xi_i p_c(I_i) \) is also a linear combination of the \( c_i \), and hence each set in the intersection defining \( \mathcal{M} \) is a halfspace. This shows that \( \mathcal{C}_\tau \) is a convex set.

It is worth noting that the set \( \mathcal{N} \) is a spectrahedron (the feasible set of a semidefinite program) because it encodes non-negativity of a univariate polynomial over a fixed interval. After restricting the set of coefficients to non-negative polynomials, we can simplify the definition of the \( A_k \)-distance: it suffices to consider sets of intervals with endpoints at the locations of samples (see Lemma 37). Hence, we can replace the supremum in the definition of \( \mathcal{M} \) by a maximum over a finite set, which shows that \( \mathcal{C}_\tau \) is also a spectrahedron. This suggests that the \( A_k \)-projection problem could be solved by a black-box application of an SDP solver. However, this would lead to a running time that is exponential in \( k \) because there are more than \( \binom{|E|}{2k} \) possible sets of intervals. While the authors of [CDSS14] introduce an encoding of the \( A_k \)-constraint with fewer linear inequalities, their approach increases the number of variables in the optimization problem to depend polynomially on \( \frac{1}{\epsilon} \), which leads to a super-linear running time.

Instead of using black-box SDP or LP solvers, we construct an algorithm that exploits additional structure in the \( A_k \)-projection problem. Most importantly, our algorithm separates the dimension of the desired degree-\( d \) polynomial from the number of samples (or equivalently, the error parameter \( \epsilon \)). This allows us to achieve a running time that is nearly-linear for a wide range of distributions. Interestingly, we can solve our SDP significantly faster than the LP which has been proposed in [CDSS14] for the same problem.

### 5.2 Separation oracles and approximately feasible polynomials

In order to work with the large number of \( A_k \)-constraints efficiently, we “hide” this complexity from the convex optimization procedure by providing access to the constraints only through a separation oracle. As we will see in Section 6, we can utilize the structure of the \( A_k \)-norm and implement such a separation oracle for the \( A_k \)-constraints in nearly-linear time. Before we give the details of our separation oracle, we first show how we can solve the \( A_k \)-projection problem assuming that we have such an oracle. We start by formally defining our notions of separation oracles.

**Definition 22 (Separation oracle).** A separation oracle \( \mathcal{O} \) for the convex set \( \mathcal{C}_\tau \) is a function that takes as input a coefficient vector \( c \in \mathbb{R}^{d+1} \) and returns one of the following two results:

1. “yes” if \( c \in \mathcal{C}_\tau \).
2. a separating hyperplane \( y \in \mathbb{R}^{d+1} \). The hyperplane \( y \) must satisfy \( y^T c' \leq y^T c \) for all \( c' \in \mathcal{C}_\tau \).

For general polynomials, it is not possible to perform basic operations such as root finding exactly, and hence we have to resort to approximate methods. This motivates the following definition of an approximate separation oracle. While an approximate separation oracle might accept a point \( c \) that is not in the set \( \mathcal{C}_\tau \), the point \( c \) is then guaranteed to be close to \( \mathcal{C}_\tau \).

**Definition 23** (Approximate separation oracle). A \( \mu \)-approximate separation oracle \( \mathcal{O} \) for the set \( \mathcal{C}_\tau = \mathcal{C}_{\tau,d,k,E} \) is a function that takes as input a coefficient vector \( c \in \mathbb{R}^{d+1} \) and returns one of the following two results, either “yes” or a hyperplane \( y \in \mathbb{R}^{d+1} \).

1. If \( \mathcal{O} \) returns “yes”, then \( \| p_c - \hat{f}_E \|_{A_k, J} \leq \tau + 2\mu \) and \( p_c(x) \geq -\mu \) for all \( x \in J \).
2. If \( \mathcal{O} \) returns a hyperplane, then \( y \) is a separating hyperplane; i.e. the hyperplane \( y \) must satisfy \( y^T c' \leq y^T c \) for all \( c' \in C_\tau \).

In the first case, we say that \( p_c \) is a \( \mu \)-approximately feasible polynomial.

Note that in our definition, hyperplanes must still be exact for the set \( C_\tau \). Although our membership test is only approximate, the exact hyperplanes allow us to use several existing separation oracle methods for convex optimization. We now formally prove that many existing methods still provide approximation guarantees when used with our approximate separation oracle.

**Definition 24** (Separation Oracle Method). A separation oracle method (SOM) is an algorithm with the following guarantee: let \( \mathcal{C} \) be a convex set that is contained in a ball of radius \( 2^L \). Moreover, let \( \mathcal{O} \) be a separation oracle for the set \( \mathcal{C} \). Then \( \text{SOM}(\mathcal{O}, L) \) returns one of the following two results:

1. a point \( x \in \mathcal{C} \).
2. “no” if \( \mathcal{C} \) does not contain a ball of radius \( 2^{-L} \).

We say that a SOM is canonical if it interacts with the separation oracle in the following way: the first time the separation oracle returns “yes” for the current query point \( x \), the SOM returns the point \( x \) as its final answer.

There are several algorithms satisfying this definition of a separation oracle method, e.g., the classical Ellipsoid method [Kha79] and Vaidya’s cutting plane method [Vai89]. Moreover, all of these algorithms also satisfy our notion of a canonical separation oracle method. We require this technical condition in order to prove that our approximate separation oracles suffice. In particular, by a straightforward simulation argument, we have the following:

**Theorem 25.** Let \( \mathcal{M} \) be a canonical separation oracle method, and let \( \mathcal{O} \) be a \((\lambda, \mu)\)-approximate separation oracle for the set \( \mathcal{C}_\tau = \mathcal{C}_{\tau,d,k,E} \). Moreover, let \( L \) be such that \( \mathcal{C}_\tau \) is contained in a ball of radius \( 2^L \). Then \( \mathcal{M}(\mathcal{O}, L) \) returns one of the following two results:

1. a coefficient vector \( c \in \mathbb{R}^{d+1} \) such that \( \| p_c - \hat{f}_E \|_{A_k,J} \leq \tau + \lambda \) and \( p_c(x) \geq -\mu \) for all \( x \in J \).
2. “no” if \( \mathcal{C} \) does not contain a ball of radius \( 2^{-L} \).

### 5.3 Bounds on the radii of enclosing and enclosed balls

In order to bound the running time of the separation oracle method, we now establish bounds on the ball radii used in Theorem 25.

**Upper bound** When we initialize the separation oracle method, we need a ball of radius \( 2^L \) that contains the set \( \mathcal{C}_\tau \).

For this, we require bounds of coefficients of polynomials which are bounded in norm. Bounds of this form were first established by Markov [Mar92].

**Lemma 26.** Let \( p_c \) be a degree-\( d \) polynomial with coefficients \( c \in \mathbb{R}^{d+1} \) such that \( p(x) \geq 0 \) for \( x \in [-1, 1] \) and \( \int_{-1}^1 p(x) \, dx \leq \alpha \), where \( \alpha > 0 \), for some \( 0 < k \leq d \). Then we have

\[
|c_i| \leq \alpha \cdot (d + 1)^2 \cdot (\sqrt{2} + 1)^d \quad \text{for all } i = 0, \ldots, d.
\]
This lemma is by no means novel, but for completeness, we include a proof of this lemma in Section B. Using this lemma, we obtain:

**Theorem 27** (Upper radius bound). Let \( \tau \leq 1 \) and let \( A \) be the \((d+1)\)-ball of radius \( r = 2^{L_u} \) where

\[
L_u = d \log(\sqrt{2} + 1) + \frac{3}{2} \log d + 1.
\]

Then \( C_{\tau,d,k,E} \subseteq A \).

**Proof.** Let \( c \in C_{\tau,d,k,E} \). From basic properties of the \( L_1 \)- and \( A_k \)-norms, we have

\[
\int_{-1}^{1} p_c \, dx = \|p_c\|_{1,J} = \|p_c\|_{A_k,J} \leq \|\hat{f}_E\|_{A_k,J} + \|p_c - \hat{f}_E\|_{A_k,J} \leq 1 + \tau \leq 2.
\]

Since \( p_c \) is also non-negative on \( J \), we can apply Lemma 26 and get

\[
|c_i| \leq 2 \cdot (d+1) \cdot (\sqrt{2} + 1)^d \quad \text{for all } i = 0, \ldots, d.
\]

Note that the above constraints define a hypercube \( B \) with side length \( s = 4 \cdot (d+1) \cdot (\sqrt{2} + 1)^d \). The ball \( A \) contains the hypercube \( B \) because \( r = \sqrt{d+1} \cdot s \) is the length of the longest diagonal of \( B \). This implies that \( C_{\tau,d,k,E} \subseteq B \subseteq A \). \( \square \)

**Lower bound** Separation oracle methods typically cannot directly certify that a convex set is empty. Instead, they reduce the volume of a set enclosing the feasible region until it reaches a certain threshold. We now establish a lower bound on volumes of sets \( C_{\tau+\eta} \) that are feasible by at least a margin \( \eta \) in the \( A_k \)-distance. If the separation oracle method cannot find a small ball in \( C_{\tau+\eta} \), we can conclude that achieving an \( A_k \)-distance of \( \tau \) is infeasible.

**Theorem 28** (Lower radius bound). Let \( \eta > 0 \) and let \( \tau \) be such that \( C_{\tau} = C_{\tau,d,k,E} \) is non-empty. Then \( C_{\tau+\eta} \) contains a ball of radius \( r = 2^{-L_\ell} \), where

\[
L_\ell = \log \frac{4(d+1)}{\eta}.
\]

**Proof.** Let \( c^* \) be the coefficients of a feasible polynomial, i.e., \( c^* \in C_{\tau} \). Moreover, let \( c \) be such that

\[
c_i = \begin{cases} 
c^*_i + \eta & \text{if } i = 0 \\
c^*_i & \text{otherwise} \end{cases}.
\]

Since \( p_{c^*} \) is non-negative on \( J \), we also have \( p_c(x) \geq \frac{\eta}{4} \) for all \( x \in J \). Moreover, it is easy to see that shifting the polynomial \( p_{c^*} \) by \( \eta \) changes the \( A_k \)-distance to \( \hat{f}_E \) by at most \( \frac{\eta}{2} \) because the interval \( J \) has length 2. Hence, \( \|p_c - \hat{f}_E\|_{A_k,J} \leq \tau + \frac{\eta}{2} \) and so \( c \in C_{\tau+\eta} \). We now show that we can perturb the coefficients of \( c \) slightly and still stay in the set of feasible polynomials \( C_{\tau+\eta} \).

Let \( \nu = \frac{\eta}{4(d+1)} \) and consider the hypercube

\[
B = \{ c' \in \mathbb{R}^{d+1} | c'_i \in [c_i - \nu, c_i + \nu] \text{ for all } i \}.
\]
Note that $B$ contains a ball of radius $\nu = 2^{-L\ell}$. First, we show that $p_{c'}(x) \geq 0$ for all $x \in J$ and $c' \in B$. We have

$$p_{c'}(x) = \sum_{i=0}^d c'_i x_i = \sum_{i=0}^d c_i x^i + \sum_{i=0}^d (c'_i - c_i) x^i \geq p_c(x) - \sum_{i=0}^d \nu |x^i| \geq \frac{\eta}{4} - (d+1) \cdot \nu > 0.$$ 

Next, we turn our attention to the $A_k$-distance constraint. In order to show that $p_{c'}$ also achieves a good $A_k$-distance, we bound the $L_1$-distance to $p_c$.

$$\|p_c(x) - p_{c'}(x)\|_{1,J} = \int_{-1}^{1} |p_c(x) - p_{c'}(x)| \, dx \leq \int_{-1}^{1} \sum_{i=0}^d \nu |x^i| \, dx \leq \int_{-1}^{1} (d+1) \nu \, dx = 2(d+1) \nu \leq \frac{\eta}{2}.$$ 

Therefore we get

$$\|p_{c'} - \hat{f}_E\|_{A_k,J} \leq \|p_{c'} - p_c\|_{A_k,J} + \|p_c - \hat{f}_E\|_{A_k,J} \leq \|p_{c'} - p_c\|_{1,J} + \tau + \frac{\eta}{2} \leq \tau + \eta.$$ 

This proves that $c' \in C_{\tau+\eta}$ and hence $B \subseteq C_{\tau+\eta}$.

### 5.4 Finding the best polynomial

We now tie the feasibility problem back to our original optimization problem of finding a non-negative polynomial with minimal $A_k$-distance. For this, we perform a binary search over the $A_k$-distance and choose our error parameters carefully in order to achieve the desired approximation guarantee. See Algorithm 3 for the corresponding pseudo code.

The main result for our $A_k$-oracle is the following:

**Theorem 29.** Let $\tau^*$ be the smallest $A_k$-distance to the empirical distribution $\hat{f}_E$ achievable with a non-negative degree-$m$ polynomial on the interval $J$, i.e., $\tau^* = \min_{h \in P^+_{m,J}} \|h - \hat{f}_E\|_{A_k,J}$. Moreover,
Algorithm 3 Finding polynomials with small $A_k$-distance.

1: function $\text{FindPolynomial}(d, k, E, \eta)$
2: \hspace{1em} $\triangleright$ Initial definitions
3: \hspace{2em} Let $\eta' = \frac{\eta}{d}$.
4: \hspace{2em} Let $L_u = d \log(\sqrt{2} + 1) + \frac{3}{2} \log d + 1$.
5: \hspace{2em} Let $L_{\ell} = \log \frac{4(d+1)}{\eta'}$.
6: \hspace{2em} Let $L = \max(L_u, L_{\ell})$.
7: \hspace{2em} Let $\mathcal{M}$ be a canonical separation oracle method.
8: \hspace{2em} Let $\mathcal{O}_\tau$ be a $\eta'$-approximate separation oracle for the set of $(\tau, d, k, E)$-feasible polynomials.
9: \hspace{2em} $\tau_{\ell} \leftarrow 0$
10: \hspace{2em} $\tau_u \leftarrow 1$
11: \hspace{2em} while $\tau_u - \tau_{\ell} \geq \eta'$ do
12: \hspace{3em} $\tau_m \leftarrow \frac{\tau_u + \tau_{\ell}}{2}$
13: \hspace{3em} $\tau_m' \leftarrow \tau_m + \eta'$
14: \hspace{3em} if $\mathcal{M}(\mathcal{O}_{\tau_m'}, L)$ returned a point then
15: \hspace{4em} $\tau_u \leftarrow \tau_m$
16: \hspace{3em} end if
17: \hspace{2em} $\tau_l \leftarrow \tau_m$  \hspace{1em} $\triangleright$ $C_{\tau_m}$ does not contain a ball of radius $2^{-L}$ and hence $C_{\tau_m}$ is empty.
18: \hspace{2em} end while
19: \hspace{2em} $c' \leftarrow \mathcal{M}(\mathcal{O}_{\tau_u + 4\eta'}, L)$ \hspace{1em} $\triangleright$ Find final coefficients.
20: \hspace{2em} $c_0 \leftarrow c_0' + \gamma$ and $c_i \leftarrow c_i'$ for $i \neq 0$ \hspace{1em} $\triangleright$ Ensure non-negativity.
21: \hspace{2em} return $c$
22: end function
let $\eta > 0$. Then \textsc{FindPolynomial} returns a coefficient vector $c \in \mathbb{R}^{m+1}$ such that $p_c(x) \geq 0$ for all $x \in J$ and $\|p_c - \hat{f}_E\|_{A_k,J} \leq \tau^* + \eta$.

\textbf{Proof.} We use the definitions in Algorithm 3. Note that $\tau^*$ is the smallest value for which $C_{\tau^*} = C_{\tau^*,d,k,E}$ is non-empty. First, we show that the binary search maintains the following invariants: $\tau_\ell \leq \tau^*$ and there exists a $(2\eta', \gamma)$-approximately $\tau_u$-feasible polynomial. This is clearly true at the beginning of the algorithm: (i) Trivially, $\tau^* \geq \tau_\ell$. (ii) For $c = (0,0,\ldots,0)^T$, we have $\|p_c - \hat{f}_E\|_{A_k,J} \leq 1 = \tau_u$ and $p_c(x) \geq 0$, so $p_c$ is $\tau_u$-feasible (and hence also approximately $\tau_u$-feasible).

Next, we consider the two cases in the while-loop:

1. If the separation oracle method returns a coefficient vector $c$ such that the polynomial $p_c$ is $(\eta', \gamma)$-approximately $\tau^*_m$-feasible, then $p_c$ is also $(2\eta', \gamma)$-approximately $\tau_m$-feasible because $\tau^*_m = \tau_m + \eta'$. Hence, the update of $\tau_u$ preserves the loop invariant.

2. If the separation oracle method returns that $C_{\tau^*_m}$ does not contain a ball of radius $2^{-L}$, then $\tau_m$ must be empty (by the contrapositive of Theorem 28). Hence, we have $\tau^* \geq \tau_m$ and the update of $\tau_{\ell}$ preserves the loop invariant.

We now analyze the final stage of \textsc{FindPolynomial} after the while-loop. First, we show that $C_{\tau_u + 3\eta'}$ is non-empty by identifying a point in the set. From the loop invariant, we know that there is a coefficient vector $v'$ such that $p_{v'}$ is a $(2\eta', \gamma)$-approximately $\tau_u$-feasible polynomial. Consider $v$ with $v_0 := v_0' + \gamma$ and $v_i := v_i'$ for $i \neq 0$. Then we have

$$\|p_v - p_{v'}\|_{1,J} = \int_{-1}^1 |p_v(x) - p_{v'}(x)| \, dx = \int_{-1}^1 \gamma \, dx = 2 \cdot \frac{\eta'}{2} = \eta'.$$

Hence, we also get

$$\|p_v - \hat{f}_E\|_{A_k,J} \leq \|p_v - p_{v'}\|_{A_k,J} + \|p_{v'} - \hat{f}_E\|_{A_k,J} \leq \|p_v - p_{v'}\|_{1,J} + \tau_u + 2\eta' \leq \tau_u + 3\eta'.$$

We used the triangle inequality in (a) and the fact that $p_{v'}$ is $(2\eta', \gamma)$-approximately $\tau_u$-feasible in (b). Moreover, we have $p_{v'}(x) \geq -\gamma$ for all $x \in J$ and thus $p_v(x) \geq 0$ for all $x \in J$. This shows that $C_{\tau_u + 3\eta'}$ is non-empty because $v \in C_{\tau_u + 3\eta'}$.

Finally, consider the last run of the separation oracle method in line 20 of Algorithm 3. Since $C_{\tau_u + 3\eta'}$ is non-empty, Theorem 28 shows that $C_{\tau_u + 4\eta'}$ contains a ball of radius $2^{-L}$. Hence, the separation oracle method must return a coefficient vector $c' \in \mathbb{R}^{d+1}$ such that $p_{c'}$ is $(\eta', \gamma)$-approximately $\tau_u + 4\eta'$-feasible. Using a similar argument as for $v$, we can make $p_{c'}$ non-negative while increasing its $A_k$-distance to $\hat{f}_E$ by only $\eta'$, i.e., we can show that $p_{c'}(x) \geq 0$ for all $x \in J$ and that

$$\|p_c - \hat{f}_E\|_{A_k,J} \leq \tau_u + 6\eta'.$$

Since $\tau_u - \tau_\ell \leq \eta'$ and $\tau_\ell \leq \tau^*$, we have $\tau_u \leq \tau^* + \eta'$. Therefore, $\tau_u + 6\eta' \leq \tau^* + 7\eta' = \tau^* + \eta$, which gives the desired bound on $\|p_c - \hat{f}_E\|_{A_k,J}$. \hfill \square

In order to state a concrete running time, we instantiate our algorithm \textsc{FindPolynomial} with Vaidya’s cutting plane method as the separation oracle method. In particular, Vaidya’s algorithm runs in time $O(TdL + d^{2+1}L)$ for a feasibility problem in dimension $d$ and ball radii bounds of $2^L$ and $2^{-L}$, respectively. $T$ is the cost of a single call to the separation oracle and $\omega$ is the matrix-multiplication constant. Then we get:
Theorem 30. Let $O$ be a $(\eta/14)$-approximate separation oracle that runs in time $T$. Then $\text{FindPolynomial}$ has time complexity $O((Td^2 + d^{\omega+2}) \log \frac{1}{\eta})$.

Proof. The running time of $\text{FindPolynomial}$ is dominated by the binary search. It is easy to see that the binary search performs $O(\log \frac{1}{\eta})$ iterations, in which the main operation is the call to the separation oracle method. Our bounds on the ball radii in Theorems 27 and 28 imply $L = O(d)$.

Combining this with the running time bound for Vaidya’s algorithm gives the time complexity stated in the theorem.

In section 6, we describe a $\mu$-approximate separation oracle that runs in time $\tilde{O}(dk \log \log \frac{1}{\mu} + s)$, where $s$ is the size of the support of the empirical on the interval $J$. Plugging this oracle directly into our algorithm $\text{FindPolynomial}$ gives a $\eta$-approximate $A_c$-projection oracle which runs in time $O((d^{-1})k \log \log \frac{1}{\eta} + \frac{sd^2 + d^{\omega+2}}{d^{\eta}})$. This algorithm is the algorithm promised in Theorem 18.

6 The separation oracle and the $A_c$-computation oracle

In this section we construct an efficient approximate separation oracle (see Definition 23) for $C_\tau$ over $J = [-1,1]$. We denote it by $\text{ApproxSepOracle}$. Let $A$ be the ball defined in section 5.3. We will show:

Theorem 31. For all $\mu > 0$, $\text{ApproxSepOracle}(c, \mu)$ is a $\mu$-approximate separation oracle for $C_\tau$ that runs in time $\tilde{O}(dk \log \log \frac{1}{\mu} + s)$, where $s$ the number of samples in $J$, assuming all queries are contained in the ball $A$.

Along the way we also develop an approximate $A_c$-computation oracle $\text{ComputeAk}$.  

6.1 Overview of $\text{ApproxSepOracle}$

$\text{ApproxSepOracle}$ consists of two parts, $\text{TestNonnegBounded}$ and $\text{AkSeparator}$. We show:

Lemma 32. For any $\tau \leq 2$, given a $c \in A$, the algorithm $\text{TestNonnegBounded}(c, \mu)$ runs in time $O(d \log^2 d (\log^2 d + \log \log 1/\mu))$ and outputs a separating hyperplane for $C_\tau$ or “yes”. Moreover, if there exists a point $x \in [-1,1]$ such that $p_c(x) < -\mu$, the output is always a separating hyperplane.

We show in the next section that whenever $c \notin C_\tau$ the output is “yes”.

Theorem 33. Given a $c \in A$ such that $p_c(x) \geq -\mu$ for all $x \in [-1,1]$, there is an algorithm $\text{AkSeparator}(c, \mu)$ which runs in time $O(s(k + \log s + \log d + \log \log d))$, and either outputs a separating hyperplane for $c$ from $C_\tau$ or returns “yes”. Moreover, if $\|p_c - \hat{f}_E\|_{A_c} > \tau + 2\mu$, the output is always a separating hyperplane.

$\text{ApproxSepOracle}$ given $\text{TestNonnegBounded}$ and $\text{AkSeparator}$ Given $\text{TestNonnegBounded}$ and $\text{AkSeparator}$, it is straightforward to design $\text{ApproxSepOracle}$.

We first run $\text{TestNonnegBounded}(c, \mu)$. If it outputs a separating hyperplane, we return the hyperplane. Otherwise, we run $\text{AkSeparator}(c, \mu)$, and again if it outputs a separating
hyperplane, we return it. If none of these happen, we return “yes”. Lemma 32 and Theorem 33 imply that APPROXSEPORACLE is correct and runs in the claimed time.

In the following sections, we prove Lemma 32 and Theorem 33. In Section 6.2 we describe TESTNONNEG and prove Lemma 32, and in Section 6.3 we describe AKSEPARATOR and prove Theorem 33.

6.2 Testing Non-negativity and Boundedness

Formally, the problem we solve here is the following testing problem:

**Definition 34** (Approximate non-negativity test). An approximate non-negativity tester is an algorithm satisfying the following guarantee. Given a polynomial \( p = \sum_{i=0}^{d} c_i x^i \) with \( \max_i |c_i| \leq \alpha \) and a parameter \( \mu > 0 \), return one of two results:

- a point \( x \in [-1, 1] \) at which \( p(x) < -\mu/2 \).
- “OK”.

Moreover, it must return the first if there exists a point \( x' \in [-1, 1] \) so that \( p(x') < -\mu \).

Building upon the classical polynomial root-finding results of [Pan01], we show:

**Theorem 35.** Consider \( p \) and \( \mu \) from Definition 34. Then there exists an algorithm \( \text{TESTNONNEG}(p, \mu) \) that is an approximate non-negativity tester and runs in time

\[
O(d \log^2 d \cdot (\log^2 d + \log \log \alpha + \log \log(1/\mu)))
\]

where \( \alpha \) is a bound on the coefficients of \( p \).

This theorem is proved in Section B.2.

We have a bound on \( c \), since we may assume that \( c \in A \), and so we can use this algorithm to efficiently test non-negativity as we require. Our algorithm TESTNONNEG simply runs TESTNONNEG for any \( c \in A \). If this returns ”yes”, then SEPARATIONTESTUNBounded outputs ”yes”. Otherwise, TESTNONNEG outputs a point \( x \in [-1, 1] \) such that \( p_c(x) \leq -\mu/2 \). It then outputs the hyperplane \( cy + \mu/2 \leq 0 \) where \( y = [1, x, x^2, \ldots, x^d] \).

**Proof of Lemma 32.** The correctness of this algorithm follows from the correctness of TESTNONNEG. We therefore only bound the running time. The worst-case runtime of this algorithm is exactly the runtime of TESTNONNEG for any \( c \in A \). Since we run TESTNONNEG only when \( \max_{i \leq d} |c_i| \leq 2(d + 1)^2(\sqrt{2} + 1)^d = 2^{O(d)}, \) the runtime of TESTNONNEG is

\[
O(d (\log^2 d (\log^2 d + \log \log 1/\mu))),
\]
as claimed. \( \square \)
6.3 An $\mathcal{A}_k$-computation oracle

We now consider the $\mathcal{A}_k$ distance computation between two functions, one of which is a polynomial and the other an empirical distribution. In this subsection, we describe an algorithm $\text{ComputeAK}$, and show:

**Theorem 36.** Given a polynomial $p$ such that $p(x) \geq -\mu$ for all $x \in [-1, 1]$ and an empirical distribution $\hat{f}$ supported on $s$ points, for any $k \leq d$, $\text{ComputeAK}(p, \hat{f}, k)$ runs in time $O(s(\log s + \log d + \log \log d))$, and compute $v$ such that $|v - \|p - \hat{f}\|_{\mathcal{A}_k}| \leq 2\mu$ and a set of intervals $I_1, \ldots, I_k$ so that

$$\sum_{i=1}^{k} |p(I_i) - \hat{f}(I_i)| = v.$$

Note that this theorem immediately implies Theorem 19.

**AkSeparator** given **ComputeAK**: Before describing $\text{ComputeAK}$, we show how to design $\text{AkSeparator}$ satisfying Theorem 33 given such a $\text{ComputeAK}$.

The algorithm $\text{AkSeparator}$ is as follows: we run $\text{ComputeAK}(p_c, \hat{f}, k)$, and let $v$ be its estimate for $\|p_c - \hat{f}\|_{\mathcal{A}_k}$, and let $I_1, \ldots, I_k$ be the intervals it produces. If $v \leq \tau$, we output “yes”.

Otherwise, suppose

$$v = \sum_{i=1}^{i} |p_c(I_i) - \hat{f}(I_i)| > \tau.$$

Note that if $\|p_c - \hat{f}\|_{\mathcal{A}_k} > \tau + 2\mu$, this is guaranteed to happen since $v$ differs from $\|p_c - \hat{f}\|_{\mathcal{A}_k}$ by at most $2\mu$. Let $\sigma_i = \text{sign}(p_c(I_i) - \hat{f}(I_i))$. Let $I_i = [a_i, b_i]$. Then

$$\sum_{i=1}^{k} |p_c(I_i) - \hat{f}(I_i)| = \sum_{i=1}^{k} \sigma_i \left( \int_{a_i}^{b_i} p(x) \, dx - \hat{f}(I_i) \right)$$

$$= \sum_{i=1}^{k} \sigma_i \left( \sum_{j=0}^{d} \frac{1}{j+1} \left( b_i^{j+1} - a_i^{j+1} \right) c_j - \hat{f}(I_i) \right),$$

and therefore,

$$\sum_{i=1}^{k} \sigma_i \sum_{j=0}^{d} \frac{1}{j+1} \left( b_i^{j+1} - a_i^{j+1} \right) c_j > \tau + \sum_{i=1}^{k} \sigma_i \hat{f}(I_i); \quad (11)$$

Note the left hand side is linear in $c$ when we fix $\sigma_i$, and this is the hyperplane $\text{AkSeparator}$ outputs, in this case.

**Proof of Theorem 33 given Theorem 36.** We first argue about the correctness of the algorithm. If $\|p_c - \hat{f}\|_{\mathcal{A}_k} \geq \tau + 2\mu$, then, $\text{ComputeAK}$ guarantees that

$$v = \sum_{i=1}^{i} |p_c(I_i) - \hat{f}(I_i)| > \tau.$$
Consider the hyperplane constructed in (11). For any \( c' \in C_{\tau} \)
\[
\sum_{i=1}^{k} \sigma_i \left( \sum_{j=0}^{d} \frac{1}{j+1} \left( b_{ij}^{j+1} - a_{ij}^{j+1} \right) c'_j - \hat{f}(I_i) \right) \leq \sum_{i=1}^{k} \left| \sum_{j=0}^{d} \frac{1}{j+1} \left( b_{ij}^{j+1} - a_{ij}^{j+1} \right) c'_j - \hat{f}(I_i) \right| 
\]
\[
= \sum_{j=0}^{d} \left| p_{c'}(I_j) - \hat{f}(I_i) \right| 
\]
\[
\leq \| p_{c'} - \hat{f} \|_{A_k} \leq \tau ,
\]
where the last inequality is from the definition of \( C_{\tau} \). Therefore this is indeed a separating hyperplane for \( c \) and \( C_{\tau} \). Moreover, given \( I_1, \ldots, I_k \) and \( v \), this separating hyperplane can be computed in time \( O(dk) \). Thus the entire algorithm runs in time
\[
O(d(k + \log^2 d + \log \log 1/\mu)) + s (\log s + \log d + \log \log d) = \tilde{O}(dk \log \log 1/\mu + s) ,
\]
as claimed.

6.3.1 A Reduction from Continuous to Discrete

We first show that the problem reduces to the following discrete problem. For a sequence of real numbers \( c_1, \ldots, c_r \) and an interval \( I = [a, b] \) in \([r]\), let \( w(I) = \sum_{a \leq i \leq b} c_i \). We show that our problem reduces to the problem \textsc{DiscreteAk}, defined below.

\textsc{DiscreteAk}: Given a sequence of \( r \) real numbers \( \{c_i\}_{i=1}^{r} \) and a number \( k \), find a set of \( k \) disjoint intervals \( I_1, \ldots, I_k' \) that maximizes
\[
\sum_{i=1}^{k} |w(I_i)| .
\]

We will denote the maximum value obtainable \( \|\{c_i\}\|_{A_k} \), i.e.
\[
\|\{c_i\}\|_{A_k} = \max_{\mathcal{I}} \sum_{I \in \mathcal{I}} |w(I)| ,
\]
where the \( \mathcal{I} \) is taken over all collections of \( k \) disjoint intervals.

We will show that it is possible to reduce the continuous problem of approximately computing the \( A_k \) distance between \( p \) and \( \hat{f} \) to solving (P) for a suitably chosen sequence of length \( O(d) \).

Suppose the empirical distribution \( \hat{f} \) is supported at \( s \) points \( a < x_1 \leq \ldots \leq x_s \leq b \) in this interval. Let \( \mathcal{X} \) be the support of \( \hat{f} \). Let \( p[\alpha, \beta] = \int_{\alpha}^{\beta} p(x) dx \). Consider the following sequences of length \( 2s + 1 \):
\[
E(i) = \begin{cases} 
1/n & \text{if } i \text{ is even}, \\
0 & \text{if } i \text{ is odd}.
\end{cases}, \quad P_{\text{disc}}(i) = \begin{cases} 
p[x_{2\ell}, x_{2\ell+1}] & \text{if } i = 2\ell + 1, \\
0 & \text{if } i \text{ is even}.
\end{cases}
\]
where for simplicity we let \( s_0 = a \) and \( s_{s+1} = b \). The two sequences are displayed in Table 2.

Then we have the following lemma:
We construct the empirical. Let $b_i$ to the endpoints of each interval $[a_i, b_i]$. If $a_i$ is even (i.e. if $P_{\text{disc}}(a_i) - E(i)$ has only a contribution from $-E(a_i)$), include the left endpoint of this interval from $J_i$, otherwise (i.e. if $P_{\text{disc}}(a_i) - E(i)$ has only a contribution from $P_{\text{disc}}(a_i)$), exclude it, and similarly for the right endpoint. Then, by observation, we have $P_{\text{disc}}(I_i) - E(I_i) = p(I_i) - \hat{f}(I_i)$, and thus this choice of $J_i$ satisfies (12), as claimed.

Now we show the other direction, i.e., that

$$\|p - \hat{f}\|_{A_k} \leq \|\{P_{\text{disc}} - E\}\|_{A_k} + 2\mu.$$  

Let $I_1, \ldots, I_k$ denote a set of disjoint intervals in $I$ achieving the maximum value on the LHS. It suffices to demonstrate a set of $k$ disjoint intervals $J_1, \ldots, J_k$ in $I$ satisfying

$$\sum_{i=1}^{k} |p(I_i) - \hat{f}(I_i)| \leq \sum_{i=1}^{k} |P_{\text{disc}}(I_i) - E(I_i)| + 2\mu.$$  

We first claim that we may assume that the endpoints of each $I_i$ are at a point in the support of the empirical. Let $a_i$ and $b_i$ be the left and right endpoints of $I_i$, respectively. Indeed, for all $i$, if $p(I_i) - \hat{f}(I_i) \geq 0$, take the interval $I'_i = (x_j, x_\ell)$ where $x_j$ is the largest point in $X$ so that $x_j \leq a_i$, and where $x_\ell$ is the smallest point in $X$ so that $x_\ell \geq b_i$. Then since $p \geq \mu$ on $[-1, 1]$ and the new

<table>
<thead>
<tr>
<th>$i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>$\cdots$</th>
<th>2s</th>
<th>2s + 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E(i)$</td>
<td>0</td>
<td>$\frac{1}{n}$</td>
<td>0</td>
<td>$\frac{1}{n}$</td>
<td>$\cdots$</td>
<td>$\frac{1}{n}$</td>
<td>0</td>
</tr>
<tr>
<td>$P_{\text{disc}}(i)$</td>
<td>$p[a, x_1]$</td>
<td>0</td>
<td>$p[x_1, x_2]$</td>
<td>0</td>
<td>$\cdots$</td>
<td>0</td>
<td>$p[x_s, b]$</td>
</tr>
</tbody>
</table>

Table 2: The sequences $E(i)$ and $P_{\text{disc}}(i)$.

Lemma 37. For any polynomial $p$ so that $p(x) \geq -\mu$ on $[-1, 1]$

$$\|p - \hat{f}\|_{A_k} \leq \|\{P_{\text{disc}} - E\}\|_{A_k} + 2\mu.$$  

Moreover, given $k$ intervals $I_1, \ldots, I_k$ maximizing $\|\{P_{\text{disc}} - E\}\|_{A_k}$, one can compute $k$ intervals $J_1, \ldots, J_k$ so that

$$\sum_{i=1}^{k} |p(J_i) - \hat{f}(J_i)| - \|\{P_{\text{disc}} - E\}\|_{A_k} \geq 2\mu$$

in time $O(k)$.

Proof. We first show that

$$\|p - \hat{f}\|_{A_k} \geq \|\{P_{\text{disc}} - E\}\|_{A_k}.$$  

Let $I_1, \ldots, I_k$ be a set of disjoint intervals in $[2 \text{ pdeg} + 1]$ achieving the maximum on the RHS. Then it suffices to demonstrate a set of $k$ disjoint intervals $J_1, \ldots, J_k$ in $I$ satisfying

$$\sum_{i=1}^{k} |p(J_i) - \hat{f}(J_i)| \geq \sum_{i=1}^{k} |P_{\text{disc}}(I_i) - E(I_i)|.$$  

(12)

We construct the $J_i$ as follows. Fix $i$, and let $I_i = [a_i, b_i]$. Define $J_i$ to be the interval from $a_i$ to $b_i$. If $a_i$ is even (i.e. if $P_{\text{disc}}(a_i) - E(i)$ has only a contribution from $-E(a_i)$), include the left endpoint of this interval from $J_i$, otherwise (i.e. if $P_{\text{disc}}(a_i) - E(i)$ has only a contribution from $P_{\text{disc}}(a_i)$), exclude it, and similarly for the right endpoint. Then, by observation, we have $P_{\text{disc}}(I_i) - E(I_i) = p(I_i) - \hat{f}(I_i)$, and thus this choice of $J_i$ satisfies (12), as claimed.
interval contains no additional points in the support of $\hat{f}$, we have $p(I'_i) - \hat{f}(I_i) \geq p(I_i) - \hat{f}(I_i) - \mu|I_i|$. Alternatively, if $p(I_i) - \hat{f}(I_i) < 0$, take the interval $I'_i = [x_j, x_{\ell}]$ where $x_j$ is the smallest point in $\mathcal{X}$ so that $x_j \geq a_i$ and $x_\ell$ is the largest point in $\mathcal{X}$ so that $x_\ell \leq b_i$. By the same reasoning as before we have that $p(I'_i) - \hat{f}(I_i) \leq p(I_i) - \hat{f}(I_i) + \mu|I_i|$, and thus $|p(I'_i) - \hat{f}(I_i)| + \mu|I_i| \geq |p(I_i) - \hat{f}(I_i)|$. Thus

$$\sum_{i=1}^{k} |p(I_i) - \hat{f}(I_i)| \leq \sum_{i=1}^{k} \left( |p(I'_i) - \hat{f}(I'_i)| + \mu|I'_i| \right) \leq \sum_{i=1}^{k} |p(I'_i) - \hat{f}(I'_i)| + 2\mu,$$

since $\sum_{i=1}^{k} |I_i| \leq 2$ as the $I_i$ are disjoint subintervals in $[-1, 1]$.

Now it is straightforward to define the $J_i$. Namely, for each $I_i$ with endpoints $x_{i_1} \leq x_{i_2}$ so that $x_{i_1}, x_{i_2} \in \mathcal{X}$, define

$$J_i = \begin{cases} [i_1, i_2] & \text{if } x_{i_1}, x_{i_2} \in \mathcal{X}; \\ [i_1 + 1, i_2] & \text{if } x_{i_1} \not\in \mathcal{X} \text{ and } x_{i_2} \in \mathcal{X}; \\ [i_1, i_2 - 1] & \text{if } x_{i_1} \in \mathcal{X} \text{ and } x_{i_2} \not\in \mathcal{X}; \\ [i_1 + 1, i_2 - 1] & \text{if } x_{i_1}, x_{i_2} \not\in \mathcal{X}. \end{cases}$$

One can check that with this definition of the $J_i$, we have $p(I_i) - \hat{f}(I_i) = P_{\text{disc}}(J_i) - E(J_i)$; moreover, all the $J_i$ are discrete and thus this choice of $J_i$ satisfies (6.3.1).

Moreover, the transformation claimed in the lemma is the transformation provided in the first part of the argument. It is clear that this transformation is computable in a single pass through the intervals $I_1, \ldots, I_k$. This completes the proof. □

### 6.3.2 Description of ComputeDiscreteAk

Thus for the rest of this section we focus on solving DiscreteAk. A very similar problem was considered in [Csu04] who showed an algorithm for the problem of computing the set of $k$ disjoint intervals $I_1, \ldots, I_k$ maximizing

$$\sum_{i=1}^{k} w(I_i)$$

which runs in time $O(d \cdot \min\{\log d, k\})$ time. We require a modified version of his algorithm which we include for completeness. We call our variant ComputeDiscreteAk.

Here is an informal description of ComputeDiscreteAk. First, we may assume the original sequence is alternating in sign, as otherwise we may merge two consecutive numbers without consequence. We start with the set of intervals $I_0 = I_{0,1} \leq \ldots \leq I_{0,r}$, where $I_{0,i} = [c_i, c_i]$ contains only the point $c_i$. We first compute $J_0$ and $m_0$, where $J_0$ is the set of $k$ intervals $I$ in $I_0$ with largest $|w(I)|$, and $m_0 = \sum_{I \in J_0} |w(I)|$. Iteratively, after constructing $I_i = \{I_{i,1}, \ldots, I_{i,r}\}$, we construct $I_{i+1}$ by finding the set $I_{i,j}$ with minimal $|w(I_{i,j})|$ amongst all intervals in $I_i$, and merging it with both of its neighbors (if it is the first or last interval and so only has one neighbor, instead discard it), that is,

$$I_{i+1} = \{I_{i,1}, \ldots, I_{i,j-2}, I_{i,j-1} \cup I_{i,j} \cup I_{i,j+1}, I_{i,j+2}, \ldots, I_{i,r}\}.$$

We then compute $J_{i+1}$ and $m_{i+1}$ where $J_{i+1}$ is the set of $k$ intervals $I$ in $I_{i+1}$ with largest $|w(I)|$, and $m_{i+1} = \sum_{I \in J_{i+1}} |w(I)|$. To perform these operations efficiently, we store the weights of the intervals we create in priority queues. We repeat this process until the collection of intervals $I_\ell$
has \( \leq k \) intervals. We output \( J_i \) and \( w_i \), where \( w_i \) is the largest amongst all \( w_i' \) computed in any iteration. An example of an iteration of the algorithm is given in Figure 1, and the formal definition of the algorithm is in Algorithm 4.

![Figure 1: An iteration of COMPUTE_DISCRETE_AK. The numbers denote the weight of each interval. The interval with smallest weight (in absolute value) is chosen and merged with adjacent intervals. Note that if weights are of alternating signs at the start, then they are of alternating signs at each iteration.](image)

The following runtime bound is not too hard to show:

**Theorem 38.** Given \( \{c_i\}_{i=1}^r \), COMPUTE_DISCRETE_AK\((\{c_i\}_{i=1}^r, k)\) runs in time \( O(r \cdot \min\{\log r, k\}) \).

The nontrivial part of the analysis is correctness.

**Theorem 39.** Given \( \{c_i\}_{i=1}^r \) and \( k \), the set of intervals returned by the algorithm COMPUTE_DISCRETE_AK\((\{c_i\}_{i=1}^r, k)\) solves the problem DISCRETE_AK.

**Proof.** Our analysis follows the analysis in [Csu04]. We call any \( I \) which attains the maximum for the DISCRETE_AK problem a **maximal subset**, or **maximal** for short. For any two collections of disjoint intervals \( I', I'' \) in \( [r] \), we say that \( I' \) is contained in \( I'' \) if all the boundary points of intervals in \( I' \) are also boundary points of intervals in \( I'' \). Figure 2 shows an example of two collections of intervals, one contained in the other. If there is a maximal \( I^* \) that is contained in \( I \) we say that \( I \) contains a maximal subset. We say that \( I' \) is **atomic** with respect to \( I'' \) if every interval in \( I' \) is also in \( I'' \). Figure 3 gives an example of two collections of intervals, one atomic with respect to the other. If there is a maximal \( I^* \) that is atomic with respect to \( I \) then we say that the maximum is atomic with respect to \( I \).

![Figure 2: \( I' \) is contained in \( I'' \) since each boundary point of all intervals in \( I' \) are boundary points of some interval in \( I'' \)](image)

We will prove the following invariant of our algorithm:

**Lemma 40.** For any \( i \geq 0 \), if \( I_i \) contains a maximal subset, then either the maximum is atomic with respect to \( I_i \) or \( I_{i+1} \) contains a maximal subset.

Before we prove this lemma, let us see how it suffices to prove Theorem 39. Clearly \( I_0 \) contains a maximal subset. By induction and Lemma 40, for all \( i \), as long as the maximum is not atomic with
Algorithm 4 Computing the discrete $A_k$ norm of a sequence.

1: function COMPUTE_DISCRETE_AK({c_i}^r_{i=1}, k)
2:     Let $I_0 \leftarrow \{[c_1, c_1], [c_2, c_2], \ldots, [c_r, c_r]\}$ be the initial set of intervals.
3:     Let $Q$ be an empty priority queue.
4:     for $I \in I_0$ do
5:         $Q.push(I, w(I))$
6:     end for
7:     $i \leftarrow 0$
8:     while $|I_i| > k$ do
9:         Let $I \leftarrow Q.deleteMin()$.
10:        if $I$ is not the leftmost or rightmost interval then
11:            Let $I_{left}$ and $I_{right}$ be its left and right neighbors, respectively.
12:            $Q.remove(I_{left})$
13:            $Q.remove(I_{right})$
14:            Let $I' = I_{left} \cup I \cup I_{right}$
15:            $Q.push(I', w(I'))$
16:        end if
17:        $i \leftarrow i + 1$
18:     end while
19:     return $w_j$ and $J_j$ where $w_j$ satisfies $w_j \geq w_i$ for all $i$.
20: end function

respect to $I_i$, $I_{i+1}$ contains a maximal subset. COMPUTE_DISCRETE_AK stops iterating at iteration $i_f$ if $I_{i_f}$ has at most $k$ intervals. At this point either the maximum was atomic with respect to some $I_i$, or $I_{i_f}$ contains a maximal subset. Let $I^*$ be any maximal subset it contains. We observe that

$$\sum_{I \in I^*} |w(I)| \leq \sum_{I \in I_{i_f}} |w(I)|,$$

and moreover, $I_{i_f}$ has $k$ pieces, so $I_{i_f}$ is itself maximal, and is clearly atomic with respect to itself.

Thus there is some $i$ so that $I_i$ contains a maximal subset that is atomic with respect to $I_i$. Call this maximal subset $J_i$. But then since it is atomic with respect to $I_i$, we have that

$$\sum_{I \in J_i} |w(I)| \leq \sum_{I \in J} |w(I)| = m_i,$$

since $J_i$ is chosen to maximize the sum over all sets of $k$ intervals which are atomic with respect to $I_i$. Since $I^*$ achieves the maximum for COMPUTE_DISCRETE_AK, we conclude that $m_i$ is indeed the maximum. Thus whatever $m_f$ we output is also the maximum, and its $J_f$ attains the maximum. This completes the proof of Theorem 39 assuming Lemma 40. \(\square\)

We now prove Lemma 40.
Proof of Lemma 40. It suffices to show that if $\mathcal{I}_i$ contains a maximal subset, but the maximum is not atomic with respect to $\mathcal{I}_i$, then $\mathcal{I}_{i+1}$ also contains a maximal subset. Thus, let $\mathcal{I}^*$ be such that

1. $\mathcal{I}^*$ is maximal
2. $\mathcal{I}^*$ is contained in $\mathcal{I}_i$, and
3. there is no $\mathcal{I}^*_1 \neq \mathcal{I}^*$ satisfying conditions (1) and (2) so that every interval in $\mathcal{I}^*_1$ is contained in some interval in $\mathcal{I}^*$.

Such a $\mathcal{I}^*$ clearly exists by the assumption on $\mathcal{I}_i$. Note that $\mathcal{I}^*$ cannot be atomic with respect to $\mathcal{I}_j$. By observation we may assume that no interval $I'$ in an maximal subset will ever end on intervals $I''$ whose $w(I'')$ have a different sign than $w(I')$, since otherwise we can easily modify the partition to have this property while still maintaining properties (1)-(3).

Let $I_j = [\beta_2, \beta_3]$ be the interval in $\mathcal{I}_i$ with minimal $|w(I)|$ amongst all $I \in \mathcal{I}_i$. WLOG assume that it is not the leftmost or rightmost interval (the analysis for these cases is almost identical and so we omit it). Let $\beta_1$ be the left endpoint of $I_{j-1}$ and $\beta_4$ be the right endpoint of $I_{j+1}$. WLOG assume that $w(I_j) < 0$.

The initial partition $\mathcal{I}_0$ had the property that the signs of the values $w(I)$ for $I \in \mathcal{I}_0$ alternated, and through a simple inductive argument it is not hard to show that for all $\mathcal{I}_i$, the signs of the values $w(I)$ for $I \in \mathcal{I}_i$ still alternate. Thus we have $w(I_{j-1}), w(I_{j+1}) \geq 0$. Since $\mathcal{I}^*$ is not atomic, there is some $I_a \in \mathcal{I}^*$ which contains at least two intervals $I_1, I_2$ of $\mathcal{I}_j$. Moreover, since the signs of the $w(I)$ of the intervals in $\mathcal{I}_j$ alternate, we may assume that $w(I_1)$ and $w(I_2)$ have different signs. Thus, by the observation, we may in fact assume that $I_a$ contains three consecutive intervals $I_1 < I_2 < I_3$, and that $w(I_1), w(I_2), w(I_3)$ have the same sign, and $w(I_2)$ has a different sign. Moreover, $|w(I_j)| \leq \min\{w(I_1), w(I_2), w(I_3)\}$. Moreover, define $I^*_a$ to be the interval which shares a left endpoint with $I_a$, and which has right endpoint the right endpoint of $I_1$, and $I^*_a$ to be the interval which shares a right endpoint with $I_a$, and which has left endpoint the left endpoint of $I_3$. We must have that $w(I^*_a), w(I^*_a)$ are the same sign as $w(I_a)$ as otherwise, say if $w(I^*_a)$’s sign was different from $w(I_a)$’s sign, we would have $|w(I_0)| \leq |w(I^*_a)|$ and so the existence of the collection of intervals $\mathcal{I}' = (\mathcal{I}^* \setminus I_a) \cup I^*_a \cup I^*_a$ violates condition (3), since it is clearly contained in $\mathcal{I}_i$, and

$$\sum_{I \in \mathcal{I}'} |w(I)| = \sum_{I \in \mathcal{I}^*} |w(I)| - |w(I_a)| + |w(I^*_a)| \geq \sum_{I \in \mathcal{I}^*} |w(I)|$$

so it is maximal.

Since $\mathcal{I}^*$ is contained in $\mathcal{I}_i$, the only boundary points that intervals in $\mathcal{I}_i$ can have in the interval $[\beta_1, \beta_4]$ are at the points $\beta_i$ for $i \in \{1, 2, 3, 4\}$. There are a few cases.

Case 1. If no interval in $\mathcal{I}^*$ has any boundary point at $\beta_2$ or $\beta_3$, then it is still contained in $\mathcal{I}_{i+1}$, by the definition of $\mathcal{I}_{i+1}$.
Case 2  If \( [\beta_2, \beta_3] \in \mathcal{I}^* \), define \( \mathcal{I}' = (\mathcal{I}^* \setminus \{[\beta_2, \beta_3], I_a\}) \cup \{I_a^1, I_a^2\} \). Then

\[
\sum_{I \in \mathcal{I}'} |w(I)| = \sum_{I \in \mathcal{I}^*} |w(I)| - |w(I_j)| + |w(I_2)| \geq \sum_{I \in \mathcal{I}^*} |w(I)|
\]

by the choice of \( I_j \), so \( \mathcal{I}' \) is maximal, contained in \( \mathcal{I}_i \), and thus its existence violates condition (3), so this case is impossible. This is illustrated in Figure 4, where for simplicity \( I_a \) contains precisely three intervals.

Figure 4: When \( I_j \) is an element of \( \mathcal{I}^* \), we can drop it and add the intervals \( I_a^1 \) and \( I_a^2 \) achieving a larger weight.

Case 3  If \( \beta_3 \) is the right endpoint of some interval \( I \in \mathcal{I}^* \), then by the same reasoning as before, we may assume that \( w(I) < 0 \). Then, let \( I' \) be the interval with the same left endpoint as \( I \) but with right endpoint \( \beta_1 \). Since then \( w(I') = w(I) - w(I_{j-1}) - w(I_j) \leq w(I) \), the partition \( \mathcal{I}' = \mathcal{I}^* \setminus I \cup I' \) is maximal, contained in \( \mathcal{I}_i \), and its existence again violates condition (3), so this case is impossible. An illustration is given in Figure 5.

Figure 5: \( \mathcal{I}^* \) can drop \( I \) and instead take \( I' \) to get a larger weight.

Case 4  If \( \beta_2 \) is the left endpoint of some interval \( I \in \mathcal{I}^* \), then symmetric reasoning to that in Case 2 results in a contradiction, so this case is also impossible.

Case 5a  If \( \beta_2 \) is the right endpoint of some interval \( I \in \mathcal{I}^* \), and no interval in \( \mathcal{I}^* \) contains \( I_{j+1} \), then we know that \( w(I) \geq 0 \). Let \( I' \) be the interval \( I \cup I_j \cup I_{j+1} \). Then, the partition \( \mathcal{I}' = \mathcal{I}^* \setminus I \cup I' \) is maximal by the same kind of reasoning as before, and \( \mathcal{I}' \) is contained in \( \mathcal{I}_{j+1} \). Thus this case is possible and consistent with the Lemma.

Case 5b  If \( \beta_2 \) is the right endpoint of some interval \( I \in \mathcal{I}^* \) and \( \beta_3 \) is the left endpoint of some interval \( I' \in \mathcal{I}^* \), then we know that \( w(I), w(I') \geq 0 \). Let \( I'' = I \cup I_j \cup I' \). Define \( \mathcal{I}' = \mathcal{I}^* \setminus \{I_a^1, I_a^2\} \)...
(\mathcal{I} \setminus \{I, I', I_0\}) \cup \{I'', I_1, I_3\}. Then \(\sum_{I \in \mathcal{I}} |w(I)| = \sum_{I \in \mathcal{I}^*} |w(I)| - |w(I_j)| + |w(I_2)| \geq \sum_{I \in \mathcal{I}^*} |w(I)|\), so again this is a maximal subset which is now contained in \(\mathcal{I}_{j+1}\).

These cases encompass all possible cases, and thus we conclude that the Lemma holds, as claimed. \(\square\)

### 6.3.3 Description of Compute\(\text{Ak}\)

Our algorithm Compute\(\text{Ak}\) simply uses Fact 41, produces the sequence \(P_{\text{disc}}(i) - E(i)\), and computes \(\|\{P_{\text{disc}} - E\}\|_{\mathcal{A}_k}\) using Compute\(\text{DiscreteAk}\).

It thus suffices to show that we can construct this sequence \(P_{\text{disc}}(i) - E(i)\) quickly when we are given the empirical distribution and the polynomial \(p\). The only difficulty lies in quickly computing the \(p[x_i, x_{i+1}]\). This problem is equivalent to efficiently evaluating the integral of \(p\) at all the points in \(\mathcal{X}\), which is in turn equivalent to efficiently evaluating a degree \(d + 1\) polynomial at \(s\) points.

To do so, we use the following, well-known fact:

**Fact 41** ([VZGG13], p. 288). Let \(x_1, \ldots, x_d\) be a set of \(d\) real numbers and let \(p\) be a polynomial of degree at most \(d\). Then there is an algorithm that computes \(p(x_1), \ldots, p(x_d)\) in time \(O(d \log d \log \log d)\).

The running time stated above is not the best known: multipoint evaluation is possible in time \(O(M(d) \log d)\), where \(M(d)\) is the time needed to multiply two degree-\(d\) polynomials. Recent results show that \(M(d) = O(d \log d \cdot 2^{O(\log^* d)})\) [Für09] and even \(M(d) = O(d \log d \cdot 8^{\log^* d})\) [HVDHL14]. To keep our running time simple, we use the time bound stated in Fact 41 with a single \(\log \log d\) factor, which is based on the algorithm of Schönhage and Strassen [SS71].

After having computed the discretized version of the problem, the algorithm then outputs the estimate that Compute\(\text{DiscreteAk}\) computed, and the processed version of the intervals, where the processing is the one promised in Lemma 37. Thus, we have:

**Proof of Theorem 36.** The correctness of the algorithm follows from Lemma 37 and the arguments given above. Thus it suffices to bound the runtime. The runtime of producing the sequence \(P_{\text{disc}}(i) - E(i)\) is bounded by computing the \(p[x_i, x_{i+1}]\), which can be done by the above in time \(O(s(\log d + \log \log d))\), and the runtime of Compute\(\text{DiscreteAk}\) on the sequence \(P_{\text{disc}}(i) - E(i)\) is \(O(s \log s)\). Hence the runtime of the entire algorithm is \(O(s(\log s + \log d + \log \log d))\), as claimed. \(\square\)

### 7 Applications

In this section we apply our main result to obtain near optimal estimators for various classes of structured distributions. As described in Table 1, we consider arbitrary mixtures of well-studied distribution families, including log-concave distributions, normal distributions, densities with bounded number of modes, and density functions in Besov spaces. We also consider mixtures of discrete structured distributions over an ordered domain, such as multi-modal distributions, monotone hazard rate distributions, Poisson and Binomial distributions. For all these classes, our sample complexity and running time matches the information-theoretic optimum, up to at most logarithmic factors.

We note that even though our algorithm is stated for distributions over a known finite interval, they are also applicable to distributions over the entire real line, such as (mixtures of) Gaussians or Poisson distributions. This follows from the fact that if \(x_{\min}\) and \(x_{\max}\) are the smallest and largest elements among \(\frac{\log(1/\delta)}{\epsilon}\) draws from any distribution, then with probability at least \(1 - \delta\), the distribution assigns probability mass at least \(1 - \epsilon\) to the interval \([x_{\min}, x_{\max}]\). Thus, at a cost
of $\frac{\log(1/\delta)}{\epsilon}$ samples, we may truncate the distribution and thereafter only consider this finite interval and rescale to $[-1, 1]$.

### 7.1 Mixture of log-concave distributions

For an interval $I \subseteq \mathbb{R}$, a function $g : I \to \mathbb{R}$ is called concave if for any $x, y \in I$ and $\lambda \in [0, 1]$ it holds $g(\lambda x + (1 - \lambda)y) \geq \lambda g(x) + (1 - \lambda)g(y)$. A function $h : I \to \mathbb{R}_+$ is called log-concave if $h(x) = \exp(g(x))$, where $g : I \to \mathbb{R}$ is concave. A density $f$ is a $k$-mixture of log-concave density functions if there exist $w_1, \ldots, w_k \geq 0$, $\sum_i w_i = 1$ and log-concave density functions $f_1, \ldots, f_k$ such that $f = \sum w_if_i$. The class of log concave distributions is very broad and contains the class of Gaussians, uniform, exponential, Gamma, Beta, and Weibull distributions. Log-concave distributions have received significant interest in economics and statistics [BB05, CSS10, DR09].

It was shown in [CDSS14a] that a $k$-mixture of log-concave density functions can be $\epsilon$-approximated in $L_1$ norm by a $t$-piecewise linear density, for $t = \tilde{O}(k/\sqrt{\epsilon})$. Using this structural result, [CDSS14a] gave an $m = \tilde{O}(t^2/\epsilon^2) = \tilde{O}(k/\epsilon^{5/2})$ sample and polynomial time algorithm to agnostically learn a $k$-mixture of log-concave distributions. This sample bound is nearly optimal, as $\Omega(k/\epsilon^{5/2})$ samples are necessary for this learning problem.

Our main result yields a sample optimal and near-linear time algorithm for this problem. In particular, this follows as a combination of Theorem 1, and a recently obtained tight structural result that removes the logarithmic factors from the previous construction of [CDSS14a]. In particular, it is shown in [DK15] that a $k$-mixture of log-concave density functions can be $\epsilon$-approximated in $L_1$ norm by a $t$-piecewise linear density, for $t = O(k/\sqrt{\epsilon})$. As a corollary, we obtain the following:

**Theorem 42.** There is an agnostic learning algorithm for the class of $k$-mixtures of log-concave distributions over the real line that uses $O(k/\epsilon^{5/2})$ samples and runs in time $O((k/\epsilon^{5/2}) \cdot \log(1/\epsilon))$.

### 7.2 Mixture of Gaussians

Let $N(\mu, \sigma^2)$ denote the normal distribution with mean $\mu$ and variance $\sigma^2$. A density $f : \mathbb{R} \to \mathbb{R}_+$ is a $k$-mixture of Gaussians if there exist $w_1, \ldots, w_k \geq 0$, $\sum_i w_i = 1$, $\mu_1, \ldots, \mu_k \in \mathbb{R}$ and $\sigma_1, \ldots, \sigma_k \in \mathbb{R}_+$ such that $f = \sum_{i=1}^k w_i N(\mu_i, \sigma_i^2)$.

The problem of parameter estimation for Gaussian mixtures was initiated in the TCS community by [Das99]. Recent work has obtained polynomial sample and time algorithms for this problem under the conditions of identifiability [MV10, BS10]. We remark that learning the parameters of a mixture of $k$ univariate Gaussians to accuracy $\epsilon$ requires $\Omega((1/\epsilon)^{6k-2})$ samples [MV10, HP15].

The problem of proper learning for Gaussian mixtures has also been recently studied in [DK14, SOAJ14] who obtain algorithms that draw $\tilde{O}(k/\epsilon^2)$ samples and run in time $O((1/\epsilon)^{3k-1})$. Another approach, due to [BSZ15], outputs a mixture of $O(k/\epsilon^3)$ Gaussians in time and sample complexity of $O(k/\epsilon^6)$.

It is well-known (see, e.g., [Tim63, Section 7.21] or [CDSS14a]) that a normal distribution is $\epsilon$-close to a $3$-piecewise polynomial of degree $O(\log(1/\epsilon))$. Using this structural result, [CDSS14a] obtain a nearly sample optimal and polynomial time agnostic learning algorithm for this problem.

As a corollary of Theorem 1, we obtain a nearly sample optimal and nearly linear time algorithm: (The sample complexity of our algorithm is better than that of [CDSS14a] by logarithmic factors.) In particular:
Theorem 43. There is an agnostic learning algorithm for $k$-mixtures of univariate Gaussians that draws $O((k/\epsilon^2) \log(1/\epsilon))$ samples and runs in time $\tilde{O}(k/\epsilon^2)$.

7.3 Densities in Besov spaces

Densities in Besov spaces constitute a broad family of distributions, including piecewise polynomials, and the exponential family. Density estimation for functions in Besov spaces has received considerable attention in the statistics and information theory literature. A lot of the early work on the topic relied on wavelet techniques, based on the fact that functions in Besov spaces are amenable to “multiscale” decompositions [DeV98, DJKP96, DJ98].

Any piecewise smooth density function $f$ in has the following decomposition,

$$f(x) = \sum_k c_{j_0,k} \phi_{j_0,k}(x) + \sum_{j=j_0}^{\infty} \sum_k d_{j_0,k} \psi_{j_0,k}(x)$$

where the $\phi$’s are scaling and the $\psi$’s are wavelet functions. The Besov space $B^\alpha_q(L_p([0,1]))$ is the following subset of such density functions

$$B^\alpha_q(L_p([0,1])) \overset{\text{def}}{=} \left\{ f : \|c_{j_0,k}\|_{\ell_p} + \left( \sum_{j=j_0}^{\infty} \left( 2^{\alpha j p} \sum_k |d_{j,k}|^p \right)^{q/p} \right)^{1/q} \right\},$$

for parameters $\alpha > \frac{1}{p} > 0$, and $q > 0$, where $\{c_{j_0,k}\}$ and $\{d_{j,k}\}$ are the scaling and wavelet coefficients in the wavelet expansion of $f$.

Nowak and Willett [WN07] showed that any density $f$ in $B^\alpha_q(L_p([0,1]))$ for $0 < q \leq p$, with $\frac{1}{p} = \alpha + \frac{1}{2}$ can be approximated up to $L_1$ error $\epsilon$ with $n = O\left( \frac{\log^2(1/\epsilon)}{\epsilon^{\alpha+1/2}} \right)$ samples. They also propose an algorithm for this problem with running time $\Omega(n^3)$.

As a corollary of our main result, we obtain a sample optimal and near-linear time agnostic algorithm for this problem. A result in [DeV98] implies that under the above assumptions on $\alpha, p, q$, any function in $B^\alpha_q(L_p([0,1]))$ can be $\epsilon$-approximated in $L_1$ norm by an $O(\epsilon^{-1/\alpha})$-piece degree-$O(\lceil \alpha \rceil)$ polynomial. Combined with our main result, we obtain an algorithm with sample complexity by a $O\left( \frac{1}{\epsilon^{2+1/\alpha}} \right)$, which is optimal up to constant factors [WN07]. Moreover, the running time of our algorithm is nearly linear in the number of samples. In particular:

Theorem 44. There is an agnostic learning algorithm for $B^\alpha_q(L_p([0,1]))$, with $0 < q < p$, $1/p = \alpha + 1/2$ with sample complexity $O\left( \frac{1}{\epsilon^{2+1/\alpha}} \right)$ and running time $\tilde{O}\left( \frac{1}{\epsilon^{2+1/\alpha}} \right)$.

7.4 Mixtures of $t$-monotone distributions

A density $f : \mathbb{R} \to \mathbb{R}_+$ is 1-monotone if it is non-increasing. It is 2-monotone if it is non-increasing and convex, and $k$-monotone for $t \geq 3$, if $(-1)^j f^{(j)}$ is non-negative, non-increasing and convex for $j = 0, \ldots, t-2$. A number of recent works in statistics studied the problem of estimating $t$-monotone density functions in the context of the MLE [BW07, GW09, BW10].

Implicit in [KL04, KL07] is the fact that any $t$-monotone bounded density function over $[0,1]$ can be approximated with an $O(1/\epsilon^{1/t})$ piecewise degree $t-1$ polynomial. Using this along with our main result yields the following guarantee on learning $t$-monotone distributions.
Theorem 45. There exists an agnostic learning algorithm for $k$-mixtures of $t$-monotone distributions that uses $O(tk/\epsilon^{2+1/t})$ samples and runs in time $\tilde{O}(kt^{2+\omega}/\epsilon^{2+1/t})$.

The above is a significant improvement in the running time compared to [CDSS14a]. Note that for $t=1, 2$, the sample complexity of our algorithm is provably optimal. This follows from known lower bounds of $\Omega(1/\epsilon^3)$ for $t=1$ [Bir87a] and of $\Omega(1/\epsilon^{5/2})$ for $t=2$ [DL01].

7.5 Mixtures of discrete distributions

Our main results applies to the discrete setting as well leading to fast algorithms for learning mixtures of discrete distributions, that can be well-approximated by piecewise polynomials.

Mixtures of $t$-modal discrete distributions and MHR distributions. A distribution over $[N]$ is unimodal if there is a $j \in [N]$ such that the pmf is non-decreasing up to $j$, and non-increasing after $j$. A distribution is $t$-modal if there is a partition of $[N]$ into at most $t$ intervals over which the conditional pmf is unimodal. It follows from [Bir87b, CDSS13] that any mixture of $k$ $t$-modal distributions is $\epsilon$-close to a $(kt/\epsilon)\log(N/kt)$-histogram. [CDSS14b] implies an algorithm for this problem that uses $n = \tilde{O}(kt\log(N)/\epsilon^3)$ samples and runs in time $\tilde{O}(n)$. As a corollary of our main result, we obtain the first sample optimal (up to constant factors) and near-linear time algorithm:

Theorem 46. There is an agnostic learning algorithm for $k$-mixtures of $t$-modal distributions over $[N]$ that draws $O\left(\frac{kt\log(N/kt)}{\epsilon^3}\right)$ samples and runs in time $O\left(\frac{kt\log(N/kt)}{\epsilon^3} \log(1/\epsilon)\right)$.

We similarly obtain a sample optimal and near-linear time algorithm for learning mixtures of MHR distributions.

For a distribution $p$ on $[N]$, the function $H(i) \triangleq \sum_{j \geq i} p(j)$ is called the hazard rate function of $p$. The distribution $p$ is a monotone hazard distribution (MHR) if $H(i)$ is non-decreasing. [CDSS13] shows that a mixture of $k$ MHR distributions over $[N]$ can be approximated up to distance $\epsilon$ using an $O(k\log(N/\epsilon)/\epsilon)$-histogram. Using this, [CDSS14b] yields a $\tilde{O}(k\log(N/\epsilon)/\epsilon^3)$ sample, $\tilde{O}(k\log(N/\epsilon)/\epsilon^3)$ time algorithm to estimate mixtures of MHR distributions. We obtain

Theorem 47. There is an agnostic learning algorithm for $k$-mixtures of MHR distributions over $[N]$ that draws $O(k\log(N/\epsilon)/\epsilon^3)$ samples and runs in time $O\left(\frac{k\log(N/\epsilon)}{\epsilon^3} \log(1/\epsilon)\right)$.

Mixtures of Binomial distributions. We consider mixtures of Poisson and Binomial distributions. The best sample complexity attainable by the techniques in [CDSS14b] for these types distributions is $\tilde{O}(1/\epsilon^3)$, since approximating a mixture of Poisson or Binomial distributions by piecewise flat distributions requires $\Theta(1/\epsilon)$ pieces.

However, a recent result of [DDS15] shows that any Poisson or Binomial distribution can be approximated to $L_1$ distance $\epsilon$ using $O(1)$ piecewise, degree $O(\log(1/\epsilon))$ polynomial. Therefore, any mixture of $k$ Poisson or Binomial distributions can be approximated with $O(k)$ piecewise, degree-$O(\log(1/\epsilon))$ polynomial. Applying our main result, we obtain

Theorem 48. There is an agnostic learning algorithm for $k$-mixtures of Binomial of Poisson distributions that uses $O\left(\frac{k}{\epsilon^2} \log(1/\epsilon)\right)$ samples and runs in time $\tilde{O}(k/\epsilon^2)$.
8 Experimental Evaluation

In addition to the strong theoretical guarantees proved in the previous sections, our algorithm also demonstrates very good performance in practice. In order to evaluate the empirical performance of our algorithm, we conduct several experiments on synthetic data. We remark that the evaluation here is preliminary, and we postpone a more detailed experimental study, including an extensive comparison with related algorithms, to future work. Nevertheless, our results here show that both the empirical sample and time complexity are nearly optimal in a strong sense. For example, no histogram learning algorithm that requires sorted samples can outperform the running time of our method by more than 30%. Similarly, our learning algorithm for piecewise linear hypotheses only adds a factor of $2 - 3\times$ overhead to the time needed to sort the samples. Moreover, the sample complexity of our algorithm matches the quantity $t \cdot \frac{d}{\epsilon^2}$ up to a small constant (roughly 2).

All experiments in this section were conducted on a laptop computer from 2010, using an Intel Core i7 CPU with 2.66 GHz clock frequency, 4 MB of cache, and 8 GB of RAM. We used Mac OS X 10.9 as operating system and g++ 4.8 as compiler with the -O3 flag (we implemented our algorithms in C++). All reported running times and learning errors are averaged over 100 independent trials. As an illustrative baseline, sorting $10^6$ double-precision floating point numbers using the std::sort algorithm from the C++ STL takes about 100 ms on the above machine.

Figure 6 shows the three distributions we used in our experiments: a mixture of two Gaussians, a mixture of two Beta distributions, and a mixture of two Gamma distributions. The three distributions have different shapes (e.g., different numbers of modes), and the support size considered for these distributions differs.

8.1 Histogram hypotheses

In order to evaluate our histogram learning algorithm (see Section 4.1), we use the following test setup. For a given unknown distribution with pdf $f$, we draw $n$ i.i.d. samples from the unknown distribution. We then give the sorted samples as input to our algorithm, which produces a histogram hypothesis $h$. We set the parameters of our algorithm so that the resulting histogram contains 80 constant pieces. As performance measures, we record the running time of our algorithm (excluding sorting) and the $L_1$ error achieved, i.e., $\|f - h\|_1$.

Figure 7 contains the running time results, both on a linear scale and on logarithmic scale. The results indicate three important points: (i) The running time of our algorithm scales nearly-linearly.
with the input size, i.e., the number of samples $n$. (ii) The constant hidden in the big-$O$ notation of our analysis is very small. In particular, the algorithm runs in less than 35 milliseconds for $10^6$ samples. Note that this is three times faster than sorting the samples. (iii) The running time of our algorithm essentially does not depend on the unknown distribution. Such robustness guarantees are very desirable for reliable performance in practice.

The $L_1$-learning error results are displayed in Figure 8. The results show that the best approximation error achievable with 80-piece histograms depends on the shape of the underlying distribution: 2-GMMs are harder to approximate than the Beta and Gamma mixtures. This shows that for large number of samples, it is beneficial to use richer hypotheses classes such as piecewise linear functions (see the next subsection). Nevertheless, our algorithm exhibits a good decay of the approximation error before the regime where $\text{OPT}_{80}$ dominates.

![Figure 7: Running times for density estimation with histogram hypotheses. The left plot shows the results on a linear scale, the right plot on a logarithmic scale. As predicted by our analysis, the running time of our algorithm scales nearly-linearly with the input size $n$. Moreover, the constant in the big-$O$ is very small: for $n = 10^6$, our algorithm takes less than 35 milliseconds, which is about three times faster than sorting the samples. The running time performance of our algorithm is also essentially independent of the unknown distribution.](image)

### 8.2 Piecewise linear hypotheses

Next, we turn our attention to the more challenging case of agnostically learning piecewise linear densities. This is an interesting case because, in contrast to the histogram algorithm, the piecewise linear algorithm requires our full set of tools developed in Sections 3 – 6. For the case of piecewise linear functions, the structure of the feasible set is still somewhat simpler than for general degree-$d$ polynomials because the non-negativity constraint on a given interval can be encoded with two linear inequalities instead, i.e., the feasible set is a polytope instead of a spectrahedron. We use this additional structure in our piecewise linear algorithm. However, we did not spend too much effort on optimizing our algorithm and resorted to an off-the-shelf linear program (LP) solver (GLPK, the GNU Linear Programming Kit) instead of a customized LP solver. We believe that the running time of our algorithm can be improved further by implementing a custom LP solver that better...
Figure 8: Approximation error for density estimation with histogram hypotheses. The left plot shows the results on a linear scale, the right plot on a logarithmic scale. The results clearly show that some distributions such as 2-GMMs are harder to approximate with 80-piecewise constant hypotheses than others. Before the optimal approximation error $O_{80}$ dominates, our algorithm nevertheless demonstrates a quickly diminishing learning error.

utilizes the structure and small size of our LPs, and also takes into account that we solve many such small LPs.

We repeat the same experimental procedure as for piecewise histogram hypotheses, but use 40 linear pieces this time. Figure 9 contains the running time results of our algorithm. Again, the results show three important points: (i) As predicted, the running time scales nearly-linearly with $n$. (ii) In spite of using an off-the-shelf LP solver, the constant factor in our running time is still good. In particular, our algorithm requires less than 0.3s seconds for $10^6$ samples. This is only three times slower than the time required for sorting the samples. We believe that with a customized LP solver, we can bring this overhead down to a factor of two. (iii) Again, the running time of our algorithm is very robust and does not depend on the shape of the unknown distribution.

Next, we consider the learning error achieved by our piecewise-linear algorithm, which is displayed in Figure 10. Compared with the plots for piecewise constant hypotheses above, the results show that piecewise linear hypotheses can approximate the unknown densities significantly better, especially for the case of the 2-GMM. Three points are worth noting: (i) The slope of the curve in the log-scale plot is about $-0.477$. Note that this matches the $\frac{t}{\epsilon^2}$ term in our learning error guarantee $O(t(d+1))$ almost perfectly. (ii) Moreover, the constant factor achieved by our algorithm is close to 1. In particular, the approximation error for the 2-GMM and $n = 10^6$ samples is roughly 0.00983. Using this as $\epsilon = 0.00983$ together with $t = 40$ and $d = 1$ in $\frac{t(d+1)}{\epsilon^2}$ gives about 830,000, which almost matches the $n = 10^6$ samples for which this error was obtained. (iii) The learning error of our algorithm is robust and essentially independent of the underlying distribution.

Acknowledgements

We thank Chinmay Hegde, Yin Tat Lee, and Aaron Sidford for helpful discussions.
Figure 9: Running times for density estimation with piecewise-linear hypotheses. The left plot shows the results on a linear scale, the right plot on a logarithmic scale. As predicted by our analysis, the running time of our algorithm scales nearly-linearly with the input size $n$. Moreover, the constant in the big-$O$ is quite small: for $n = 10^6$, our algorithm takes less than 0.3 seconds, which is only three times slower than sorting the samples. Note that this means that no algorithm that relies on sorting the samples can be more than 4 times faster than our algorithm when the total running time with sorting is taken into account. As before, the running time of our algorithm is also essentially independent of the unknown distribution.

References


Figure 10: Approximation error for density estimation with piecewise-linear hypotheses. The left plot shows the results on a linear scale, the right plot on a logarithmic scale. The slope of the curve in the log-scale plot is roughly $-0.477$, which almost exactly matches the asymptotic guarantee for our algorithm. Moreover, the average learning error for 2-GMMs with $n = 10^6$ samples is about 0.00983. Substituting this into the theoretical guarantee $\frac{C(d+1)}{\epsilon^2}$ gives a sample requirement of roughly 830,000, i.e., very close to the $10^6$ samples our algorithm required to achieve this error. As for the running time, the learning error is also robust and essentially independent of the underlying distributions.


Appendix

A Analysis of the General Merging Algorithm: Proof of Theorem 17

This section is dedicated to the proofs of Theorem 17. These proof is similar to that of Theorem 10, and we include it for completeness.

Recall the statement of Theorem 17.

Theorem 17. Let $O_p$ and $O_c$ be $\eta$-approximate $A_k$-projection and $A_k$-computation oracles for $D$. Algorithm GENERAL-MERGING$(f, t, \alpha, \epsilon, \delta)$ uses $n = O((\alpha dt + \log 1/\delta)/\epsilon^2)$ samples, runs in time $O \left( (R_p(n) + R_c(n)) \log(\frac{n}{\alpha t}) \right)$, and outputs a hypothesis $h$ and an interval partition $I$ such that $|I| \leq 2\alpha \cdot t$ and with probability at least $1 - \delta$ we have

$$\|h - f\|_1 \leq 3 \cdot \text{OPT}_{D,t} + \frac{\text{OPT}_{D,t} + \epsilon}{\alpha - 1} + 2\epsilon + \eta.$$  \hspace{1cm} (10)

Proof. We first bound the running time. The number of iterations of the algorithm is $O(\log(n/\alpha t))$ by the same argument as for histograms, since the number of intervals reduce by at least $1/4$-th in each iteration. In each iteration, we compute the closest functions in $D$ and the $A_{d+1}$ distances, whose run-time is bounded by $R_p(n) + R_c(n)$, by definition.

We now prove the error guarantee. Let $T = \{I_1, \ldots, I_{t'}\}$ be the partition of $I$ returned by GENERAL-MERGING, and let $h$ be the function returned. The desired bound on $t'$ is immediate since the algorithm terminates only when $t' \leq 2\alpha t$. We now prove (10).
Let $h^* \in D_t$ be such that $\|h^* - f\|_1 = \text{OPT}_{D,t}$. Let $I^* = \{I_1^*, \ldots, I_t^*\}$ be a partition with at most $t$ pieces such that $h^* \in D_{I_i^*}$ for all $i$. Call the end-points of $I_j^*$’s as jumps of $h^*$. For any interval $J \subset I$ let $\Gamma(J)$ be the number of jumps of $h^*$ in the interior of $J$. Since we draw $n = \Omega((\alpha d t + \log 1/\delta)/\epsilon^2)$ samples, Corollary 4 implies that with probability at least $1 - \delta$,

$$\|\hat{f} - f\|_{A_{(2\alpha + 1)(d+1)t}} \leq \epsilon.$$  

We condition on this event throughout the analysis. We split the total error into three terms based on the final partition $I$:

**Case 1:** Let $\mathcal{F}$ be the set of intervals in $I$ with no jumps in $h^*$, i.e., $\mathcal{F} = \{J \in I | \Gamma(J) = 0\}$.

**Case 2a:** Let $\mathcal{J}_0$ be the set of intervals in $I$ that were created in the initial partitioning step of the algorithm and contain a jump of $h^*$, i.e., $\mathcal{J}_0 = \{J \in I | \Gamma(J) > 0 \text{ and } J \in \mathcal{I}_0\}$.

**Case 2b:** Let $\mathcal{J}_1$ be the set of intervals in $I$ that contain at least one jump, and were created by merging two other intervals, i.e., $\mathcal{J}_1 = \{J \in I | \Gamma(J) > 0 \text{ and } J \notin \mathcal{I}_0\}$.

Notice that $\mathcal{F}, \mathcal{J}_0, \mathcal{J}_1$ form a partition of $I$, and thus

$$\|h - f\|_1 = \|h - f\|_{1,\mathcal{F}} + \|h - f\|_{1,\mathcal{J}_0} + \|h - f\|_{1,\mathcal{J}_1}.$$  

We bound the error from above in the three cases separately. In particular, we will show:

**Case 1**: By the triangle inequality we have

$$\|h - f\|_{1,\mathcal{F}} \leq 3 \cdot \|f - h^*\|_{1,\mathcal{F}} + 2 \cdot \|\hat{f} - f\|_{A_{|\mathcal{F}|(d+1)},\mathcal{F}} + \eta \frac{\epsilon}{2\alpha t} |\mathcal{F}|,$$  

**Case 2a**: Let $\mathcal{J}_0$ be the set of intervals in $I$ that were created in the initial partitioning step of the algorithm and contain a jump of $h^*$, i.e., $\mathcal{J}_0 = \{J \in I | \Gamma(J) > 0 \text{ and } J \in \mathcal{I}_0\}$.

**Case 2b**: Let $\mathcal{J}_1$ be the set of intervals in $I$ that contain at least one jump, and were created by merging two other intervals, i.e., $\mathcal{J}_1 = \{J \in I | \Gamma(J) > 0 \text{ and } J \notin \mathcal{I}_0\}$.

Notice that $\mathcal{F}, \mathcal{J}_0, \mathcal{J}_1$ form a partition of $I$, and thus

$$\|h - f\|_{1,\mathcal{F}} \leq 3 \cdot \|f - h^*\|_{1,\mathcal{F}} + 2 \cdot \|\hat{f} - f\|_{A_{|\mathcal{F}|(d+1)},\mathcal{F}} + \eta \frac{\epsilon}{2\alpha t} |\mathcal{F}|,$$  

We bound the error from above in the three cases separately. In particular, we will show:

$$\|h - f\|_{1,\mathcal{F}} \leq 3 \cdot \|f - h^*\|_{1,\mathcal{F}} + 2 \cdot \|\hat{f} - f\|_{A_{|\mathcal{F}|(d+1)},\mathcal{F}} + \eta \frac{\epsilon}{2\alpha t} |\mathcal{F}|.$$  

Using these results along with the fact that $\|f - h^*\|_{1,\mathcal{F}} + \|f - h^*\|_{1,\mathcal{J}_1} \leq \text{OPT}_{D,t}$ and $\alpha > 2$, we have

$$\|h - f\|_1 \leq 3 \cdot \text{OPT}_{D,t} + \frac{\text{OPT}_{D,t} + \epsilon}{\alpha - 1} + 2 \|\hat{f} - f\|_{A_{|\mathcal{F}|(d+1)}} + \|\hat{f} - f\|_{A_{|\mathcal{J}_0|d}}$$

$$\leq 3 \cdot \text{OPT}_{D,t} + \frac{\text{OPT}_{t} + \epsilon}{\alpha - 1} + \epsilon + \eta,$$  

where (a) follows from Fact 6(d) and since $(|\mathcal{F}| + |\mathcal{J}_1| + |\mathcal{J}_0|) \leq 2\alpha t$, and (b) follows from the VC inequality. Thus, it suffices to prove Equations (14)–(16).

**Case 1.** We first consider the set of intervals in $\mathcal{F}$. By the triangle inequality we have

$$\|h - f\|_{1,\mathcal{F}} \leq \|f - h^*\|_{1,\mathcal{F}} + \|h - h^*\|_{1,\mathcal{F}}.$$  

52
For any interval $J \in \mathcal{F}$, since $h$ and $h^*$ are both in $\mathcal{D}$, they have at most $d$ sign changes, and

$$
\|h - h^*\|_{1,J} = \|h - h^*\|_{A_{d+1},J} \leq \|h - \hat{f}\|_{A_{d+1},J} + \|\hat{f} - h^*\|_{A_{d+1},J}.
$$

By the definition of $h$ and the projection oracle,

$$
\|h - \hat{f}\|_{A_{d+1},J} \leq \min_{h' \in \mathcal{D}_J} \|h' - \hat{f}\|_{A_{d+1},J} + \frac{\eta}{2\alpha t} \leq \|h^* - \hat{f}\|_{A_{d+1},J} + \frac{\eta}{2\alpha t}.
$$

Therefore,

$$
\|h - h^*\|_{1,J} \leq 2 \cdot (\|h^* - \hat{f}\|_{A_{d+1},J} + \frac{\eta}{2\alpha t}).
$$

Again by the triangle inequality,

$$
\|h^* - \hat{f}\|_{A_{d+1},J} \leq \|h^* - f\|_{A_{d+1},J} + \|f - \hat{f}\|_{A_{d+1},J}.
$$

Summing over the intervals in $\mathcal{F}$,

$$
\sum_{J \in \mathcal{F}} \|h^* - \hat{f}\|_{A_{d+1},J} \leq \sum_{J \in \mathcal{F}} \|h^* - f\|_{A_{d+1},J} + \sum_{J \in \mathcal{F}} \|f - \hat{f}\|_{A_{d+1},J} + \sum_{J \in \mathcal{F}} \|f - \hat{f}\|_{A_{\mathcal{F}|(d+1),\mathcal{F}}}.
$$

Combining these, we obtain,

$$
\|h - f\|_{1,\mathcal{F}} \leq 3 \cdot \|f - h^*\|_{1,\mathcal{F}} + 2 \cdot \|f - \hat{f}\|_{A_{\mathcal{F}|(d+1),\mathcal{F}}} + \frac{\eta}{2\alpha t} |\mathcal{F}|,
$$

which is precisely (14).

**Case 2a.** We now analyze the error for the intervals $\mathcal{J}_0$. The set $\mathcal{I}_0$ contains only singletons and intervals with no sample points. By definition, with probability 1, only the intervals in $\mathcal{I}_0$ that contain no samples may contain a jump of $h^*$. The singleton intervals containing the sample points do not include jumps, and are hence covered by Case 1. Since $\mathcal{J}_0$ does not contain any samples, our algorithm assigns

$$
h(J) = \hat{f}(J) = 0
$$

for any $J \in \mathcal{J}_0$. Hence,

$$
\|h - f\|_{1,\mathcal{J}_0} = \|f\|_{1,\mathcal{J}_0},
$$

and

$$
\|h - f\|_{1,\mathcal{J}_0} = \|f\|_{1,\mathcal{J}_0}
$$

$$
= \sum_{J \in \mathcal{J}_0} |f(J)|
$$

$$
= \sum_{J \in \mathcal{J}_0} |f(J) - \hat{f}(J)|
$$

$$
\leq \|f - \hat{f}\|_{A_{\mathcal{J}_0|(d+1),\mathcal{J}_0}},
$$

where the last step simply follows from non-negativity of $f - \hat{f}$ over $\mathcal{J}_0$. 

53
Case 2b. We finally consider $J_1$, the set of intervals created by merging in some iteration of our algorithm that also contain jumps. As before, our first step is the following triangle inequality:

$$\|h - f\|_{1,J_1} \leq \|h - h^*\|_{1,J_1} + \|h^* - f\|_{1,J_1}.$$

Consider an interval $J \in J_1$ with $\Gamma(J) \geq 1$ jumps of $h^*$. Since $h \in D_J$, $h - h^*$ has at most $d \cdot \Gamma(J)$ sign changes in $J$. Therefore,

$$(a) \|h - h^*\|_{1,J_1} \leq \|h - h^*\|_{A_d \Gamma(J)+1,J} \leq (b) \|h - \hat{f}\|_{A_d \Gamma(J)+1,J} + \|\hat{f} - f\|_{A_d \Gamma(J)+1,J} + \|f - h^*\|_{A_d \Gamma(J)+1,J} \leq (c) \Gamma(J) \|h - \hat{f}\|_{A_d + 1,J} + \|\hat{f} - f\|_{A_d \Gamma(J)+1,J} + \|f - h^*\|_{1,J_1},$$

where (a) follows from Fact 6(a), (b) is the triangle inequality, and inequality (c) uses Fact 6(c) along with the fact that $\Gamma(J) \geq 1$ and $d \geq 1$. We start by bounding the $A_{d+1}$ distance in the first term above.

Lemma 49. For any $J \in J_1$, we have

$$\|h - \hat{f}\|_{A_{d+1},J} \leq \text{OPT}_{D,t} + \epsilon + \frac{\eta}{2(\alpha - 1)t}.$$

Before proving this lemma, we use it to complete Case 2b. Summing (7) over $J \in J_1$ and plugging in the lemma,

$$\|h - h^*\|_{1,J_1} \leq \left( \sum_{J \in J_1} \Gamma(J) \right) \cdot \left( \frac{\text{OPT}_{D,t} + \epsilon}{(\alpha - 1)t} + \frac{\eta}{2(\alpha - 1)t} \right) + \sum_{J \in J_1} \|\hat{f} - f\|_{A_d \Gamma(J)+1,J} + \|f - h^*\|_{1,J_1} \leq (a) \left( \frac{\text{OPT}_{D,t} + \epsilon}{(\alpha - 1)} \right) + \frac{\eta}{2(\alpha - 1)} + \|\hat{f} - f\|_{A_{d+1},J_1} + \|f - h^*\|_{1,J_1},$$

where the first term in (a) uses the fact that $\sum_{J \in J_1} \Gamma(J) \leq t$ and the second term uses this in conjunction with Fact 6(d).

We now prove Lemma 49.

Proof of Lemma 49. Each iteration of our algorithm merges pairs of intervals except those with the $\alpha t$ largest errors. Therefore, if two intervals were merged, there were at least $\alpha t$ other interval pairs with larger error. We will use this fact to bound the error on the intervals in $J_1$.

Suppose an interval $J \in J_1$ was created in the $j$th iteration of the while loop of our algorithm, i.e., $J = I'_{i,j+1} = I_{2i-1,j} \cup I_{2i,j}$ for some $i \in \{1, \ldots, s_j/2\}$. Recall that the intervals $I'_{i,j+1}$, for $i \in \{1, \ldots, s_j/2\}$, are the candidates for merging at iteration $j$. Let $h'$ be the distribution given by applying the projection oracle to the empirical distribution over each candidate interval $I_{j+1}' = \{I'_{i,j+1}, \ldots, I'_{s_j/2,j+1}\}$. Note that $h'(x) = h(x)$ for $x \in J$ since $J$ remains intact through the remainder of the algorithm.

As with the histogram estimation, for a class $D$ with at most $d$ sign changes, let $e_d(g, J) = \min_{g' \in D} \|g - g'\|_{A_{d+1}}$. Let $\mathcal{L}$ be the set of candidate intervals $I'_{i,j+1}$ in the set $I_{j+1}'$ with the largest $\alpha \cdot t$ errors $\|h' - \hat{f}\|_{A_{d+1}}$. By the guarantee of projection oracle,

$$\|h' - \hat{f}\|_{A_{d+1},I'_{i,j+1}} \leq e_d(\hat{f}, I'_{i,j+1}) + \frac{\eta}{2\alpha t}. $$

54
Let $\mathcal{L}_0$ be the intervals in $\mathcal{L}$ that do not contain any jumps of $h^*$. Since $h^*$ has at most $t$ jumps, $|\mathcal{L}_0| \geq (\alpha - 1)t$.

Therefore,

$$\sum_{I' \in \mathcal{L}_0} \|h' - \hat{f}\|_{A_{d+1},I'} \leq \sum_{I' \in \mathcal{L}_0} \left( e_d(\hat{f},I') + \frac{n}{2\alpha t} \right)$$

$$\leq \sum_{I' \in \mathcal{L}_0} \left( \|h^* - \hat{f}\|_{A_{d+1},I'} + \frac{n}{2\alpha t} \right)$$

$$\leq \|f - h^*\|_{1,\mathcal{L}_0} + \|f - \hat{f}\|_{A_{(d+1)\alpha},\mathcal{L}_0} + \frac{n}{2}$$

$$\leq \text{OPT}_{\mathcal{D},t} + \epsilon + \eta/2.$$ 

Since $h'$ is $h$ on the interval $J$, combining with $|\mathcal{L}_0| \geq (\alpha - 1)t$, we obtain

$$\|h' - \hat{f}\|_{A_{d+1},J} = \|h - \hat{f}\|_{A_{d+1},J} \leq \frac{\text{OPT}_{t} + 2\epsilon}{(\alpha - 1)t} + \frac{\eta}{2(\alpha - 1)t} ,$$

completing the proof of the lemma.

\[\square\]

B Additional Omitted Proofs

B.1 Proof of Fact 26

We first require the following classical lemma, first proved by Markov [Mar92]. For completeness, we include an elegant proof by the mathoverflow user fedja\textsuperscript{5}. We remark that the bounds in the following fact are essentially tight.

\textbf{Fact 50} ([Mar92]). Let $p(x) = \sum_{j=0}^{d} c_j x^j$ be a degree-$d$ polynomial so that $|p(x)| \leq 1$ for all $x \in [-1,1]$. Then $\max_{j} |c_j| \leq (\sqrt{2} + 1)^d$ for all $j = 0, \ldots, d$.

\textit{Proof.} We first claim that $|c_j| \leq \max_{z \in \mathbb{D}} |p(z)|$ where $\mathbb{D}$ is the unit complex disc. To see this, we notice that by Cauchy’s integral formula,

$$c_j = \frac{1}{j!} p^{(j)}(0) = \frac{1}{2\pi i} \int_{|\zeta| = 1} \frac{p(\zeta)}{\zeta^{j+1}} d\zeta ,$$

where we also changed the order of differentiation and integration and used

$$\frac{d}{dx^j} \frac{p(\zeta)}{\zeta - x} = \frac{j! \cdot p(\zeta)}{(\zeta - x)^{j+1}} .$$

\textsuperscript{5}See \url{http://mathoverflow.net/questions/97769/approximation-theory-reference-for-a-bounded-polynomial-having-bounded-coefficie}
Therefore, we get
\[
|c_j| = \frac{1}{2\pi} \left| \int_{|\zeta|=1} \frac{p(\zeta)}{\zeta^{j+1}} d\zeta \right|
\leq \frac{1}{2\pi} \int_{|\zeta|=1} \left| \frac{p(\zeta)}{\zeta^{j+1}} \right| d\zeta
\leq \max_{|\zeta|=1} |p(\zeta)|.
\]
Consider the function
\[
F(z) = z^{-m}p\left(\frac{z + z^{-1}}{2}\right).
\]
On the domain \(\{z : |z| \geq 1\}\), this function is analytic. So by the maximum modulus principle, it is bounded by its value on the unit circle. Since for all \(z \in \mathbb{D}\), \((z + z^{-1})/2 = \Re(z)\), we conclude that \(|F(z)| \leq \max_{x \in [-1,1]} p(x) \leq 1\) by assumption. Thus we have that
\[
p\left(\frac{z + z^{-1}}{2}\right) \leq z^d
\]
for all \(|z| > 1\). Fix any \(w \in \mathbb{D}\). It is straightforward to see that \(w = (z + z^{-1})/2\) for some \(z \in \mathbb{C} \setminus \{0\}\); by symmetry of \(z\) and \(z^{-1}\) we conclude that this also holds for some \(z\) with \(|z| \geq 1\). For each \(w\), arbitrarily choose such a \(z\) and denote it \(z_w\). Moreover, for all \(|z| > (\sqrt{2} + 1)\), we have
\[
\left|\frac{z + z^{-1}}{2}\right| \geq \frac{|z| - |z^{-1}|}{2} > \frac{\sqrt{2} + 1 - \frac{1}{\sqrt{2}+1}}{2} \geq 1
\]
and thus we conclude that for all \(w \in \mathbb{D}\) we have that its corresponding \(z_w\) satisfies \(|z_w| \leq \sqrt{2} + 1\) and therefore \(|p(w)| = |p((z_w + z_w^{-1})/2)| \leq z_w^d \leq (\sqrt{2} + 1)^d\), as claimed. \(\square\)

The above statement is for polynomials that are uniformly bounded on \([-1,1]\). We will be interested in bounds for polynomials that integrate to a fixed constant. In order to relate these bounds, we use the following classical result.

**Fact 51** (Bernstein’s Inequality [Che82]). Let \(p\) be a degree-\(d\) polynomial and let \(p'\) be its derivative. Then
\[
\max_{x \in [-1,1]} |p'(x)| \leq d^2 \cdot \max_{x \in [-1,1]} |p(x)|.
\]

With these results, we are now ready to prove Lemma 26.

**Proof of Lemma 26.** Consider the degree-(\(d + 1\)) polynomial \(P\) such that \(P(-1) = 0\) and \(P' = p\). This implies that \(P(x) = \int_{-1}^{x} p(y) \, dy\). Since \(p\) is non-negative on \([-1,1]\), the bound on \(\int_{-1}^{1} p(y) \, dy\) then gives
\[
\max_{x \in [-1,1]} |P(x)| \leq \alpha \cdot (\sqrt{2} + 1)^d.
\]
Using Bernstein’s Inequality (Fact 51), we can convert this bound into a bound on \(P' = p\), i.e., we get that \(|p(x)| \leq t \cdot (d + 1)\) for all \(x \in [-1,1]\). Combining this uniform bound on \(p\) with Fact 50 gives the desired bound on the coefficients of \(p\). \(\square\)
B.2 Proof of Lemma 34

Our approach to proving Lemma 34 is relatively straightforward. Assume we had an algorithm $A$ that finds the roots of $p$ exactly. Then one could perform a non-negativity test by running $A$ to find the roots of $p'$, which correspond to the extrema of $p$. Given the extrema of $p$, it suffices to check whether $p$ is non-negative at those points and the endpoints of the interval.

However, such an exact root-finding algorithm $A$ does not exist in general. Nevertheless, there are efficient algorithms for finding the approximate roots of $p$ in certain regimes. We leverage these results to construct an efficient non-negativity test. Before we proceed, we remark briefly that we could also utilize the univariate SOS algorithm [Sho87, Las01, Par03], which is arguably more elementary than our approach here, but slower.

Formally, we build on the following result.

**Fact 52** ([Pan01], Part II, Theorem 1.1). Let $\mathbb{D}$ denote the complex unit disc. For all $\nu > 0$, there exists an algorithm $\text{FindRoots}(q, \beta)$ satisfying the following guarantee: given any degree-$d$ polynomial $q(z): \mathbb{C} \to \mathbb{C}$ with roots $z_1, \ldots, z_d$ such that $z_i \in \mathbb{D}$ for all $i$ and $\beta \geq d \log d$, returns $z_1^*, \ldots, z_d^*$ so that $|z_j^* - z_j| \leq 2^{2-\beta/d}$ for all $j$. Moreover, $\text{FindRoots}$ runs in time $O(d \log^2 d \cdot (\log^2 d + \log \beta))$.

Our polynomials do not necessarily have all roots within the complex unit disc. Moreover, we are only interested in real roots. However, it is not too hard to solve our problems with the algorithm from Fact 52. We require the following structural result:

**Fact 53** ([Hen88], Sect. 6.4). Let $q(x) = x^d + c_{d-1}x^{d-1} + \ldots + c_1 x + c_0$ be a monic polynomial of degree $d$ (i.e., the leading coefficient is 1). Let $\rho(q)$ denote the norm of the largest zero of $q$. Then

$$\rho(q) \leq 2 \max_{1 \leq i \leq d} |c_{d-i}|^{1/i}.$$ 

In order to use the result above, we process our polynomial $p$ so that it becomes monic and still has bounded coefficients. We achieve this by removing the leading terms of $p$ with small coefficients. This then allows us to divide by the leading coefficient while increasing the other coefficients by a controlled amount only. Formally, we require the following definitions.

**Definition 54** (Truncated polynomials). For any degree-$d$ polynomial $p = \sum_{i=0}^{d} c_i x^i$ and $\nu > 0$ let

$$\Delta = \Delta(p, \nu) = \max \left\{ i : |c_i| \geq \frac{\nu}{2d} \right\},$$

and let $\Pi = \Pi_\nu$ be the operator defined by

$$(\Pi p)(x) = \sum_{i=0}^{\Delta(p, \nu)} c_i x^i.$$ 

Formally, $\Pi$ acts on the formal coefficient representation of $p$ as $q = \sum c_i x^i$. It then returns a formal representation $\sum_{i=0}^{\Delta(p, \nu)} c_i x^i$. In a slight abuse of notation, we do not distinguish between the formal coefficient representation of $p$ and the polynomial itself. Then Facts 52 and 53 give us the following:
Lemma 55. There exists an algorithm \textsc{FastApproxRoots}(p,v,\mu) with the following guarantee. Let \( p \) be a polynomial as in Definition 34, and let \( v, \mu > 0 \) such that \( v \leq \frac{1}{2\alpha d} \) (where \( \alpha \) and \( d \) are as in Def. 34). Then \textsc{FastApproxRoots} returns approximate roots \( x_1^*, \ldots, x_{\Delta(p,v)}^* \in \mathbb{R} \) so that for all real roots \( y \) of \( \Pi_v p \), there is some \( j \) so that \( |y - x_j^*| \leq \mu \). Moreover, \textsc{FastApproxRoots} runs in time \( O(d \log^2 d \cdot (\log^2 d + \log \log \alpha + \log \log (1/\nu) + \log \log (1/\mu))) \).

Proof. \textsc{FastApproxRoots}(p,v,\mu) proceeds as follows. We find \( \Delta = \Delta(p,v) \) and \( \Pi_p = \Pi_v p \) in time \( O(d) \) by a single scan through the coefficients \( c_i \) of \( p \). Let \( q_1(x) = \frac{1}{c_{\Delta}}(\Pi_p)(x) \). Note that the roots of \( q_1 \) are exactly the roots of \( \Pi_p \). Then, by Theorem 53, we have that

\[
A \overset{\text{def}}{=} 2 \max_{1 \leq i \leq \Delta} \left| \frac{c_{\Delta-i}}{c_{\Delta}} \right|^{1/i} \geq \rho(q_1).
\]

The quantity \( A \) is also simple to compute in a single scan of the \( c_i \). Notice that we have

\[
A \leq \max \left( 2 \max_{1 \leq i \leq \Delta} \left| \frac{c_{\Delta-i}}{c_{\Delta}} \right|, 1 \right) \leq \frac{2\alpha d}{\nu \mu B}
\]

by the definition of \( \Delta \) and the assumption that the \( c_i \) are bounded by \( \alpha \) (Definition 34). Let \( B \) denote the right hand side of the expression above. If we let \( q(x) = q_1(Ax) \), we have that the roots of \( q \) all lie within the complex unit disc. Let \( z_1, \ldots, z_{\Delta} \) be the roots of \( \Pi_p \). Then the roots of \( q \) are exactly \( z_1/A, \ldots, z_{\Delta}/A \). Run \textsc{FindRoots}(q,2d + d \log B + d \log (1/\mu)), which gives us \( z_1^*, \ldots, z_{\Delta}^* \) so that for all \( i \), we have \( |z_i^* - z_i/A| < \mu/B \). Thus, for all \( i \), we have

\[
|Az_i^* - z| \leq A\frac{\mu}{B} \leq \mu.
\]

\textsc{FastApproxRoots}(p,v,\mu) returns the numbers \( x_1^* = \Re(Az_1^*) \). For any real root \( x \) of \( \Pi_p \), there is some \( z_i^* \) so that \( |Az_i^* - x| < \mu \), and thus \( |x_i^* - x| < \mu \) as well. Thus, we output numbers which satisfy the conditions of the Lemma. Moreover, the runtime of the algorithm is dominated by the runtime of \textsc{FindRoots}(q,2d + d \log B + d \log (1/\mu)), which runs in time

\[
O(d \log^2 d \cdot (\log^2 d + \log \log \alpha + \log \log (1/\nu) + \log \log (1/\mu)))
\]

This completes the proof. \( \square \)

Proof of Lemma 34. Let \( \nu = \frac{\mu}{2} \), and let \( \nu' = \frac{\mu}{4\alpha d (d+1)} \). Set

\[
r = (\Pi_v p)(x) = \sum_{i=1}^{\Delta(p,v)} c_i x^i.
\]

We can compute the coefficients of \( r \) in time \( O(d) \). Moreover, \( \Pi(r')(x) = r'(x) \). Let \( x_1, \ldots, x_d \), where \( d' \leq \Delta \), be the roots of \( r'(x) \) in \([-1,1] \). These points are exactly the local extrema of \( r \) on \([-1,1] \). Our algorithm \textsc{TestNonneg}(p,\mu) then is simple:

1. Run \textsc{FastApproxRoots}(r,\nu',\mu) and let \( x_1^*, \ldots, x_{\Delta}^* \) be its output.
2. Let \( J = \{i : x^*_i \in [-1, 1]\} \) and construct the set \( S = \{-1, 1\} \cup \{x_i : i \in J\} \).

3. Denote the points in \( S \) by \( x_0 = -1 \leq x_1 \leq \ldots \leq x_{d'} - 1 \leq x_d' = 1 \), where \( d' \leq \Delta + 1 \).

4. Evaluate the polynomial \( p \) at the points in \( S \) using the fast algorithm from Fact 41.

5. If at any of these points the polynomial evaluates to a negative number, return that point. Otherwise, return “OK”.

The running time is dominated by the call to \textsc{FastApproxRoots}. By Lemma 55, this algorithm runs in time \( O(d \log^2 d \cdot (\log^2 d + \log \log \alpha + \log \log(1/\mu))) \) as claimed.

It suffices to prove the correctness of our algorithm. Clearly, if \( p \) is nonnegative on \([-1, 1]\), it will always return “OK”. Suppose there exists a point \( y \in I \) so that \( p(y) < -\mu \).

For a function \( f \), and an interval \( I = [a, b] \), let \( |f|_{\infty, I} = \sup_{x \in I} |f(x)| \). Then,

\[
\|p - r\|_{\infty, [-1, 1]} \leq \sup_{x \in [-1, 1]} \left| \sum_{i=1}^{d} c_i x_i \right| (d - \Delta) \cdot \frac{\mu}{4d} \leq \mu/4 , \tag{19}
\]

where the inequality (a) follows from the choice of \( \Delta \). Thus \( r(y) < -3\mu/4 \). Since the points \( x_0, x_1, \ldots, x_{d'} \) are extremal for \( r \) on \( I \), there exists a \( 0 \leq j \leq d' \) so that \( r(x_j) < -3\nu/4 \). If \( j = 0 \) (resp. \( j = m' \)), so if \( r(-1) < -3\mu/4 \) (resp. \( r(1) < -3\mu/4 \)), then by Equation (19), we have \( p(-1) < \mu/2 \) (resp. \( p(1) < -\mu/2 \)). Thus our algorithm correctly detects this, and the polynomial fails the non-negativity test as intended.

Thus assume \( j \in \{1, \ldots, \Delta\} \). By Lemma 55, we know that there is a \( x^*_j \) so that \( |x^*_j - x_j| < \nu' \).

Since \( x_j \in I \), either \( \ell \in J \) or \( |x_j + 1| < \nu' \) or \( |x_j - 1| < \nu' \), so in particular, there is a point \( s \in S \) so that \( |x_j - s| < \nu' \). Since for all \( x \in [-1, 1] \), we have

\[
|p'(x)| \leq \sum_{i=1}^{d} |ic_i x^i| \leq \alpha d(d + 1)
\]

by the bound on the coefficients of \( p \) (see Definition 34). By a first order approximation, we have that

\[
|p(x_j) - p(s)| \leq \alpha d(d + 1)|x_j - s| \leq \mu/4
\]

where the last inequality follows by the definition of \( \nu' \). Thus, we have that \( p(s) < -\mu/2 \), and we will either return \( s \) or some other point in \( s' \in S \) with \( p(s') \leq p(s) \). Thus our algorithm satisfies the conditions on the theorem.

\[\square\]

\section*{C Learning discrete piecewise polynomials}

Throughout this paper we focused on the case that the unknown distribution has a density \( f \) supported on \([-1, 1]\), and that the error metric is the \( L_1 \)-distance with respect to the Lebesgue measure on the real line. We now show that our algorithm and analysis naturally generalize to the case of discrete distributions.

In the discrete setting, the unknown distribution is supported on the set \([N] \overset{\text{def}}{=} \{1, \ldots, N\} \), and the goal is to minimize the \( \ell_1 \)-distance between the corresponding probability mass functions. The
\( \ell_1 \)-norm of a function \( f : [N] \to \mathbb{R} \) is defined to be \( \| f \|_1 = \sum_{i=1}^N |f(i)| \) and the \( \ell_1 \)-distance between \( f, g : [N] \to \mathbb{R} \) is \( \| f - g \|_1 \).

In the following subsections, we argue that our algorithm also applies to the discrete setting with only minor adaptations. That is, we can agnostically learn discrete piecewise polynomial distributions with the same sample complexity and running time as in the continuous setting.

### C.1 Problem statement in the discrete setting

Fix an interval \( I \subseteq [N] \). We say that a function \( p : I \to \mathbb{R} \) is a degree-\( d \) polynomial if there is a degree-\( d \) real polynomial \( q : \mathbb{R} \to \mathbb{R} \) such that \( p(i) = q(i) \) for all \( i \in I \). We say that \( h : [N] \to \mathbb{R} \) is a \( t \)-piecewise degree-\( d \) polynomial if there exists a partition of \( [N] \) into \( t \) intervals so that on each interval, \( h \) is a degree-\( d \) polynomial. Let \( \mathcal{P}^{\text{disc}}_{t,d} \) be the set of \( t \)-piecewise degree-\( d \) polynomials on \( [N] \) which are nonnegative at every point in \( [N] \). Fix a distribution (with probability mass function) \( f : [N] \to \mathbb{R} \). As in the continuous setting, define \( \text{OPT}^{\text{disc}}_{t,d} \) as \( \min_{g \in \mathcal{P}^{\text{disc}}_{t,d}} \| g - f \|_1 \).

As before, our goal is the following: given access to \( n \) i.i.d. samples from \( f \), to compute a hypothesis \( h \) so that probability at least \( 9/10 \) over the samples, we have \( \| h - f \|_1 \leq C \cdot \text{OPT}^{\text{disc}}_{t,d} + \epsilon \), for some universal constant \( C \). As before, we let \( \hat{f} \) denote the empirical after taking \( n \) samples.

### C.2 The algorithm in the discrete setting

Our algorithm in the discrete setting is nearly identical to the algorithm in the continuous setting, and the analysis is very similar as well. Here, we only present the high-level ideas of the discrete algorithm and highlight the modifications necessary to move from a continuous to a discrete distribution.

#### C.2.1 The \( A_k \)-norm and general merging in the discrete setting

We start by noting that the notion of the \( A_k \)-norm and the VC inequality also hold in the discrete setting. In particular, the \( A_k \)-norm of a function \( f : [N] \to \mathbb{R} \) is defined as

\[
\| f \|_{A_k} = \max_{I_1, \ldots, I_k} \sum_{i=1}^k |f(I_i)| ,
\]

where the maximum ranges over all \( I_1, \ldots, I_k \) which are disjoint sub-intervals of \( [N] \).

The basic properties of the \( A_k \)-norm (i.e., those in Lemma 6) still hold true. Moreover, it is well-known that the VC inequality (Theorem 2) still holds in this setting. These properties of the \( A_k \)-norm are the only ones that we use in the analysis of GENERALMERGING. Therefore, it is readily verified that the same algorithm is still correct, and has the same guarantees in the discrete setting, assuming appropriate approximate \( A_k \)-projection and \( A_k \)-computation oracles for polynomials on a fixed subinterval of \( [N] \).

#### C.2.2 Efficient \( A_k \)-projection and computation oracles for polynomials

We argue that, as in the continuous setting, we can give efficient \( A_k \)-projection and computation oracles for non-negative polynomials of degree \( d \) on a discrete interval \( I \), using an \( O(d) \)-dimensional convex program.
**The Convex Program**  As in the continuous case, it can be shown that the set of non-negative polynomials $p$ on $I$ satisfying $\|p - \hat{f}\|_{A_k} \leq \tau$ is convex (as in Lemma 21), for any fixed $\tau > 0$ (since $\| \cdot \|_{A_k}$ is a norm). Moreover, using explicit interpolation formulas, it is easy to show that every polynomial in this feasible region has a representation with bounded coefficients (the analogue of Theorem 27), and that the feasible region is robust to small perturbations in the coefficients (the analogue of Theorem 28). Thus, it suffices to give an efficient separation oracle for the feasible set.

**The Separation Oracle**  Recall that the separation oracle in the continuous case consisted of two components: (i) a non-negativity checker (Subsection 6.2), and (ii) a fast $A_k$-computation oracle (Subsection 6.3). We still use the same approach for the discrete setting.

To check that a polynomial $p : I \to \mathbb{R}$ with bounded coefficients is non-negative on the points in $I$, we proceed as follows: we use Fast-Approx-Roots to find all the real roots of $p$ up to precision $1/4$, then evaluate $p$ on all the points in $I$ which have constant distance to any approximate root of $p$. Since $p$ cannot change sign in an interval without roots, this is guaranteed to find a point in $I$ at which $p$ is negative, if one exists. Moreover, since $p$ has at most $d$ roots, we evaluate $p$ at $O(d)$ points; using Fact 41, this can be done in time $O(d \log d \log \log d)$.

Finally, to compute the $A_k$-distance between $p = \sum_{j=0}^{d} c_j x^j$ and $\hat{f}$ on an interval $I$, we use the same reduction as in Section 6.3 with minor modifications. The main difference is that between two points $x_i, x_{i+1}$ in the support of the empirical distribution, the quantity $p[x_i, x_{i+1}]$ (see section 6.3) is now defined to be

$$p[x_i, x_{i+1}] = \sum_{\ell=x_{i+1}}^{x_{i+1}-1} p(\ell)$$

$$= \sum_{\ell=x_{i+1}}^{x_{i+1}-1} \sum_{j=0}^{d} c_j \ell^j$$

$$= \sum_{j=0}^{d} c_j \left( \sum_{\ell=x_{i+1}}^{x_{i+1}-1} \ell^j \right).$$

Notice that the above is still a linear expression in the $c_j$, and there are simple closed-form expressions for $\left( \sum_{\ell=0}^{\beta} \ell^j \right)$ for all integers $\alpha, \beta$ and for all $0 \leq j \leq d$. Following the arguments in Section 6.3 with this substituted quantity, one can show that the quantity returned by ApproxSepOracle in the discrete setting is still a separating hyperplane for $p$ and the current feasible set. Moreover, ApproxSepOracle still runs in time $O(d)$.  

61