SIDESTEPPING HARDNESS IN STATISTICAL PROBLEMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Brian Axelrod
July 2022

This dissertation is online at: https://purl.stanford.edu/dj989fk4653

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Gregory Valiant, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Omer Reingold, Co-Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Li-Yang Tan

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Preface

This thesis examines two kinds of hardness in computational problems involving noise and data. The first kind of hardness is computational—the minimum amount of computational resources required to compute a solution. The second kind of hardness is statistical—the minimum amount of data required to produce an output that is correct with sufficiently high probability. In particular, this thesis is focused on how to circumvent the limitations imposed by certain hardness results.

We start out by examining strategies to sidestep computational hardness. We start by examining minimum risk motion planning (MRMP), the problem of a robot finding a path in an environment where the obstacles are estimated instead of known apriori. As we show in chapter 2, this problem is hard even under conditions that make the path-finding problem with obstacles known apriori easy. However, this does not mean that solving instances of this problem are infeasible. We identify a parameter that controls the hardness of an instance of the problem. When this parameter is small, we are able to find an optimal solution efficiently. Even though MRMP may be infeasibly difficult in the worst case, many practical instances that we care about turn out to be "simple" in the sense this parameter is small, allowing us make progress.

We sidestep hardness in a different way for the non-parametric statistical problem of log-concave maximum likelihood in chapter 4. The original algorithm for log-concave maximum likelihood requires computing a high-dimensional triangulation. Unfortunately, the size of a triangulation can grow exponentially with dimension, making the problem computationally infeasible even in the medium dimensional setting. We draw a connection between the log-concave maximum likelihood problem and the much simpler problem of exponential family maximum likelihood. Exponential family maximum likelihood, which includes recovering a Gaussian by computing the sample mean, yields a simple algorithm that applies to all distributions in the exponential family. It turns out that solutions to the log-concave maximum likelihood problem are members of an exponential family, at least in general position for a sufficiently small neighborhood of parameter values. While these caveats are important, this structure allows a *modified* version of the exponential family maximum likelihood algorithm to successfully compute a solution. Not only does this algorithm converge relatively quickly, but it also does not require computing the exponential large triangulation. This yields the first polynomial time algorithm for solving log-concave maximum likelihood problem.

The final portion of this thesis examines statistical hardness. In chapter 5 we examine sample amplification, a stylized problem related to data augmentation and the synthesis of artificial data. Sample amplification is the problem of taking n samples drawn iid from some distribution, and synthesizing a dataset of size $n + m$ that is statistically indistinguishable from a true dataset. At first glance, one might endeavor to recover ("learning" in the parlance of the field) the distribution and append m new samples from said recovered distribution. It turns out that the task of learning a distribution is substantially *harder* than the task of sample amplification. In fact, even for simple distributions like multivariate Gaussians, it is possible to enlarge a dataset by one sample with square root as many samples as is required to recover a distribution well enough to produce a single, high quality sample. We show that taking steps to actively decorrelate the dataset is necessary to achieve this result. In effect, one must do something substantially different than learning a distribution to achieve this improvement in sample complexity. Effectively, be examining a slightly different problem formulation, we are able to sidestep the hardness lower bounds that come from the distribution learning problem.

Acknowledgments

My PhD could not have happened in a vacuum, in that many helped me both along the way and to even have the opportunity to pursue it in the first place. This section thanks a small sampling of people who contributed both directly to the work contained in this thesis, and enabled me to have an opportunity to pursue it.

I would like to start out by thanking my parents, Irina and Boris Axelrod. Their support of me started before I was born, when they immigrated the first time. Not only did they sacrifice familiarity and proximity to family, but they were never able to return to their original careers after moving west. I credit them not only for supporting my education at every turn, but also being excellent role models. I can only strive to have as much grit, resourcefulness and compassion as you two did while raising my brother and me.

I was also extremely fortunate in the industry roles I was able to hold before the completion of my PhD. While few of them were research focused, they helped me support myself while giving me context that would be difficult to obtain in academia alone. In particular I would like to thank the many individuals who not only took a risk hiring the younger and less experienced versions of me but and mentored me at the various places I have worked. In no particular order, this includes Jenny Yang, Mark Leon, David Gofman, Steve Cousins, Kaijen Hsiao, Michel Laverne, Ashish Goel, Debadeepta Dey, Gireeja Ranade and Alex Lemon. There are two in particular I would like to note. Steve Cousins for hiring me at Willow Garage right after high school, an opportunity that really launched my robotics career. Kaijen Hsiao has been a great friend and mentor for me ever since I graduated high school. She introduced me to the Learning and Intelligent Systems group at MIT, offered sage advice many times over the years, and created the team at Waymo which I am excited to be joining soon.

At MIT, I was very lucky to spend four years with Leslie Pack Kaelbling and Tomás Lozano-Pérez in the Learning and Intelligent Systems groups. It was under your patient guidance that I first learned how to become a researcher and develop my own unique flavor of research. I want to thank you for not just cultivating your students as researchers but also looking out for us and making us feel at home.

I also wanted to thank Michel Laverne. Not only was it a pleasure to work with you, but you

were also very patient with me as you co-authored my first paper with me. Having gained a bit more experience in that department since, I have developed an appreciation of just how much time and patience that required.

At Stanford I was extremely lucky to have been advised by Gregory Valiant and Omer Reingold. You me even though I had no theory background and you guided me in learning a new area. You encouraged me to pursue ambitious projects, supported my work even when it was far from your field of expertise and were patient with as my productivity oscillated. Thank you for welcoming me to the Stanford theory group, which was such an integral part of my PhD.

I wanted to thank all the students of the theory group, including but not limited to, Annie Marsden, Shashwat Silas, Ofir Geri, Reyna Hullet, Shivam Garg, Neha Gupta, Margalit Glasgow, Benedikt Bünz, Saba Eskandarian, Alex Porter, Clement Cannonne, Rad Niazadeh. Not only was it a pleasure to collaborate with many of it, I learned so much from all of you and am proud to call many of you close friends.

No discussion of my time at Stanford would be complete without mentioning the cycling team which has been such a large part of my time at Stanford. Thank you John Vaughen, Shifrah Aron-Dine, Dave Miller, Catherine Kirkos, Heidi Poppe, Maxime Cauchois, Katia Tkachenko, Lauren Cantwell and all the others that not only rode with me but also signed up for hare-brained adventures and taught me so much about different areas. I'd also like to give a special thank the folks that joined me for my longer bikepacking trips. Tony Wang, Tristan Ballard, Davis Voiceščuks and Rennie Kendrick, thank you for joining me on these! I certainly have both epic stories and some of my favorite memories from these trips.

Finally, I wanted to take a moment to thank Karen Ouyang. You've been a great friend, even as both our lives have changed so much since we met in middle school.

Contents

List of Figures

Chapter 1

Introduction

The purpose of this chapter is to further elucidate the structure of this thesis and explain in further detail how each chapter relates to the theme. Each chapter will begin with an introduction that motivates the problem and addresses related work so we defer that discussion until later sections.

1.0.1 Minimum Risk Motion Planning

Chapters 2 and 3 consider the minimum risk motion planning problem and are based on joint work with Luke Shimanuki and under the supervision of Leslie Pack Kaelbling and Tomás Lozano-Pérez.

First we recall the setting of minimum risk motion planning. Consider a robot that wants to get from point a to point b, but its sensors do not measure the location and shape of obstacles perfectly. This means that when the robot comes close to the measured location of an obstacle, there is some risk of collision. Minimum risk motion planning is the problem of finding a path from point a to point with the minimum risk of collision.

However, before making any attempts to circumvent hardness lower bounds on the problem, we must first identify if the problem is hard at all in the first place. For many classical algorithmic problems there is a straightforward way of doing so—if the problem is NP-hard we consider it "hard" and if it solvable in polynomial time we consider it "easy". This binary classification breaks down when examining motion planning problems in robotics, however, because motion planning is almost always NP-hard. Even though motion planning is NP-hard, we have algorithms which are very effective at solving practical instances of motion planning problems. These algorithms can generally be split into two parts, the execution of which is often interleaved. The first part identifies or incrementally constructs a graph which contains a solution. The second part does a graph search within this graph. The idea behind this abstraction is that the graph search is *always* "easy", and while in the worst case, finding a graph that contains the solutions is hard, it is often "easy" for most practical instances of motion planning problems. This abstraction lets the algorithm designer

focus on techniques for finding the graph and while using standard graph searches. We formalize a method of quantifying hardness with this abstraction as the *graph-restriction* hardness. That is, how hard is it to solve the problem when given a graph containing the solution?

Unfortunately, there is a gap in hardness between the more classical motion planning problem with known obstacle locations and the minimum risk motion planning problem with estimated locations. This is the result shown in chapter 2. However, the intention is not to lose at hope at this stage. The reduction helps us gain insight into what exactly makes the problem hard in the worst case.

Chapter 3 builds on this and identifies a parameter which controls the hardness of the graph search problem. Small values of this parameter block constructions with correlations that allow us to show that the problem is hard. Let α denote this parameter. We show an algorithm with run time parameterized by another parameter β which can be used to solve the graph search problem. The algorithm runs in polynomial time with exponent a function of β. For problem instances where $\beta \geq \alpha$ the algorithm returns the optimal solution. For instances where $\beta < \alpha$, we can show the algorithm returns an approximate solution, with the quality of approximation determined by the ratio between β and α. In essence, even though we showed the problem is $NP - hard$ we are able to optimally solve these easier instances efficiently. When the problem instance is only marginally harder than what is solvable by the allocated computational resources, we return a good quality approximation.

Furthermore, we believe that most practical instances of minimum risk motion planning have small values of the parameter α. The parameter is also natural enough that one can evaluate what values they expect this parameter to take for a particular application, effectively sidestepping the NP-hardness result in many practical settings.

1.0.2 Log-Concave Maximum Likelihood

Chapter 4 involves sidestepping hardness via avoiding a hard subproblem and is joint work with Gregory Valiant, Ilias Diakonikolas, Anastasios Sidiropoulos and Alistair Stewart. Log-concave maximum likelihood is a non-parametric statistical technique for recovering a certain class of distributions from samples. Again, a more thorough definition and motivation of the problem is present in the start of chapter 4.

Here the hardness in the original algorithm for log-concave maximum likelihood comes from enumerating a triangulation for n points in d dimensions, which high dimensional geometry tells us can be of size $n^{\Theta(d)}$. The original algorithm solved a convex optimization, which required computing a volume by summing calculations over each simplex in the triangulation. We propose a different algorithm for computing the log-concave MLE with runtime polynomial in both n and d. We achieve this result by [slightly] modifying the optimization formulation in a way that does not change the optimum but results in a problem which is both well conditioned and yields a stochastic gradient

oracle which does not need to enumerate the triangulation. It turns out there is a strong connection between exponential families and the class of distributions which show up as solutions to log-concave MLE problems, which makes it straightforward to understand why the algorithm is correct and efficient for anyone familiar with the theory of exponential families.

1.0.3 Sample Amplification

Chapter 5 gives us one last perspective on sidestepping hardness, this time in terms of the size of the dataset required to achieve some aim. Chapter 5 examines the opportunities opened up by changing the problem formulation in a way that makes a great deal of difference to the algorithm designer, but not to an individual using the output of the algorithm. This chapter is based on joint work that began with Shivam Garg, Vatsal Sharan and Greg Valiant and expanded to include Yanjun Han after the first paper in the series.

Consider a data augmentation application. A practitioner may take an input dataset and expand it with newly generated samples. This new, "synthetic", dataset is then used to train a model, often with better performance or sample complexity than if one used the original dataset. One way of formalizing some of these tasks is with the concept of distribution learning. One has successfully learned a distribution p in total variation distance if given n samples drawn iid from p, one can output a single new sample close in total variation distance to p. This sample can be added to the original dataset yielding a new, larger "synthetic" dataset. The idea behind sample amplification is that this problem is unnecessarily hard from a statistical perspective if one is only interested in constructing a synthetic dataset from the original dataset. Sample amplification defines the alternative task: given n samples drawn iid from p, output a dataset of size $n+1$ close in total variation distance to a true dataset from p. It turns out this task can be accomplished successfully with *substantially* smaller datasets than distribution learning. This formulation does not require that we preserve the original samples. Indeed, for most of the regime where distribution learning is not possible, we show that it is necessary to take an active decorrelation step that modifies the original dataset.

Chapter 5 provides algorithms and lower bounds for two simple instances of this problem where sample amplification can be done with only square root as many samples as is required for learning. Later work joint with Yanjun Han showed that this square root advantage of sample complexity is present is a very wide range of cases, covering both exponential families with bounded higher moments and certain sparse models.

These methods suggest a different way of looking at data augmentation and construction of synthetic datasets. The goal of these synthetic datasets is to make it easier for the downstream algorithm to extract information during statistical analysis or training of a machine learned model or classifier. To ensure the correctness of the downstream method, certain statistics (the sufficient statistic in the case of exponential families) and properties of the dataset must be preserved. If one is operating without unlimited data, this may require a synthetic dataset which is not a superset of the

original dataset. One must be cognizant of the correlations between new samples and old samples, and it may be necessary to modify the old samples to decorrelate them from the new samples.

The author hopes this encourages more theoretical study of the generation of synthetic datasets. Practitioners have found this to be an incredibly powerful tool, even for high stakes applications. There would be great value in theoretical guarantees for statistical statistical analysis based on synthetic data as well as better design principles for generating synthetic datasets.

Chapter 2

Hardness Of Minimum Risk Motion Planning

2.1 Introduction

Navigation under uncertainty is one of the most basic problems in robotics. While there are many methods to plan a trajectory between two points in a known environment with strong theoretical guarantees, few of them generalize to obstacles with locations estimated by noisy sensors. It has proven much harder to provide strong completeness, runtime, and optimality guarantees in this setting.

While some of the original work addressing planning under uncertainty was able to capture the additional richness of this problem by modeling it as a partially observable Markov decision process (POMDP) [32], it has proven difficult to solve POMDPs for complicated real world problems despite large advances in POMDP solvers [91, 122]. In fact, solving POMDPs is PSPACE-complete in the finite horizon and undecidable otherwise, suggesting that it likely not possible to find a general, efficient algorithm for solving POMDPs [105].

Luckily, navigating among uncertain obstacles is a significantly more restricted problem class than POMDPs, giving us hope that we might find an algorithm that is efficient in practice and gives strong theoretical guarantees such as completeness and safety.

Prior work proposed solving an approximation of the navigation under uncertainty problem [9, 10]. Instead of trying to compute a path that minimizes the true probability of collision under any distribution of obstacles, they propose solving a restricted problem where the obstacles are limited to the structured class of PGDF (Polytopes with Gaussian Distributed Faces) distributions and the collision probability is approximated using a shadow (the geometric equivalent of a confidence interval). While shadow bounds are inherently loose (they overestimate the probability of collision

when the obstacle is likely to be far away from the estimated location) they greatly decrease the computational complexity of bounding the probability of collision, since only space visited by the robot close to the obstacle affects the probability bound.

This prior work proposed the following question: Is there an efficient algorithm that, given a graph embedded in $\mathbb{R}^n$ and a set of obstacles, can find the path with minimal risk as computed via a shadow bound [10]? The cost function derived from the shadow approximation is only influenced by the portion of the trajectory close to the obstacle and has submodular structure with respect to the graph. The fact that similar approximations have worked well for motion planning, and the existence of efficient algorithms for certain classes of submodular minimization problems gave the hope that it might be possible to find an efficient algorithm for this problem as well.

While motion planning is hard in general, practical and efficient algorithms have proven very successful under some assumptions [93]. One such body of work are the sampling-based motion-planning methods. These algorithms often have the assumption that the problem can be split into two pieces: First use a practically (though often not worst-case) efficient method to generate a small graph that contains a solution; then use a standard, efficient graph search algorithm to find the solution in this graph. Algorithms based on this scheme have been successful even for high-dimensional planning problems for robots with many degrees of freedom.

There are several other classes of practically efficient algorithms (including grid based and optimization-based planners) that rely on the assumption that part of the problem may be solved much more efficiently in the average case than in the worst case. We discuss this further in the background section.

This paper answers the question of whether an efficient algorithm exists under these assumptions in the negative when an exact solution of FPTAS (Fully Polynomial Time Approximation Scheme) is required.

Theorem 1. *Safe path planning in the presence of uncertain obstacles in 2 dimensions is NP-hard.*

A more formal statement of this result is presented in Section 2.3. We prove this by reducing from MAXQHORNSAT using a construction based on and very similar to that used by prior work for the minimum constraint removal problem (MCR) [64]. We also show, via reduction from 3-SAT, that both safe path planning and MCR remain hard in three dimensions even when each obstacle overlaps with only a constant number of other obstacles, answering the question posed by [78]. Our first contribution is modifying the reduction to 2D MCR in prior work to instead reduce to the 2D safe path planning problem [64]. Our second contribution is presenting a new reduction from 3-SAT to a restricted version of safe path planning and MCR in 3D.

While the proofs in this paper use PGDF obstacles, we do not use any property that is unique to PGDF obstacles. The results essentially depend on only several properties of the model. This includes tail behavior (probability of collision being small far away from the obstacle), monotonicity (larger shadows have a larger probabilities), and the correlations between the events that nearby

locations are in collision. Since these properties are rather natural, we believe the same results would apply to many other reasonable classes of realistic models of estimated obstacles.

The proofs presented in this paper illuminate what makes this problem more difficult than the standard motion-planning problem with known obstacles. Searching for the minimum-risk path does not have a Markov-like structure. Unlike in the shortest-path problem on a graph, the risk of the second half of a trajectory is very much affected by the first half. This means that the problem is lacking the Bellman property, as identified in prior work [115]. The absence of a Markov-like property for the risk over the path has important ramifications for the complexity of the problem. In particular, the collision risk at different points along a trajectory can be highly correlated.

2.2 Background

Motion planning for robotics has been extensively studied in many different settings. One important high-level distinction between settings is whether the environment and state are known exactly or estimated.

2.2.1 Complexity in Motion Planning

The story of motion-planning algorithms in robotics has been one of walking the fine boundaries of complexity classes. On one hand, motion planning is PSPACE-hard in $\mathbb{R}^3$ and $\mathbb{R}^2$ with respect to the number of degrees of freedom of a robot (and thus dimension of its configuration space) [112, 61, 79]. However, while prior work on singly-exponential time (with respect to number of degrees of freedom) roadmaps leads to a polynomial-time algorithm when the number of degrees of freedom is fixed, a different set of algorithms is used in practice [26]. The robotics community has been able to find practically efficient methods that provide meaningful theoretical guarantees weaker than completeness (finding a solution if one exists). Sampling-based planners such as Rapidly-Exploring Random Trees (RRTs) and Probabilistic Roadmaps (PRMs) are both practically efficient and *probabilistically complete* under some regularity conditions [94, 93, 87, 89]. Given effective heuristics, graph-based planners have also proved efficient and provide *resolution completeness* [93].

Searching for optimal plans, as opposed to simply feasible plans, further increases the difficulty. In a classic result, prior work shows that the 3-d Shortest-Path Problem is NP-hard for a simple robot in terms of the number of obstacles [27]. This ruled out results similar to Canny's roadmap algorithm that showed fixed parameter tractability in the feasible motion planning case.

However, the community has been able to find practically efficient algorithms regardless of these worst-case results. A modified version of the original sampling-based algorithms allows them to return nearly optimal solutions in the limit and graph-based planning algorithms are able to provide bounds on the suboptimality of their solutions [86, 4].

Another motion-planning problem that lacks a Markov property is the minimum constraint

removal problem (MCR), where the objective is to find a path that collides with the fewest obstacles. This problem was shown to be NP-hard in Cartesian spaces of three dimensions, and shortly later, in two dimensions [77, 78, 64]. Further work improves on these results by showing that MCR remains hard when obstacles are restricted to line segments or axis-aligned rectangles [62]. Prior work observes that MCR is in P when obstacles are connected and non-overlapping, and the author suggests that the hardness seen in MCR is caused when an obstacle intersects with $O(n)$ other obstacles [78]. These works further pose the open question of whether MCR remains hard when each obstacle overlaps with only a constant number of other obstacles [64, 78, 62]. We then ask an analogous question: Is safe path planning in the presence of uncertain obstacles tractable when obstacles intersect only a constant number of other obstacles?

One form of this question in 3D is answered in the negative in a paper whose results are presented in this thesis [119]. These questions are stated more formally and answered in the negative in Section 2.3. The results in two dimensions and the results with bounded obstacle overlap were first presented by Axelrod and Shimanuki are presented in this chapter [120].

2.2.2 Planning under Uncertainty

While planning under uncertainty has been broadly studied in robotics, few methods have formal guarantees on solution quality and efficient runtime. We survey some of the related work below.

Many works assume some sort of uncertainty about the environment, but do not propose a model in which to rigorously quantify the uncertainty in the environment and provide guarantees about the success probability of the trajectory. Instead they often rely on heuristics that seem to provide the desired behavior.

One line of work focuses on uncertainty in the robot's position. Here the model of the robot itself is "inflated" before the collision checking, ensuring that any slight inaccuracy in the position estimate or tracking of the trajectory does not result in a collision.

Work that focuses on uncertainty in the environment sometimes does the exact opposite. They often inflate the occupied volume of the obstacle with a "shadow" and ensure that any planned trajectory avoids the shadow [83, 95].

A more general approach that handles either or both of localization and obstacle uncertainty is belief-space planning. Belief space is the set of all possible beliefs about or probability distributions over the current state. Belief-space planning converts the uncertain domain in state space to belief space, then plans in belief space using trees or control systems [109, 25, 108].

Another line of work uses synthesis techniques to construct a trajectory intended to be safe by construction. If the system is modeled as a Markov decision process with discrete states, a safe plan can be found using techniques from formal verification [56, 66]. Other authors have used techniques from Signal Temporal Logic combined with an explicitly modeled uncertainty to generate plans that are heuristically safe [114].

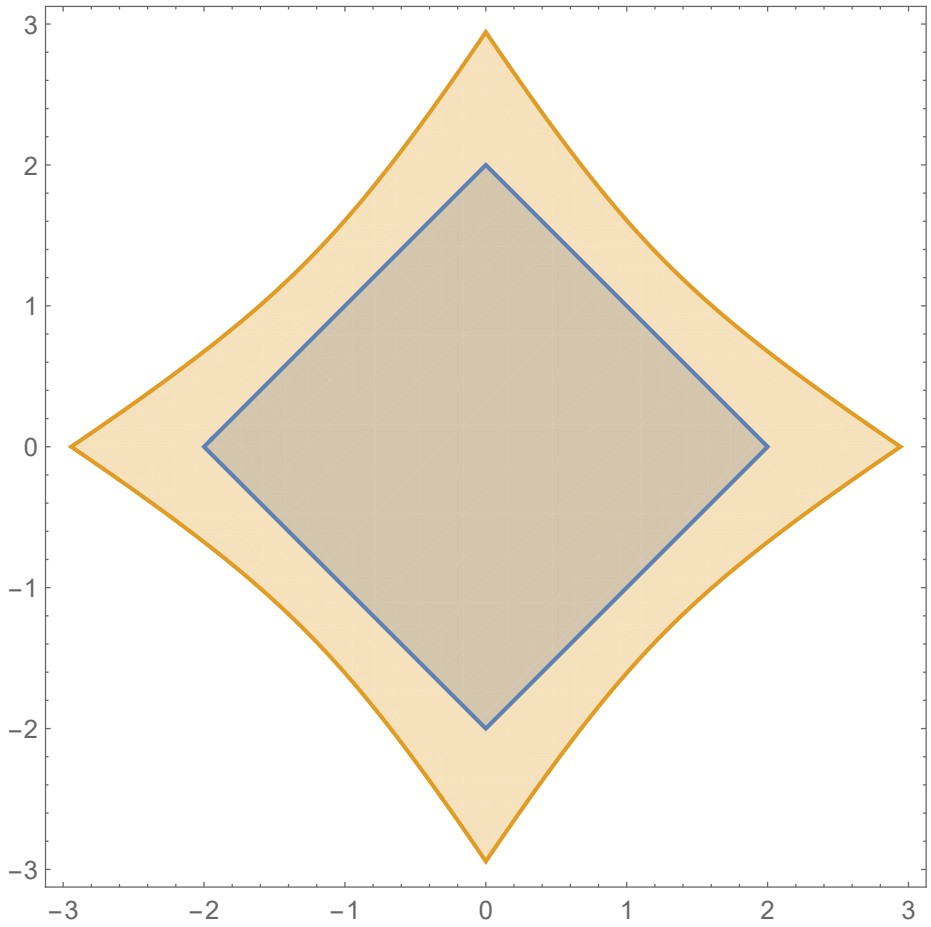

Figure 2.1: The orange set is a shadow of the obstacle. The blue set is the obstacle represented by the mean parameters.

Other work by applies an approximate minimum constraint removal algorithm to motion planning under obstacle uncertainty by randomly sampling many draws for each obstacle and finding the path that intersects with the fewest samples. With this approach, he demonstrates low runtime and error on average although with poor worst case performance [78, 77].

In previous work, formalized the notion of a shadow in a way that allowed the construction of an efficient algorithm to bound the probability that a trajectory will collide with the estimated obstacles [9, 10].

We can now define a shadow rigorously:

Definition 1 (ϵ-shadow)**.** *A set $S \subseteq \mathbb{R}^d$ is an ϵ-shadow of a random obstacle $O \subseteq \mathbb{R}^d$ if $Pr[O \subseteq S] \geq 1 - \epsilon$.*

Shadows are important because they allow for an efficient method to upper-bound the probability of collision. If there exists an $\epsilon-$shadow of an obstacle that does not intersect a given trajectory's swept volume, then the probability of the obstacle intersecting with the trajectory is at most ϵ. An example of a shadow for an obstacle is shown in figure 2.1.

2.3 Preliminaries

2.3.1 Notation

In this section we will cover definitions and notation conventions that will be used in this paper. A vector will be marked in bold as in $\mathbf{u}$, in contrast to a scalar u. $\wedge$, $\vee$, and $\neg$ are the logical AND, OR, and NOT operators, respectively. The power set (set of all subsets) of S is denoted by $\mathcal{P}(S)$. A function mapping into the power set of $\mathbb{R}^n$ outputs subsets of $\mathbb{R}^n$. We will use e_i to denote the ith standard basis vector.

2.3.2 Random Obstacle Model

In order to attempt to provide formal non-collision guarantees one must first model the uncertainty in the environment. At a high level we assume that each episode of the robot's interaction happens in the following sequence:

1. A distribution of obstacles is fixed and known to the robot. Usually this is the conditional distribution for the obstacles given the sensor observations.

2. A set of obstacles is now drawn from this distribution. These obstacles now remain static for the duration of the episode.

3. The robot computes, commits to and executes a trajectory.

4. The probability of collision in question is exactly the probability that this trajectory collides with at least one of the obstacles.

It is important that the obstacle distribution captures the fact that collision probabilities in different locations can be correlated. Consider the following toy example where a range sensor reports that an obstacle is 10 meters in front of the robot. At time $t = 1$ the robot drives forward 10 meters and then at $t = 2$ drives backwards 2 meters. If the robot did not crash at time $t = 1$, it is unlikely to collide at time $t = 2$! Using a more realistic model that captures correlations allows systems to be both safer and less conservative.

In this work we restrict ourselves to polytopes with Gaussian-distributed faces (PGDFs)—a model that is able to capture such correlations [9]. Under the PGDF assumption, obstacles are the intersections of halfspaces with parameters drawn from multivariate normal distributions. More formally a PGDF $O \subset \mathbb{R}^n$ is $O = \bigcap^{i} {\alpha_\mathbf{i}}^T \mathbf{x} \leq 0$, where $\alpha_\mathbf{i} \sim \mathcal{N}(\mu_\mathbf{i}, \mathbf{\Sigma_i})$. We can use homogeneous coordinates to create obstacles not centered about the origin.

One reason that PGDF obstacles are important is that we have methods of computing shadows for PGDF obstacles efficiently [9].

We note that this formulation differs from the notion of "risk-zones" evaluated by prior work, where the cost of a trajectory is proportional to the amount of time spent within a risk-zone [116, 115]. These problems share the lack of an optimal substructure—the subpaths of an optimal path are not necessarily optimal. Prior work provides a generalization of Dijkstra's algorithm that finds minimum-risk plans in their domain efficiently, but as we will show, there are no such techniques for our problem [115].

2.3.3 Algorithmic Question

Now that we have defined shadows and PGDF obstacles, we can define what it means for a path to be safe. Suppose the robot and obstacles exist in $\mathbb{R}^d$ (d is usually 2 or 3). This is commonly referred to as the task space. Furthermore, suppose the configuration space of the robot is parametrized in $\mathbb{R}^k$ (usually corresponding to the k degrees of freedom of the robot).

Since planning usually happens in the robot's configuration space, but the obstacles are in task space, we need to be able to convert between the two.

Definition 2 (Embedding Map)**.** *A function $f : \mathbb{R}^k \to \mathcal{P}(\mathbb{R}^d)$ is an embedding map if it maps robot configurations into the subset of $\mathbb{R}^d$ that is occupied by the robot at that configuration.*

The embedding map can usually be constructed by combining the forward kinematics and robot model.

Definition 3. *A configuration space trajectory $\tau : [0, 1] \to \mathbb{R}^k$ is a map from a "time" index into the trajectory to the robot configuration at that point in the trajectory.*

Definition 4. *A task-space trajectory $\tau' : [0,1] \to \mathcal{P}(\mathbb{R}^d)$ is defined as the map between an index into the trajectory and the space occupied by the robot at that point in the trajectory.*

Alternatively, if given a configuration space trajectory τ, $\tau'(t) = f(\tau(t))$ where f is the embedding map.

For the rest of the paper we will only concern ourselves with task-space trajectories, noting that it is easy to go from a configuration space trajectory to a task-space trajectory using the embedding map.

Definition 5. *The swept volume X of a task-space trajectory τ is the set of task-space points touched by the robot while executing trajectory τ.*

Said differently, $X = \bigcup_{t \in [0,1]} \tau(t)$.

This allows us to formally define what it means for a trajectory to be safe.

Definition 6 (ϵ-safe trajectory)**.** *Given a joint distributions over random obstacles, a task-space trajectory is ϵ-safe if the corresponding swept volume has at* most *ϵ probability of intersecting at least one obstacle.*

This leads to the following algorithmic question, of finding safe plans for a known distribution of PGDF obstacles.

Problem 1 (ϵ-safe Planning Problem)**.** *Given the parameters of PGDF distributions for each obstacle and initial and end points $\mathbf{s}, \mathbf{t}$ in configuration space, find an ϵ-safe trajectory from $\mathbf{s}$ to $\mathbf{t}$.*

Note that there exists reductions between the safe planning problem and finding a path that minimizes the risk of collision. Since the probability ϵ is confined to $[0,1]$, a binary search over ϵ yields an efficient algorithm that can approximately compute the minimum risk given an $\epsilon-$safe planner. For convenience, our proofs will consider the approximate minimum-risk planning problem, though the construction applies directly to $\epsilon-$safe planning as well.

Problem 2 ($(1+\alpha)$-approximate minimum-risk planning problem)**.** *Given the parameters of PGDF distributions for each obstacle and initial and end points $\mathbf{s}, \mathbf{t}$ in configuration space, return a $((1+\alpha)\epsilon_*)$-safe trajectory from $\mathbf{s}$ to $\mathbf{t}$, where ϵ_* is the minimum ϵ for which an ϵ-safe trajectory exists.*

2.3.4 Graph Restriction

We start by considering the class of motion-planning algorithms that first construct a graph embedded in the robot's configuration space, and then run a graph-search algorithm to find a path within the graph. This class of algorithms has been shown to be practical in the known environment by sampling-based planners such as PRM and RRG. Conditioned on there being a nonzero probability of sampling a solution, these algorithm are guaranteed to find a collision-free path with probability approaching 1 as the number of iterations approaches infinity [94, 86].

More formally, this condition can be articulated as the existence of a path in the δ-interior of the free space X_{free}.

Definition 7 (δ-interior [86])**.** *A state $\mathbf{x} \in X_{free}$ is in the δ-interior of X_{free} if the closed ball of radius δ around $\mathbf{x}$ lies entirely in X_{free}. The a set is in the δ-interior of X_{free} if every point in the set is in the δ-interior of X_{free}*

This condition is necessary because it guarantees that finding a plan does not require waiting for a zero probability event. However this formulation does not extend well to the domain with uncertain obstacles; there is no concept of "free space" because the locations of the obstacles are not known. Instead we will use the equivalent view of inflating the path instead of shrinking the free space.

Definition 8. *For $\delta > 0$, the δ-inflation of the set $\mathbf{X}$ is the set $Y = \bigcap_{\mathbf{x} \in X} \{\mathbf{y} \mid d(\mathbf{x}, \mathbf{y}) \leq \delta\}$.*

For the proofs in this paper, the particular choice of metric d does not matter. We note that in the deterministic setting, if a trajectory is in the δ-interior of X_{free}, then the δ-inflation of the trajectory is entirely in X_{free}. This allows us to consider problems with the following regularity condition: there exists a δ-inflated task-space trajectory that has a low risk of collision.

Definition 9 (ϵ-safe δ-inflated task-space trajectory)**.** *A task-space trajectory is an ϵ-safe δ-inflated trajectory if its δ-inflation intersects an obstacle with probability at most ϵ.*

We want to find an algorithm that satisfies the completeness and safety guarantees defined below.

Definition 10 (Probabilistically Complete $(1+\alpha)$-approximate Safe Planning Algorithm)**.** *A planning algorithm takes a set of PGDF obstacles O, a start state $\mathbf{s}$, and a goal state $\mathbf{t}$ as input and generates a path as output. A planning algorithm is probabilistically complete and $(1+\alpha)$-approximate safe if, with n samples, the probability that it finds a $((1+\alpha)\epsilon_*)$-safe trajectory approaches 1 as n approaches ∞, where ϵ_* is the minimum ϵ for which an ϵ-safe trajectory exists.*

We also consider a special case where each obstacle overlaps with only a constant number of other obstacles.

Definition 11. *Two obstacles O_i, O_j overlap if there exists any point in space that both obstacles have a significant probability of intersecting. That is, there exists $\mathbf{x} \in \mathbb{R}^d$ such that $Pr[\mathbf{x} \in O_i] \geq \epsilon$ and $Pr[\mathbf{x} \in O_j] \geq \epsilon$ for $\epsilon = \frac{\epsilon_*}{|O|}$.*

Definition 12 (Probabilistically Complete κ-overlap $(1+\alpha)$-approximate Safe Planning Algorithm)**.** *A probabilistically complete κ-overlap $(1+\alpha)$-approximate safe planning algorithm is a planning algorithm that is probabilistically complete and $(1+\alpha)$-approximate safe for cases where the number of other obstacles that each obstacle overlaps with is at most κ.*

Prior work provides an extension of the RRT algorithm to the probabilistic domain using the shadow approximation [9]. The uniqueness of paths between any two vertices in a tree makes finding the optimal (restricted to the tree) path trivial. However, while the paths it generates are indeed safe, the algorithm is not probabilistically complete.

However, the following extension of the RRG algorithm is probabilistically complete [8].

Algorithm 1 SAFE_RRG

Input: End points $\mathbf{s}, \mathbf{t} \in \mathbb{R}^d$, set of PGDF obstacles O, and number of samples n.

Output: A $((1+\alpha)\epsilon_*)$-safe trajectory from $\mathbf{s}$ to $\mathbf{t}$, where ϵ_* is the minimum ϵ for which an ϵ-safe trajectory exists.

1: G = CONSTRUCT_RRG($\mathbf{s}$, $\mathbf{t}$, n)

2: **return** GRAPH_SEARCH($G, O, \mathbf{s}, \mathbf{t}$)

We note that as n increases, the probability that there is a sample near any given point x in the space approaches 1. Here, GRAPH_SEARCH is a $(1+\alpha)$-approximate safe graph-search algorithm as defined below.

Definition 13. *A $(1+\alpha)$-approximate safe graph-search algorithm is a procedure $\phi(G, O, \mathbf{s}, \mathbf{t})$, where G is a graph, O is a set of PGDF obstacles, and $\mathbf{s}$ and $\mathbf{t}$ are the start and end nodes in G, respectively. It returns a $((1+\alpha)\epsilon_*)$-safe trajectory in G, where ϵ_* is the minimum ϵ for which an ϵ-safe trajectory exists.*

Theorem ([8])**.** *SAFE_RRG is probabilistically complete and $(1+\alpha)$-approximate safe as long as GRAPH_SEARCH is complete and $(1+\alpha)$-approximate safe.*

However, no graph-search procedure, beyond the naïve, exponential-time search procedure, is provided [8]. This means that, while the probability of success increases with more samples, the worst-case running time is exponential. Sampling-based motion-planning algorithms work in practice in the known environment because efficient graph-search algorithms can quickly find collision-free paths within a graph. In order for the *SAFE_RRG* class of algorithms to be practical, we would need a corresponding graph-search algorithm in the probabilistic domain. Because the cost of a path depends on what set of shadows it intersects, the state space of the graph search is not just the current node but also includes the accumulated risk incurred due to each obstacle. This means that the typical approaches for searching graphs with known obstacles, which make use of dynamic programming, cannot be applied in the same manner to graphs with unknown obstacles.

2.4 Results

Unfortunately, as will be shown in the remainder of this paper, Problem 1 and Problem 2 are NP-HARD with respect to $n = \Theta(|G| + |O|)$, the size of the input, even with a point robot in two

dimensions and given a graph containing the solution. We note that while our results preclude an FPTAS, it does not preclude all approximation algorithms.

Theorem 2. *Unless $P = NP$, there is no $(1+\alpha)$-approximate ϵ-safe graph-search algorithm that runs in $POLY(n)$, time when restricted to graphs that embed in $\mathbb{R}^d$, $d = O(k)$ where k is the number of obstacles and $\alpha = \Theta(\frac{1}{n})$.*

We can strengthen this result to show that the minimum-risk planning problem is hard in general, that is, even when not restricted to a graph.

Theorem 3. *The $(1+\alpha)$-approximate minimum-risk planning problem is NP-hard. That is, unless $P = NP$, there is no $(1+\alpha)$-approximate Safe Planning Algorithm for $\mathbb{R}^2$ that runs in $POLY(n)$ when $\alpha = \Theta\left(\frac{1}{n^2}\right)$, even when provided a graph containing the solution.*

We show Theorem 2 and Theorem 3 by constructing a minimum risk planning problem in 2 dimensions which solves MAXQHORNSAT (an NP-complete problem). The proof follows the outline of the MCR hardness proof presented in prior work [64]. The main contribution of the work in this theorem is the connection to the minimum risk planning problem and the construction of the uncertain obstacles. However, the construction is not natural in the sense that certain constructed obstacles are used to correlate collision probabilities in disparate parts of the space. Some obstacles will be split by others and there is a high degree of overlap. We also show that the problem remains hard in 3D even when each obstacle overlaps with only a constant number of other obstacles.

Theorem 4. *The κ-overlap $(1+\alpha)$-approximate minimum-risk planning problem is NP-hard for $\kappa = O(1)$ in 3 dimensions. That is, unless $P = NP$, there is no κ-overlap $(1+\alpha)$-approximate Safe Planning Algorithm for $\mathbb{R}^3$ that runs in $POLY(n)$ when $\alpha = \Theta(\frac{1}{n^2})$, even when provided a graph containing the solution and when restricted to cases where each obstacle overlaps with at most κ other obstacles.*

Furthermore, the proof can be extended to apply to MCR as well.

Theorem 5. *The κ-overlap minimum constraint removal problem is NP-hard for $\kappa = O(1)$ in 3 dimensions.*

We show Theorems 4 and 5 by constructing a minimum risk planning problem and related MCR problem in 3 dimensions which solves 3-SAT. We believe the construction presented here is of particular value because it is very natural – the construction is simple and does not require that any obstacle be immediately adjacent to more than a constant number of other obstacles.

2.5 Hardness Results in $\mathbb{R}^2$

2.5.1 Maximum Quadratic Horn Clause Satisfiability

Maximum Quadratic Horn Clause Satisfiability (MAXQHORNSAT) is an NP-complete problem whose input is a Boolean formula given in conjunctive normal form. It consists of the intersection of many clauses, each consisting of at most two literals (i.e. is quadratic), and each clause contains at most one positive literal (i.e. is a Horn clause) [82]. In other words, it is of the form $((x_0 \vee \neg x_1) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_2) \wedge ...)$. While QHORNSAT, the decision problem of determining the satisfiability of the input formula, is in P [65], MAXQHORNSAT, the problem of determining the maximum number of clauses that can be satisfied, is NP-hard [82]. MAXQHORNSAT was used in prior work to show that minimum constraint removal (MCR), a similar problem, is NP-hard [64]. Our reduction is based on that used in said work and will use a similar construction modified to apply to the safe planning problem [64]. We will consider a formula with n_v variables, n_n clauses with two negative literals, n_p clauses with one positive literal and one negative literal, and n_s clauses with only a single literal. We also define the total number of clauses $n_c = n_s + n_p + n_n$ and the total size of the problem $n = n_v + n_c$.

2.5.2 Proof Outline

We prove Theorem 2 using a reduction from MAXQHORNSAT. Given a MAXQHORNSAT instance, we construct an $\mathbb{R}^2$ $(1+\alpha)$-approximate minimum-risk planning problem and graph containing the solution. Our construction will have two kinds of obstacles. High-risk obstacles will induce a sufficiently high risk that any "reasonable" solution will go through the minimal number of these obstacles. Low-risk obstacles will affect the collision probability much less and will be used to count how many clauses are satisfied. The sum of the potential risk of all low-risk obstacles will be less than that of a single high-risk obstacle. This means the optimal solution will always choose to avoid a high-risk obstacle whenever possible, regardless of how many low-risk obstacles it must pass in order to do so. This creates a measurable gap between the optimal solution and the next best one. The construction can be split into three pieces. Figure 2.2 describes the spatial relationship between the pieces described in the following caption.

1. Construct a portion of the graph for the algorithm to assign every variable by taking either the *left* or *right* branch, corresponding to setting the value of each variable to *true* or *false*, respectively. A high-risk obstacle corresponding to each branch will ensure that the optimal path only goes down one of the branches.

2. Construct a portion of the graph for the algorithm to select a literal from each clause to try to satisfy by taking either the *left* or *right* branch, corresponding to selecting the first or second

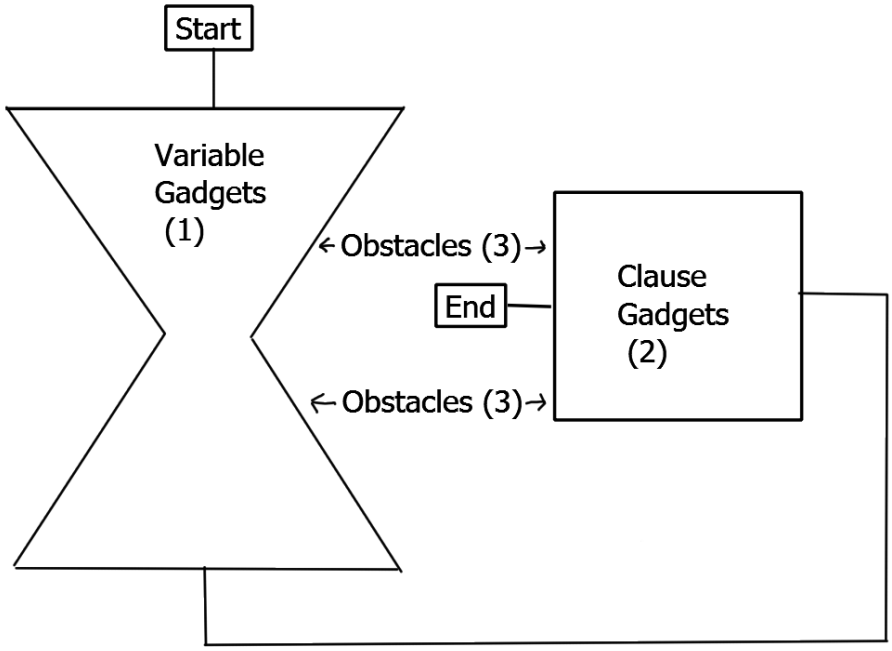

Figure 2.2: The spatial relationship between the different gadgets.

literal, respectively. Choosing a branch which corresponds to a different assignment than in the first part would result in passing by an extra high-risk obstacle.

3. Construct low-risk obstacles such that there will be additional collision risk each time the selected literal is not satisfied by the chosen variable assignment.

The solution to this planning problem can then be transformed into a solution to the original MAXQHORNSAT instance in polynomial time (via observing which nodes were visited), demonstrating that $(1+\alpha)$-approximate safe graph search is at least as hard as MAXQHORNSAT.

2.5.3 Obstacle Templates

Throughout this reduction we will construct a number of obstacles using a few common templates, so for simplicity of notation we will define a few parameterized types of obstacles. These obstacle templates will fall into two categories based on how much we want them to affect the cost of a trajectory: low-risk obstacles and high-risk obstacles. In figures, high-risk obstacles will be denoted in blue and low-risk obstacles will be denoted in green.

The first kind of obstacle, shown in Figure 2.3, is a low-risk obstacle parameterized by a line segment $(\mathbf{u}, \mathbf{v})$. It is a long, thin obstacle that runs parallel to $(\mathbf{u}, \mathbf{v})$ and has one edge with uncertain

position such that there is a risk of collision with points along $(\mathbf{u}, \mathbf{v})$.

$$\widehat{C}(\mathbf{u}, \mathbf{v}, \alpha) = \left\{ \mathbf{x} \;\middle|\; \begin{array}{l} \mathbf{x} \in \mathbb{R}^2 \\ \frac{1}{|\mathbf{v}-\mathbf{u}|}(\mathbf{v}-\mathbf{u})^T(\mathbf{x}-\mathbf{u}) \geq -\epsilon_C \\ \frac{1}{|\mathbf{u}-\mathbf{v}|}(\mathbf{u}-\mathbf{v})^T(\mathbf{x}-\mathbf{v}) \geq -\epsilon_C \\ \alpha \leq \frac{1}{|\mathbf{v}-\mathbf{u}|}(R_{\frac{\pi}{2}}(\mathbf{v}-\mathbf{u}))^T(\mathbf{x}-\mathbf{u}) \leq \epsilon_C \end{array} \right\}$$

for small constant ϵ_C, and where R_θ is the 2D rotation matrix for a clockwise rotation with angle θ. Note that $\widehat{C}$ defines a rectangular obstacle, where the position of one edge is parameterized by α. Then we can define a distribution over such obstacles

$$C(\mathbf{u}, \mathbf{v}) = \widehat{C}(\mathbf{u}, \mathbf{v}, \alpha) \text{ where } \alpha \sim \mathcal{N}(\mu_C, \sigma_C^2)$$

for small constants μ_C and σ_C. It guarantees that any point within distance ϵ_C of $(\mathbf{u}, \mathbf{v})$ has a risk of collision of at most $r_c = \Phi\left(-\frac{1}{\sigma_C}(\mu_C - \epsilon_C)\right)$ and at least $r'_c = \Phi\left(-\frac{1}{\sigma_C}(\mu_C + \epsilon_C)\right)$, and any point with distance further than $z_C\epsilon_C$ from $(\mathbf{u}, \mathbf{v})$ for some constant $z_C > 1$ has a risk of collision of at most $r_f = \Phi\left(-\frac{1}{\sigma_C}(\mu_C + \frac{1}{\sqrt{2}}z_C\epsilon_C)\right)$ (lower bounded by $r'_f = 0$ because risk becomes arbitrarily small as distance from the obstacle increases), where Φ is the cumulative distribution function of the standard normal distribution. Note that given some value of ϵ_C we can set μ_C, σ_C, and z_C to achieve any desired values of r_c, r'_c, and r_f. In particular, we can make r'_c/r_c arbitrarily close to 1 by decreasing ϵ_B, and make r_f/r_c arbitrarily close to 0 by increasing z_B.

The next kind of obstacle is identical to the line-segment obstacle defined above and shown in Figure 2.3 except it is a high-risk obstacle, so it has a higher probability of intersecting with points near the line segment (so it can be thought of as having a higher weight in terms of affecting the risk of a path).

$$\widehat{V}(\mathbf{u}, \mathbf{v}) = \left\{ \mathbf{x} \;\middle|\; \begin{array}{l} x \in \mathbb{R}^2 \\ \frac{1}{|\mathbf{v}-\mathbf{u}|}(\mathbf{v}-\mathbf{u})^T(\mathbf{x}-\mathbf{u}) \geq -\epsilon_V \\ \frac{1}{|\mathbf{u}-\mathbf{v}|}(\mathbf{u}-\mathbf{v})^T(\mathbf{x}-\mathbf{v}) \geq -\epsilon_V \\ \alpha \leq \frac{1}{|\mathbf{v}-\mathbf{u}|}(R_{\frac{\pi}{2}}(\mathbf{v}-\mathbf{u}))^T(\mathbf{x}-\mathbf{u}) \leq \epsilon_V \\ \alpha \sim \mathcal{N}(\mu_V, \sigma_V^2) \end{array} \right\}$$

for small constant ϵ_V. Note that $\widehat{V}$ defines a rectangular obstacle, where the position of one edge is parameterized by α. Then we can define a distribution over such obstacles

$$V(\mathbf{u}, \mathbf{v}) = \widehat{V}(\mathbf{u}, \mathbf{v}, \alpha) \text{ where } \alpha \sim \mathcal{N}(\mu_V, \sigma_V^2)$$

for small constants μ_V and σ_V. As before, it guarantees that any point within distance ϵ_V of $(\mathbf{u}, \mathbf{v})$

Figure 2.3: The illustrations of the obstacle template $C(u, v)$. Note that it is a PGDF obstacle with the height of the obstacle being the only part that is random. Figure 2.3a illustrates a probability density function of collision. The distance between u and the top line or between v and the top line is ϵ_C. Figure 2.3b illustrates several obstacles drawn from the obstacle template. Note that if u is in collision so is v and vice versa.

has a risk of collision of at most $r_{Vc} = \Phi(-\frac{1}{\sigma_V}(\mu_V - \epsilon_V))$ and at least $r'_{Vc} = \Phi(-\frac{1}{\sigma_V}(\mu_V + \epsilon_V))$, and any point with distance further than $z_V\epsilon_V$ from $(\mathbf{u}, \mathbf{v})$ for some constant $z_V > 1$ has a risk of collision of at most $r_{Vf} = \Phi(-\frac{1}{\sigma_V}(\mu_V + \frac{1}{\sqrt{2}}z_V\epsilon_V))$ (lower bounded by $r'_{Vf} = 0$). Again, given some value of ϵ_V we can set μ_V, σ_V, and z_V to achieve any desired values of r_{Vc}, r'_{Vc}, and r_{Vf}, and so let us set the constants such that

$$r_{Vc} = 5n_c r_c$$
$$r'_{Vc} = 5n_c r'_c$$
$$r_{Vf} = 5n_c r_f.$$

The final type of obstacle, shown in Figure 2.4, is another low-risk obstacle parameterized by a single point $\mathbf{q}$ and a horizontal direction h (either 1 or -1, corresponding to right and left, respectively). It is a small obstacle that sits to the side of $\mathbf{q}$ in the direction specified by h and has one edge with uncertain position such that there is risk of collision with $\mathbf{q}$ and points nearby q.

$$\widehat{B}(\mathbf{q}, h, \alpha) = \left\{ x \;\middle|\; \begin{array}{l} \mathbf{x} \in \mathbb{R}^2 \\ \mathbf{e_2}^T\mathbf{q} - \epsilon_B \leq \mathbf{e_2}^T\mathbf{x} \leq \mathbf{e_2}^T\mathbf{q} + \epsilon_B \\ (\mathbf{e_1}^T\mathbf{q} + \alpha)h \leq \mathbf{e_1}^T\mathbf{x}h \leq (\mathbf{e_1}^T\mathbf{q} + \epsilon_B)h \end{array} \right\}$$

for small constant ϵ_B (also recall that $\mathbf{e_i}$ refers to the ith standard basis vector). Note that $\widehat{B}$ defines a rectangular obstacle, where the position of one edge is parameterized by α. Then we can

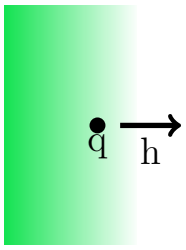

Figure 2.4: An example of the $B(q, h, \alpha)$ obstacle template. The distance between q and the left edge of the obstacle is ϵ_B

define a distribution over such obstacles

$$B(\mathbf{q}, h) = \widehat{B}(\mathbf{q}, h, \alpha) \text{ where } \alpha \sim \mathcal{N}(\mu_B, \sigma_B^2)$$

for small constants μ_B and σ_B. It guarantees that any point within a ball of radius ϵ_B around $\mathbf{q}$ has a risk of collision of at most $r_c = \Phi\left(-\frac{1}{\sigma_B}(\mu_B - \epsilon_B)\right)$ and at least $r'_c = \Phi\left(-\frac{1}{\sigma_B}(\mu_B + \epsilon_B)\right)$, and any point outside a ball of radius $z_B\epsilon_B$ around $\mathbf{q}$ for some constant $z_B > 1$ has a risk of collision of at most $r_f = \Phi\left(-\frac{1}{\sigma_B}(\mu_B + \frac{1}{\sqrt{2}}z_B\epsilon_B)\right)$. As before, note that given some value of ϵ_B we can set μ_B, σ_B, and z_B to achieve any desired values of r_c, r'_c, and r_f. Since these will also be low-risk obstacles, let us say that they will take on the same risk values as with the obstacles defined by C above.

2.5.4 Variable Gadgets

First, for each variable i in the MAXQHORNSAT problem, we construct a section of the graph where the choice of path corresponds to choosing either a true or false value of variable i. We illustrate this in Figure 2.5 and formalize this below. For variable i we construct vertices

$$\begin{aligned}
\mathbf{u_i^v} &= (0, 3i) \\
\mathbf{a_i^v} &= (-(n_v - i) - 1, 3i + 1) \\
\mathbf{b_i^v} &= (n_v - i + 1, 3i + 1) \\
\mathbf{v_i^v} &= (0, 3i + 2)
\end{aligned}$$

and edges

$$\begin{aligned}
&(\mathbf{u_i^v}, \mathbf{a_i^v}) \\
&(\mathbf{u_i^v}, \mathbf{b_i^v}) \\
&(\mathbf{a_i^v}, \mathbf{v_i^v}) \\
&(\mathbf{b_i^v}, \mathbf{v_i^v}).
\end{aligned}$$

Each pair of consecutive loops is connected by an additional edge $(\mathbf{v_i^v}, \mathbf{u_{i+1}^v})$ for all i..

This entire set of variable gadgets will be mirrored at the bottom, with the positive-negative clause gadgets (see Section 2.5.5 between them. The bottom set of variable gadgets will be needed for the negative-negative clause gadgets (see Section 2.5.5). Then for each variable i we construct a mirrored loop with vertices

$$\begin{aligned}
\mathbf{u_i^{v\prime}} &= (0, 7n_v + 3n_p - 3i) \\
\mathbf{a_i^{v\prime}} &= (-(n_v - i) - 1, 7n_v + 3n_p - 3i + 1) \\
\mathbf{b_i^{v\prime}} &= (n_v - i + 1, 7n_v + 3n_p - 3i + 1) \\
\mathbf{v_i^{v\prime}} &= (0, 7n_v + 3n_p - 3i + 2)
\end{aligned}$$

and edges

$$\begin{aligned}
&(\mathbf{u_i^{v\prime}}, \mathbf{a_i^{v\prime}}) \\
&(\mathbf{u_i^{v\prime}}, \mathbf{b_i^{v\prime}}) \\
&(\mathbf{a_i^{v\prime}}, \mathbf{v_i^{v\prime}}) \\
&(\mathbf{b_i^{v\prime}}, \mathbf{v_i^{v\prime}}) .
\end{aligned}$$

Likewise, consecutive loops are connected by an additional edge $(\mathbf{v_i^{v\prime}}, \mathbf{u_{i+1}^{v\prime}})$ for all i.

In order to ensure that the resulting path selects the same variable assignment in the top set and the mirrored set, for each variable i, we construct an obstacle that has risk of colliding with the *true* path in both versions, and another obstacle that has a risk of colliding with the *false* path in both versions. These obstacles are given by

$$\begin{aligned}
&V(\mathbf{a_i^{v}}, \mathbf{a_i^{v\prime}}) \\
&V(\mathbf{b_i^{v\prime}}, \mathbf{b_i^{v}}).
\end{aligned}$$

Because the collision risks are correlated, selecting the same value in the bottom gadget as in the top gadget will incur no additional risk of colliding with the corresponding obstacle, but selecting a different value will incur the additional risk of colliding with the other obstacle.

2.5.5 Clause Gadgets

There are three types of clause gadgets, each of which will need to be handled separately: single literals, each with only a single positive or negative literal, positive-negative clauses, each with one positive literal and one negative literal, and negative-negative clauses, each with two negative literals.

Single Literal

This first case is the simplest, as there is no choice to make about which literal to satisfy, so we do not need to add any additional components to the graph. For each single-literal clause j, let c_j^s

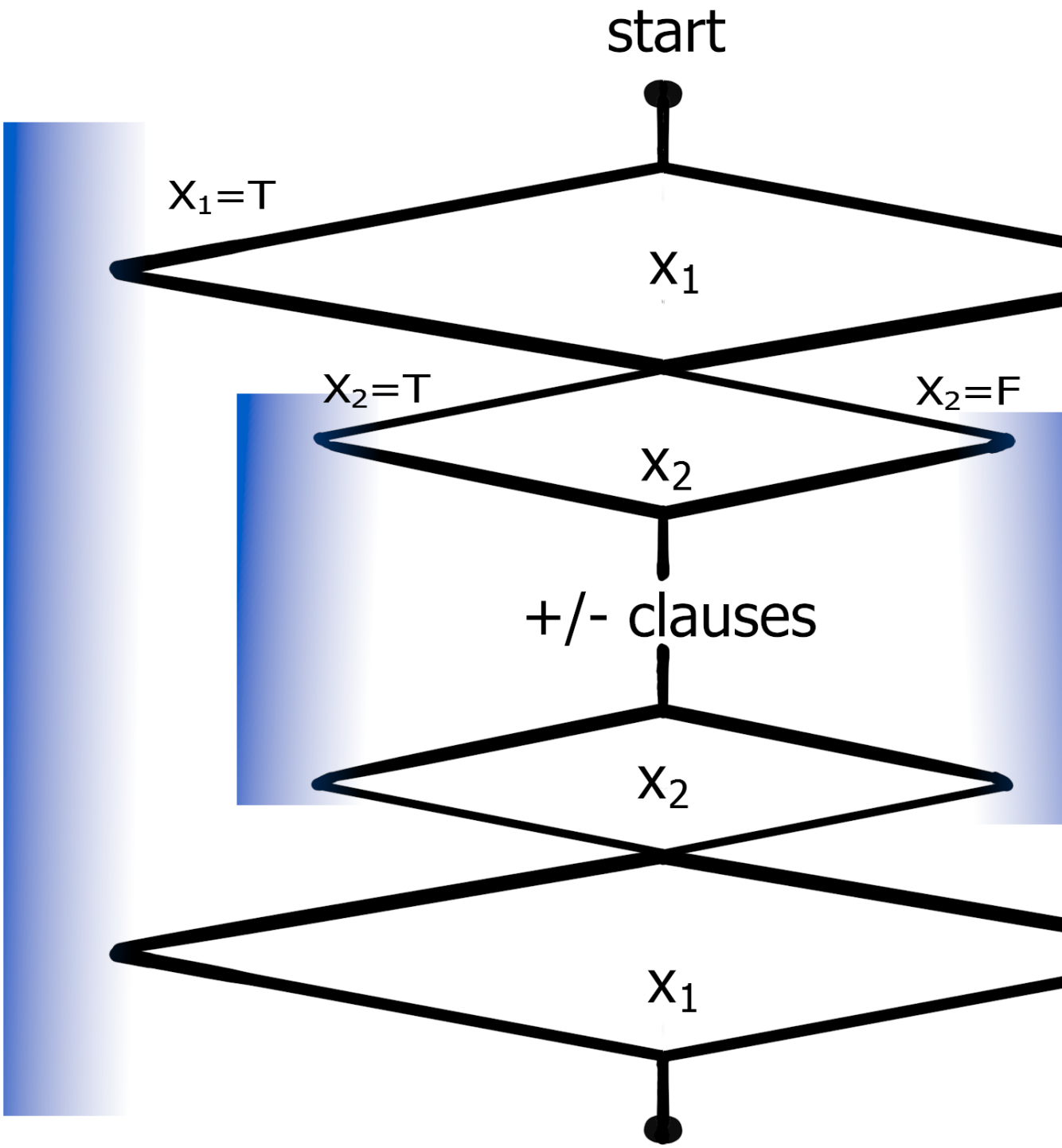

Figure 2.5: The variable gadget loops (solid lines) and obstacles (gradient-shaded rectangles). A path assigns a *true* or *false* value to each variable by selecting a branch through the variable gadget loop to traverse. It must also select the same variable assignment in the mirrored gadgets at the bottom in order to avoid additional collision risk. Notice that only high-risk obstacles are used in these gadgets, as they are constructed with the V template. There are two mirrored copies of each loop, with the positive-negative clause (or $+/-$ clause) gadgets in the center. The positive-negative clause gadgets will be constructed in Section 2.5.5. Also notice how loops closer to the center have

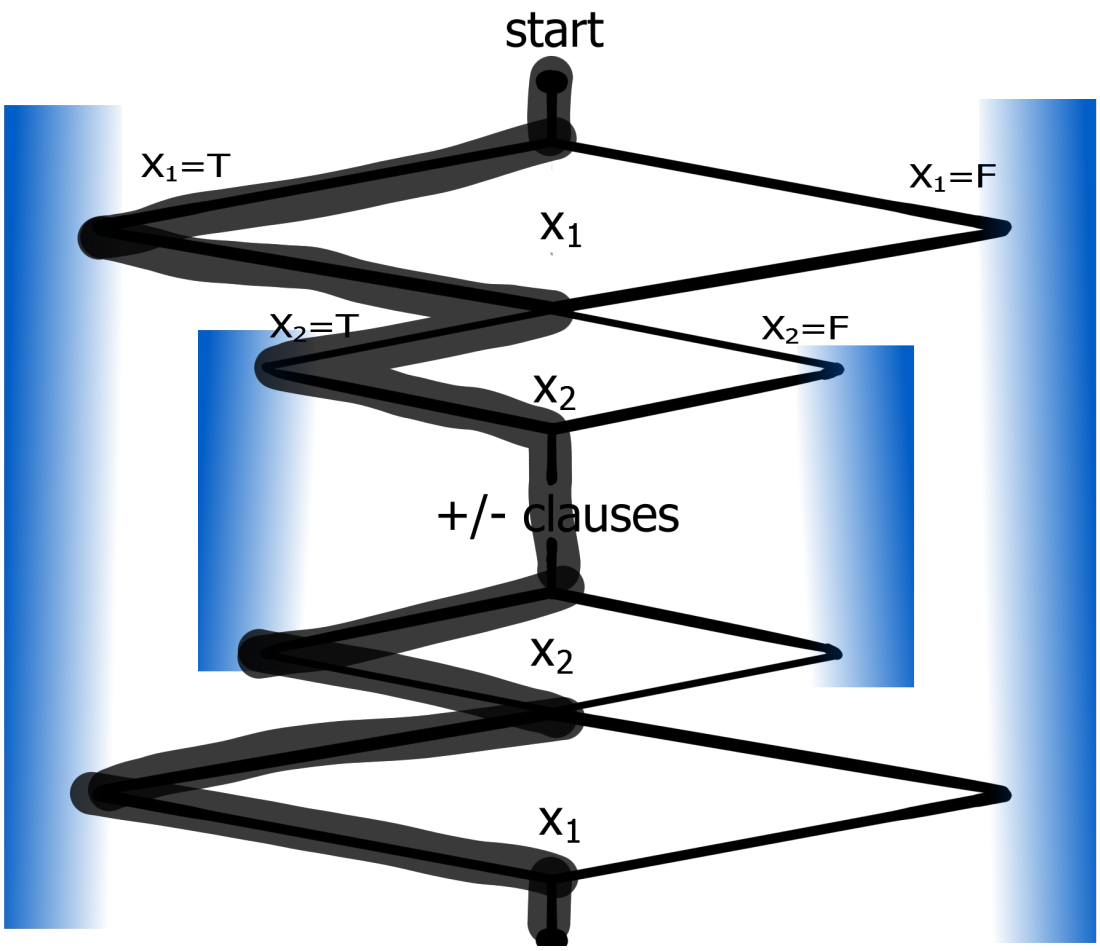

Figure 2.6: An example of a trajectory corresponding to $X_1 = T, X_2 = T$. Note that the top and bottom clauses match, otherwise excessive risk would be incurred.

denote the variable specified by the literal. Then we construct obstacles such that setting variable c_j^s to the opposite value in the variable assignment gadgets will incur additional risk. For a positive literal, the obstacles are constructed as

$$C(\mathbf{a}^{\mathbf{v}}_{\mathbf{c}^{\mathbf{s}}_{\mathbf{j}}}, \mathbf{u}^{\mathbf{v}}_{\mathbf{c}^{\mathbf{s}}_{\mathbf{j}}+\mathbf{1}})$$
$$B(\mathbf{b}^{\mathbf{v}}_{\mathbf{c}^{\mathbf{s}}_{\mathbf{j}}}, 1)$$

and for a negative literal, the obstacles are constructed as

$$C(\mathbf{u}^{\mathbf{v}}_{\mathbf{c}^{\mathbf{s}}_{\mathbf{j}}+\mathbf{1}}, \mathbf{b}^{\mathbf{v}}_{\mathbf{c}^{\mathbf{s}}_{\mathbf{j}}})$$
$$B(\mathbf{a}^{\mathbf{v}}_{\mathbf{c}^{\mathbf{s}}_{\mathbf{j}}}, -1).$$

Notice that each path through the variable gadget loop will incur risk of colliding with one of these obstacles, but at the end of the loop it will pass near the same obstacle that is near the branch corresponding to a satisfying assignment. Therefore selecting a variable assignment that satisfies this clause will risk collision with only one of these obstacles, whereas selecting a variable assignment that does not satisfy this clause will risk collision with both obstacles.

Positive and Negative Literal

For each clause j with one positive literal and one negative literal, we construct a loop below the top set of variable gadgets, with vertices

$$\mathbf{u}^{\mathbf{p}}_{\mathbf{j}} = (0, 4n_v + 3j)$$
$$\mathbf{a}^{\mathbf{p}}_{\mathbf{j}} = (-1, 4n_v + 3j + 1)$$
$$\mathbf{b}^{\mathbf{p}}_{\mathbf{j}} = (1, 4n_v + 3j + 1)$$
$$\mathbf{v}^{\mathbf{p}}_{\mathbf{j}} = (0, 4n_v + 3j + 2)$$

and edges

$$(\mathbf{u}^{\mathbf{n}}_{\mathbf{i}}, \mathbf{a}^{\mathbf{n}}_{\mathbf{i}})$$
$$(\mathbf{u}^{\mathbf{n}}_{\mathbf{i}}, \mathbf{b}^{\mathbf{n}}_{\mathbf{i}})$$
$$(\mathbf{a}^{\mathbf{n}}_{\mathbf{i}}, \mathbf{v}^{\mathbf{n}}_{\mathbf{i}})$$
$$(\mathbf{b}^{\mathbf{n}}_{\mathbf{i}}, \mathbf{v}^{\mathbf{n}}_{\mathbf{i}}).$$

As before, we also construct edges connecting consecutive loops $(\mathbf{v}^{\mathbf{n}}_{\mathbf{i}}, \mathbf{u}^{\mathbf{n}}_{\mathbf{i+1}})$ for all j, an edge from the end of the last variable gadget in the top set to the first positive-negative clause loop $(\mathbf{v}^{\mathbf{v}}_{\mathbf{n_v}}, \mathbf{u}^{\mathbf{n}}_{\mathbf{0}})$, and an edge from the last positive-negative clause loop to the first variable gadget in the mirrored set $(\mathbf{v}^{\mathbf{n}}_{\mathbf{n_n}}, \mathbf{u}^{\mathbf{v'}}_{\mathbf{n_v}})$.

Each loop gives a planning algorithm two paths corresponding to two literals to try to satisfy.

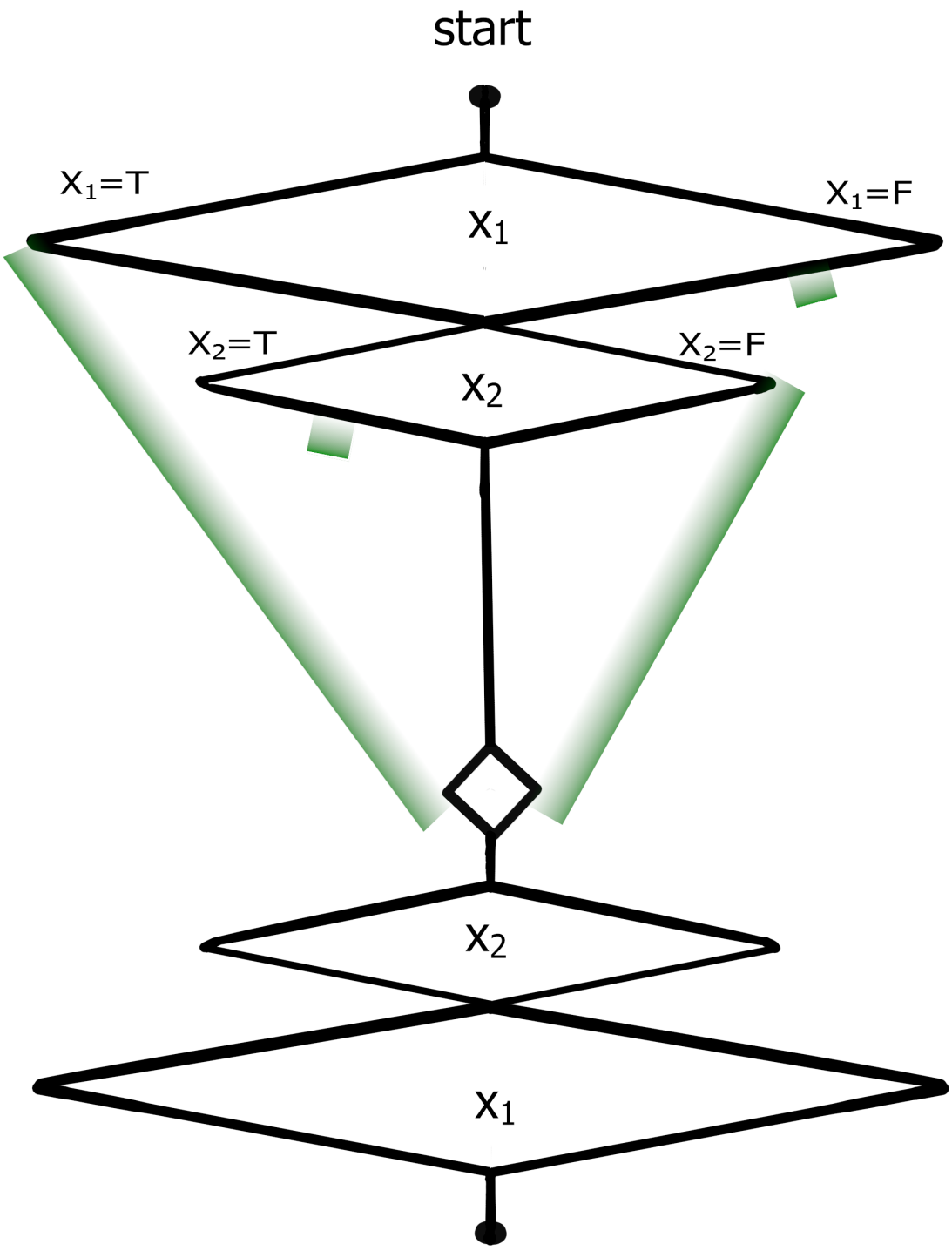

Figure 2.7: An example positive-negative clause gadget, for $X_1 \vee \neg X_2$, illustrating the variable gadget loops (obstacles not shown) and positive-negative clause gadget loops and obstacles. A path assigns a *true* or *false* value to each variable by selecting a branch through the variable gadget loop, thereby risking collision with an obstacle. Then, there is no additional risk for taking the branch through the positive-negative clause gadget loop that corresponds to a satisfied literal. Notice that only low-risk obstacles are used in this gadget. For clarity, we have drawn the B-template obstacles along the edge of the branch rather than at the corner of the branch, as it is constructed in the template.

Arbitrarily, for each such positive-negative clause j, let the left path correspond to the positive literal c^p_{jp} and the right path correspond to the negative literal c^p_{jn}. For each one, we construct two obstacles, each near one of the two paths of the corresponding variable gadget loop. However, for the obstacle near the path corresponding to assigning a value to the variable that satisfies the literal, we extend it to also be near the path for this literal in this clause gadget loop.

$$C\left(\mathbf{a^P_j}, \mathbf{a^v_{c^P_{jp}}}\right)$$
$$C\left(\mathbf{b^v_{c^P_{jn}}}, \mathbf{b^P_j}\right)$$

$$B\left(\mathbf{b^v_{c^P_{jp}}}, 1\right)$$
$$B\left(\mathbf{a^v_{c^P_{jn}}}, -1\right)$$

Therefore, taking a path corresponding to a satisfied literal risks collision with just that one obstacle, whereas a path corresponding to a non-satisfied literal risks collision with both obstacles.

Two Negative Literals

For each clause j with two negative literals, we construct a loop to the side of the other gadgets, with vertices

$$\mathbf{u^n_j} = (2n_v + 3n_s - 3j, 4n_v)$$
$$\mathbf{a^n_j} = (2n_v + 3n_s - 3j - 1, 4n_v + 1)$$
$$\mathbf{b^n_j} = (2n_v + 3n_s - 3j - 1, 4n_v - 1)$$
$$\mathbf{v^n_j} = (2n_v + 3n_s - 3j - 2, 4n_v)$$

and edges

$$(\mathbf{u^P_i}, \mathbf{a^P_i})$$
$$(\mathbf{u^P_i}, \mathbf{b^P_i})$$
$$(\mathbf{a^P_i}, \mathbf{v^P_i})$$
$$(\mathbf{b^P_i}, \mathbf{v^P_i}).$$

We construct edges connecting consecutive loops $(\mathbf{v^P_i}, \mathbf{u^P_{i+1}})$ for all j. We also construct an intermediate vertex $\mathbf{d} = (2n_v + 3n_s, 7n_v + 3n_p + 2)$ to connect the last variable gadget in the mirrored set to the first negative-negative clause loop with two edges $(\mathbf{v^v_0}, \mathbf{d})$ and $(\mathbf{d}, \mathbf{u^P_0})$.

Each loop gives a planning algorithm two paths corresponding to two literals to try to satisfy. For each such negative-negative clause j, let c^n_{j1} denote the variable specified by the first literal and c^n_{j2} denote the variable specified by the second literal. For each literal, we construct two obstacles, each near one of the two paths of the corresponding variable gadget loop (for the literal corresponding to the top path of the negative-negative clause loop, we will use the top set of variable gadgets, and

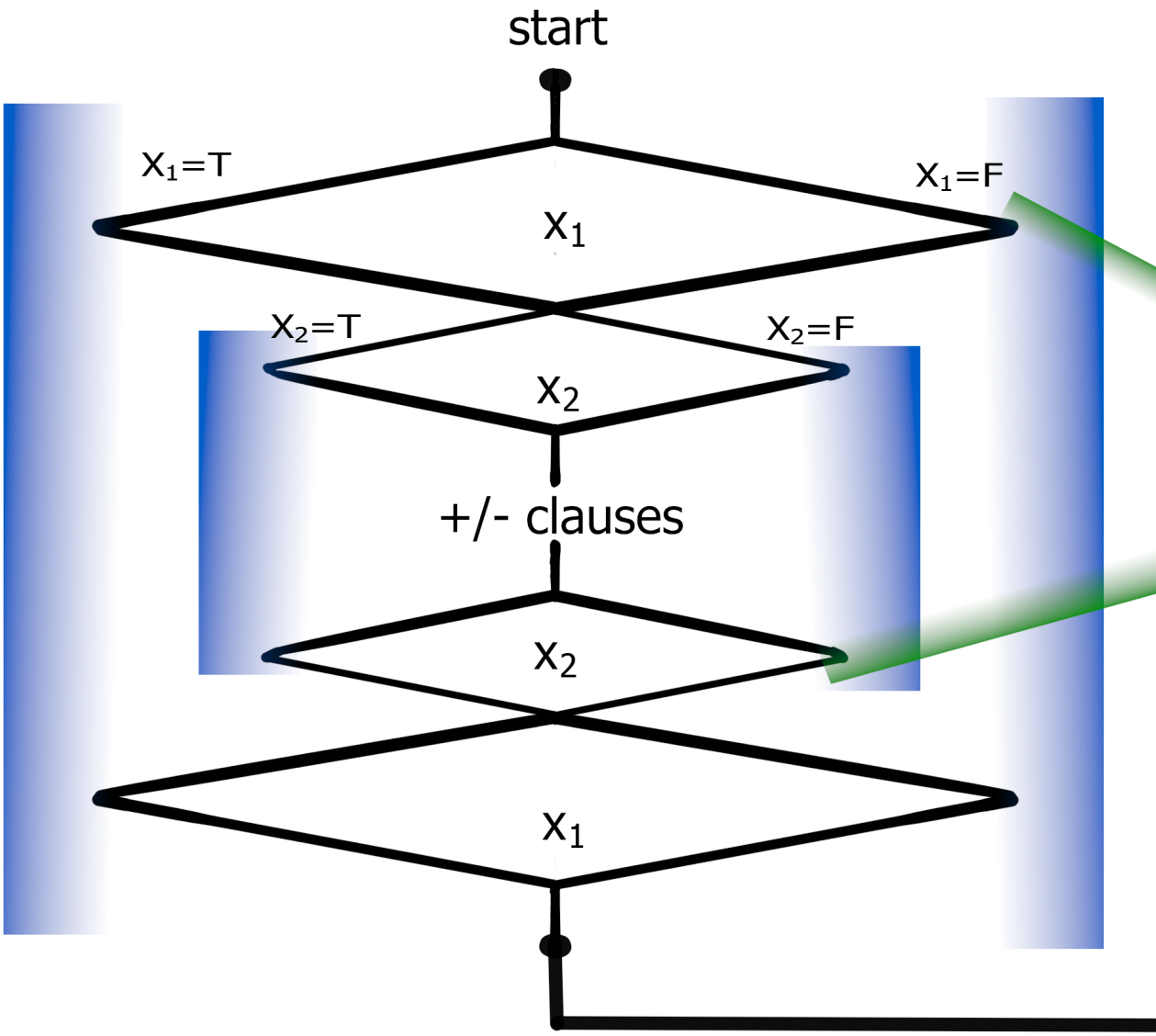

Figure 2.8: An example negative-negative clause gadget, for $\neg X_1 \vee \neg X_2$, illustrating the variable gadget loops and obstacles (blue) and negative-negative clause gadget loops and obstacles (green). A path assigns a *true* or *false* value to each variable by selecting a branch through the variable gadget loop. It must also select the same variable assignment in the mirrored gadgets at the bottom in order to avoid additional collision risk. Then, there is no additional risk for taking the branch through the negative-negative clause gadget loop that corresponds to a satisfied literal.

for the literal corresponding to the bottom path of the negative-negative clause loop, we will use the mirrored set of variable gadgets). However, for the obstacle near the path corresponding to a negative assignment assigning (which satisfies the literal), we extend it to also be near the path for this literal in this clause gadget loop.

$$C(\mathbf{b}^{\mathbf{v}}_{\mathbf{c^n_{j1}}}, \mathbf{a}^{\mathbf{n}}_{\mathbf{j}})$$
$$C(\mathbf{b}^{\mathbf{n}}_{\mathbf{j}}, \mathbf{a}^{\mathbf{v'}}_{\mathbf{c^n_{j2}}})$$

$$B(\mathbf{a}^{\mathbf{v}}_{\mathbf{c^n_{j1}}}, -1)$$
$$B(\mathbf{a}^{\mathbf{v'}}_{\mathbf{c^n_{j2}}}, -1)$$

Therefore, taking a path corresponding to a satisfied literal risks collision with just that one obstacle, whereas a path corresponding to a non-satisfied literal risks collision with both obstacles.

2.5.6 Path Risk Encoding MAXQHORNSAT

We define the $(1+\alpha)$-approximate safe graph search problem as $\phi(G, O, \mathbf{s}, \mathbf{t})$, where G is the set of vertices and edges constructed in the variable and clause gadgets above, O is the set of obstacles constructed in the variable and clause gadgets above, and

$$\mathbf{s} = \mathbf{u}^{\mathbf{v}}_{\mathbf{0}}$$
$$\mathbf{t} = \mathbf{v}^{\mathbf{n}}_{\mathbf{n_s}}.$$

G and O were constructed such that there will exist a gap between the risk of a path corresponding to an optimal assignment and a path corresponding to a suboptimal assignment. Each variable gadget loop in the top set passes near a single high-risk obstacle, and there will exist a path through the mirrored set that does not pass near any additional high-risk obstacles, and passing near an additional high-risk obstacle incurs more risk than passing near every low-risk obstacle, so a minimum-risk path will pass near exactly n_v high-risk obstacles. Any path must also pass near at least one obstacle for each clause gadget, and it will pass near an additional obstacle for each unsatisfied clause, so a minimum-risk path will pass near $(n_s + n_p + n_n + \delta)$ low-risk obstacles, where δ is the minimum number of unsatisfied clauses. Recall that there exists a gap between the induced risk close to an obstacle r_c and far away from the obstacle r_f (or r_{Vc} and r_{Vf} in the case of high-risk obstacles). Then a path corresponding to an optimal solution to the MAXQHORNSAT problem will incur risk at most

$$n_v r_{Vc} + n_v r_{Vf} + (n_c + \delta) r_c + (3n_c - \delta) r_f$$

and a path corresponding to a suboptimal solution to the MAXQHORNSAT problem will incur risk

at least

$$n_v r'_{Vc} + n_v r'_{Vf} + (n_c + \delta + 1)r'_c + (3n_c - \delta - 1)r'_f.$$

This allows us to compute a lower bound on the ratio of the risks between a suboptimal solution and an optimal solution as

$$\begin{aligned}
&\frac{n_v r'_{Vc} + n_v r'_{Vf} + (n_c + \delta + 1)r'_c + (3n_c - \delta - 1)r'_f}{n_v r_{Vc} + n_v r_{Vf} + (n_c + \delta)r_c + (3n_c - \delta)r_f} \\
\geq &\frac{n_v r'_{Vc} + (n_c + \delta + 1)r'_c}{n_v r_{Vc} + n_v r_{Vf} + (n_c + \delta)r_c + (3n_c - \delta)r_f} \\
= &\frac{5n_c n_v r'_c + (n_c + \delta + 1)r'_c}{5n_c n_v r_c + 5n_c n_v r_f + (n_c + \delta)r_c + (3n_c - \delta)r_f} \\
\geq &\left(\frac{r'_c}{r_c \beta} + \frac{r'_c}{20 n_c n_v r_c}\right) + \frac{r'_c}{20 n_c n_v r_c} \\
\geq &1 + \theta\left(\frac{1}{n^2}\right)
\end{aligned}$$

if we set the obstacle constants such that $20n_c n_v + \beta \geq 20n_c n_v \beta$, where $\beta = 1 + 3\frac{r_f}{r_c}$

Each gadget can be constructed in polynomial time, and the number of gadgets is polynomial, so this reduction can be constructed in polynomial time. Thus, any algorithm that can approximate the minimum-risk planning problem in a graph to a factor better than $1 + \Theta(\frac{1}{n^2})$ can also solve MAXQHORNSAT with polynomial overhead. □

2.5.7 Hardness of Continuous Planning Problem

We prove Theorem 3 by extending the above reduction to still apply even without the graph restriction (thereby reducing to $(1+\alpha)$-approximate minimum-risk planning). Given the graph G and set of obstacles O constructed above, we surround G with additional obstacles such that a path cannot deviate from G by more than $2\epsilon_P$, for some small constant ϵ_P. We divide the space of $\mathbb{R}^2$ into a grid with cells of size $\epsilon_P \times \epsilon_P$ and construct a square obstacle in every cell that does not intersect with the graph and is within a window containing all of the gadgets.

$$C_{ij} = \left\{ \mathbf{x} \;\middle|\; \begin{array}{l} \mathbf{x} \in \mathbb{R}^2 \\ i\epsilon_P \leq \mathbf{e_1}^T \mathbf{x} \leq (i+1)\epsilon_P \\ j\epsilon_P \leq \mathbf{e_2}^T \mathbf{x} \leq (j+1)\epsilon_P \end{array} \right\}$$

$$\text{for all } i, j \in \mathbb{Z}$$

$$O' = O \cup \left\{ C_{ij} \;\middle|\; \begin{array}{l} i, j \in \mathbb{Z} \\ -\frac{1}{\epsilon_P} 2n_v \leq i, j \leq \frac{1}{\epsilon_P}(8n_v + 4n_p) \\ C_{ij} \cap G = \emptyset \end{array} \right\}$$

Note that there are a polynomial number of such obstacles, so if ϵ_P is sufficiently small, the solution to the resulting $(1+\alpha)$-approximate minimum-risk planning problem or lack thereof is approximately equivalent to that for the original $(1+\alpha)$-approximate safe graph search problem, and so $(1+\alpha)$-approximate minimum-risk planning is also NP-hard. □

2.6 Hardness with Constraints on Overlapping Obstacles

Hauser observed that his 3D MCR reduction as well as the 2D MCR reduction presented in [64] required that each obstacle be allowed to overlap with $O(n)$ other obstacles [78]. Similarly, we note that the reduction we present above for 2D motion planning under obstacle uncertainty also requires that each obstacle overlap with $O(n)$ other obstacles. This is a relatively unnatural problem instance, as most real-world problems will not have this degree of overlap. In this section, we show that the problem remains hard in 3D even when each obstacle only overlaps with a constant number of other obstacles.

2.6.1 3SAT

3SAT is an NP-complete problem that is commonly used to prove the hardness of other problems [121]. The problem input is a Boolean formula given in conjunctive normal form, where each clause consists of three literals, or in other words, it is of the form $((x_0 \vee \neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge \ldots)$. The algorithm must then decide whether there exists any variable assignment that satisfies the formula. We will consider a 3SAT problem with k variables $x_0, x_1, \ldots$ and m clauses, where each clause j is of the form $(x_{j_u} \vee \neg x_{j_v} \vee x_{j_w})$.

2.6.2 Proof Outline

We prove Theorem 4 using a reduction from 3SAT. Given a 3SAT instance, we construct a κ-overlap $(1+\alpha)$-approximate minimum-risk planning problem as follows.

1. Construct a set of variable assignment layers where each branch corresponds to a variable assignment.
2. Construct a set of clause layers where each branch corresponds to selecting a literal to satisfy.
3. For each variable assignment layer, construct a pair of obstacles for each variable that will encode whether the variable is set to *true* or *false*. There will be additional collision risk for a path that selects different values in each variable assignment layer, as well as for a path that selects a literal that is not satisfied by the value selected in the preceding variable assignment layer.

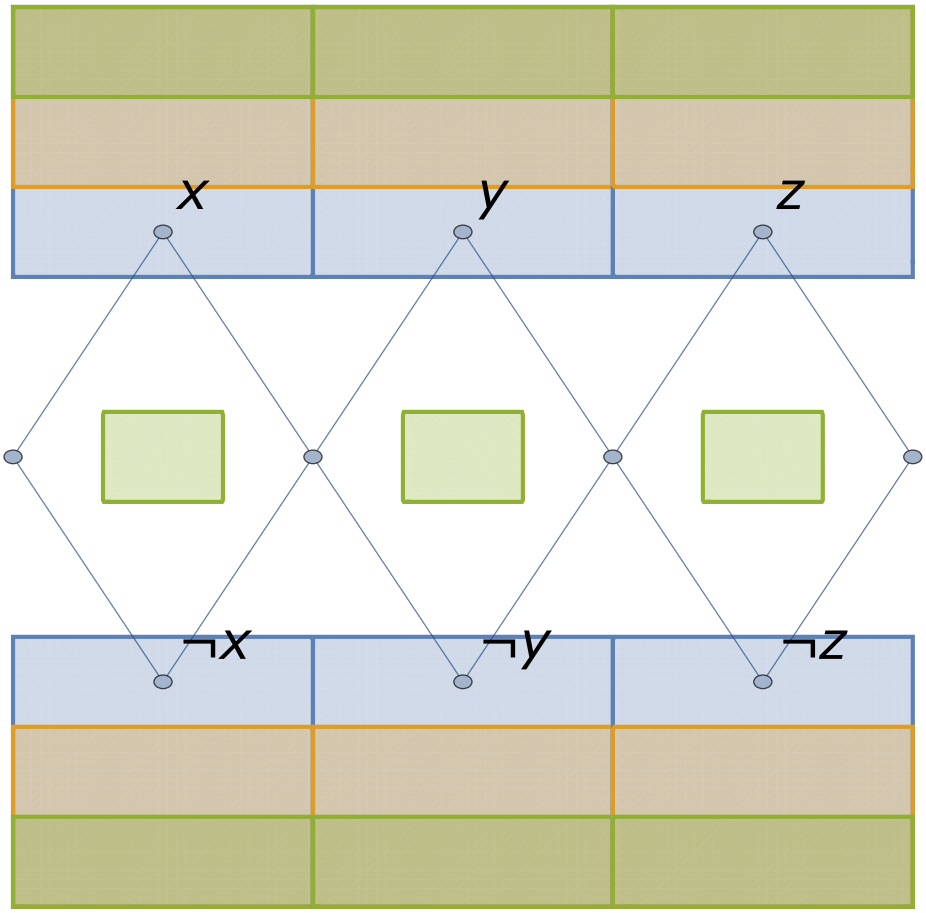

Figure 2.9: A path through this gadget must go near either the *true* or *false* obstacle for each variable, thereby selecting a variable assignment.

The solution to the planning problem can then be transformed into a solution to the 3SAT instance in polynomial time, demonstrating that the κ-overlap $(1+\alpha)$-approximate minimum-risk planning problem is at least as hard as 3SAT.

2.6.3 Proof

Variable Gadgets

First, we will construct a variable assignment layer for each clause j. A variable assignment layer consists of two PGDF obstacles for each variable i in the 3SAT problem.

Note that we define a PGDF obstacle as the intersection of halfspaces of the form $\alpha^T x \leq 0$ for α normally distributed and x represented in homogeneous coordinates. Here we will work with just

one face and standard coordinates for convenience. That is, each obstacle i will be defined as

$$o = \left\{ \mathbf{x} \mid {\alpha_\mathbf{i}}^T \mathbf{x} \leq 1, \alpha_\mathbf{i} \sim \mathcal{N}(\mu_\mathbf{i}, \mathbf{\Sigma_i}) \right\}.$$

For obstacle i, the *true* obstacle will be defined as the intersection of

$$\begin{aligned} {\alpha_\mathbf{i}}^T \mathbf{x} &\leq 1 \\ i \leq {\mathbf{e_2}}^T \mathbf{x} &\leq i+1 \\ 2j - \frac{3}{2} < {\mathbf{e_3}}^T \mathbf{x} &\leq 2j + \frac{3}{2} \end{aligned}$$

where $\alpha_\mathbf{i} \sim \mathcal{N}(2\mathbf{e_1}, \mathbf{e_1}{\mathbf{e_1}}^T)$. The "negative" obstacle will similarly be defined with

$$\begin{aligned} {\beta_\mathbf{i}}^T \mathbf{x} &\leq 1 \\ i \leq {\mathbf{e_2}}^T \mathbf{x} &\leq i+1 \\ 2j - \frac{3}{2} < {\mathbf{e_3}}^T \mathbf{x} &\leq 2j + \frac{3}{2} \end{aligned}$$

where $\beta_\mathbf{i} \sim \mathcal{N}(-2\mathbf{e_1}, \mathbf{e_1}{\mathbf{e_1}}^T)$.

Intuitively the covariance $\mathbf{e_1}{\mathbf{e_1}}^T$ means that $\alpha_\mathbf{i}$ has variance 1 in the direction of the normal of the face. This is important because it means that there is no variance in the orientation of the face. Also note that each obstacle overlaps with the corresponding obstacle in the layer below and the layer above, which we will later show to be important in ensuring that variable assignments are consistent across layers.

Then we will construct the variable assignment graph, as illustrated in figure 2.9. Said formally, indexing over the variable with index i, we embed nodes in locations

$$\begin{aligned} &(2j-1)\mathbf{e_3} + i\mathbf{e_2} \\ &(2j-1)\mathbf{e_3} + (i+\frac{1}{2})\mathbf{e_2} \pm \mathbf{e_1} \\ &(2j-1)\mathbf{e_3} + (i+1)\mathbf{e_2}. \end{aligned}$$

We then draw edges from $(2j-1)\mathbf{e_3} + i\mathbf{e_2}$ to both of $(2j-1)\mathbf{e_3} + (i+\frac{1}{2})\mathbf{e_2} \pm \mathbf{e_1}$, and from both of $(2j-1)\mathbf{e_3} + (i+\frac{1}{2})\mathbf{e_2} \pm \mathbf{e_1}$ to $(2j-1)\mathbf{e_3} + (i+1)\mathbf{e_2}$.

Clause Gadgets

For each clause j we will construct an additional graph "layer" in between consecutive pairs of variable layers that lets the algorithm choose which literal to satisfy, as illustrated in figure 2.10.

Recall that each clause j is of the form $x_{j_u} \vee \neg x_v \vee x_w$. Without loss of generality, let $j_u < j_v < j_w$.

Indexing over j, construct nodes at

$$2j\mathbf{e_3}, 2j\mathbf{e_3} + \left(j_u + \frac{1}{2}\right)\mathbf{e_2}$$
$$\left(2j + \frac{1}{3}\right)\mathbf{e_3} + \left(j_u + \frac{1}{2}\right)\mathbf{e_2} \pm \mathbf{e_1}$$
$$\left(2j + \frac{2}{3}\right)\mathbf{e_3} + \left(j_u + \frac{1}{2}\right)\mathbf{e_2}$$
$$\left(2j + \frac{2}{3}\right)\mathbf{e_3}$$

drawing edges between consecutive nodes, and letting '$\pm$' represent '-' if x_{j_u} is given in negated form and '+' otherwise. Then construct nodes at

$$2j\mathbf{e_3} + \left(j_u + \frac{1}{2}\right)\mathbf{e_2}$$
$$2j\mathbf{e_3} + \left(j_v + \frac{1}{2}\right)\mathbf{e_2}$$
$$\left(2j + \frac{1}{3}\right)\mathbf{e_3} + \left(j_v + \frac{1}{2}\right)\mathbf{e_2} \pm \mathbf{e_1}$$
$$\left(2j + \frac{2}{3}\right)\mathbf{e_3} + \left(j_v + \frac{1}{2}\right)\mathbf{e_2}$$
$$\left(2j + \frac{2}{3}\right)\mathbf{e_3} + \left(j_u + \frac{1}{2}\right)\mathbf{e_2}$$

(the first and last were already constructed previously), drawing edges between consecutive nodes, and similarly setting '$\pm$' based on the negation of literal x_{j_v}. Then construct nodes at

$$2j\mathbf{e_3} + \left(j_v + \frac{1}{2}\right)\mathbf{e_2}$$
$$2j\mathbf{e_3} + \left(j_w + \frac{1}{2}\right)\mathbf{e_2}$$
$$\left(2j + \frac{1}{3}\right)\mathbf{e_3} + \left(j_w + \frac{1}{2}\right)\mathbf{e_2} \pm \mathbf{e_1}$$
$$\left(2j + \frac{2}{3}\right)\mathbf{e_3} + \left(j_w + \frac{1}{2}\right)\mathbf{e_2}$$
$$\left(2j + \frac{2}{3}\right)\mathbf{e_3} + \left(j_v + \frac{1}{2}\right)\mathbf{e_2}$$

(the first and last were already constructed previously), drawing edges between consecutive nodes, and similarly setting '$\pm$' based on the negation of literal x_{j_w}. Intuitively, this creates three possible routes through the graph, each going near the obstacle corresponding to a particular value assigned to a variable.

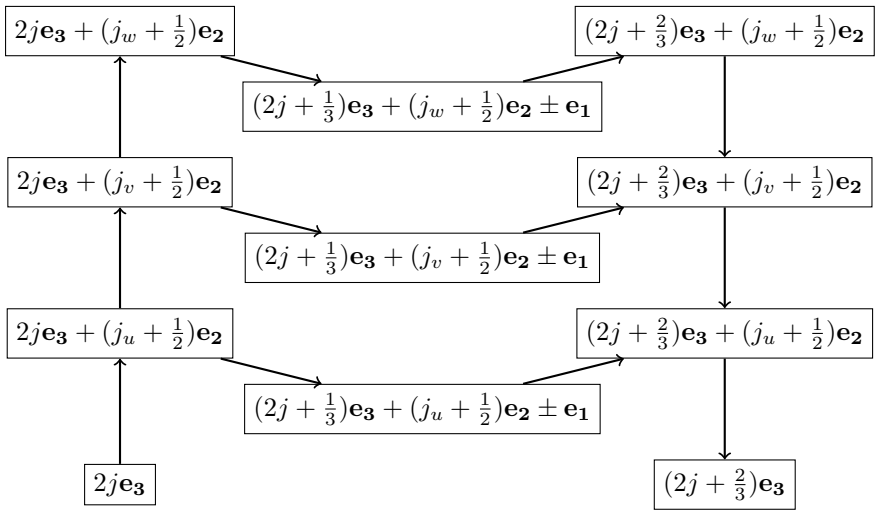

Figure 2.10: A path through this gadget must select one of three paths to go through, each going near the obstacle for the corresponding literal.

A path through this gadget must pick one of the literals in the clause to satisfy and pass near the obstacle that corresponds to that variable and the value the literal requires it to have. In doing so, it may incur risk of intersecting with the obstacle. If this variable was assigned to the value the literal specifies, then the path would have already gone near this obstacle so no further risk is incurred. However, if the literal contradicts the variable assignment, the path will incur additional risk for going near this obstacle.

Full Reduction

Now we combine the variable and clause gadgets, as seen in figure 2.11. As in Section 2.5.7, we construct a grid of deterministic obstacles to force the path to remain near the graph. Given the graph G and set of obstacles O constructed above, we surround G with additional obstacles such that a path cannot deviate from G by more than $2\epsilon_P$, for some small constant ϵ_P. We divide the space of $\mathbb{R}^{\not\vdash}$ into a grid with cells of size $\epsilon_P \times \epsilon_P \times \epsilon_P$ and construct a cubic obstacle in every cell

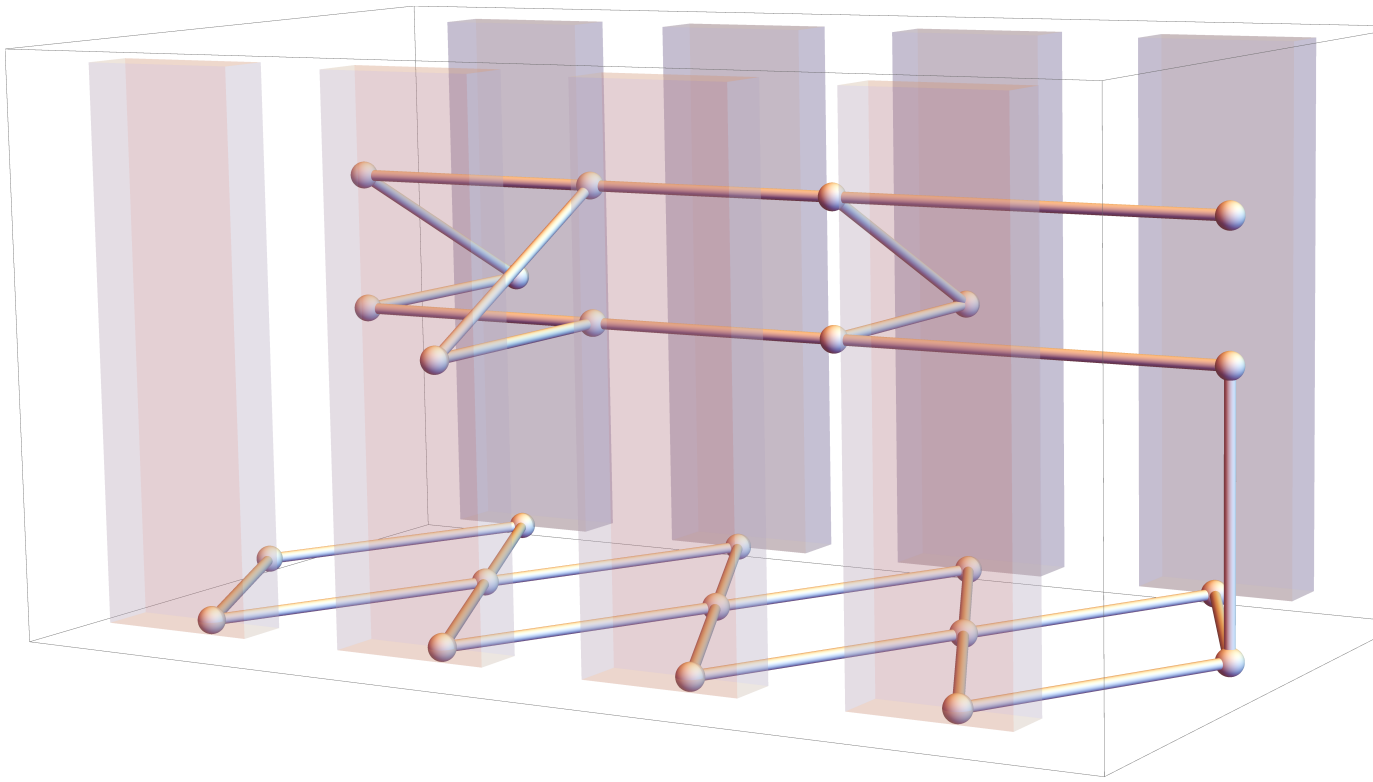

Figure 2.11: The bottom layer is the first variable assignment layer. The top layer is the first clause gadget. There would usually be many more clause gadgets stacked on top with additional variable gadget layers in between.

that does not intersect with the graph and is within a window containing all of the gadgets.

$$C_{\mathbf{z}} = \left\{ \mathbf{x} \;\middle|\; \begin{array}{l} \mathbf{x} \in \mathbb{R}^3 \\ z_i \epsilon_P \leq x_i \leq (z_i + 1)\epsilon_P \;\forall\; i \in \{1,2,3\} \end{array} \right\}$$
$$\text{for all } \mathbf{z} \in \mathbb{Z}^3$$

$$O' = O \cup \left\{ C_{\mathbf{z}} \;\middle|\; \begin{array}{l} z \in \mathbb{Z}^3 \\ -\frac{4}{\epsilon_P} \leq \frac{1}{\sqrt{3}}|\mathbf{z}| \leq \frac{1}{\epsilon_P}(k+m+4) \\ C_{\mathbf{z}} \cap G = \emptyset \end{array} \right\}$$

Path Risk Encoding 3SAT

This graph was constructed such that there will exist a gap between the risk of a satisfying assignment and of a non-satisfying assignment. First we note that for the PGDF obstacle model as well as most reasonable alternative formulations, there exists a gap between the induced risk close to the obstacle and far away from the obstacle. In particular, there is some r_c that lower-bounds the risk computed from the shadow approximation for the closer points and r_f that upper-bounds the computed risk for the further point. A path through the variable assignment portion of the graph will go near km obstacles for the k variable assignments it makes, each repeated m times. Then it will be "close" to km obstacles and "far" from the other km obstacles. Therefore, it will incur risk $kmr_c + kmr_f$.

If a path through the variable assignment portion encodes a satisfying assignment to the 3SAT problem, there will exist a path through the remainder of the graph that will not incur any additional cost. If there is no satisfying assignment, then any path through the remaining portion must go near an obstacle that it did not go near in the variable assignment portion, so for some variable i, the optimal path must go close to both the *true* and *false* obstacles, incurring cost at least $(km+1)r_c + (km-1)r_f$. This allows us to compute a lower bound on ratio between the two risks:

$$\begin{aligned} &\frac{(km+1)r_c + (km-1)r_f}{kmr_c + kmr_f} \\ =&\frac{kmr_c + kmr_f + r_c - r_f}{kmr_c + kmr_f} \\ =&1 + \frac{r_c - r_f}{kmr_c + kmr_f} \\ =&1 + \Theta\left(\frac{1}{km}\right). \end{aligned}$$

We note that each obstacle only overlaps with at most a constant number of other obstacles. In particular, each obstacle will overlap with the two corresponding obstacles in the layer above and the layer below, as well as the constant number of deterministic obstacles forming dividers between layers.

Each gadget can be constructed in polynomial time, and the number of gadgets is polynomial, so this reduction can be constructed in polynomial time. Thus any algorithm that can approximate the κ-overlap $(1+\alpha)$-approximate minimum-risk planning problem (regardless of whether a graph containing the solution is provided) to a factor better than $1+\Theta\left(\frac{1}{km}\right)$ can also solve 3SAT with polynomial overhead. □

Extension for Minimum Constraint Removal

This reduction can also be extended to apply to the Minimum Constraint Removal (MCR) Problem, proving Theorem 5 and answering the question posed by Hauser[78]. We replace each uncertain obstacle in the ϵ-safe planning problem with an MCR obstacle covering the r_c-shadow of the obstacle. As before, a path corresponding to a satisfying assignment will collide with km obstacles, whereas a path corresponding to a nonsatisfying assignment must collide with at least $km+1$ obstacles. Then the ratio between the costs of a nonsatisfying path and a satisfying path is lower bounded by

$$\frac{km+1}{km} = 1+\Theta\left(\frac{1}{km}\right).$$

Therefore, MCR remains NP-hard even when each obstacle is connected and intersects no more than κ obstacles (κ held constant).

2.7 Conclusions and future work

We have shown that the minimum-risk planning problem on graphs is NP-hard, even in two dimensions. Furthermore, the fact that it remains hard after restriction to a small graph indicates that exact algorithms reducing to a graph search are likely to be impractical in the uncertain domain. However, barring stronger hardness-of-approximation results, it is possible that there is a practical approximation algorithm for solving the minimum-risk planning problem on graphs. There is also the potential for algorithms that demonstrate fixed parameter tractability.

There is also the related direction of investigating models of uncertainty over obstacles. We focus on the PGDF model in this work because it captures certain desirable characteristics and has been used in prior work. However, the PGDF model has certain surprising characteristics, particularly near the tails of the distribution [9]. Perhaps there is a model that is a better fit for obstacle estimates in practice, that also permits efficient algorithms. In exploring this direction, it is important to note that we do not strongly invoke the structure of PGDF obstacles.

Chapter 3

Feasible Minimum Risk Motion Planning

3.1 Introduction

Planning in an uncertain environment is one of the fundamental problems facing autonomous systems. Drones and autonomous cars, for example, must all navigate environments that are not known a priori. Even after they are able to observe the environment, observations are often noisy and incomplete. Unfortunately, motion planning in this context is especially difficult when one wants to guarantee a low collision probability. One must reconstruct a model of the environment from sensor observations, quantify uncertainty in this estimation process, and then compute a path that ensures a low collision probability. For the rest of the paper, the term *safe* is used to refer to a trajectory with low collision probability over the randomness of the uncertain environment. Then the goal of Minimum Risk Motion Planning (MRMP) is to find such a safe trajectory. Note that this differs from many other works in planning under uncertainty which focus on uncertainty in estimation of the robot position, orientation or ability to execute a trajectory precisely.

In this paper we build on previous work characterizing collision probabilities and focus on developing methods for computing safe trajectories. Unfortunately, since many instances of this problem are NP hard, we cannot hope to find a provably efficient algorithm that works in general. Instead we identify a parameter that controls long range correlations between collision probabilities, and in turn, the hardness of the planning problem. When this parameter is a small constant, we demonstrate a provably efficient, optimal, algorithm with rigorous guarantees. *Fortunately, this parameter seems to be small for many practical instances.* When the parameter cannot be controlled, we demonstrate a tradeoff between computation time and the suboptimality of the result. The key contributions of this paper are twofold:

1. Defining a parameter that controls the hardness of MRMP and allows us to identify solvable MRMP problems. We demonstrate experimentally that this parameter is small for certain problems of interest.

2. Describing an algorithm that efficiently solves MRMP when the parameter is small. When the parameter is large, the algorithm allows trading off runtime and optimality.

In particular, this represents a different paradigm of working with computational complexity in robotics. Many works in robotics can be divided into two categories based on the hardness of the problem they are trying to solve: (a) the problem is efficiently (polynomial-time) solvable and the work presents an algorithm that guarantees an efficient runtime and optimal solution (b) the problem is provably computationally difficult so the work presents an algorithm that depends on heuristics and may only sometimes be efficient and/or produce an optimal solution. This paper takes a more fine grained approach. By identifying a parameter that quantifies how hard a problem instance, we can understand the exact tradeoff between runtime and solution quality. As a result, we are able to present an algorithm with strong theoretical guarantees that is, as demonstrated in the experimental section, also practical and efficient at solving instances which could not be solved by prior approaches.

3.1.1 Approximations in Planning with Environment Uncertainty

The hardness of planning with environmental uncertainty has led to various approximations to make the problem more computationally feasible. One line of work models this uncertainty as a partially observable Markov decision problem (POMDP) [32]. Unfortunately, solving POMDPs is PSPACE-hard, implying that an efficient, general algorithm does not exist [105]. Furthermore, even solving POMDPs in practice has proven difficult for complicated problems, despite advances in approximate POMDP solvers [91, 122].

However, there are many works that suggest that navigation among uncertain obstacles is an easier problem than solving general POMDPs. There is a long line of work that approaches this problem, trading off performance, safety guarantees, completeness, and limitations of the model [25, 83, 95, 81, 124, 106, 107, 56, 66]. One way to avoid making these tradeoffs is to find a restricted setting where the problem is easier to solve. Unfortunately, positive results in this area are sparse. Consider the following two negative results for context. The first is that planning on a graph with uncertain obstacles remains hard, even in two dimensions and even if obstacles are guaranteed to not overlap with more than a fixed number of other obstacles [120]. The second concerns the related minimum constraint removal problem. Finding the minimal set of obstacles to remove to make travel between two points feasible remains NP-hard even if (a) the obstacles are constrained to be axis-aligned rectangles or (b) obstacles are line segments such that no three intersect at one point [63]. In contrast to this prior work, this paper will present conditions under which the problem *is*

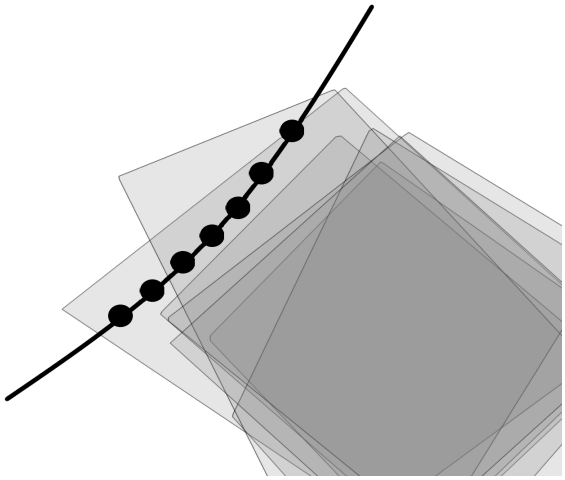

Figure 3.1: Estimating the risk by summing the risk of individual waypoints can overestimate the risk since the collision probability of adjacent waypoints is often correlated in practice. In the above cartoon, we can see that if a draw contains one waypoint, it is likely to contain many. Even though the collision risk of the trajectory is 20%, the sum of the risks of the waypoints is greater than 100%.

efficiently solvable.

In this paper, we focus on navigation among uncertain obstacles drawn from known distributions. In the scenario we examine, the robot first fixes a trajectory based on a known obstacle distribution. Then obstacles are drawn exactly once from the distribution and the robot executes the plan. The *risk* of the plan is defined as the probability that the trajectory collides with an obstacle. Ideally, an algorithm would always efficiently find the least-risk solution and never return a solution that is less safe than advertised.

There are several straightforward approaches to this problem, that while efficient, do not necessarily yield solutions that meet practical needs. The first would be to penalize each waypoint along a discretized path with its [unconditional] likelihood of collision. This approach corresponds to the assumption that collision probabilities at different waypoints are independent. The sum of the individual risks can be taken as a proxy for the total risk. Unfortunately, this has several undesirable properties. It is sensitive to the coarseness of the waypoint discretization. Taking a trajectory and discretizing more finely increases the computed risk bound even though the real risk does not change. At one extreme, using this calculation would show that a robot that does not change position is certain to collide! In reality, one expects collision probabilities at different points of a trajectory to potentially have significant correlations, making a union bound a poor strategy by which to bound the collision probability. This issue is illustrated in Figure 3.1.

This dependence on the discretization can be remedied by using the concept of a shadow. For every obstacle, a "shadow" (depicted in Figure 3.2) is a volume that contains the obstacle with

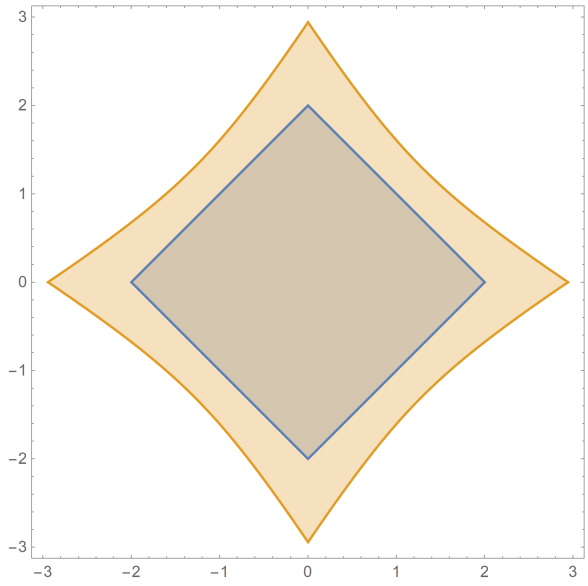

Figure 3.2: When the true position of the blue square is unknown, we can identify the orange "shadow" as a region that contains the blue square with some probability [10].

a fixed probability. As long as the robot follows a trajectory that avoids the shadows, the risk of the trajectory is at most the sum of the probabilities that each shadow contains its obstacle. If one can decide the probabilities corresponding to each shadow ahead of time, MRMP becomes identical to a standard motion planning problem. Shadows effectively capture the notion that collision probabilities are strongly correlated between points that are close together. Unfortunately, choosing these probabilities a priori is crucial for the reduction to the standard motion planning problem. If it is not known a priori which obstacles the optimal trajectory must pass near, the shadows must be chosen conservatively in a way that can have many undesired consequences. For example, even obstacles far away from the final trajectory which do not have a sizeable effect on the true collision probability may end up affecting the choice of trajectory and computed risk bound. The incompleteness resulting from allowing equal risk for every obstacle is illustrated in Figure 3.3. Unfortunately, planning without fixed shadows is harder than deterministic motion planning, even when allowing for certain approximations. Even when the obstacles are polytopes composed of Gaussian-distributed faces (defined in [10]), with paths restricted to a small graph and with a point robot in a low dimensional space, the problem is NP-hard [120].

In this work we identify a parameter that determines the hardness of such problems. When this parameter is small, we compute an optimal solution efficiently. We believe this parameter is small for most practical instances of planning among uncertain obstacles.

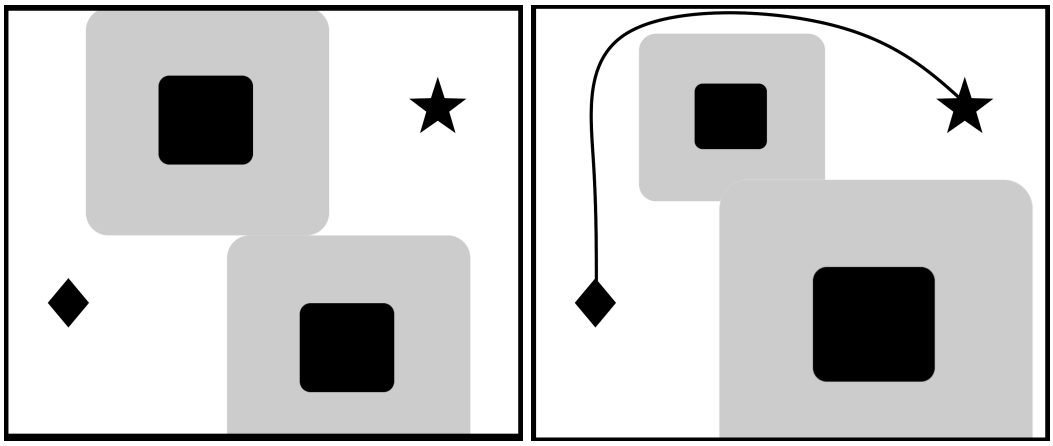

Figure 3.3: Planning a safe trajectory requires deciding how to balance the risk among the possible obstacles. In the first image, an equal risk is assigned to both obstacles and no path between the diamond and the star exists. In the second, more risk is assigned to the upper obstacle and less to the lower one creating a passage for the robot.

3.1.2 Related Work

Since planning under uncertainty is both ubiquitous and provably hard, many works rely on heuristics, rarely providing formal guarantees for both solution quality and runtime.

One line of work focuses on uncertainty in the robot's position. Here the model of the robot itself is "inflated" before the collision checking, ensuring that any slight inaccuracy in the position estimate or tracking of the trajectory does not result in a collision. Work that focuses on uncertainty in the environment sometimes does the mirror image. They often inflate the occupied volume of the obstacle with a "shadow" and ensure that any planned trajectory avoids the shadow [7, 83, 95]. Both of these approaches work well when the obstacles are spaced out because there is still room to pass between them, but they become incomplete in more crowded domains when a trajectory must reason about which objects force the robot to incur more risk. The planner does not know beforehand how much it can expand each shadow while still allowing a feasible trajectory.

A more general approach that handles either or both of localization and obstacle uncertainty is belief-space planning. Belief space is the set of all possible beliefs about or probability distributions over the current state. Belief-space planning converts the uncertain domain in state space to belief space, then plans in belief space using trees [109, 25] or control systems [108]. However, the dimensionality of the belief space can become quite large in domains with many uncertain variables such as obstacles.

Another line of work uses synthesis techniques to construct trajectories safe by construction. If the system is modeled as a Markov decision process with discrete states, a safe plan can be found using techniques from formal verification [56, 66]. Other authors have used techniques from Signal

Temporal Logic combined with an explicitly modeled uncertainty to generate plans that are safe [114], though the ϵ in said paper does not correspond to the notion of safety used in this paper.

Recent work by Hauser applies an approximate minimum constraint removal algorithm to motion planning under obstacle uncertainty by randomly sampling many draws for each obstacle and finding the path that intersects with the fewest samples [78]. With this approach, he demonstrates low runtime and suboptimality on average although with poor worst case performance. He notes that his greedy minimum constraint removal algorithm is optimal when the optimal plan does not require entering an obstacle multiple times, and similarly his solution for motion planning under uncertainty is optimal when it is not required to risk collision with an obstacle multiple times.

Building on previous works using "shadows", earlier work formalized the notion of a shadow in a way that allowed the construction of an efficient algorithm to bound the probability that a trajectory will collide with the estimated obstacles [10]. However, the proposed RRT-based planning method making use of this algorithm does not return a solution with probability approaching 1 as the number of iterations approaches infinity (i.e. it lacks probablistic completeness). This is because this problem lacks optimal substructure: subpaths of an optimal path are not necessarily optimal, since the collision risk at different points along a trajectory can be highly correlated if it passes near the same obstacle multiple times.

Other earlier work showed that planning in this domain is NP-hard in fixed dimension, and that even the problem of searching a graph for a safe path is NP-hard [120]. The reduction they construct forces the planner to solve MAXQHORNSAT by encoding variable assignment and clause satisfaction into collision risks for the different obstacles, taking advantage of the fact that collisions can be correlated even between distant portions of a trajectory. Hence, this suggests that the hardness of the problem stems from long-distance dependencies between potential collisions.

3.1.3 Summary of Formal and Experimental Results

As mentioned in section 3.1.2, planning under obstacle uncertainty is hard because the risk of collision at potentially distant segments of a trajectory can be correlated. However, in many domains, these correlated risks tend to be somewhat localized to nearby portions of the trajectory. In this paper we identify a parameter that controls this, the *collision horizon*. It functions as a measure of how far apart these correlated collisions are in terms of the number of obstacle shadows entered in between them. Intuitively, the collision horizon captures the number of obstacles that are interacting with each other in such a way that a planner must reason about their collision risks jointly; or alternatively, how much collision history is needed to define a Markov state, that is, one that fully determines the marginal risk at future states.

We present a parameterized polynomial-time algorithm M_h, for finding the mininum risk path in a graph, for which the following informal theorems hold (the formal statements are presented later in the paper):

Theorem 6. *On problems with collision horizon at most h, M_h returns the minimal collision risk plan.*

Theorem 7. *On problems with collision horizon h', where $h' > h$, M_h produces a solution that is suboptimal at most by the risk incurred by correlated collisions in the optimal trajectory that are separated by more than h other collisions.*

We also show that minimum constraint removal, the problem of finding trajectories with the fewest collisions, is also solved by our algorithm with the same guarantees.

Theorem 8. *The greedy and exact algorithms presented by [78] are equivalent to M_0 and M_∞, respectively, applied to MCR.*

Hence, M_h can be seen as interpolating between the greedy and exact algorithms on MCR problems (but not on minimum-risk planning).

These definitions and theorems are stated formally in sections 3.3.2 and 3.4.

We demonstrate M_h on a manipulation domain and an autonomous driving domain. We find that M_0 is near-optimal and that M_1 is optimal on all problem instances, indicating that the collision horizon tends to be small in these domains. We find similarly on an MCR manipulation domain.

3.2 Theoretical Guarantees for Motion Planning

Before we present a solution to minimum risk motion planning, it is helpful to understand the conditions under which we are able to effectively solve motion planning with known obstacle ocations and extents. While motion planning has been shown to be PSPACE-hard, the community has developed algorithms which are able to solve many practical motion planning problems [112]. Even though we cannot guarantee that any algorithm is both efficient (polynomial time) and complete (guaranteed to find a solution), we can provide lighter guarantees. The goal of this section is to provide background on different theoretical guarantees relevant to motion planning with obstacle uncertainty.

3.2.1 Completeness

An algorithm is said to be *complete* if it is always able to find a solution if one exists. Unfortunately, many algorithms which depend on heuristics are not complete and can fail in certain scenarios. For example, optimization based motion planners can fail to find a solution if they are not initialized with a trajectory whose homotopy class contains a valid trajectory.

When working with sampling based motion planners, we work with a criteria known as *probabilistic completness*, a property introduced with the RRT algorithm [94]. While we cannot guarantee that any algorithm efficiently returns a valid solution if one exists, we can ensure the algorithm is probabilistically complete.

Definition 14 (Probabilistically Complete Motion Planning Algorithm)**.** *A motion planning algorithm takes a set of obstacles, a start state* $\mathbf{s}$*, and a goal state* $\mathbf{t}$ *as input and generates a trajectory that does not intersect any obstacle as output. A planning algorithm is probabilistically complete if, with* n *samples, the probability that it finds a safe trajectory approaches* 1 *as* n *approaches* ∞*.*

Probabilistic completeness essentially means the algorithm will eventually find a solution if one exists.

Many sampling based algorithms, including RRTs, guarantee probabilistic completeness as long as there exists a solution that meets certain conditions. In particular, RRTs and many other sampling based motion planners are only probabilistically complete if there exists a solution trajectory in the topological interior of free configuration space. Since a similar condition is necessary for the algorithm presented in this paper, we develop this condition without explicitly relying on topology below.

This condition can be articulated as the existence of a path in the δ-interior of the free space X_{free} where X_{free} is a bounded subset of $\mathbb{R}^n$.

Definition 15 (δ-interior [86])**.** *A state* $x \in X_{free}$ *is in the* δ*-interior of* X_{free} *if the closed ball of radius* δ *around* x *lies entirely in* X_{free}*.*

A sampling based algorithm can only succeed if it always has a non-zero probability of sampling a waypoint leading to more progress. One way to ensure this is requiring the existence of a solution where every waypoint has a ball of nonzero diameter around it, guaranteeing that said ball has non-zero measure and the algorithm will eventually draw a sample in said ball.

When there exists a path in the δ-interior of free space for some $\delta > 0$, many sampling based motion planners are probabilistically complete. However this formulation does not extend well to the domain with uncertain obstacles; there is no concept of "free space" because the locations of the obstacles are not known. Instead we will use the equivalent view of inflating the path instead of shrinking the free space.

Definition 16 (δ-inflation)**.** *The* δ*-inflation of the set* X *is the set* $Y = \bigcup_{x \in X} \{y \mid d(x, y) \leq \delta\}$*, where* $d(x, y)$ *is any distance metric.*

We note that in the deterministic setting, if a trajectory is in the δ-interior of X_{free}, then the δ-inflation of the trajectory is entirely in X_{free}. This allows us to consider problems with the following regularity condition: there exists a δ-inflated trajectory that has a low risk of collision.

Definition 17 (ϵ-safe δ-inflated trajectory)**.** *A trajectory* τ *is an* ϵ*-safe* δ*-inflated trajectory if its* δ*-inflation intersects an obstacle with probability at most* ϵ*.*

While the standard RRT requires the existence of a trajectory in the interior of free space in order to be probabilistically complete, the algorithm in this paper requires the existence of an $\epsilon-$safe $\delta-$inflated trajectory in order to be probabilistically complete.

3.2.2 Graph Restriction Hardness

When solving motion planning without uncertainty, once the algorithm has identified a graph that contains a solution, the problem is essentially solved. Algorithms like Djikstra's algorithm and A* can be applied out of the box to find the minimum cost path within said graph.

We refer to the hardness of finding a solution restricted to a graph the *graph restriction complexity* of a problem. Since we can apply Djikstra's algorithm to solve motion planning without uncertainty, the graph restriction complexity is P. This is crucial to enabling the fast solving of motion planning in practice. Sampling based planners tackle the problem in two [sometimes alternating] phases. The first phase involves sampling a graph. The second involves checking if the graph contains a solution, and finding the lowest cost one if it does. Motion planning problems that are "easy" for sampling based planners are ones where the first phase is easy. The hardness of the second phase is exactly the graph restriction complexity.

One could hope that with the right approximations, a similar pattern could work for motion planning with obstacle uncertainty. In [10], the authors develop a notion of confidence intervals around obstacles with the aim of using them as an approximation enabling efficient planning. Unfortunately, the graph restriction complexity of planning with obstacle uncertainty with shadows is still NP-hard, even in two dimensions [120]. In other words, even if you are able to efficiently identify a graph containing the solution, it is not easy to find the solution in this graph unless P=NP.

This paper presents an algorithm that shows that the graph-restriction complexity of MRMP is P when the collision horizon is constant. Under these conditions, the same paradigm as for standard motion planning applies. One can sample a graph in configuration space just like with a standard sampling based motion planner and then use the presented graph search algorithm in order to solve MRMP. This allows us to adapt the standard sampling based motion planners to be able to solve MRMP.

3.3 Definitions

3.3.1 Formal Problem Definition

Random Obstacle Model

Up until this point, we have been talking about uncertain obstacles in general. In this section, we define the abstraction we use to work with uncertain obstacles. Note that we are not working with uncertainty in the robot pose, only in the environment. Further, we will be restricting our discussion to methods that first construct a graph embedded in configuration space, such as a PRM [87] or RRG [85], then search the graph for safe plans.

A useful reference here are the shadows defined by [10]. The paper defined shadows controlled by a single parameter (the probability that the shadow contains the obstacle). Shadows more likely

to contain the obstacle strictly contain shadows less likely to contain the obstacle. The algorithm in this paper works with the *monotone risk model*, which includes the shadows used in [10]. Under this model, every node in the planning graph is assigned a risk for each obstacle. For a given path and particular obstacle, the risk of any node colliding with the obstacle is bounded by a weakly monotone submodular function over the nodes in the path. When using shadows, this function becomes simply the maximum over the risks for each individual node being in collision with that obstacle. Note that it does not matter if risks are associated with nodes or edges since a graph with risks at edges may be transformed to one with risks at nodes by constructing a node for every edge.

Definition 18 (Monotone Risk Model)**.** *Given a graph $G = (V, E)$ (with edges E and vertices V), a function $f : \mathcal{P}(E) \to L$ (with $\mathcal{P}$ denoting the power set, and $L \subset \mathbb{R}$ is a finite set over which the risk is discretized) is a* Monotone Risk Model *if f is a weakly monotone submodular set function* or *f is a weakly monotone accumulation of Monotone Risk Models, i.e. $f(E') = g(f_1(E'), f_2(E'))$, where f_1, f_2 are Monotone Risk Models and g is weakly monotone, that is, $g(w, x) \geq g(y, z)$ if $w \geq y$ and $x \geq z$. Typically, each $f_i(E')$ would represent the risk that a given path, represented as a set of edges $E' \subseteq E$, is in collision with obstacle i. Then the overall risk of the path can be obtained by combining the risk for each obstacle in some monotonic manner.*

A natural monotone accumulation would be the OR operation, treating the inputs as collision probabilities for independent events (i.e. $g(x, y) = x + y - xy$). In practice —and in our experiments —a union-bound accumulation, that is, summation over probabilities ($g(x, y) = x + y$), is often preferred for its simplicity. We note that our theorems and algorithms apply to both of these, as well as any other monotone accumulation model. Henceforth we let $R[x \cup y]$ denote this accumulation over the risks of events x and y.

Furthermore, the discrete minimum constraint removal problem (MCR), can also be expressed as a minimization of a monotone risk model. Each obstacle i corresponds to a monotone risk model f_i such that $f_i(E') = 1$ if any edge $e \in E'$ is in collision with obstacle i, and $f_i(E') = 0$ otherwise. Then the sum of the risk models for all obstacles, which corresponds exactly to the cost function defined in the MCR problem, is also a monotone risk model.

In the deterministic case, an obstacle is typically a set of points in task space that the robot would like to avoid. A useful abstraction is to define for each obstacle a mapping from robot trajectories to 1 or 0, indicating whether the swept volume of a robot executing that trajectory collides with the obstacle or not. We extend this notion to the uncertain case, defining a random obstacle and then constructing a mapping instead from robot trajectories to collision *probabilities*.

Definition 19 (Obstacle)**.** *An obstacle is defined as a random volume in task space (usually $\mathbb{R}^3$) drawn from a known distribution. We note that an obstacle also corresponds to a random volume in configuration space (all configurations that result in the robot colliding with the obstacle). Each obstacle o is associated with a mapping $f_o : \mathcal{P}(E) \to [0, \infty)$, where f_o is a Monotone Risk Model denoting the risk of a path colliding with obstacle o.*

There are many ways to describe the risk of collision with a given obstacle as a weakly monotone submodular set function over edges, but just as an example, under the model presented in [10], the risk for a given edge corresponds to the risk of the minimal shadow that intersects with the edge. Then the risk of an entire path, or the risk of the minimal shadow that intersects with the path, is equal to the maximum over the risks for the individual edges. In this paper, we will be working with a discrete set of shadows for each obstacle. Then we define a risk level as the risk associated with a particular shadow.

Definition 20 (Risk Level)**.** *A risk level l_o^i is the probability that obstacle o is not fully contained within its i'th shadow.*

Informally, the shadows for a set of risk levels can be thought of as regions enclosed by contour lines encircling an obstacle, with higher risk levels associated with smaller shadows of higher risk. We note this model is not limited to the shadows constructed in [10]. For example, it applies to shadows computed for different distributions using a similar technique.

It is convenient to work with trajectories as swept volumes in task space (i.e. the space that a robot moves through when following a given trajectory). For a given set of distributions over obstacles, the risk of a trajectory is the probability that the trajectory intersects *any* obstacle. We frequently abuse notation, representing trajectories either in configuration space, task space, or nodes and edges in a graph. We now define an $\epsilon-$safe trajectory.

Definition 21 (ϵ-safe trajectory)**.** *Given a joint distribution over random obstacles O, a trajectory τ is ϵ-safe if, for any sample $s_1...s_n \sim O$, $R[\tau \cap s_i \neq \varnothing \; \forall \; i] \leq \epsilon$. That is, a trajectory is ϵ-safe if the probability that it collides with* any *obstacle is less than ϵ.*

Note that for obstacles $o_1...o_n$, the probability of collision is bounded by the sum of the individual collision probabilities. Thus if $\sum f_{o_i}(\tau) \leq \epsilon$, τ is $\epsilon-$ safe. This allows us to quantify the safety of a trajectory by using a monotone risk model.

Algorithmic Question

This leads to the following algorithmic question, of finding safe plans for a known distribution of obstacles.

Problem 3 (ϵ-safe Planning Problem)**.** *Given the obstacle distributions O and initial and end points $\mathbf{s}, \mathbf{t}$ in configuration space, find an ϵ-safe trajectory from $\mathbf{s}$ to $\mathbf{t}$ if one exists, otherwise FAIL.*

While the problem is known in general to be intractable [120], and risk-constrained extensions to practical sampling-based motion planning algorithms that build up a tree lack probabilistic completeness [10], [8] proposed a risk-constrained extension to the RRG algorithm [85] which is probabilistically complete. Unfortunately, that algorithm requires executing a risk-constrained graph search, which is also known to be intractable if completeness is required [120]. (The conditions and

guarantees for probabilistic completeness of the algorithm presented in this paper are discussed in detail in section 3.2). Nevertheless, we would still like to find a practical algorithm that can solve this problem.

Definition 22 (minimum-risk graph-search algorithm)**.** *A minimum-risk graph-search algorithm is a procedure $\phi(G, O, \mathbf{s}, \mathbf{t})$, where G is a graph, O is a set of obstacles, and $\mathbf{s}$ and $\mathbf{t}$ are the start and end nodes in G, respectively. It returns an ϵ_*-safe trajectory in G, where ϵ_* is the minimum ϵ for which an ϵ-safe δ-inflated trajectory exists.*

The δ-inflation condition informally means that there must be a nonzero probability of eventually sampling a satisfying trajectory, and is more formally discussed in section 3.2 above. We note that an algorithm optimizing risk can be directly used in place of an algorithm satisfying bounded risk. Simply run the risk-minimizing algorithm and do not return the solution if the risk is too high.

3.3.2 Formal Definition of Collision Horizon

Because the minimum-risk graph-search problem is intractable, we would now like to define a parameterized version of the problem that is tractable when the parameter is fixed. We identify a parameter that drives the hardness of the minimum-risk graph-search problem. More specifically, we define the collision horizon which, loosely speaking, captures the length of dependencies between obstacles.

We first define the collision coverage, illustrated in Figure 3.4, which describes the distance between correlated collisions for a single obstacle.

Definition 23 (collision coverage)**.** *The collision coverage $H_o^{(\tau)}$ for a given trajectory τ and obstacle $o \in O$ in minimum-risk graph-search problem instance $(G, O, \mathbf{s}, \mathbf{t})$ is the number of obstacles (including o itself) for which τ enters a higher-risk shadow between the first time it enters a given shadow of o and the last time. More formally, treating τ as a sequence of edges, let $\zeta_{o'}^{(\tau)}$ denote the set of indices t for edges for which $\tau(t)$ enters a higher-risk shadow of obstacle o', i.e. $\zeta_{o'}^{(\tau)} = \{t \mid f_{o'}(\tau(t)) > f_{o'}(\tau(t-1))\}$. Then*

$$H_o^{(\tau)} = \sum_{o' \in O} \mathbb{1}(\exists t \in \zeta_{o'}^{(\tau)} \; s.t. \; \min \zeta_o^{(\tau)} \leq t \leq \max \zeta_o^{(\tau)}).$$

This leads us to define the collision horizon for the overall problem instance.

Definition 24 (collision horizon)**.** *The collision horizon h of a given minimum-risk graph-search problem instance $(G, O, \mathbf{s}, \mathbf{t})$ is the maximum collision coverage of any obstacle of a minimum-risk trajectory, or more formally, $h = \min_{\tau \in T_*} \max_{o \in O} H_o^{(\tau)}$, where T_* is the set of minimal-risk trajectories from $\mathbf{s}$ to $\mathbf{t}$.*

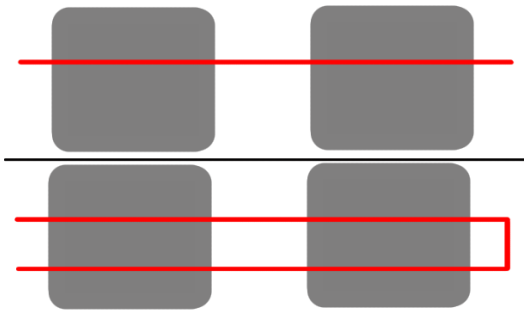

Figure 3.4: In the upper half of the figure the path does not return to any obstacles and has a collision coverage of 0. The second path returns to the first obstacle after going through the second, leading to a coverage of 1.

Definition 25 (h-horizon minimum-risk graph-search algorithm)**.** *A h-horizon minimum-risk graph-search algorithm is a procedure $\phi(G, O, \mathbf{s}, \mathbf{t})$, where G is a graph, O is a set of obstacles, and $\mathbf{s}$ and $\mathbf{t}$ are the start and end nodes in G, respectively. When the collision horizon of the problem is at most h, it returns an ϵ_*-safe trajectory in G, where ϵ_* is the minimum ϵ for which an ϵ-safe δ-inflated trajectory exists.*

We will now present such an algorithm.

3.4 Algorithm

In this section, we present a polynomial-time h-horizon minimum-risk graph-search algorithm. The algorithm is based on Dijkstra's algorithm minimizing the collision risk, but instead of nodes corresponding to robot configurations, nodes correspond to a robot configuration and a risk-level memory limited to h obstacles. The other differentiator between our algorithm and a standard graph search is an update rule that takes advantage of the structure of shadows to expand multiple nodes simultaneously.

Recall that shadows are split up into risk levels. Once the robot enters a higher risk-level shadow for a particular obstacle, it does not incur any additional risk for time spent in lower risk-level shadows for that same obstacle. The risk memory keeps track of the maximum risk-level shadow entered for each of some number of obstacles. Then we define a state as a pair $(\mathbf{u}, C)$, where $\mathbf{u}$ is a robot configuration and C is a risk memory with $|C| \leq h$. Suppose edge $(\mathbf{u}, \mathbf{v})$ in the original graph enters some set C' of shadows. Then the augmented graph will have a corresponding set of edges $((\mathbf{u}, C), (\mathbf{v}, C''))$ for each C of size at most h, and for each C'' that can be obtained from C by adding (i.e. memorizing) the additional shadows in C' then removing (i.e. forgetting) some number of remembered shadows such that the resulting $|C''| \leq h$. Each of these edges would have a cost equal to the marginal risk of entering the shadows in C' conditioned on having already entered

the shadows in C. This allows the cost function to only increase when the search enters a higher risk-level shadow than those which have been recently visited.

We note that standard Dijkstra's algorithm with this graph augmentation on its own would be sufficient for our theoretical results below to hold, but the large branching factor stemming from the choice of which collision to forget causes a substantial, albeit still polynomial, increase in runtime. Instead, we introduce an optimization via an extension to the expansion rule that takes advantage of the structure imposed by shadows. In order to define the expansion rule we use the term "open" to refer to a node which is in the queue to be expanded, and "closed" to refer to nodes which have already been expanded. While the search must retain multiple paths to each robot configuration, potentially with different risks, not all paths are useful. Consider the situation where we already have a path to configuration $\mathbf{u}$ with computed risk 0.5 with a 0.2 risk shadow for obstacle one and 0.2 risk for obstacle two. The next node to be expanded reaches the same configuration with computed risk 0.5 with a 0.1 risk for obstacle one and a 0.1 risk for obstacle two. The computed risk for any path going through this new node is either worse or unchanged relative to if it had instead went through the first node, since any future marginal cost would be at least as high for the new node as for the first one. Then even though the new node technically has a different state, we don't have to expand it since it will not lead to any paths that are better than those that go through the first node.

We formally encode this idea by defining a partial ordering between risk memories such that $C \leq C'$ iff, for every obstacle o that is represented in C, the risk level for o in C is at most the risk level for o in C' (defaulting to false if o is not represented in C'). Then when deciding whether to expand a state $(\mathbf{u}, C)$, instead of just checking whether the exact state is closed, we check for every subset $M \leq C$ where $|M| \leq h$, whether any closed state at vertex $\mathbf{u}$ has a risk memory that M precedes. This works because $C \leq C'$ implies that any future marginal cost starting from $(\mathbf{u}, C)$ would be at least the future marginal cost starting from $(\mathbf{u}, C')$. This is effectively treating each state as a collection of all states for that vertex with a preceding risk memory. Pseudocode for this algorithm is provided in Algorithm 2.

With k obstacles, L risk levels, and E edges, it can visit at most $(kL)^h$ unique states for each vertex (since there are $(kL)^h$ potential memories of size h, and each additional visited state corresponds to at least one new memory of size h that precedes the state's memory but none of the prior states' memories), so it can query at most $E(kL)^h$ edges. Since each edge can take $O\left(\binom{k}{h}(kL)^h h\right)$ to check whether it can be skipped (for each subset of size h, check against the memory for each visited state at that vertex, which requires comparing h risk levels for each check), the overall algorithm runs in time $O\left(\binom{k}{h}(kL)^{2h} h^2 E \log V kL\right)$, which is polynomial when h is fixed.

This algorithm correctly computes the cost of any trajectory segment that does not incur correlated collision risk at points separated by more than h other obstacles. Thus, the following theorem bounding the suboptimality of the algorithm holds.

Algorithm 2 MEMORY_SEARCH (M_h)

Input: Graph $G = (V, E)$, obstacles O, end points $\mathbf{s}, \mathbf{t}$, and collision horizon h.
Output: A trajectory from $\mathbf{s}$ to $\mathbf{t}$ through G.

```
closed = {}
open = PriorityQueue({(s, [], {})})
while not open.empty() do
    // u is the configuration
    // τ is the trajectory that reaches u
    // C is the risk memory
    // i.e. (obstacle, collision level) pairs
    u, τ, C = open.pop()
    if u = t then
        return τ // Reached goal
    end if
    // Check whether any closed state's
    // memory precedes C, or every
    // subset of C of size at most h
    // precedes the memory of some
    // closed state
    if (∄(w, C'') ∈ closed s.t. w = u, C'' ≤ C) and (∃M ≤ C s.t. (|M| ≤ h and ∄ (w, C'') ∈
    closed s.t. w = u, M ≤ C'')) then
        closed.insert(u, C)
        // Add each outgoing edge to open
        for v ∈ E[u] do
            C' = {(o, max(C[o], f_o((u, v)))
            for each o ∈ O}
            open.insert(∑_{o∈O} C'[o], (v, τ + (u, v), C'))
        end for
    end if
end while
```

Theorem 9. *Let ϵ be the associated cost of the trajectory generated by $M_h(G, O, \mathbf{s}, \mathbf{t})$ and let τ_* be any optimal trajectory, with associated cost ϵ_*. Then*

$$\epsilon \leq \sum_{o \in O, \tau_i \in S_o^{(\tau_*)}} f_o(\tau_i) \leq \epsilon_* + \sum_{o \in O} (|S_o^{(\tau_*)}| - 1) f_o(\tau_*)$$

where $S_o^{(\tau_*)}$ is the trajectory τ_* split into the fewest segments such that for each $\tau_i \in S_o^{(\tau_*)}$, $H_o^{(\tau_i)} \leq h$.

The proof for Theorem 9 is in section 3.5. And the following corollary holds when the collision horizon is fixed.

Theorem 10. *M_h returns an optimal trajectory when the collision horizon of the problem is less than h.*

Thus, the problem is tractable when the collision horizon is small, and when it is unbounded,

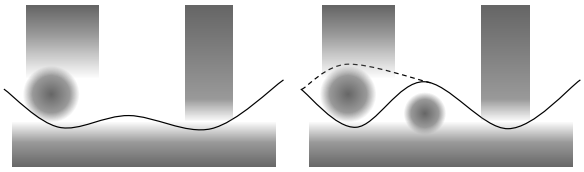

Figure 3.5: An example where M_0 is optimal (left) and one where it is suboptimal (right). The shading indicates the obstacle shadows, so the probability a trajectory collides with a given obstacle is given by the darkness of the maximally shaded point it goes through. In both cases the optimal trajectory is the solid black line, which risks collision with the long obstacle at the bottom —even though the trajectory on the right risks collision with the bottom obstacle twice, these collisions are correlated (if the first "dip" is in collision, then so is the second, and vice versa), so the overall collision risk is the same as on the left. However, in the suboptimal case, M_0 will instead pick the dotted subtrajectory, because by itself it is safer than the corresponding subpath of the optimal trajectory. Hence, this problem instance has collision horizon 1, and so M_1 will solve it correctly.

we have an algorithm that produces a solution whose suboptimality is limited by how much the optimal trajectory exceeds a collision horizon of h. This bound is especially helpful for domains where obstacle extents (i.e. the distribution over the obstacle boundaries) have infinite support, because while the collision horizon of these problems are large due to significant overlap of shadows, they impact the quality of the solution only by the amount of change in risk level over the areas that interact with more than h other obstacles, which is typically only the large, low-risk shadows. Hence, we can still find a near-optimal solution. Some examples of the behavior of this algorithm on different kinds of problems are in Figure 3.5.

3.4.1 Application to MCR

We would now like to consider the related problem of *minimum constraint removal*, or the problem of finding a plan that collides with the smallest number of obstacles. This problem was studied by Hauser, who presented two algorithms for solving it [78]. The first is an exact solver, which retains the set of past collisions in the state space of the search, leading to optimal solutions but a potentially exponentially large search space. The other algorithm is a greedy solver, which is restricted to visiting each vertex at most once. As a result, the greedy method is faster, but can be suboptimal in certain cases.

We so far have described an algorithm for planning under obstacle uncertainty. However, as noted by [78] and by [120], there appear to be strong connections between planning under obstacle uncertainty and minimum constraint removal. In fact, a minimum constraint removal problem can be described as a planning under obstacle uncertainty problem, where each obstacle only has a single shadow and a fixed cost for colliding with it. Intuitively, this can be thought of as treating the objective of minimizing collisions as equivalent to the objective of minimizing collision risk when each obstacle has some fixed probability of existing.

As such, our algorithm can also be used to solve minimum constraint removal problems. Moreover, M_0 is equivalent to the greedy algorithm proposed by [78], and M_∞ is equivalent to his exact algorithm. Thus, the collision horizon parameter of our algorithm can be seen as interpolating between the greedy and exact algorithms, providing a tradeoff between optimality and runtime based on the application.

3.5 Proofs

Proof of Theorem 9:

Proof. Let ϵ be the associated cost of the trajectory generated by $M_h(G, O, \mathbf{s}, \mathbf{t})$ and let τ_* be any optimal trajectory, with associated cost ϵ_*. Because each obstacle is distributed independently from other obstacles, we begin by considering the risk incurred by each one separately. For a given obstacle o, suppose $S_o^{(\tau_*)}$ is the trajectory τ_* split into the fewest segments such that for each $\tau_i \in S_o^{(\tau_*)}$, $H_o^{(\tau_i)} \leq h$. For each time τ_i enters an obstacle level with edge (u, v), the planning tree generated by M_h must contain a state $\hat{s}_u$ at vertex u with memory containing the preceding h collisions in τ_i since such a state is reachable (given that τ_i reaches it) and would not be skipped unless another previously closed state at u already contained the preceding h collisions. Because $H_o^{(\tau_i)} \leq h$, we know that the preceding h collisions are sufficient to determine the marginal risk of each collision. Then M_h will at some point expand edge $(\hat{s}_u, \hat{s}_v)$, where $\hat{s}_v$ is the state at vertex v still with memory containing the preceding h collisions in τ_i and with cost from o no more than $f_o(\tau_i)$. Hence, M_h computes the marginal risk of this obstacle for this subtrajectory correctly. Then the total computed risk from obstacle o for trajectory τ_* is at most

$$\sum_{\tau_i \in S_o^{(\tau_*)}} f_o(\tau_i)$$

Hence, the algorithm would assign the overall risk of trajectory τ_* as at most

$$\widetilde{\epsilon} \leq \sum_{o \in O, \tau_i \in S_o^{(\tau_*)}} f_o(\tau_i)$$

Because M_h greedily expands nodes in order of computed cost, it would only select a different trajectory if its computed cost $\hat{\epsilon} \leq \widetilde{\epsilon}$. We know that M_h can only overestimate the cost of a trajectory (due to not taking into account the optimal set of past collisions), not underestimate, so the cost of the trajectory it returns is at most $\hat{\epsilon}$. Finally, since $f_o(\tau_*) = \max_{\tau_i \in S_o^{(\tau_*)}} f_o(\tau_i)$ and

$$\epsilon_* = \sum_{o \in O} f_o(\tau_*) = \sum_{o \in O} \max_{\tau_i \in S_o^{(\tau_*)}} f_o(\tau_i)$$

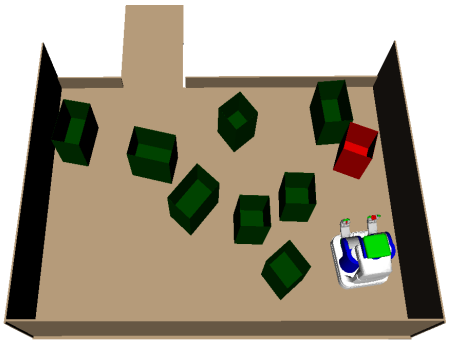

Figure 3.6: The robot is tasked to pick up the red box and carry it out of the room through the hallway at the top. The green boxes (of which there are between 8 and 24) are obstacles with known position but Gaussian distributed extents. This is then discretized into shadows for 3 risk levels.

we are left with the following bound:

$$\epsilon \leq \hat{\epsilon} \leq \widetilde{\epsilon} \leq \epsilon_* + \sum_{o \in O} (|S_o^{(\tau_*)}| - 1) f_o(\tau_*)$$

□

3.6 Empirical Results

In this section we compare our algorithm to various baselines and illustrate how the collision horizon parameter infuences behavior. We consider two baselines: First, we implemented the naive and commonly used approach of setting all the shadows to be equal, often referred to as constructing buffers, and then we ran a normal motion planner on it. Our implementation iteratively adapts the buffer size to select the smallest risk level that allows a solution. Second, we compare to a method proposed by [78] which samples obstacle instances from the distribution and then uses an approximate minimum constraint removal planner to find the path that collides with the fewest sampled obstacles. Note that avoiding 90% of samples does not guarantee a collision rate of less than 10%. In fact, the number of samples required for such an empirical collision rate estimate to be useful is quite large and depends on the dimension of the space. As such, we slightly modify it to construct each shadow as an individual obstacle rather than sampling actual obstacle instances. This ensures the soundness of the algorithm while also reducing the runtime due to needing fewer obstacles.

3.6.1 Moving Boxes Domain

Our first domain is a pick-and-place motion planning problem among uncertain obstacles. An example problem instance is depicted in Figure 3.6. We sample approximately 600 robot base poses and draw edges to form a graph embedded in the configuration space of the robot base. We then sample up to 4 feasible grasps, each of which creates an edge to a copy of the original graph (a copy is necessary because the collisions at each node are different based on whether the robot is holding a box and how it is grasping it). The performance of each method on this domain is compared in Figure 3.7. We also measured the effect of the collision horizon on these performance metrics, depicted in

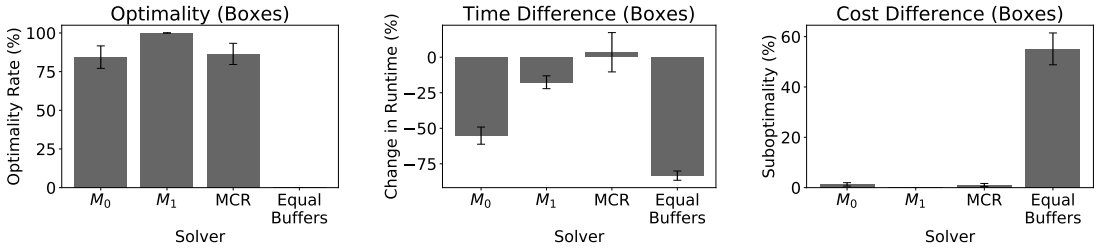

Figure 3.7: The optimality rate (percent of problem instances where the algorithm returns an optimal solution), runtime, and planning risk of each method. Runtime and cost are depicted as the difference compared to the optimal planner M_{24} to control for the variance in difficulty of different problem instances. Note that the Equal Buffers algorithm, which assigns equal risk to every obstacle, was not able to find the optimal solution in any problem instance.

Figure 3.8. We find that our algorithm is already near-optimal at $h = 0$, and the gap becomes

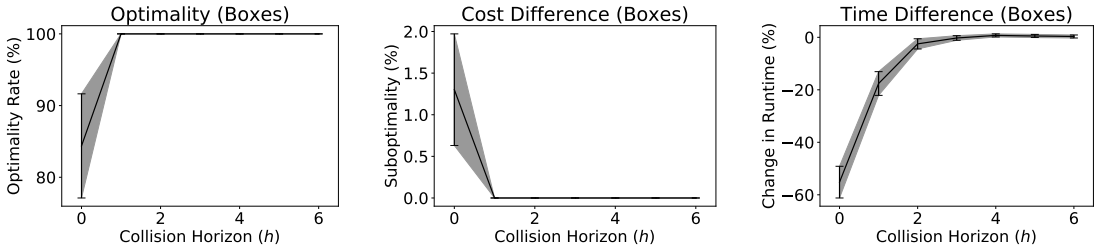

Figure 3.8: The optimality, runtime, and planning risk of M_h for each collision horizon. Runtime and cost are depicted as the difference compared to the optimal planner M_{24} to control for the variance in difficulty of different problem instances. Increasing the collision horizon past 6 up to 24 shows no noticeable change in behavior, so we have cropped the graphs for clarity.

entirely closed with $h = 1$. This suggests that this domain tends to have a very small collision horizon. The obstacle-sampling approach behaves similarly to M_0, which is unsurprising since the minimum constraint removal planner used is equivalent to M_0. The MCR algorithm runs slower, however, because it cannot take advantage of the fact that the obstacles correspond to quantile

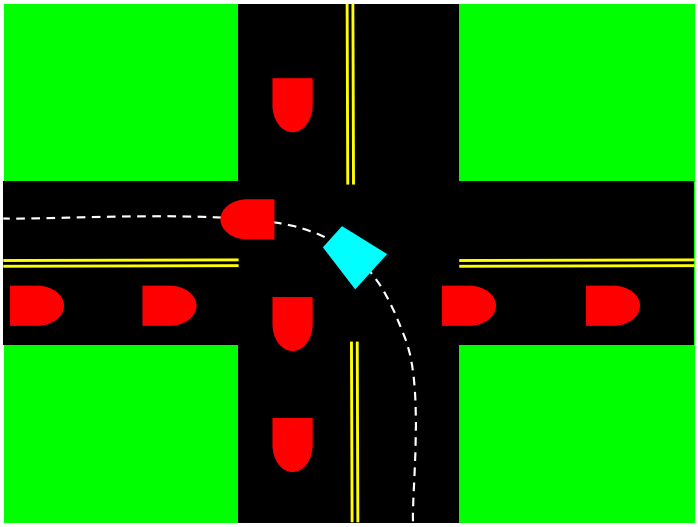

Figure 3.9: The robot (blue trapezoidal vehicle) making an unprotected left turn along the dotted white curve. Each obstacle (red rounded vehicles) exists in space-time and has uncertain speed. There is cross traffic going to the right blocking the robot's path before entering the intersection, oncoming traffic going downwards blocking the robot's path before exiting the intersection, and an obstacle vehicle in front. The robot must choose when it is safest to cut between vehicles, keeping in mind that going too fast risks collision with the front vehicle. There are a total of 12 obstacle vehicles.

shadows. Setting the shadows to be equal is a fast solution, but is very suboptimal. Overall, M_1 appears to be the most generally attractive, as it is optimal in every instance and slightly faster than the exact search. We also notice that the runtime did not increase as much as we would expect as h increased.

3.6.2 Driving Domain

We also evaluate our method on a driving domain. In this problem, the vehicle is attempting to make an unprotected left turn, and there is both cross and oncoming traffic. An example of this domain is depicted in Figure 3.9.

The geometric curve the vehicle will follow is fixed, but the vehicle has the option to proceed forward or wait at each timestep. Hence, the graph is a 2D lattice where one dimension is progress along the curve (40 steps) and the other dimension is time (100 timesteps). This graph is fed as input to the graph search algorithms, each of which returns a trajectory that indicates when the robot should be moving and when it should be stopping. Practically speaking, a solution trajectory makes two choices: which vehicles to cut between when entering the intersection, and which vehicles to cut between when leaving the intersection. The performance of these methods are compared in Figure 3.10. We also measured the effect of the collision horizon on these performance metrics,

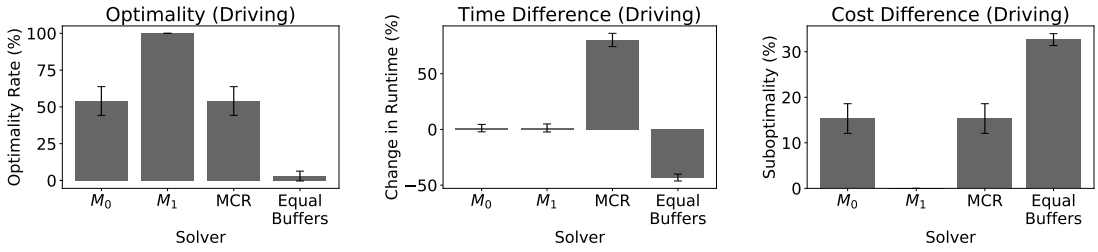

Figure 3.10: The optimality rate (percent of problem instances where the algorithm returns an optimal solution), runtime, and planning risk of each method. Runtime and cost are depicted as the difference compared to the optimal planner M_{12} to control for the variance between problem instances.

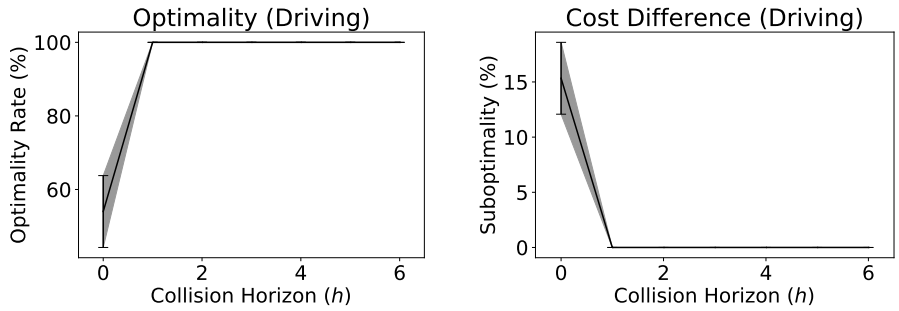

Figure 3.11: The optimality and planning risk of M_h for each collision horizon. Cost is depicted as the percent difference compared to the optimal planner M_{12} to control for the variance in difficulty between problem instances. There was no significant difference in runtime across different values of h, so the runtime graph is omitted. Increasing the collision horizon past 6 shows no noticeable change in behavior, so we have cropped the graphs for clarity.

depicted in Figure 3.11. We find that M_0 and running MCR on sampled obstacles produce plans of similar quality, although M_0 is significantly faster. As before, setting the shadows to be equal is suboptimal in nearly all of the problems in this domain because there are too many obstacles, so it must choose an overly conservative shadow for each one. Overall, M_1 appears to be the most generally attractive option, as it is optimal in every instance, although in this domain increasing the collision horizon does not appear to increase runtime significantly.

3.6.3 Minimum Constraint Removal for Manipulation Planning

We also evaluate our algorithm as a minimum constraint removal planner. Our experimental domain is identical to the one in Section 3.6.1, but the obstacles are deterministic and the task is instead to find the path with the fewest collisions. This task is very practically relevant in manipulation planning domains to determine which obstacles must be moved out of the way in order to perform

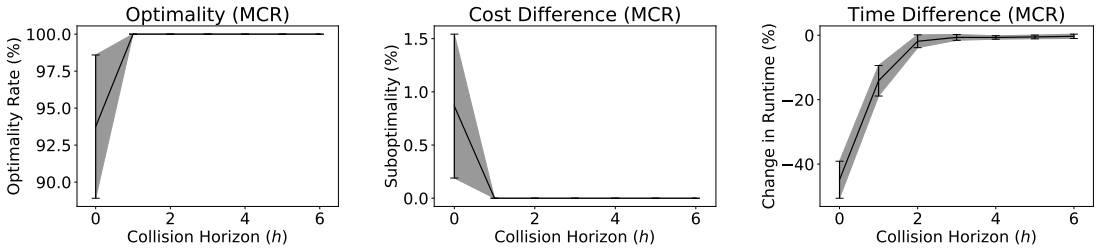

Figure 3.12: The optimality, runtime, and planning risk of M_h for each collision horizon. Runtime and cost are depicted as the difference compared to the optimal planner M_{24} to control for the variance in difficulty of different problem instances. Increasing the collision horizon past 6 up to 24 shows no noticeable change in behavior, so we have cropped the graphs for clarity.

a given operation.

As described before, Hauser presented two algorithms, a greedy planner and an exact planner, which are equivalent to M_0 and M_{24}, respectively (note that M_{24} is equivalent to M_∞ when there are at most 24 obstacles) [78]. Our algorithm is compared for different settings of the collision horizon in Figure 3.12.

Similar to minimum-risk planning, we find that M_0 is already near optimal, and that M_1 closes the gap. As a result, it is unclear whether the collision horizon is bounded for this domain, or if it is just highly likely to be small. As before, M_1 strikes a good balance of optimal performance and quick runtime.

3.7 Conclusion and Future Work

We have shown that while searching a graph for minimal risk plans is NP-hard, it becomes tractable when the collision horizon is bounded. Furthermore, we present a practical algorithm that efficiently finds optimal plans for fixed collision horizon and finds approximately-optimal plans with a natural suboptimality bound when the collision horizon is higher. This demonstrates that approximate planning under obstacle uncertainty is tractable in practical domains, which can lead to improved robustness in many robotic planning domains. Furthermore, it shows that the collision horizon is the source of the hardness of the problem, suggesting that the field should look towards this parameter to find further improvements and applications.

One caveat is that the runtime of the algorithm scales poorly with the collision horizon, although well with general problem size in domains with limited collision horizon. A potential direction for future work would be to further explore the relationship between the collision horizon and the hardness of the problem. In particular, our result provides a runtime with n in the base of the exponential dependency on the collision horizon, placing it in the complexity class XP instead of FPT (fixed-parameter tractability), which would require a runtime of the form $f(h)poly(n)$. Either

finding more efficient algorithms that reduce the exponential runtime to one with a fixed base, ideally something like 2^h rather than the n^h our algorithm has, or showing that such an algorithm does not exist would be a significant step in understanding the nature of these problems.

Another limitation is that our algorithm operates on problems with discrete risk levels. In practice, this is reasonable because the obstacle shadows can be discretized based on the precision that is required. However, the number of risk levels has a substantial effect on runtime especially with higher collision horizons. A line of future work would be to generalize this approach to a continuous notion of risk levels.

Finally, we have shown that some realistic domains tend to have small collision horizons. An interesting open question is what determines this, and whether we can define special classes of problems that have provably fixed collision horizon.

Chapter 4

Log-Concave Maximum Likelihood

4.1 Introduction

A distribution on $\mathbb{R}^d$ is log-concave if the logarithm of its probability density function is concave:

Definition 26 (Log-concave Density)**.** *A probability density function $f : \mathbb{R}^d \to \mathbb{R}_+$, $d \in \mathbb{Z}_+$, is called* log-concave *if there exists an upper semi-continuous concave function $\phi : \mathbb{R}^d \to [-\infty, \infty)$ such that $f(x) = e^{\phi(x)}$ for all $x \in \mathbb{R}^d$. We will denote by $\mathcal{F}_d$ the set of upper semi-continuous, log-concave densities with respect to the Lebesgue measure on $\mathbb{R}^d$.*

Log-concave densities form a broad nonparametric family encompassing a wide range of fundamental distributions, including the uniform, normal, exponential, logistic, extreme value, Laplace, Weibull, Gamma, Chi and Chi-Squared, and Beta distributions (see, e.g., [11]). Log-concave probability measures have been extensively investigated in several scientific disciplines, including economics, probability theory and statistics, computer science, and geometry (see, e.g., [123, 5, 100, 132, 118]). The problem of *density estimation* for log-concave distributions is of central importance in the area of non-parametric estimation (see, e.g., [132, 118, 117]) and has received significant attention during the past decade in statistics [41, 59, 57, 38, 88, 12, 75] and computer science [34, 35, 2, 28, 49, 53, 31].

One reason the class of log-concave distributions has attracted this attention, both from the theoretical and practical communities, is that log-concavity is a very natural "shape constraint," which places significantly fewer assumptions on the distribution in question than most parameterized classes of distributions. In extremely high-dimensional settings when the amount of available data is not too much larger than the dimensionality, fitting a multivariate Gaussian (or some other parametric distribution) to the data might be all one can hope to do. For many practical settings, however, the dimensionality is modest (e.g., 5-20) and the amount of data is significantly larger (e.g., hundreds of thousands or millions). In such settings, making a strong assumption on the parametric

form of the underlying distribution is unnecessary—there is sufficient data to fit a significantly broader class of distributions, and log-concave distributions are one of the most natural such classes. From a practical perspective, even in the univariate setting, computing the log-concave density that maximizes the likelihood of the available data is a useful primitive, with the R implementation of Rufibach and Duembgen having over 39,000 downloads [60]. As we discuss below, the amount of data required to *learn* a log-concave distribution scales exponentially in the dimension, in contrast to most parametric classes of distributions. Nevertheless, for the many practical settings with modest dimensionality and large amounts of data, there *is* sufficient data to learn. The question now is computational: how does one compute the best-fit log-concave distribution? We focus on this algorithmic question:

Is there an efficient algorithm to compute the log-concave MLE for datapoints in $\mathbb{R}^d$?

Obtaining an understanding of the above algorithmic question is of interest for a number of reasons. First, the log-concave MLE is *the* prototypical statistical estimator for the class, is fully automatic (in contrast to kernel-based estimators, for example), and was very recently shown to achieve the minimax optimal sample complexity for the task of learning a log-concave distribution (up to logarithmic factors) [31, 42]. The log-concave MLE also has an intriguing geometry that is of interest from a purely theoretical standpoint [41, 113]. Developing an efficient algorithm for computing the log-concave MLE is of significant theoretical interest, and would also allow this general non-parametric class of distributions to be leveraged in the many practical settings where the dimensionality is moderate and the amount of data is large. We refer the reader to the recent survey [117] for a more thorough justification for why the log-concave MLE is a desirable distribution to compute.

4.1.1 Our Results and Techniques

The main result of this paper is the first efficient algorithm to compute the multivariate log-concave MLE. For concreteness, we formally define the log-concave MLE:

Definition 27 (Log-concave MLE)**.** *Let $X_1, \ldots, X_n \in \mathbb{R}^d$. The log-concave MLE, $\widehat{f}_n = \widehat{f}_n(X_1, \ldots, X_n)$, is the density $\widehat{f}_n \in \mathcal{F}_d$ which maximizes the log-likelihood $\ell(f) \stackrel{\text{def}}{=} \sum_{i=1}^n \ln(f(X_i))$ over $f \in \mathcal{F}_d$.*

As shown in [41], the log-concave MLE $\widehat{f}_n$ exists and is unique. Our main result is the first efficient algorithm to compute it up to any desired accuracy.

Theorem 11 (Main Result)**.** *Fix $d \in \mathbb{Z}_+$ and $0 < \epsilon, \tau < 1$. There is an algorithm that, on input any set of points $X_1, \ldots, X_n$ in $\mathbb{R}^d$, and $0 < \epsilon$, $\tau < 1$, runs in $\mathrm{poly}(n, d, 1/\epsilon, \log(1/\tau))$ time and with probability at least $1 - \tau$ outputs a succinct description of a log-concave density $h^* \in \mathcal{F}_d$ such that $\ell(h^*) \geq \ell(\widehat{f}_n) - \epsilon$.*

Our algorithm does *not* require that the input points $X_1, \ldots, X_n$ in $\mathbb{R}^d$ are i.i.d. samples from a log-concave density, i.e., it efficiently solves the MLE optimization problem for any input set of

points. We also note that the succinct output description of h^* allows for both efficient evaluation and efficient sampling. That is, we can efficiently approximate the density at a given point (within multiplicative accuracy), and efficient sample from a distribution that is close in total variation distance.

Recent work [31, 42] has shown that the log-concave MLE is minimax optimal, within a logarithmic factor, with respect to squared Hellinger distance. In particular, the minimax rate of convergence with n samples is $\tilde{\Theta}_d\big(n^{-2/(d+1)}\big)$. Combining this sample complexity bound with our Theorem 11, we obtain the first sample near-optimal and computationally efficient *proper* learning algorithm for multivariate log-concave densities. See Theorem 15 in section 4.8.

Technical Overview Here we provide an overview of our algorithmic approach. Notably, our algorithm does *not* require the assumption that the input points are samples from a log-concave distribution. It runs in $\mathrm{poly}(n, d, 1/\epsilon)$ on *any* set of input points and outputs an ϵ-accurate solution to the log-concve MLE. Our algorithm proceeds by convex optimization: We formulate the problem of computing the log-concave MLE of a set of n points in $\mathbb{R}^d$ as a convex optimization problem that we solve via an appropriate first-order method. It should be emphasized that one needs to overcome several non-trivial technical challenges to implement this plan.

The first difficulty lies in choosing the right (convex) formulation. Previous work [41] considered a convex formulation of the problem, though that formulation seems to inherently lead to an exponential time algorithm. Given our convex formulation, a second difficulty arises: we do not have direct access to the (sub-)gradients of the objective function and the naive algorithm to compute a subgradient at a point takes exponential time. Hence, a second challenge is how to obtain an efficient algorithm for this task. One of our main contributions is a randomized polynomial time algorithm to approximately compute a subgradient of the objective function. Our algorithm for this task leverages structural results on log-concave densities established in [31] combined with classical algorithmic results on approximating the volume of convex bodies and uniformly sampling from convex sets [84, 99, 98].

We now proceed to explain our convex optimization formulation. Our starting point is a key structural property of the log-concave MLE, shown in [41]: The logarithm of the log-concave MLE $\ln \widehat{f}_n$, is a "tent" function, whose parameters are the values $y_1, \ldots, y_n$ of the log density at the n input points $x^{(1)}, \ldots, x^{(n)}$, and whose log-likelihoods correspond to polyhedra. Our conceptual contribution lies in observing that while tent distributions are not an exponential family, they "locally" retain many properties of exponential families (Definition 29). This high-level similarity can be leveraged to obtain a convex formulation of the log-concave MLE that is similar in spirit to the standard convex formulation of the exponential family MLE [130]. Specifically, we seek to maximize the log-likelihood of the probability density function obtained by normalizing the log-concave function whose logarithm is the convex hull of the log densities at the samples. This objective function is a concave function of the parameters, so we end up with a (non-differentiable) convex optimization

problem. The crucial observation is that the subgradient of this objective at a given point y is given by an expectation under the current hypothesis density at y.

Given our convex formulation, we would like to use a first-order method to efficiently find an ϵ-approximate optimum. We note that the objective function is not differentiable everywhere, hence we need to work with subgradients. We show that the subgradient of the objective function is bounded in ℓ_2-norm at each point, i.e., the objective function is Lipschitz. Another important structural result (Lemma 2) allows us to essentially restrict the domain of our optimization problem to a compact convex set of appropriately bounded diameter $D = \text{poly}(n, d)$. This is crucial for us, as the diameter bound implies an upper bound on the number of iterations of a first-order method. Given the above, we can in principle use a projected subgradient method to find an approximate optimum to our optimization problem, i.e., find a log-concave density whose log-likelihood is ϵ-optimal.

It remains to describe how we can efficiently compute a subgradient of our objective function. Note that the log density of our hypothesis can be considered as an unbounded convex polytope. The previous approach to calculate the subgradient in [41] relied on decomposing this polytope into faces and obtaining a closed form for the underlying integral over these faces (that gives their contribution to the subgradient). However, this convex polytope is given by n vertices in d dimensions, and therefore the number of its faces can be $n^{\Omega(d)}$. So, such an algorithm cannot run in polynomial time.

Instead, we note that we can use a linear program (see proof of Lemma 1) to evaluate a function proportional to the hypothesis density at a point in time polynomial in n and d. To use this oracle for the density in order to produce samples from the hypothesis density, we use Markov Chain Monte Carlo (MCMC) methods. In particular, we use MCMC to draw samples from the uniform distribution on super-level sets and estimate their volumes. With appropriate rejection sampling, we can use these samples to obtain samples from a distribution that is close to the hypothesis density. See Lemma 3. (We note that it does not suffice to simply run a standard log-concave density sampling technique such as hit-and-run [97]. These random walks require a hot start which is no easier than the sampling technique we propose.)

Since the subgradient of the objective can be expressed as an expectation over this density, we can use these samples to sample from a distribution whose expectation is close to a subgradient. We then use stochastic subgradient descent to find an approximately optimal solution to the convex optimization problem. The hypothesis density this method outputs has log-likelihood close to the maximum.

4.1.2 Related Work

There are two main strands of research in density estimation. The first one concerns the learnability of high-dimensional parametric distributions, e.g., mixtures of Gaussians. The sample complexity of learning parametric families is typically polynomial in the dimension and the challenge is to design computationally efficient algorithms. The second research strand — which is the focus of this

paper — considers the problem of learning a probability distribution under various non-parametric assumptions on the shape of the underlying density, typically focusing on the univariate or small constant dimensional regime. There has been a long line of work in this vein within statistics since the 1950s, dating back to the pioneering work of [72] who analyzed the MLE of a univariate monotone density. Since then, shape constrained density estimation has been an active research area with a rich literature in mathematical statistics and, more recently, in computer science. The reader is referred to [16] for a summary of the early work and to [74] for a recent book on the subject.

The standard method used in statistics for density estimation problems of this form is the MLE. See [24, 110, 134, 76, 73, 21, 22, 67, 33, 14, 80, 59, 13, 69, 15, 90, 132, 38, 88, 12, 75, 31] for a partial list of works analyzing the MLE for various distribution families. During the past decade, there has been a body of algorithmic work on shape constrained density estimation in computer science with a focus on both sample and computational efficiency [45, 46, 44, 34, 35, 36, 1, 2, 51, 52, 43, 50, 53, 54]. The majority of this literature has studied the univariate (one-dimensional) setting which is by now fairly well-understood for a wide range of distributions. On the other hand, the *multivariate* setting is significantly more challenging and wide gaps in our understanding remain even for $d = 2$.

For the specific problem of learning a log-concave distribution, a line of work in statistics [41, 59, 57, 38, 12] has characterized the global consistency properties of the log-concave multivariate MLE. Regarding finite sample bounds, [88, 42] gave a sample complexity *lower bound* of $\Omega_d\left((1/\epsilon)^{(d+1)/2}\right)$ for $d \in \mathbb{Z}_+$ that holds for *any* estimator, and [88] gave a near-optimal sample complexity *upper bound* for the log-concave MLE for $d \leq 3$. [53] established the first finite sample complexity upper bound for learning multivariate log-concave densities under global loss functions. Their estimator (which is different than the MLE and seems hard to compute in multiple dimensions) learns log-concave densities on $\mathbb{R}^d$ within squared Hellinger loss ϵ with $\tilde{O}_d\left((1/\epsilon)^{(d+5)/2}\right)$ samples. [31] showed a sample complexity upper bound of $\tilde{O}_d\left((1/\epsilon)^{(d+3)/2}\right)$ for the multivariate log-concave MLE with respect to squared Hellinger loss, thus obtaining the first finite sample complexity upper bound for this estimator in dimension $d \geq 4$. Building on their techniques, this bound was subsequently improved in [42] to a near-minimax optimal bound of $\tilde{O}_d\left((1/\epsilon)^{(d+1)/2}\right)$. Alas, the computational complexity of the log-concave MLE has remained open in the multivariate case. Finally, we note that a recent work [47] obtained a non-proper estimator for multivariate log-concave densities with sample complexity $\tilde{O}_d((1/\epsilon)^{d+2})$ (i.e., at least quadratic in that of the MLE) and runtime $\tilde{O}_d((1/\epsilon)^{2d+2})$.

On the empirical side, recent work [111] proposed a non-convex optimization approach to the problem of computing the log-concave MLE, which seems to exhibit superior performance in practice in comparison to previous implementations (scaling to 6 or higher dimensions). Unfortunately, their method is of a heuristic nature, in the sense that there is no guarantee that their solution will converge to the log-concave MLE.

4.2 Preliminaries

Notation. We denote by $X_1, \ldots, X_n \in \mathbb{R}^d$ the sequence of samples. We denote by $S_n = \text{Conv}(\{X_i\}_{i=1}^n)$ the convex hull of $X_1, \ldots, X_n$, and by X the $d \times n$ matrix with columns vectors $X_1, \ldots, X_n$. We write $\mathbb{1}$ for the all-ones vector of the appropriate length. For a set $Y \subset Z$, $\mathbb{1}_Y$ denotes the indicator function for Y.

Tent Densities. We start by defining tent functions and tent densities:

Definition 28 (Tent Function)**.** *For $y = (y_1, \ldots, y_n) \in \mathbb{R}^n$ and a set of points $X_1, \ldots, X_n$ in $\mathbb{R}^d$, we define the tent function $h_{X,y} : \mathbb{R}^d \to \mathbb{R}$ as follows:*

$$h_{X,y}(x) = \begin{cases} \max\{z \in \mathbb{R} \text{ such that } (x,z) \in \text{Conv}(\{(X_i, y_i)\}_{i=1}^n)\} & \text{if } x \in S_n \\ -\infty & \text{if } x \notin S_n \end{cases}$$

The points (X_i, y_i) are referred to as *tent poles*. See Figure 4.1 for the graph of an example tent function.

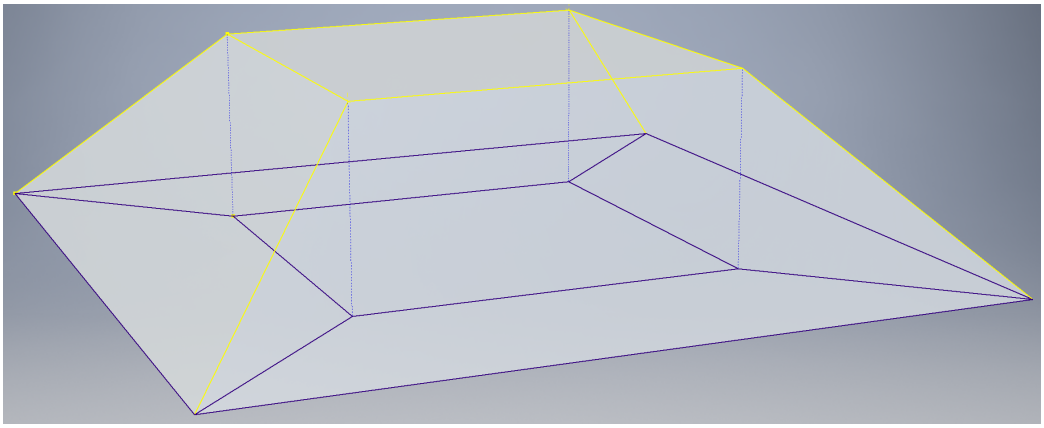

Figure 4.1: An example of a tent function and its corresponding regular subdivision. Notice that the regular subdivision is *not* a regular triangulation.

Let $p_{X,y}(x) = c \exp(h_{X,y}(x))$ with c chosen such that $p_{X,y}(x)$ integrates to one. We refer to $p_{X,y}$ as a *tent density* and the corresponding distribution as a *tent distribution*. Note that the support of a *tent distribution* must be within the convex hull of $X_1, \ldots, X_n$. For the remainder of the paper, we choose a scaling such that $\mathbb{1}^T y = 0$. This scaling is arbitrary, and has no significant effect on either the algorithm or its analysis.

Tent densities are notable because they contain solutions to the log-concave MLE [41]. The solution to the log-concave MLE over $X_1, \ldots, X_n$ is always a tent density, because tent densities with tent poles $X_1, \ldots, X_n$ are the minimal log-concave functions with log densities $y_1, \ldots, y_n$ at points $X_1, \ldots, X_n$.

The algorithm which we present can be thought of as an optimization over tent functions. In Section 4.4.2, we will show that tent distributions retain important properties of exponential families which will be useful to establish the correctness of our algorithm.

Regular Subdivisions. Given a tent function $h_{X,y}$ with $h_{X,y}(X_i) = y_i$, its associated *regular subdivision* $\Delta_{X,y}$ of X is a collection of subsets of $X_1, \ldots, X_n \in \mathbb{R}^d$ whose convex hulls are the regions of linearity of $h_{X,y}$. See Figure 4.1 for an illustration of a tent function and its regular subdivision. We refer to these polytopes of linearity as *cells*. We say that $\Delta_{X,y}$ is a *regular triangulation* of X if every cell is a $d-$dimensional simplex.

It is helpful to think of regular subdivisions in the following way: Consider the hyperplane H in $\mathbb{R}^{d+1}$ obtained by fixing the last coordinate. Consider the function $h_{X,y}$ as a polytope and project each face onto H. Each cell is a projection of a face, and together the cells partition the convex hull of $X_1, \ldots, X_n$. Observe that regular subdivisions may vary with y. Figure 4.2 provides one example of how changing the y vector changes the regular subdivision.

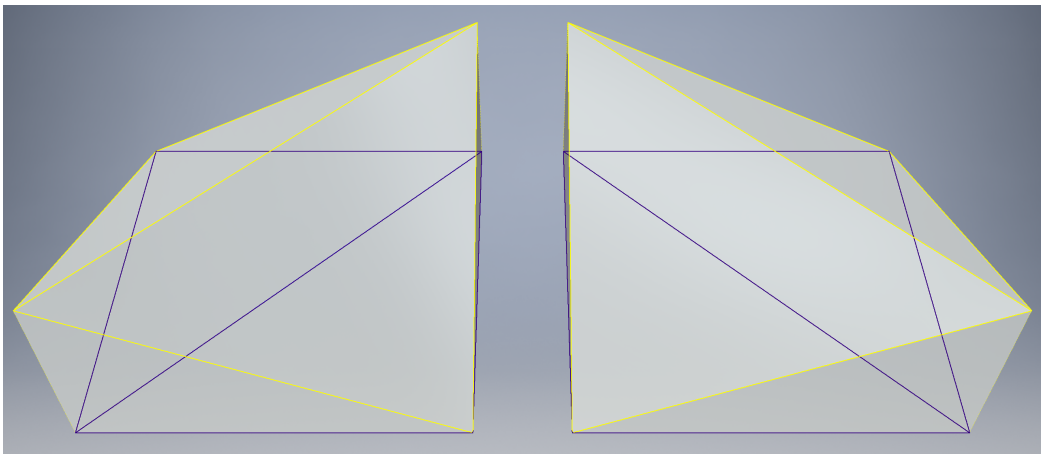

Figure 4.2: Changing the height of the tent poles can change the induced regular subdivision (shown in purple).

For a given regular triangulation Δ, the associated *consistent neighborhood* N_Δ is the set of all $y \in \mathbb{R}^n$, such that $\Delta_{X,y} = \Delta$. That is, consistent neighborhoods are the sets of parameters where the *regular triangulation* remains fixed. Note that these neighborhoods are open and their closures cover the whole space. See Figure 4.2 for an example of how crossing between consistent neighborhoods results in different subdivisions. We note that for fixed X, when y is chosen in general position, $\Delta_{X,y}$ is always a regular triangulation.

4.3 Exponential Families

In this section, we give a brief overview of exponential families that covers just the material necessary to appreciate the connection between exponential families and the log-concave maximum likelihood problem. We refer to [130] for a more complete treatment of exponential families.

An *exponential family* parameterized by $\theta \in \mathbb{R}^n$ with *sufficient statistic* $T(x)$, with carrier density h measurable and non-negative is a family of probability distributions of the form

$$p_\theta(x) = \exp(\langle T(x), \theta \rangle - A(\theta))h(x).$$

The *log-partition* function $A(\theta)$ is defined to normalize the integral of the density

$$A(\theta) = \log \int \exp(\langle T(x), \theta \rangle)h(x)dx.$$

It makes sense to restrict our attention to values of θ that give a valid probability density. The set of *Canonical Parameters* Θ is defined such that $\Theta = \{\theta \mid A(\theta) < \infty\}$.

We say that an exponential family is *minimal* if $\theta_1 \neq \theta_2$ implies $p_{\theta_1} \neq p_{\theta_2}$. This is necessary and sufficient for statistical identifiability.

One reason exponential families are well studied is that we have an algorithm that computes the maximum likelihood estimate via a convex program.

The maximum likelihood parameters $\theta^\star$ for a set of iid samples $X_1, \ldots, X_n$ are:

$$\begin{aligned}\theta^\star &= \arg\max_\theta \prod_i p_\theta(X_i) = \arg\max_\theta \log \prod_i p_\theta(X_i) \\ &= \arg\max_\theta \sum_i \langle T(X_i), \theta \rangle - nA(\theta) - \sum_i \log h(x_i) = \arg\max_\theta \left\langle \frac{1}{n}\sum_i T(X_i), \theta \right\rangle - A(\theta) \qquad (4.1)\end{aligned}$$

We refer to the optimization in Equation (4.1) as the *exponential maximum likelihood optimization.* The last equation helps highlight why $T(x)$ is referred to as the sufficient statistic. No other information is needed about the data points to compute both the likelihood and the maximum likelihood estimator.

One reason why exponential families are important is that the geometry of the optimization in Equation (4.1) has several nice properties.

Fact 1. *$A(\theta)$ of exponential families satisfies the following properties: (a) $A(\theta) \in C^\infty$ on Θ. (b) $A(\theta)$ is convex. (c) $\Delta A(\theta) = \mathbb{E}_{x \sim p(\theta)}[T(x)]$. (d) If the exponential family is minimal, $A(\theta)$ is strictly convex.*

Note that properties $(b), (c)$ are very similar to the definition of locally exponential families. The fact that tent distributions maintain some of these properties is exactly what enables the efficient algorithm in this paper.

4.4 Locally Exponential Convex Programs

In this section, we lay the foundations for the algorithm presented in the next section. We present the "locally" exponential form of tent distributions and show it has the necessary properties to enable efficient computation of the log-concave MLE. Though they form a broader class of distributions, "locally" exponential distributions share some important properties of exponential families. Namely, the log-likelihood optimization is convex, and the expectation of the sufficient statistic is a subgradient. This will allows us to formulate a convex program which we will be able to solve in polynomial time.

Definition 29. *Let T be some function (possibly parametrized by y) and let $q_y = \exp\left(\langle T(x), y\rangle - A(y)\right)$ be a family of probability densities parametrized by y with $A(y)$ acting to normalize the density so it integrates to* 1*. We say that the family $\{q_y\}$ is* locally-exponential *if the following hold: (1) $A(y)$ is convex in y, and (2) $\mathbb{E}_{x\sim q_y}[T(x)] \in \partial_y A(y)$.*

Note that the above definition differs from an exponential family in that for exponential families T may not depend on y.

In this section, we derive a sufficient statistic, the *polyhedral statistic*, that shows that tent distributions are in fact locally exponential. More formally, we show:

Lemma 1. *For tent poles $X_1, \ldots, X_n$, there exists a function $T_{X,y} : \mathbb{R}^d \to \mathbb{R}^n$ (the polyhedral statistic) such that $p_{X,y}(x) = \exp\left(\langle T_{X,y}(x), y\rangle - A(y)\right)$ corresponds to the family of tent-distributions such that $\{p_{X,y}\}$ is locally exponential. Furthermore, $T_{X,y}$ is computable in time* $\mathrm{poly}(n, d)$.

Since we know that the log-concave MLE is a tent distribution, and all tent-distributions are log-concave, we know that the optimum of the maximum likelihood convex program in Equation (4.2) corresponds to the log-concave MLE.

$$\text{MLE of tents } = \max_y \sum_i h_{X,y}(X_i) - \log \int \exp h_{X,y}(x)dx = \max_y \sum_i y_i - A(y) \tag{4.2}$$

Combining the above with the fact that the sufficient statistic allows us to compute the stochastic subgradient suggests that Algorithm 3 can compute the log-concave MLE in polynomial time.

Algorithm 3 ComputeLogConcaveMLE($X_1, \ldots, X_n, \epsilon$

Input: $y \leftarrow 0$; $c \leftarrow 8n^2 d \log(2nd)$; $m \leftarrow \frac{2c^2}{\epsilon^2}$

Output: A set of log-likelihoods y that approximates the log-concave MLE.

1: s
2: **for** $i \leftarrow 1, m$ **do**
3: $\quad \eta \leftarrow c/\sqrt{i}$
4: $\quad s \sim p_{X,y}$ {Using Lemma 3}
5: $\quad y \leftarrow y + \eta\left(\frac{1}{n}\mathbb{1} - T_{X,y}(s)\right)$ {T computed via Lemma 1. $\frac{1}{n}\mathbb{1}$ follows from Equation (4.2)}
6: **end for**
7: **return** y

Proving lemma 1 and highlighting the connection to exponential families will be the focus of this section. Section 4.5 will fill in the remaining details by establishing polynomial time methods for sampling from tent distributions and bounding the number of iterations for stochastic gradient descent to converge.

A reader familiar with exponential families may note that Equation (4.2) and algorithm (3) share a superficial similarity with the exponential family maximum likelihood problem. We develop this connection further in section 4.4.1, generalizing some tools that were originally limited to exponential families. Note this applies even though the log-concave MLE is a *non-parametric* statistics problem.

4.4.1 Analogy Between Log-Concave MLE and Exponential Family MLE

In the case of exponential families, at each time step, the algorithm maintains a distribution (from the hypothesis class) and generates a single sample from this distribution. The sufficient statistic of the exponential family can then be used to compute a subgradient. The computational efficiency follows from the convexity of the log-likelihood function, and existence of efficient samplers and procedures for computing the sufficient statistic. We portray this stochastic gradient method for exponential families, together with the analogous form of our algorithm for log-concave distributions.

Exponential Family MLE	**Log-Concave MLE**
Optimization Formulation:	Optimization Formulation:
$\max_y \langle \mu, y \rangle - \log \int \exp\left(\langle T(x), y \rangle\right) dx$	$\max_y \langle \mathbb{1}, y \rangle - \log \int \exp\left(\langle T_{X,y}(x), y \rangle\right) dx$

Algorithm 4 Compute Exp. Family MLE	**Algorithm 5** Compute Log-Concave MLE
Input: Points $X_1, ...X_n \in \mathbb{R}^d$ whose likelihood is to be maximized.	**Input:** Points $X_1, ...X_n \in \mathbb{R}^d$ whose likelihood is to be maximized.
Output: Parameter vector y for the exponential family that [approximately] maximizes the likelihood of the dataset.	**Output:** Log-likelihood vector y for the tent function with poles at $X_1, ..., X_n$ that [approximately] maximizes their likelihood.
$y \leftarrow y_{init}$	$y \leftarrow 0$
for all $i \in [m]$ **do**	**for all** $i \leftarrow 1, m$ **do**
$s \sim p(y)$ {sample}	$s \sim p(X, y)$
$y \leftarrow y + \eta_i \left(\mu - T(s)\right)$ {subgradient}	$y \leftarrow y + \eta_i \left(\frac{1}{n}\mathbb{1}_n - T_{X,y}(s)\right)$
end for	**end for**
return y	**return** y

4.4.2 The Polyhedral Sufficient Statistic

Consider a regular triangulation Δ corresponding to tent distribution parametrized by X and y. The *polyhedral statistic* is the function

$$T_{X,y}(x) : S_n \to [0,1]^n,$$

that expresses x as a convex combination of corners of the cell containing x in Δ_y. That is $x = XT_{X,y}(x)$ where $||T_y(x)||_1 = 1$ and $T_y(x)_i = 0$ if X_i is not a corner of the cell containing x. The polyhedral statistic gives an alternative way of writing tent functions and tent densities:

$$h_{X,y}(x) = \langle T_y(x), y \rangle \qquad p_{X,y}(x) = \exp(\langle T_y(x), y \rangle) \ .$$

If we restrict y such that $\sum_i y_i = 0$ and define $A(y) = \log \int_x p_{X,y}(x)dx$, then we can see that for every consistent neighborhood N_Δ we have an exponential family of the form

$$\exp\left(\langle T_y(x), \theta \rangle - A(y)\right) \quad \text{for } \theta \in N_\Delta \ . \tag{4.3}$$

While Equation (4.3) shows how subsets of tent distributions are exponential families, it also helps highlight why tent distributions are *not* an exponential family. The sufficient statistic depends on y through the regular subdivision. This means that tent distributions do not admit the same factorized form as exponential families since the sufficient statistic depends on y.

Note that we can use any ordering of $X_1, \ldots, X_n$ to define the polyhedral sufficient statistic everywhere including on regular subdivisions that are *not* regular triangulations. Also note that, assuming that no $X_i = X_j, i \neq j$, eliminating the last coordinate using the constraint $\mathbb{1}_n^T \theta = 0$ makes each exponential family minimal. In other words, over regions where the regular subdivision does not change (for example the consistent neighborhoods), tent distributions are minimal exponential families. This means the set of tent distribution can be seen as the finite union of a set of minimal exponential families. We refer to Equation (4.4) as the exponential form for tent densities:

$$p_{X,y}(x) = \exp\left(\langle T_{X,y}(x), y\rangle - A(y)\right) \mathbb{1}_{S_n}(x). \tag{4.4}$$

Both the polyhedral statistic and tent density queries can be computed in polynomial time with the packing linear program presented in Equation (4.5). For a point x, the value of y yields the log-density and the vector α corresponds to polyhedral statistic.

$$\max y \text{ s.t. } (x, y) = \sum_i \alpha_i (X_i, y_i), \sum_i \alpha_i = 1, \alpha_i \geq 0 \tag{4.5}$$

Note that the above combined with tent distributions being exponential families on consistent neighborhoods gives us that the properties from Lemma 1 hold true on consistent neighborhoods. We extend the proof to the full result below.

Proof. Convexity follows by iteratively applying known operations that preserve convexity of a function. Since a sum of convex functions is convex (see, e.g., page 79 of [23]), it suffices to show that the function $G(y) = \ln(\int \exp(h_{X,y}(x))\mathrm{d}x)$ is convex. Since $h_{X,y}(x)$ is a convex function of y, by definition, $\exp(h_{X,y}(x))$ is log-convex as a function of y. Since an integral of log-convex functions is log-convex (see, e.g., page 106 of [23]), it follows that $\int \exp(h_y(x))\mathrm{d}x$ is log-convex. Therefore, G is convex. We have therefore established that Equation (4.2) is convex, as desired.

$\mathbb{E}_{x\sim p_{X,y}}[T_{X,y}(x)] \in \partial_y A(y)$: Note that when y is in the interior of a consistent neighborhood, the polyhedral statistic LP has a unique solution and $\mathbb{E}_{x\sim p_{X,y}}[T(x)] \in \partial_y A(y)$ (by Fact 1). When y is on the boundary the solution set to the LP corresponds to the convex hull of solutions corresponding to each adjacent consistent neighborhood. This corresponds to the convex hull of limiting gradients from each neighboring consistent neighborhood and is the set of subgradients. □

4.5 Algorithm and Analysis

Recall that we compute the log-concave MLE via a first-order method on the optimization formulation presented in Equation (4.2). The complete method is presented in Algorithm 3. The algorithm is based on the stochastic gradient computation presented in the previous section, a standard application of the stochastic gradient method, and a sampler that we describe later in this section. Theorem 11 follows from bounding the rate of convergence of the stochastic subgradient method and the efficiency of the sampling procedure. We outline these two components below.

4.5.1 The Stochastic Subgradient Method

Recall that algorithm 3 is simply applying the stochastic subgradient method to the following convex program with $\mathbb{1}^T y = 0$: $h(y) = \left\langle \frac{1}{n}\mathbb{1}_n, y \right\rangle - A(y)$. We require a slight strengthening of the following standard result, see, e.g., Theorem 3.4.11 in [58]:

Fact 2. *Let $\mathcal{C}$ be a compact convex set of diameter* $\mathrm{diam}(\mathcal{C}) < \infty$. *Suppose that the projections $\pi_{\mathcal{C}}$ are efficiently computable, and there exists $M < \infty$ such that for all $y \in \mathcal{C}$ we have that $\|g\|_2 \leq M$ for all stochastic subgradients. Then, after $K = \Omega\left(M \cdot \mathrm{diam}(\mathcal{C}) \log(1/\tau)/\epsilon^2\right)$ iterations of the projected stochastic subgradient method (for appropriate step sizes), with probability at least $1 - \tau$, we have that $F\left(\bar{y}^{(K)}\right) - \min_{y \in \mathcal{C}} F(y) \leq \epsilon$, where $\bar{y}^{(K)} = (1/K)\sum_{i=1}^{K} y^{(i)}$.*

We note that Fact 2 assumes that, in each iteration, we can efficiently calculate an *unbiased* stochastic subgradient, i.e., a vector $g^{(k)}$ such that $\mathbb{E}[g^{(k)}] \in \partial_y F(y^{(k)})$. Unfortunately, this is not the case in our setting, because we can only *approximately* sample from log-concave densities. However, it is straightforward to verify that the conclusion of Fact 2 continues to hold if in each iteration we can compute a random vector $\widetilde{g}^{(k)}$ such that $\|\mathbb{E}[\widetilde{g}^{(k)}] - g^{(k)}\|_2 < \delta \stackrel{\text{def}}{=} \epsilon/(2\mathrm{diam}(\mathcal{C}))$, for some $g^{(k)} \in \partial_y F(y^{(k)})$. This slight generalization is the basic algorithm we use in our setting.

We now return to the problem at hand. We note that since T represents the coefficients of a convex combination $||T(x)|| < 1$ for all x, bounding M by 1.

Lemma 2 will show that $\mathrm{diam}(\mathcal{C}) = O(2n^2 d \log(2nd))$. This implies that if we let $c = 8n^2 d \log(2nd)$ and run SGD for $\frac{2c^2}{\epsilon^2}$ iterations, the resulting point will have objective value within ϵ of the log-concave MLE.

Lemma 2. *Let $X_1, \ldots, X_n$ be a set of points in $\mathbb{R}^d$ and $\hat{f}$ be the corresponding log-concave MLE. Then, we have that $R_\infty \stackrel{\text{def}}{=} \frac{\max_{i\in[n]} \hat{f}(X_i)}{\min_{i\in[n]} \hat{f}(X_i)} \leq (2nd)^{2nd}$. Converting to an ℓ_2 norm yields a bound on the diameter of $\mathcal{C}$:* $\mathrm{diam}(\mathcal{C}) \leq 2n^2 d \log(2nd)$.

Let us briefly sketch the proof of Lemma 2. The main idea is to show that if R_∞ were too high, then $\widehat{f}_n$ would have a lower likelihood than the uniform distribution on the convex hull of the samples S_n. More specifically, if the maximum value M of the density $\widehat{f}_n$ is large, then the volume

of the set $\{x \in \mathbb{R}^d : \widehat{f}_n(x) \geq M/R\}$ is small. For a fixed R, this set contains S_n and thus R_∞ must be large compared to $M\text{vol}(S_n)$. Since $\widehat{f}_n$ has likelihood at least as high as the uniform distribution over S_n, R must be small compared to $M\text{vol}(S_n)$. Combining these two observations yields a bound on R.

We now proceed with the complete proof.

Proof of Lemma 2. Let $V = \text{vol}(S_n)$ be the volume of the convex hull of the sample points and $M = \max_x \widehat{f}_n(x)$ be the maximum pdf value of the MLE. By basic properties of the log-concave MLE (see, e.g., Theorem 2 of [41]), we have that $\widehat{f}_n(x) > 0$ for all $x \in S_n$ and $\widehat{f}_n(x) = 0$ for all $x \notin S_n$. Moreover, by the definition of a tent function, it follows that $\widehat{f}_n$ attains its global maximum value and its global non-zero positive value in one of the points X_i.

We can assume without loss of generality that $\widehat{f}_n$ is not the uniform distribution on S_n, since otherwise $R_\infty = 1$ and the lemma follows. Under this assumption, we have that $R_\infty > 1$ or $\ln R_\infty > 0$, which implies that $M > 1/V$. The following fact bounds the volume of upper level sets of any log-concave density:

Fact 3 (see, e.g., Lemma 8 in [31])**.** *Let $f \in \mathcal{F}_d$ with maximum value M_f. Then for all $w > 0$, we have* $\text{vol}(L_f(M_f e^{-w})) \leq w^d/M_f$.

By Fact 3 applied to the MLE $\widehat{f}_n$, for $w = \ln R_\infty$, we get that $\text{vol}(L_{\widehat{f}_n}(M/R_\infty)) \leq (\ln R_\infty)^d/M$. Since the pdf value of $\widehat{f}_n$ at any point in the convex hull S_n is at least that of the smallest sample point X_i, i.e., M/R_∞, it follows that S_n is contained in $L_{\widehat{f}_n}(M/R_\infty)$. Therefore, $V \leq (\ln R_\infty)^d/M$. On the other hand, the log-likelihood of $\widehat{f}_n$ is at least the log-likelihood of the uniform distribution U_{S_n} on S_n. Since at least one sample point X_i has pdf value $\widehat{f}_n(X_i) = M/R_\infty$ and the other $n-1$ sample points have pdf value $\widehat{f}_n(X_i) \leq M$, we have that $\ln(M/R_\infty) + (n-1)\ln M \geq \ell(\widehat{f}_n) \geq \ell(U_{S_n}) = n\ln(1/V)$, or $n\ln M - \ln R_\infty \geq -n\ln V$, and therefore $\ln(MV) \geq (\ln R_\infty)/n$. This gives that $R_\infty^{1/n} \leq MV$. Combining this expression with $V \leq (\ln R_\infty)^d/M$ from above yields that $R_\infty \leq (\ln R_\infty)^{nd}$.

Since $\ln x < x$, $x \in \mathbb{R}$, setting $x = R_\infty^{\frac{1}{2nd}}$ gives that $\ln R_\infty < 2nd \cdot R_\infty^{\frac{1}{2nd}}$ or $(\ln R_\infty)^{nd} < (2nd)^{nd} \cdot R_\infty^{1/2}$. By the above, we deduce that $R_\infty \leq (2nd)^{nd} \cdot R_\infty^{1/2}$ or $R_\infty \leq (2nd)^{2nd}$. This completes the proof of Lemma 2. □

4.5.2 Efficient Sampling and Log-Partition Function Evaluation

In this section, we establish the following result, which gives an efficient algorithm for sampling from the log-concave distribution computed by our algorithm. Before we present the complete algorithm, we start with a sketch to orient the reader.

Sketch of Efficient Sampling and Log-Partition Function Evalaution

Lemma 3 (Efficient Sampling)**.** *There exist algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfying the following: Let $\delta, \tau > 0$, let $X = X_1, \ldots, X_n \in \mathbb{R}^d$, let $y \in \mathbb{R}^n$ be a parameter of a tent-density in exponential form. Then the following conditions hold:*

(1) On input X, y, δ, and τ, algorithm $\mathcal{A}_1$ outputs a random vector $Z \in \mathbb{R}^d$, distributed according to some probability distribution with density $\widetilde{\phi}$, such that $\|\widetilde{\phi} - p_{X,y}\|_1 = O(\delta)$, in time $\mathrm{poly}(n, d, \|y\|_\infty, 1/\delta, \log(1/\tau))$, with probability at least $1 - \tau$.

(2) On input X, y, δ, and τ, algorithm $\mathcal{A}_2$ outputs some $\gamma' > 0$, such that $\gamma'/(1 + O(\delta)) \leq \int \exp(h_{X,y}(x))\mathrm{d}x \leq \gamma' \cdot (1 + O(\delta))$, in time $\mathrm{poly}(n, d, \|y\|_\infty, 1/\delta, \log(1/\tau))$, with probability at least $1 - \tau$.

The algorithm used to show lemma 3 is presented in algorithm 6.

Algorithm 6 Algorithm to sample from $p_{X,y}$

Input: Sequence of points $X = \{X_i\}_{i=1}^n$ in $\mathbb{R}^d$, vector $y \in \mathbb{R}^n$, parameter $0 < \delta < 1$.

Output: A random vector $Z \in \mathbb{R}^d$ sampled from a probability distribution with density function $\widetilde{\phi}$, such that $\|\widetilde{\phi} - p_{X,y}\|_1 \leq \delta$.

1: Let $m = \lceil 1 + 2\|y\|_\infty \rceil$.

2: Let $M = \max_{x \in \mathbb{R}^d} \exp(h_{X,y}(x))$.

3: **for all** $i \in [m]$ **do**

4: Let $L_i = \{x \in \mathbb{R}^d : \exp(h_{X,y}(x)) \geq M \cdot 2^{-i}\}$.

5: Compute an estimate $\widetilde{\mathrm{vol}}(L_i)$ of $\mathrm{vol}(L_i)$ such that

$$\mathrm{vol}(L_i)/(1+\delta) \leq \widetilde{\mathrm{vol}}(L_i) \leq \mathrm{vol}(L_i)(1+\delta).$$

6: **end for**

7: **for** $i \leftarrow 1, m$ **do**

8: $\eta \leftarrow c/\sqrt{i}$

9: $s \sim p_{X,y}$ {Using Lemma 3}

10: $y \leftarrow y + \eta\left(\frac{1}{n}\mathbb{1} - T_{X,y}(s)\right)$ {T computed via Lemma 1. $\frac{1}{n}\mathbb{1}$ follows from Equation (4.2)}

11: Let u_i be the uniform probability distribution on L_i.

12: Let $\widetilde{u}_i$ be an efficiently samplable probability distribution such that

$$\|\widetilde{u}_i - u_i\|_1 \leq \delta.$$

13: **end for**

14: Let $\widetilde{c} = \sum_{i=1}^m 2^{-i}\widetilde{\mathrm{vol}}(L_i) + 2^{-m}\widetilde{\mathrm{vol}}(L_m)$.

15: Let $\widehat{D}$ be the probability distribution on $[m]$ with

$$\Pr_{I \sim \widetilde{D}}[I = i] = \begin{cases} \widetilde{\mathrm{vol}}(L_i) \cdot 2^{-i}/\widetilde{c} & \text{if } i \in \{1, \ldots, m-1\} \\ 2 \cdot \widetilde{\mathrm{vol}}(L_m) \cdot 2^{-m}/\widetilde{c} & \text{if } i = m \end{cases}$$

16: Sample $I \sim \widetilde{D}$ and sample $Z \sim \widetilde{u}_I$.

17: For any $x \in \mathbb{R}^d$ let $G_{X,y}(x) = M \cdot 2^{-\lfloor \log_2(M/\exp(h_{X,y}(x)))\rfloor}$.

18: With probability $1 - \exp(h_{X,y}(Z))/G_{X,y}(Z)$ return to line 16.

19: **return** Z

Algorithm 6 operates in two stages (The formal analysis is presented in section 4.6.) First, it slices the tent distribution into level sets and computes their volume. Properties of log-concave distributions allow us to guarantee that we obtain a good approximation of the the density with these slices. We then derive a linear program which can be used as a separation oracle for tent

densities. This allows us to compute their volume using a classic result for volume estimation [84]. In the second stage we sample from the "sliced" distribution above. We first draw a single random number to choose a level set, weighted by the volume computed in the first stage. We then draw a sample uniformly at random from the corresponding level set and return that as our sample. Please see section 4.6 for a complete exposition, proof and pseudocode.

4.6 Sampling Algorithm

The algorithm used in the proof of Lemma 3 is concerned mainly with part (1) in its statement. The pseudocode of this sampling procedure is given in Algorithm 6.

Using the notation from Algorithm 6, part (2) is easier to describe and we thus omit the pseudocode. We note that the following exposition of Algorithm 6 assumes that the input vector y is bounded. In the execution of Algorithm 3, $\|y\|_\infty$ is bounded linearly by the number of SGD iterates. Thus, the dependence of the sampling runtime on $\|y\|_\infty$ increases the overall runtime by at most a polynomial.

We now present the proof of Lemma 3. The pseudocode of the sampling procedure is given in Algorithm 6. As stated in Section 4.5.2, Algorithm 6 uses subroutines for approximating the volume of a convex body given by a membership oracle, and a procedure for sampling from the uniform distribution supported on such a body. For these procedures we use the algorithms by [84], which are summarized in Theorems 12 and 13 respectively.

Theorem 12 ([84])**.** *The volume of a convex body K in $\mathbb{R}^d$, given by a membership oracle, can be approximated to within a relative error of δ with probability $1-\tau$ using*

$$d^5 \cdot \mathrm{poly}(\log d, 1/\delta, \log(1/\tau))$$

oracle calls.

Theorem 13 ([84])**.** *Given a convex body $K \subset \mathbb{R}^d$, with oracle access, and some $\delta > 0$, we can generate a random point $u \in K$ that is distributed according to a distribution that is at most δ away from uniform in total variation distance, using*

$$d^5 \cdot \mathrm{poly}(\log d, 1/\delta)$$

oracle calls.

For all $X = X_1, \ldots, X_n \in \mathbb{R}^d$, $y \in \mathbb{R}^n$, and $x \in \mathbb{R}^d$, we use the notation $H_{X,y}(x) = \exp(h_{X,y}(x))$.

In order to use the algorithms in Theorems 12 and 13 in our setting, we need a membership oracle for the superlevel sets of the function $H_{X,y}$. Such an oracle can clearly be implemented using the

LP (4.5). We also need a separation oracle for these superlevel sets, which is given in the following lemma:

Lemma 4 (Efficient Separation)**.** *There exists a* $\mathrm{poly}(n, d)$ *time separation oracle for the superlevel sets of* $H_{X,y}(x) = \exp(h_{X,y}(x))$.

Proof. To construct our separation oracle, we will rely on the covering LP that is dual to the packing LP used to evaluate a tent function. The dual to the packing LP looks for the hyperplane that is above all the (X_i, y_i) that has minimal y at x. More specifically, it is the following LP:

$$\begin{array}{ll} \text{minimize} & \beta_0 + \sum_{j=1}^{d} \beta_j x_j \\ \text{subject to} & \beta \in \mathbb{R}^{d+1}, \beta_0 + \sum_{j=1}^{d} \beta_j X_{i,j} \geq y_i, i \in [n] \ , \end{array} \tag{4.6}$$

where $X_{i,j}$ is the j-th coordinate of the vector X_i. Now suppose that we are interested in a super level set $L_{H_{X,y}}(l)$. We can use the above LP to compute $h_{X,y}(x)$ (and thus $H_{X,y}(x)$) and check if it is in the superlevel set. Suppose that it is not, then there will be a solution $\beta \in \mathbb{R}^{d+1}$ whose value is below $\ln l$, say $\ln l - \delta$ for some $\delta > 0$. Consider an x' in the halfspace $\beta_0 + \sum_{j=1}^{d} \beta_j x'_j \leq \ln l - \delta/2$ which has x in the interior. Since x does not appear in the objective, β is a feasible solution for the dual LP (4.6) with y, x', and so $h_y(x') \leq \ln l - \delta/2$, which implies that x' is not in the superlevel set. Therefore, $\beta_0 + \sum_j \beta_j x'_j = \ln l - \delta/2$ is a separating hyperplane for x and the level set. This completes the proof. □

Given all of the above ingredients, we are now ready to prove the main result of this section.

Proof of Lemma 3. We first prove part (1) of the assertion. To that end we analyze the sampling procedure described in Algorithm 6. Recall that $m = 1 + \lceil \|y\|_\infty \rceil$, and for any $i \in [m]$, we define the superlevel set

$$L_i = \{x \in \mathbb{R}^d : H_{X,y}(x) \geq M_{H_{X,y}} \cdot 2^{-i}\} \ .$$

For any $x \in \mathbb{R}^d$ recall that

$$G_{X,y}(x) = M_{H_{X,y}} 2^{-\lfloor \log_2(M_{H_{X,y}}/H_{X,y}(x)) \rfloor} \ .$$

For any $A \subseteq \mathbb{R}^d$, let $\chi_A : \mathbb{R}^d \to \{0, 1\}$ be the indicator function for A. It is immediate that for all $x \in \mathbb{R}^d$,

$$\begin{aligned} G_{X,y}(x) &= M_{H_{X,y}} \sum_{i=1}^{\infty} 2^{-i} \chi_{L_i}(x) \\ &= M_{H_{X,y}} \sum_{i=1}^{m} 2^{-i} \chi_{L_i}(x) + 2^{-m} \chi_{L_i}(m) \qquad \text{(since } H_{X,y}(x) = 0 \text{ for all } x \notin L_m\text{)} \end{aligned}$$

Let

$$c = \sum_{i=1}^{m} 2^{-i}\mathrm{vol}(L_i) + 2^{-m}\mathrm{vol}(L_m).$$

We have

$$\int_{\mathbb{R}^d} G_{X,y}(x)\mathrm{d}x = M_{H_{X,y}} \left(\sum_{i=1}^{m} 2^{-i}\mathrm{vol}(L_i) + 2^{-m}\mathrm{vol}(L_m) \right) = M_{H_{X,y}} c. \tag{4.7}$$

Let

$$\widehat{G}_{X,y}(x) = G_{X,y}(x)/(M_{H_{X,y}} c).$$

It follows by (4.7) that $\widehat{G}_{X,y}$ is a probability density function.

Let D be the probability distribution on $\{1, \ldots, m\}$, where

$$\Pr_{I\sim D}[I = i] = \begin{cases} \mathrm{vol}(L_i) \cdot 2^{-i}/c & \text{if } i \in \{1, \ldots, m-1\} \\ 2 \cdot \mathrm{vol}(L_m) \cdot 2^{-m}/c & \text{if } i = m \end{cases}$$

For any $i \in [m]$, let u_i be the uniform probability density function on L_i. To sample from $\widehat{G}_{X,y}$, we can first sample $I \sim D$, and then sample $Z \sim u_I$.

Recall that $\widehat{p}_{X,y} : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ is the probability density function obtained by normalizing $H_{X,y}$; that is, for all $x \in \mathbb{R}^d$ we have

$$p_{X,y}(x) = H_{X,y}(x)/c',$$

where

$$c' = \int_{\mathbb{R}^d} H_{X,y}(x)\mathrm{d}x.$$

Consider the following random experiment: first sample $Z \sim \widehat{G}_y$, and then accept with probability $H_{X,y}(Z)/G_{X,y}(Z)$; conditioning on accepting, the resulting random variable $Z \in \mathbb{R}^d$ is distributed according to $\widehat{H}_{X,y}$. Note that since for all $x \in \mathbb{R}^d$, $G_{X,y}(x)/2 \leq H_{X,y}(x) \leq G_{X,y}(x)$, it follows that we always accept with probability at least $1/2$. Let α be the probability of accepting. Then

$$\alpha = \int_{\mathbb{R}^d} \widehat{G}_{X,y}(x)(H_{X,y}(x)/G_{X,y}(x))\mathrm{d}x,$$

and thus

$$\begin{aligned} \int_{\mathbb{R}^d} H_{X,y}(x)\mathrm{d}x &= \int_{\mathbb{R}^d} G_{X,y}(x)(H_{X,y}(x)/G_{X,y}(x))\mathrm{d}x \\ &= M_{H_{X,y}} c \int_{\mathbb{R}^d} \widehat{G}_{X,y}(x)(H_{X,y}(x)/G_{X,y}(x))\mathrm{d}x \\ &= M_{H_{X,y}} c\alpha\ . \end{aligned} \tag{4.8}$$

By Theorem 12, for each $i \in [m]$, we compute an estimate, $\widetilde{\mathrm{vol}}(L_i)$, to $\mathrm{vol}(L_i)$, to within relative error δ, using $\mathrm{poly}(d, 1/\delta, \log(1/\tau'))$ oracle calls, with probability at least τ', where $\tau' = \tau/n^b$, for some constant $b > 0$ to be determined; moreover, by Theorem 13, we can efficiently sample, using $\mathrm{poly}(d, 1/\delta)$ oracle calls, from a probability distribution $\widetilde{u}_i$ with $\|u_i - \widetilde{u}_i\| \leq \delta$. Each of these oracle calls is a membership query in some superlevel set of $H_{X,y}$. This membership query can clearly be implemented if we can compute that value H_y at the desired query point x, which can be done in time $\mathrm{poly}(n, d)$ using LP (4.5). Thus, each oracle call takes time $\mathrm{poly}(n, d)$. Let

$$\widetilde{c} = \sum_{i=1}^{m} 2^{-i}\widetilde{\mathrm{vol}}(L_i) + 2^{-m}\widetilde{\mathrm{vol}}(L_m). \tag{4.9}$$

Since for all $i \in [m]$, $\mathrm{vol}(L_i)/(1+\delta) \leq \widetilde{\mathrm{vol}}(L_i) \leq \mathrm{vol}(L_i)(1+\delta)$, it is immediate that

$$c/(1+\delta) \leq \widetilde{c} \leq c(1+\delta)\ .$$

Recall that Algorithm 6 uses the probability distribution $\widetilde{D}$ on $[m]$, where

$$\Pr_{I\sim\widetilde{D}}[I = i] = \begin{cases} \widetilde{\mathrm{vol}}(L_i) \cdot 2^{-i}/\widetilde{c} & \text{if } i \in \{1, \ldots, m-1\} \\ 2 \cdot \widetilde{\mathrm{vol}}(L_m) \cdot 2^{-m}/\widetilde{c} & \text{if } i = m \end{cases}$$

Consider the following random experiment, which corresponds to Step 5 of Algorithm 6: We first sample $I \sim \widetilde{D}$, and then we sample $Z \sim \widetilde{u}_I$. The resulting random vector $Z \in \mathbb{R}^d$ is distributed according to

$$\widetilde{G}_{X,y}(x) = \frac{1}{\widetilde{c}}\left(\sum_{i=1}^{m} 2^{-i}\widetilde{\mathrm{vol}}(L_i)\widetilde{u}_i(x) + 2^{-m}\widetilde{\mathrm{vol}}(L_m)\widetilde{u}_m(x)\right).$$

Next, consider the following random experiment, which captures Steps 5–7 of Algorithm 6: We sample $Z \sim \widetilde{G}_{X,y}$, and we accept with probability $H_{X,y}(Z)/G_{X,y}(Z)$. Let $\widetilde{H}_{X,y}$ be the resulting probability density function supported on $\mathbb{R}^d$ obtained by conditioning the above random experiment on accepting. Let $\widetilde{\alpha}$ be the acceptance probability. We have

$$\widetilde{\alpha} = \int_{\mathbb{R}^d} (H_{X,y}(x)/G_{X,y}(x))\widetilde{G}(x)\mathrm{d}x.$$

We have

$$\begin{aligned}
\|D_i - \widetilde{D}_i\|_1 &= \sum_{i=1}^{m-1} 2^{-i} \cdot \left| \frac{\mathrm{vol}(L_i)}{c} - \frac{\widetilde{\mathrm{vol}}(L_i)}{\widetilde{c}} \right| + 2 \cdot 2^{-m} \cdot \left| \frac{\mathrm{vol}(L_m)}{c} - \frac{\widetilde{\mathrm{vol}}(L_m)}{\widetilde{c}} \right| \\
&= \sum_{i=1}^{m-1} 2^{-i} \cdot \left| \frac{\mathrm{vol}(L_i)}{c} - \frac{\mathrm{vol}(L_i)(1+\delta)}{c/(1+\delta)} \right| + 2 \cdot 2^{-m} \cdot \left| \frac{\mathrm{vol}(L_m)}{c} - \frac{\mathrm{vol}(L_m)(1+\delta)}{c/(1+\delta)} \right| \\
&\leq \sum_{i=1}^{m-1} 2^{-i} \frac{\mathrm{vol}(L_i)}{c} 3\delta + 2 \cdot 2^m \frac{\mathrm{vol}(L_m)}{c} 3\delta \\
&= 3\delta.
\end{aligned}$$

It follows that

$$\|\widehat{G}_{X,y} - \widetilde{G}_{X,y}\|_1 \leq \|D_i - \widetilde{D}_i\| + \max_i \|u_i - \widetilde{u}_i\|_1 \leq 3\delta + \delta \leq 4\delta,$$

and so

$$|\alpha - \widetilde{\alpha}| \leq \int_{\mathbb{R}^d} \frac{H_{X,y}(x)}{G_{X,y}(x)} \left| \widehat{G}_{X,y}(x) - \widetilde{G}_{X,y}(x) \right| \mathrm{d}x \leq \int_{\mathbb{R}^d} \left| \widehat{G}_{X,y}(x) - \widetilde{G}_{X,y}(x) \right| \mathrm{d}x \leq \|\widehat{G}_{X,y} - \widetilde{G}_{X,y}\|_1 \leq 4\delta.$$

Note that $p_{X,y}(x)/\alpha = \widehat{G}_{X,y}(x) \frac{H_{X,y}(x)}{G_{X,y}(x)}$ and $\widetilde{H}_{X,y}(x)/\widetilde{\alpha} = \widetilde{G}_{X,y}(x) \frac{H_{X,y}(x)}{G_{X,y}(x)}$ and so

$$\begin{aligned}
\|\widetilde{H}_{X,y} - p_{X,y}\|_1 &\leq \alpha \left(\|\widetilde{H}_{X,y}/\alpha - p_{X,y}/\alpha\|_1 + \|p_{X,y}/\widetilde{\alpha} - p_{X,y}/\alpha\|_1 \right) && \text{(by the triangle inequality)} \\
&= \alpha \left(\|\widetilde{H}_{X,y}/\alpha - p_{X,y}/\alpha\|_1 + |1/\widetilde{\alpha} - 1/\alpha| \right) \\
&= \alpha \int_{\mathbb{R}^d} (H_{X,y}(x)/G_{X,y}(x)) |\widetilde{G}_{X,y}(x) - p_{X,y}(x)| + |\alpha - \widetilde{\alpha}|/\widetilde{\alpha} \\
&\leq \|p_{X,y} - \widetilde{G}_{X,y}\|_1 + 2|\alpha - \widetilde{\alpha}| \\
&\leq 12\delta,
\end{aligned}$$

which establishes that the random vector Z that Algorithm 6 outputs is distributed according to a probability distribution $\widetilde{\phi}$ such that $\|\widetilde{\phi} - p_{X,y}\|_1 \leq 10\delta$, as required.

In order to bound the running time, we observe that all the steps of the algorithm can be implemented in time $\mathrm{poly}(n, d, \|y\|_\infty, 1/\delta, \log(1/\tau))$. The most expensive operation is approximating the volume of a superlevel set L_i and sampling for L_i, using Theorems 12 and 13. By the above discussion, using LP (4.5) and Lemma 4 each of these operations can be implemented in time $\mathrm{poly}(n, d, 1/\delta, \log(1/\tau))$. The algorithm succeeds if all the invocations of the algorithm of Theorem 12 are successful; by the union bound, this happens with probability at least $1 - \tau' \mathrm{poly}(n) = 1 - \tau' n^b \mathrm{poly}(n) \geq 1 - \tau$, where the inequality follows by choosing some sufficiently large constant $b > 0$. This establishes part (1) of the Lemma.

It remains to prove part (2). By (4.8) we have that $\gamma = M_{H_{X,y}} c\alpha$. Algorithm $\mathcal{A}_2$ proceeds as follows. First, we compute $M_{H_{X,y}}$. By the convexity of $h_{X,y}$, it follows that the maximum value of $M_{H_{X,y}}$ is attained on some sample point x_i; that is, $M_{H_{X,y}} = \max_{i\in[n]} H_{X,y}(x_i)$. Since we can evaluate H_y in polynomial time using LP (4.5), it follows that we can also compute $M_{H_{X,y}}$ in polynomial time. Next, we compute $\widetilde{c}$ using formula 4.9. Arguing as in part (1), this can be done in time $\mathrm{poly}(n, 1/\delta, \log(1/\tau))$, and with probability at least $1 - \tau/2$. Finally, we estimate $\widetilde{\alpha}$. The value of $\widetilde{\alpha}$ is precisely the acceptance probability of the random experiment described in Steps 5–7 of Algorithm 6. Since $\alpha \geq 1/2$, and $|\alpha - \widetilde{\alpha}| \leq 4\delta$, it follows that for $\delta < 1/16$, we can compute an estimate $\bar{\alpha}$ of the value of $\widetilde{\alpha}$, to within error $1 + O(\delta)$, with probability at least $1 - \tau/2$, after $O(\log(1/\tau))$ repetitions of the random experiment. The output of algorithm $\mathcal{A}_2$ is $\gamma' = M_{H_{X,y}} \widetilde{c}\bar{\alpha}$. We obtain that, with probability at least $1 - \tau$, we have

$$\gamma' = M_{H_{X,y}} \widetilde{c}\bar{\alpha} \leq M_{H_{X,y}} c(1+\delta)\alpha(1+O(\delta)) = \gamma(1+O(\delta)) \ ,$$

and

$$\gamma' = M_{H_{X,y}} \widetilde{c}\bar{\alpha} \geq M_{H_{X,y}} (c/(1+\delta))(\alpha/(1+O(\delta))) = \gamma/(1+O(\delta)) \ ,$$

which concludes the proof. □

4.7 Learning Multivariate Log-Concave Densities

In this section, we combine our Theorem 11 with known sample complexity bounds to give the first computationally efficient and sample near-optimal proper learner for multivariate log-concave densities.

Recall that the squared Hellinger loss between two distributions with densities $f, g : \mathbb{R}^d \to \mathbb{R}_+$ is $h^2(f,g) = (1/2) \cdot \int_{\mathbb{R}^d} (\sqrt{f(x)} - \sqrt{g(x)})^2 dx$. Combined with the known rate of convergence of the log-concave MLE with respect to the squared Hellinger loss [31, 42], Theorem 11 implies the following:

Theorem 14. *Fix $d \in \mathbb{Z}_+$ and $0 < \epsilon, \tau < 1$. Let $n = \tilde{\Omega}\left((d^2/\epsilon)\ln(1/\tau)\right)\big)^{(d+1)/2}$. There is an algorithm that, given n iid samples from an unknown log-concave density $f_0 \in \mathcal{F}_d$, runs in $\mathrm{poly}(n)$ time and outputs a log-concave density $h^* \in \mathcal{F}_d$ such that with probability at least $1 - \tau$, we have that $h^2(h^*, f_0) \leq \epsilon$.*

We note that Theorem 15 yields the first efficient proper learning algorithm for multivariate log-concave densities under a global loss function. The proof follows by combining Theorem 11 with the following lemma:

Lemma 5. *Let $n = \Omega_d\left((1/\epsilon)\ln(1/(\epsilon\tau))\right)^{(d+1)/2}$. Let $\widehat{f}_n$ be the MLE of n samples drawn from $f_0 \in \mathcal{F}_d$. Let h^* be a log-concave density that is supported on the convex hull of the samples with*

$\ell(h^*) \geq \ell(\widehat{f}_n) - \epsilon/16$. *Then with probability at least* $1-\tau$ *over the samples,* $h^2(h^*, f_0) \leq \epsilon$.

We write f_n for the empirical density over the samples $X_1, \ldots, X_n$. The proof is a minor modification of the arguments in Section 3 of [31], using the following lemma [42]:

Lemma 6 (Theorem 4 from [42])**.** *For any* $t > 0$*, we have except with probability* $2\exp(-2t^2)$ *that for any convex set* C*,*

$$|f_n(C) - f_0(C)| \leq O_d(n^{-2/(d+1)}) + t/\sqrt{n} \; .$$

Proof. The proof follows Section 3 of [31], except that we need to replace Lemma 10 of that paper with Lemma 8 and that we use h^* in place of $\widehat{f}_n$. We will sketch the proof here and highlight the modified components of that proof.

Lemma 10 of [31] had that, except with probability $\tau/3$, for all convex sets C, $|f_n(C) - f_0(C)| \leq \epsilon/32 \ln(100n^4/\tau^2)$. We take $n = \Omega_d\left((1/\epsilon)\ln(1/(\epsilon\tau))^{(d+1)/2}\right.$ and $t = \sqrt{\ln(6/\tau)/2}$ in Lemma 8 and so $n^{-2/(d+1)} = O_d(\epsilon/\ln(1/\epsilon\tau)) = O_d(\epsilon/\ln(n/\tau)$ and $t/\sqrt{n} \leq \sqrt{\ln(\tau)}(\epsilon/(\ln(\epsilon\tau)))^{-(d+1)/2} \leq O(\epsilon/\ln(n/\tau))$ for $d \geq 2$. With a sufficiently large constant in the Ω_d, we obtain that $|f_n(C) - f_0(C)| \leq \epsilon/K \ln(100n^4/\tau^2)$ except with probability $\tau/3$ where K is a constant large enough to make the subsequent proof work.

This gives the improved sample complexity. We now need to argue that replacing $\widehat{f}_n$ with h^* does not affect the proof.

Corollary 9 of [31] gave that except with probability $\tau/10$, all samples lie in a set S, which is the set where $f_0(x) \geq p_{\min}$ for $p_{\min} = M_{f_0}/(n^4 100/\tau^2)$, where we use the notation M_f for the maximum value of a density f.. When this holds both $\widehat{f}_n$ and h^* are supported on S. Examination of the proof of Lemma 18 from [31] shows that we can relax the inequality $\ell(f) \leq \ell(f_0)$ to $\ell(f) \leq \ell(f_0) - \epsilon/16$ for any f with maximum value M_f has $M_f = \Omega(\ln(100n^4/\tau^2))$. In partuclar, since $\ell(h^*) \geq \ell(\widehat{f}_n)) - \epsilon/16 \geq \ell(f_0) - \epsilon/16$, we have $M_{h^*} = O(\ln(100n^4/\tau^2))$.

Then we define $g_h(x)$ supported on S as the normalisation of $\max\{p_{\min}, h^*(x)\}$ for $x \in S$. The proof of Lemma 17 in [31] required only that $\widehat{f}_n$ is supported on S and so we can obtain the same result for g_h and $h^*(x)\}$ i.e. that $g_h(x) = \alpha \max\{p_{\min}, h^*(x)\}$ for $1 - \epsilon/32 \leq \alpha \leq 1$ and that the total variation distance is small,

$$d_{\mathrm{TV}}(g_h, h^*) \leq 3\epsilon/64 \; . \tag{4.10}$$

Note that since the superlevel sets of $\ln \max\{p_{\min}, h^*(x)\}$ are convex, we can use our application of Lemma 8 to bound the error in it's expectation as

$$\begin{aligned} |\mathbb{E}_{X\sim f_0}[1_S \ln(\max\{h^*(X), p_{\min}\})] - \mathbb{E}_{X\sim f_n}[1_S \ln(\max\{h^*(X), p_{\min}\})]| &\leq (M_{h^*} - p_{\min})\epsilon/K \ln(100n^4/\tau^2) \\ &\leq \epsilon/4 \end{aligned} \tag{4.11}$$

for large enough K.

We now follow the proof of Lemma 19 in [31]. We have that

$$\begin{aligned}
\mathbb{E}_{X\sim f_0}[\ln g_h(X)] &= \mathbb{E}_{X\sim f_0}[1_S(x)\ln(\alpha\max\{p_{\min}, h^*(x)\})] \\
&\geq \mathbb{E}_{X\sim f_0}[1_S(x)\ln\max\{p_{\min}, h^*(x)\}] - \epsilon/16 && \text{(since } a > 1-\epsilon/32\text{)} \\
&\geq \mathbb{E}_{X\sim f_0}[1_S\ln(\max\{h^*(X), p_{\min}\})] - \epsilon/16 \\
&\geq \mathbb{E}_{X\sim f_n}[1_S\ln(\max\{h^*(X), p_{\min}\})] - 3\epsilon/16 && \text{by (4.15)} \\
&\geq \frac{1}{n}\sum_i \ln h^*(X_i) - 3\epsilon/16 \\
&\geq \frac{1}{n}\sum_i \ln \widehat{f}_n(X_i) - \epsilon/4 \\
&\geq \frac{1}{n}\sum_i \ln f_0(X_i) - \epsilon/4 \\
&\geq \mathbb{E}_{X\sim f_0}[\ln f_0(X)] - 3\epsilon/8. && \text{(using Lemma 14 of [31])} \quad (4.12)
\end{aligned}$$

Thus, we obtain that

$$\mathrm{KL}(f_0||g) = \mathbb{E}_{X\sim f_0}[\ln f_0(X)] - \mathbb{E}_{X\sim f_0}[\ln g_h(X)] \leq 3\epsilon/8. \quad (4.13)$$

For the next derivation, we use that the Hellinger distance is related to the total variation distance and the Kullback-Leibler divergence in the following way: For probability functions $k_1, k_2 : \mathbb{R}^d \to \mathbb{R}$, we have that $h^2(k_1, k_2) \leq d_{\mathrm{TV}}(k_1, k_2)$ and $h^2(k_1, k_2) \leq \mathrm{KL}(k_1||k_2)$. Therefore, we have that

$$\begin{aligned}
h(f_0, h^*) &\leq h(f_0, g_h) + h(g_h, h^*) \\
&\leq \mathrm{KL}(f_0||g_h)^{1/2} + d_{\mathrm{TV}}(g_h, h^*)^{1/2} \\
&= (3\epsilon/8)^{1/2} + (3\epsilon/64)^{1/2} && \text{(by (4.17) and (4.14))} \\
&\leq \epsilon^{1/2}\,,
\end{aligned}$$

concluding the proof. □

4.8 Learning Multivariate Log-Concave Densities

In this section, we combine our Theorem 11 with known sample complexity bounds to give the first computationally efficient and sample near-optimal proper learner for multivariate log-concave densities. In essence, we show that the approximation found by our algorithm also approximates the true log-concave MLE in an information theoretic distance in the parameter range of interest.

Recall that the squared Hellinger loss between two distributions with densities $f, g : \mathbb{R}^d \to \mathbb{R}_+$ is $h^2(f, g) = (1/2)\cdot\int_{\mathbb{R}^d}(\sqrt{f(x)} - \sqrt{g(x)})^2 dx$. Combined with the known rate of convergence of

the log-concave MLE with respect to the squared Hellinger loss [31, 42], Theorem 11 implies the following:

Theorem 15. *Fix $d \in \mathbb{Z}_+$ and $0 < \epsilon, \tau < 1$. Let $n = \tilde{\Omega}\left((d^2/\epsilon)\ln(1/\tau)\right)^{(d+1)/2}$. There is an algorithm that, given n iid samples from an unknown log-concave density $f_0 \in \mathcal{F}_d$, runs in $\mathrm{poly}(n)$ time and outputs a log-concave density $h^* \in \mathcal{F}_d$ such that with probability at least $1-\tau$, we have that $h^2(h^*, f_0) \leq \epsilon$.*

We note that Theorem 15 yields the first efficient proper learning algorithm for multivariate log-concave densities under a global loss function. The proof follows by combining Theorem 11 with the following lemma:

Lemma 7. *Let $n = \Omega_d\left((1/\epsilon)\ln(1/(\epsilon\tau))\right)^{(d+1)/2}$. Let $\widehat{f}_n$ be the MLE of n samples drawn from $f_0 \in \mathcal{F}_d$. Let h^* be a log-concave density that is supported on the convex hull of the samples with $\ell(h^*) \geq \ell(\widehat{f}_n) - \epsilon/16$. Then with probability at least $1-\tau$ over the samples, $h^2(h^*, f_0) \leq \epsilon$.*

We write f_n for the empirical density over the samples $X_1, \ldots, X_n$. The proof is a minor modification of the arguments in Section 3 of [31], using the following lemma [42]:

Lemma 8 (Theorem 4 from [42])**.** *For any $t > 0$, we have except with probability $2\exp(-2t^2)$ that for any convex set C,*

$$|f_n(C) - f_0(C)| \leq O_d(n^{-2/(d+1)}) + t/\sqrt{n}\,.$$

Proof. The proof follows Section 3 of [31], except that we need to replace Lemma 10 of that paper with Lemma 8 and that we use h^* in place of $\widehat{f}_n$. We will sketch the proof here and highlight the modified components of that proof.

Lemma 10 of [31] had that, except with probability $\tau/3$, for all convex sets C, $|f_n(C) - f_0(C)| \leq \epsilon/32\ln(100n^4/\tau^2)$. We take $n = \Omega_d\left((1/\epsilon)\ln(1/(\epsilon\tau))\right)^{(d+1)/2}$ and $t = \sqrt{\ln(6/\tau)/2}$ in Lemma 8 and so $n^{-2/(d+1)} = O_d(\epsilon/\ln(1/\epsilon\tau)) = O_d(\epsilon/\ln(n/\tau)$ and $t/\sqrt{n} \leq \sqrt{\ln(\tau)}(\epsilon/(\ln(\epsilon\tau)))^{-(d+1)/2} \leq O(\epsilon/\ln(n/\tau))$ for $d \geq 2$. With a sufficiently large constant in the Ω_d, we obtain that $|f_n(C) - f_0(C)| \leq \epsilon/K\ln(100n^4/\tau^2)$ except with probability $\tau/3$ where K is a constant large enough to make the subsequent proof work.

This gives the improved sample complexity. We now need to argue that replacing $\widehat{f}_n$ with h^* does not affect the proof.

Corollary 9 of [31] gave us that, except with probability $\tau/10$, all samples lie in a set S, which is the set where $f_0(x) \geq p_{\min}$ for $p_{\min} = M_{f_0}/(n^4 100/\tau^2)$, where we use the notation M_f for the maximum value of a density f.. When this holds both $\widehat{f}_n$ and h^* are supported on S. Examination of the proof of Lemma 18 from [31] shows that we can relax the inequality $\ell(f) \leq \ell(f_0)$ to $\ell(f) \leq \ell(f_0) - \epsilon/16$ for any f with maximum value M_f has $M_f = \Omega(\ln(100n^4/\tau^2))$. In partuclar, since $\ell(h^*) \geq \ell(\widehat{f}_n)) - \epsilon/16 \geq \ell(f_0) - \epsilon/16$, we have $M_{h^*} = O(\ln(100n^4/\tau^2))$.

Then we define $g_h(x)$ supported on S as the normalisation of $\max\{p_{\min}, h^*(x)\}$ for $x \in S$. The proof of Lemma 17 in [31] required only that $\widehat{f}_n$ is supported on S and so we can obtain the same result for g_h and $h^*(x)\}$ i.e. that $g_h(x) = \alpha \max\{p_{\min}, h^*(x)\}$ for $1 - \epsilon/32 \leq \alpha \leq 1$ and that the total variation distance is small,

$$d_{\mathrm{TV}}(g_h, h^*) \leq 3\epsilon/64 \,. \tag{4.14}$$

Note that since the superlevel sets of $\ln \max\{p_{\min}, h^*(x)\}$ are convex, we can use our application of Lemma 8 to bound the error in it's expectation as

$$\begin{aligned} |\mathbb{E}_{X\sim f_0}[1_S \ln(\max\{h^*(X), p_{\min}\})] - \mathbb{E}_{X\sim f_n}[1_S \ln(\max\{h^*(X), p_{\min}\})]| &\leq (M_{h^*} - p_{\min})\epsilon/K \ln(100n^4/\tau^2) \\ &\leq \epsilon/4 \end{aligned} \tag{4.15}$$

for large enough K.

We now follow the proof of Lemma 19 in [31]. We have that

$$\begin{aligned} \mathbb{E}_{X\sim f_0}[\ln g_h(X)] &= \mathbb{E}_{X\sim f_0}[1_S(x) \ln(\alpha \max\{p_{\min}, h^*(x)\})] \\ &\geq \mathbb{E}_{X\sim f_0}[1_S(x) \ln \max\{p_{\min}, h^*(x)\}] - \epsilon/16 && \text{(since } a > 1 - \epsilon/32\text{)} \\ &\geq \mathbb{E}_{X\sim f_0}[1_S \ln(\max\{h^*(X), p_{\min}\})] - \epsilon/16 \\ &\geq \mathbb{E}_{X\sim f_n}[1_S \ln(\max\{h^*(X), p_{\min}\})] - 3\epsilon/16 && \text{by (4.15)} \\ &\geq \frac{1}{n}\sum_i \ln h^*(X_i) - 3\epsilon/16 \\ &\geq \frac{1}{n}\sum_i \ln \widehat{f}_n(X_i) - \epsilon/4 \\ &\geq \frac{1}{n}\sum_i \ln f_0(X_i) - \epsilon/4 \\ &\geq \mathbb{E}_{X\sim f_0}[\ln f_0(X)] - 3\epsilon/8. && \text{(using Lemma 14 of [31])} \end{aligned} \tag{4.16}$$

Thus, we obtain that

$$\mathrm{KL}(f_0||g) = \mathbb{E}_{X\sim f_0}[\ln f_0(X)] - \mathbb{E}_{X\sim f_0}[\ln g_h(X)] \leq 3\epsilon/8. \tag{4.17}$$

For the next derivation, we use that the Hellinger distance is related to the total variation distance and the Kullback-Leibler divergence in the following way: For probability functions $k_1, k_2 : \mathbb{R}^d \to \mathbb{R}$,

we have that $h^2(k_1, k_2) \leq d_{\mathrm{TV}}(k_1, k_2)$ and $h^2(k_1, k_2) \leq \mathrm{KL}(k_1||k_2)$. Therefore, we have that

$$\begin{aligned} h(f_0, h^*) &\leq h(f_0, g_h) + h(g_h, h^*) \\ &\leq \mathrm{KL}(f_0||g_h)^{1/2} + d_{\mathrm{TV}}(g_h, h^*)^{1/2} \\ &= (3\epsilon/8)^{1/2} + (3\epsilon/64)^{1/2} \qquad \text{(by (4.17) and (4.14))} \\ &\leq \epsilon^{1/2}\,, \end{aligned}$$

concluding the proof. □

4.9 Conclusions

This chapter presents a different way of looking at tent distributions, a class of distributions that arise during log-concave maximum likelihood estimation. We define the *polyhedral statistic* in order to draw a connection to exponential families.

While tent distributions with this statistic do not form an exponential family, in general position in parameter space, there always exists a sufficiently small neighborhood for which the contained distributions do form an exponential family. In fact, for a set of fixed tent poles, the corresponding tent distributions form a union of a finite number of exponential families. It turns out this property with the polyhedral statistic suffices for us to use the textbook exponential family maximum likelihood maximum likelihood algorithm, albeit with slightly slower convergence rate.

This comparison to exponential families also gives us a bit more insight into the log-concave MLE problem. For example, it gives us the natural characterization of the solution to the log-concave MLE as the distribution which has uniform expected polyhedral statistic.

The work does leave a question open, beyond the obvious one of improving the exact obtained polynomial. Does this locally-exponential property enable efficient solutions to other problems of interest?

Chapter 5

Sample Amplification

5.1 Learning, Testing, and Sample Amplification

How much do you need to know about a distribution, D, in order to produce a dataset of size m that is indistinguishable from a set of independent draws from D? Do you need to *learn* D, to nontrivial accuracy in some natural metric, or does it suffice to have access to a smaller dataset of size $n < m$ drawn from D, and then "amplify" this dataset to create one of size m? In this work we formalize this question, and show that for two natural classes of distribution, discrete distributions with bounded support, and d-dimensional Gaussians, non-trivial data "amplification" is possible even in the regime in which you are given too few samples to learn.

From a theoretical perspective, this question is related to the meta-question underlying work on distributional property testing and estimation: *To answer basic hypothesis testing or property estimation questions regarding a distribution D, to what extent must one first learn D, and can such questions be reliably answered given a relatively modest amount of data drawn from D?* Much of the excitement surrounding distributional property testing and estimation stems from the fact that, for many such testing and estimation questions, a surprisingly small set of samples from D suffices—significantly fewer samples than would be required to learn D. These surprising answers have been revealed over the past two decades. The question posed in our work fits with this body of work, though instead of asking how much data is required to perform a hypothesis test, we are asking how much data is required to *fool* an optimal hypothesis test—in this case an "identity tester" which knows D and is trying to distinguish a set of m independent samples drawn from D, versus m datapoints constructed in some other fashion.

From a more practical perspective, the question we consider also seems timely. Deep neural network based systems, trained on a set of samples, can be designed to perform many tasks, including testing whether a given input was drawn from a distribution in question (i.e. "discrimination"), as well as sampling (often via the popular Generative Adversarial Network (GAN) approach). There

are many relevant questions regarding the extent to which current systems are successful in accomplishing these tasks, and the question of how to quantify the performance of these systems is still largely open. In this work, however, we ask a different question: Suppose a system *can* accomplish such a task—what would that actually mean? If a system can produce a dataset that is indistinguishable from a set of m independent draws from a distribution, D, does that mean the system knows D, or are there other ways of accomplishing this task?

5.1.1 Formal Problem Definition

We begin by formally stating two essentially equivalent definitions of sample amplification and then provide an illustrative example. Our first definition states that a function f mapping a set of n datapoints to a set of m datapoints is a valid amplification procedure for a class of distributions $\mathcal{C}$, if for all $D \in \mathcal{C}$, letting X_n denote the random variable corresponding to n independent draws from D, the distribution of $f(X_n)$ has small total variation distance[1] to the distribution defined by m independent draws from D.

Definition 30. *A class $\mathcal{C}$ of distributions over domain S admits an* (n, m) amplification procedure *if there exists a (possibly randomized) function $f_{\mathcal{C},n,m} : S^n \to S^m$, mapping a dataset of size n to a dataset of size m, such that for every distribution $D \in \mathcal{C}$,*

$$D_{TV}\left(f_{\mathcal{C},n,m}(X_n), D^m\right) \leq 1/3,$$

where X_n is the random variable denoting n independent draws from D, and D^m denotes the distribution of m independent draws from D. If no such function $f_{\mathcal{C},n,m}$ exists, we say that $\mathcal{C}$ does not admit *an (n, m) amplification scheme.*

Crucially, in the above definition we are considering the random variable $f(X_n)$ whose randomness comes from the randomness of X_n, as well as any randomness in the function f itself. For example, every class of distributions admits an (n, n) amplification procedure, corresponding to taking the function f to be the identity function. If, instead, our definition had required that the *conditional* distribution of $f(X_n)$ given X_n be close to D^m, then the above definition would simply correspond to asking how well we can *learn* D, given the n samples denoted by X_n.

Definition 30 is also equivalent, up to the choice of constant $1/3$ in the bound on total variation distance, to the following intuitive formulation of sample amplification as a game between two parties: the "amplifier" who will produce a dataset of size m, and a "verifier" who knows D and will either accept or reject that dataset. The verifier's protocol, however, must satisfy the condition that given m independent draws from the true distribution in question, the verifier must accept with probability at least $3/4$, where the probability is with respect to both the randomness of the set of samples,

[1] We overload the notation $D_{TV}(\cdot, \cdot)$ for total variation distance, and also use it when the argument is a random variable instead of the distribution of the random variable, whenever convenient.

and any internal randomness of the verifier. We briefly describe this formulation, as it parallels the pseudo-randomness framework, and a number of natural directions for future work—such as if the verifier is computationally bounded, or only has sample access to D—are easier to articulate in this setting.

Definition 31. *The* sample amplification *game consists of two parties, an* amplifier *corresponding to a function $f_{n,m} : S^n \to S^m$ which maps a set of n datapoints in domain S to a set of m datapoints, and a* verifier *corresponding to a function $v : S^m \to \{ACCEPT, REJECT\}$. We say that a verifier v is* valid *for distribution D if, when given as input a set of m independent draws from D, the verifier accepts with probability at least $3/4$, where the probability is over both the randomness of the draws and any internal randomness of v:*

$$\Pr_{X_m \leftarrow D^m}[v(X_m) = ACCEPT] \geq 3/4.$$

A class $\mathcal{C}$ of distributions over domain S admits an (n, m) amplification procedure *if, and only if, there is an amplifier function $f_{\mathcal{C},n,m}$ that, for every $D \in \mathcal{C}$, can "win" the game with probability at least $2/3$; namely, such that for every $D \in \mathcal{C}$ and valid verifier v_D for D*

$$\Pr_{X_n \leftarrow D^n}[v_D(f_{\mathcal{C},n,m}(X_n)) = ACCEPT] \geq 2/3,$$

where the probability is with respect to the randomness of the choice of the n samples, X_n, and any internal randomness in the amplifier and verifier, f and v.

As was the case in Definition 30, in the above definition it is essential that the verifier only observes the output $f(X_n)$ produced by the amplifier. If the verifier sees both the amplified samples, $f(X_n)$ in addition to the original data, X_n, then the above definition also becomes equivalent to asking how well the class of distributions in question can be *learned* given n samples.

Example 1. *Consider the class of distributions $\mathcal{C}$ corresponding to i.i.d. flips of a coin with unknown bias p. We claim that there are constants $c' \geq c > 0$ such that $(n, n + cn)$ sample amplification is possible, but $(n, n + c'n)$ amplification is not possible. To see this, consider the amplification strategy corresponding to returning a random permutation of the original samples together with cn additional tosses of a coin with bias $\hat{p}$, where $\hat{p}$ is the empirical bias of the n original samples. Because of the random permutation, the total variation distance between these samples and $n + cn$ i.i.d. tosses of the p-biased coin is a function of only the distribution of the total number of heads. Hence this is equivalent to the distance between $Binomial(n + cn, p)$, and the distribution corresponding to first drawing $h \leftarrow Binomial(n, p)$, and then returning $h + Binomial(cn, h/n)$. It is not hard to show that the total variation distance between these two can be bounded by any small constant by taking c to be a sufficiently small constant. Intuitively, this is because both distributions have the same mean,*

Figure 5.1: Sample amplification can be viewed as a game between an "amplifier" that obtains n independent draws from an unknown distribution D and must output a set of $m > n$ samples, and a "verifier" that receives the m samples and must ACCEPT or REJECT. The verifier knows the true distribution D and is computationally unbounded but does not know the amplifier's training set (the set of n input samples). An amplification scheme is successful if, for every verifier, with probability at least 2/3 the verifier will accept the output of the amplifier. [In the setting illustrated above, observant readers might recognize that one of the images in the "Output" set is a painting which was sold in October, 2018 for over $400k by Christie's auction house, and which was "painted" by a Generative Adversarial Network (GAN) [40]].

they are both unimodal, and have variances that differ by a small constant factor for small constant c. For the lower bound, to see that amplification by more than a constant factor is impossible, note that if it were possible, then one could learn p to error $o(1/\sqrt{n})$, with small constant probability of failure, by first amplifying the original samples and then returning the empirical estimate of p based on the amplified samples.

In the above setting, this constant factor amplification is not surprising, since the amplifier can learn the distribution to non-trivial accuracy. It is worth observing, however, that the above amplification scheme corresponding to a $(n, n+1)$ amplifier will return a set of $n+1$ samples, whose total variation distance from $n+1$ i.i.d. samples is only $O(1/n)$; this is despite the fact that the amplifier can only learn the distribution to total variation distance $\Theta(1/\sqrt{n})$.

5.1.2 Summary of Results

Our main results provide tight bounds on the extent to which sample amplification is possible for two fundamental settings, unstructured discrete distributions, and d-dimensional Gaussians with unknown mean and fixed covariance. Our first result is for discrete distributions with support size at most k. In this case, we show that sample amplification is possible given only $O(\sqrt{k})$ samples from the distribution, and tightly characterize the extent to which amplification is possible.[2] Note that learning the distribution to small total variation distance requires $\Theta(k)$ samples in this case.

Theorem 16. *Let $\mathcal{C}$ denote the class of discrete distributions with support size at most k. For sufficiently large k, and $m = n + O\left(\frac{n}{\sqrt{k}}\right)$, $\mathcal{C}$ admits an (n, m) amplification procedure.*

[2]This addresses a variant of an open problem posed in the Frontiers in Distribution Testing workshop at FOCS 2017 (`https://sublinear.info/index.php?title=Open_Problems:85`).

This bound is tight up to constants, i.e., there is a constant c, such that for every sufficiently large k, $\mathcal{C}$ does not admit an $\left(n, n + \frac{cn}{\sqrt{k}}\right)$ amplification procedure.

Our amplification procedure for discrete distributions is extremely simple: roughly, we generate additional samples from the empirical distribution of the initial set of n samples, and then randomly shuffle together the original and the new samples. For technical reasons, we do not exactly sample from the empirical distribution but from a suitable modification which facilitates the analysis.

Our second result concerns d-dimensional Gaussian distributions with unknown mean and fixed covariance. We show that we can amplify even with only $O(\sqrt{d})$ samples from the distribution. In contrast, learning to small constant total variation distance requires $\Theta(d)$ samples. Unlike the discrete setting, however, we do not get optimal amplification in this setting by generating additional samples from the empirical distribution of the initial set of n samples, and then randomly shuffling together the original and new samples. Moreover, we show a lower bound proving that, for $n = o(d/\log d)$ there is no $(n, n+1)$ amplification procedure which always returns a superset of the original n samples. Curiously, however, the procedure that generates new samples from the empirical distribution, and then randomly shuffles together the new and old samples, is able to amplify at $n = \Omega(d/\log d)$, even though learning is not possible until $n = \Theta(d)$. Additionally, as n goes from $10\frac{d}{\log d}$ to $1000\frac{d}{\log d}$, this amplification procedure goes from being unable to amplify at all, to being able to amplify by nearly $\sqrt{d}$ samples. This is formalized in the following proposition.

Proposition 1. *Let $\mathcal{C}$ denote the class of $d-$dimensional Gaussian distributions with unknown mean μ and covariance Σ. There is an absolute constant, c, such that for sufficiently large d, if $n \leq \frac{cd}{\log d}$, there is no $(n, n+1)$ amplification procedure that always returns a superset of the original n points.*

On the other hand, there is a constant c' such that for any ϵ, for $n = \frac{d}{\epsilon \log d}$, and for sufficiently large d, there is an $\left(n, n + c' n^{\frac{1}{2}-9\epsilon}\right)$ amplification protocol for $\mathcal{C}$ that returns a superset of the original n samples.

The above proposition suggests that to be able to amplify at input size $n = o(d/\log d)$, one must modify the input samples. A naive way to modify the input samples is to discard all the original n samples and generate m new samples from the distribution $N(\hat{\mu}, \Sigma)$, where $\hat{\mu}$ is empirical mean $\hat{\mu}$ of the original set X_n. However this does not even give an (n, n) amplification procedure for any value of n. To achieve optimal amplification in the Gaussian case, the amplifier first computes the empirical mean $\hat{\mu}$ of the original set X_n, and then draws $m - n$ new samples from $N(\hat{\mu}, \Sigma)$. We then shift the original n samples to "decorrelate" the original set and the new samples; intuitively, this step hides the fact that the $m - n$ new samples were generated based on the empirical mean of the original samples. The final set of returned samples consists of the shifted versions of the n original samples along with the $m - n$ freshly generated ones. This procedure gives $(n, n + O(\frac{n}{\sqrt{d}}))$ amplification, and we also show that this amplification is tight up to constant factors.

Theorem 17. *Let $\mathcal{C}$ denote the class of $d-$dimensional Gaussian distributions $N(\mu, \Sigma)$ with unknown mean μ and fixed covariance Σ. For all $d, n > 0$ and $m = n + O\left(\frac{n}{\sqrt{d}}\right)$, $\mathcal{C}$ admits an (n, m) amplification procedure.*

This bound is tight up to constants, i.e., there is a fixed constant c such that for all $d, n > 0$, $\mathcal{C}$ does not admit an (n, m) amplification procedure for $m \geq n + \frac{cn}{\sqrt{d}}$.

5.1.3 Open Directions

From a technical perspective, there are a number of natural open directions for future work, including establishing tight bounds on amplification for other natural distribution classes, such as d dimensional Gaussians with unknown mean and covariance. More conceptually, it seems worth getting a broader understanding of the range of potential amplification algorithms, and the settings to which each can be applied.

Weaker or More Powerful Verifiers? Our results showing that non-trivial amplification is possible even in the regime in which learning is not possible, rely on the modeling assumption that the verifier gets no information about the amplifier's training set, X_n (the set of n i.i.d. samples). If this dataset is revealed to the verifier, then the question of amplification is equivalent to learning. This prompts the question about a middle ground, where the verifier has some information about the set X_n, but does not see the entire set; this middle ground also seems the most practically relevant (e.g. how much do I need to know about a GAN's training set to decide whether it actually understands a distribution of images?).

> *How does the power of the amplifier vary depending on how much information the verifier has about X_n? If the verifier is given a uniformly random subsample of X_n of size $n' \ll n$, how does the amount of possible amplification scale with n'?*

Rather than considering how to increase the power of the verifier, as the above question asks, it might also be worth considering the consequences of decreasing either the computational power, or information theoretic power of the verifier.

> *If the verifier, instead of knowing distribution D, receives only a set of independent draws from D, how much more power does this give the amplifier? Alternately, if the verifier is constrained to be an efficiently computable function, does this provide additional power to the amplifier in any natural settings?*

Better Amplifiers in the Discrete Setting. In the discrete distribution setting, our amplification results are tight (to constant factors) in a worst-case sense, and our amplifier essentially just returns the original n samples, together with additional samples drawn from the empirical distribution of those n samples, and then randomly permutes the order of these datapoints. This begs

the question: *In the case of discrete distributions, is there any benefit to considering more sophisticated amplification schemes?* Below we sketch one example motivating a more clever amplification approach.

Example 2. *Consider obtaining n samples corresponding to independent draws from a discrete distribution that puts probability $p \gg 1/n$ on a single domain element, and with probability $1-p$ draws a sample from the uniform distribution over some infinite discrete domain. If $p < 2/3$, then the amplification approach that adds samples from the empirical distribution of the data to the original set of samples, will fail. Indeed, with probability at least $1/3$ it will introduce a second sample of one of the "rare" elements, and such samples can be rejected by the verifier. For this setting, the optimal amplifier would always introduce extra samples corresponding to the element of probability p.*

The above example motivates a more sophisticated amplification strategy for the discrete distribution setting. Approaches such as Good-Turing frequency estimation, or more modern variants of it, adjust the empirical probabilities to more accurately reflect the true probabilities (see e.g. [71, 102, 126]). Indeed, in a setting such as Example 2, based on the fact that only one domain element is observed more than once, it is easy to conclude that the total probability mass of all the elements observed just once, is likely at most $O(1/n)$, which implies that a successful amplification scheme cannot duplicate any of them. While inserting samples from a Good-Turing adjusted empirical distribution will not improve the amplification in a worst-case sense for discrete distributions with a bounded support size, such schemes seem *strictly* better than the schemes we currently analyze. The following question outlines one potential avenue for quantifying this, along the lines of the recent work on "instance optimal" distribution testing and estimation (see e.g. [3, 103, 126]):

> *Is there an "instance optimal" amplification scheme, which, for every distribution, D, amplifies as well as could be hoped? Specifically, to what extent is there an amplification scheme which performs nearly as well as a hypothetical optimal scheme that knows distribution D up to relabeling/permuting the domain?*

Potential Applications of Sample Amplification. An interesting future direction is to examine if sample amplification is a useful primitive in settings where the samples are given as input to downstream analysis. Amplification does not add any new information to the original data, but it could still make the original information more easily accessible to certain types of algorithms which interact with the data in limited ways. For example, many popular algorithms and heuristics are not information theoretically optimal, despite their widespread use. It seems worth examining if amplification schemes could improve the statistical efficiency of these commonly used methods. Since the amplified samples are "good" in an information theoretic sense (they are indistinguishable from true samples), the performance of downstream algorithms *cannot* be significantly hurt. Below, we provide a toy example of a setting where amplification improves the accuracy of a standard downstream estimator.

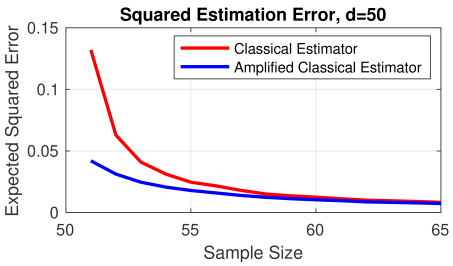

Figure 5.2: Toy example illustrating potential benefit of feeding amplified samples into a commonly used estimator. See Example 3 for a description of the specific setup.

Example 3. *Given labeled examples,* $(x_1, y_1), \ldots, (x_n, y_n)$ *drawn from a distribution,* D*, with* $x_i \in \mathbb{R}^d$ *and* $y_i \in \mathbb{R}$*, a natural quantity to estimate is the fraction of variance in* y *explainable as a linear function of* x*:*

$$\inf_{\theta \in \mathbb{R}^d} \mathbb{E}_{(x,y)\sim D}[(\theta^T x - y)^2].$$

The standard unbiased estimator for this quantity is the training error of the least-squares linear model, scaled by a factor of $\frac{1}{n-d}$*. This scaling factor makes this estimate unbiased, although the variance is large when* n *is not much larger than* d*. Figure 5.2 shows the expected squared error of this estimator on raw samples, and on* $(n, n+2)$ *amplified samples, in the case where* $x_i \sim N(0, I_d)$*, and* $y_i = \theta^T x_i + \eta$ *for some model* $\|\theta\|_2 = 1$ *and independent noise* $\eta \sim N(0, \frac{1}{4})$*—hence the true value for the "unexplainable variance" is* $1/4$*. Here, the amplification procedure draws two additional datapoints,* x *from the isotropic Gaussian with mean equal to the empirical mean, and labels them according to the learned least-squares regression model* $\hat{\theta}$ *with independent noise of variance* $5/n$ *times the empirical estimate of the unexplained variance.*

One potential limitation to applications of amplification is that our existing results show that it is only possible to amplify the sample size by sub-constant factors (for the settings considered). If the algorithm using the amplified data is limited, however, then we could hope for much larger amplification factors. This is reminiscent of the open problem in the previous section on whether larger amplification is possible against weaker classes of verifiers.

In practice, there is already growing interest in using generative models for data augmentation to improve classification accuracy [6, 68, 133, 136]. Given our results which show that amplification is significantly easier than learning, such pipelines might be more effective than one would initially suspect. It is also worth thinking more generally about how to design modular data analysis or learning pipelines, where a first component of the pipeline could be an amplifier tailored to the specific data distribution, followed by more generic learning algorithms that do not attempt to leverage structural properties of the data distribution. Such modular pipelines might prove to be significantly easier to develop and maintain, in practice.

Implications for Generative Modelling. The sample amplification framework has some connections to generative modelling. Generative models such as GANs aim to produce new samples from an unknown distribution D given a training set of size n drawn from D. It is tempting to try to relate the amplification setting to GANs by viewing the amplifier and verifier as analogs of the generator and discriminator, respectively. This is *not* an accurate correspondence: For GANs, the discriminator typically evaluates examples individually (or in small batches), and often has seen the same training set as the generator, whereas our verifier explicitly evaluates a full set of samples without knowledge of the training samples. The samples generated by a generative model are often evaluated by humans (either manually or algorithmically). This evaluation is usually aimed at understanding the quality of output samples *conditioned on the training data*—if some of the output samples are copies of the training set, this is not satisfactory—which again corresponds to learning rather than sample amplification.

Despite these differences, some ways that generative models are actually used, do closely mirror the amplification setting. For example, when generative models are used to augment a training set that is used to learn a classifier, both the generated samples and the original dataset are fed into the learning algorithm. The learning algorithm does not necessarily distinguish between "new" and "old" samples. In this setting, it does make sense to evaluate the set of "new" and "old" samples together, as a single set of "amplified" samples, rather than evaluating the "new" samples conditioned on the "old" ones. This exactly corresponds to our amplification formulation. Given that amplification is often easier than learning, it might be worthwhile trying to develop more techniques that are explicitly trying to amplify, rather than learn.

A second, distinct connection between amplification and GANs, relates to the question of how humans (or algorithms) can evaluate the samples produced by a GAN. The gap between learning (evaluating the generated samples conditioned on the training set), and amplification (evaluating the generated samples without knowledge of the training set), suggests that in order to truly evaluate the samples produced by a GAN, we would need to closely inspect the training data used by the GAN. This is obviously impractical in many settings, and motivates some of the questions described above concerning how much access a verifier needs to the input examples in order for there to be a gap between learning, and amplifying.

5.1.4 Related Work

The question of deciding whether a set of samples consists of independent draws from a specified distribution is one of the fundamental problems at the core of distributional property testing. Interest in this problem was sparked by the seminal work of Goldreich and Ron [70], who considered the specific problem of determining whether a set of samples was drawn from a uniform distribution of support size k. This sparked a line of work on the slightly more general problem of "identity testing" whether a set of samples was drawn from a specified distribution, D, versus a

distribution with distance at least ϵ from D [17, 104, 127, 55]. Beyond the specific question of identity testing, there is an enormous body of work on other distributional property testing questions, including the "tolerant" version of identity testing, as well as the multi-distribution analogs (see e.g. [18, 128, 37, 103, 20, 96, 55]). In the majority of these works, the assumption is that the given samples consist of independent draws from some fixed distribution, and the common theme in these results is that such tests can typically be accomplished with far less data than would be required to learn the distribution. While the identity testing problem is clearly related to the amplification problem we consider, these appear to be quite distinct problems. In particular, in the identity testing setting, the main technical challenge is understanding what statistics of a set of i.i.d. samples are capable of distinguishing samples drawn from the prescribed distribution, versus samples drawn from any distribution that is at least ϵ-far from that distribution. In contrast, in the amplification setting, the core question is how the amplifier can leverage a set of independent samples from D to generate a larger set of (presumably) non-independent samples that can successfully masquerade as a set of independent samples drawn from D; of course, the catch is that the amplifier must do this in the data regime in which it is impossible for it to learn much about D.

Within this line of work on distributional property testing and estimation, there is also a recent thread of work on designing estimators for specific properties (such as entropy, or distance to uniformity), whose performance given n independent draws from the distribution in question is comparable to the expected performance of a naive "plugin" estimator (which returns the property value of the empirical distribution) based on $m > n$ independent draws [126, 135]. The term "data amplification" has been applied to this line of work, although it is a different problem from the one we consider. In particular, we are considering the extent to which the samples can be used to create a larger set of samples; the work on property estimation is asking to what extent one can craft superior estimators whose performance is comparable to the performance that a more basic estimator would achieve with a larger sample size.

The recent work on *sampling correctors* [30] also considers the question of how to produce a "good" set of draws from a given distribution. That work assumes access to independent draws from a distribution, D, which is close to having some desired structural property, such as monotonicity or uniformity, and considers how to "correct" or "improve" those samples to produce a set of samples that appear to have been drawn from a different distribution D' that possesses the desired property (or is closer to possessing the property). Part of that work also considers the question of whether such a protocol requires access to additional randomness.

Our formulation of sample amplification as a game between an amplifier and a verifier, closely resembles the setup for *pseudo-randomness* (see [125] for a relatively recent survey). There, the pseudo-random generator takes a set of n independent fair coin flips, and outputs a longer string of $m > n$ outcomes. The verifier's job is to distinguish the output of the generator from a set of m independent tosses of the fair coin (i.e. truly random bits). In contrast to our setting, in

pseudo-randomness, both players know that the distribution in question is the uniform distribution, the catch is that the generator does not have access to randomness, and the verifier is computationally bounded. Beyond the superficial similarity in setup, we are not aware of any deeper connections between our notion of amplification and pseudorandomness.

Finally, it is also worth mentioning the work of Viola on the complexity of sampling from distributions [129]. That work also considers the challenge of generating samples from a specified distribution, though the problem is posed as the computational challenge of producing samples from a specified distribution, given access to uniformly random bits. One of the punchlines of that work is that there are distributions, such as the distribution over pairs (x, y) where x is a uniformly random length-n string, and $y = \mathit{parity}(x)$, where small circuits can sample from the distribution, yet no small circuit can compute $y = \mathit{parity}(x)$ given x. A different way of phrasing that punchline is that there are distributions that are easy to sample from, for which it is much harder to sample from their conditional distributions (e.g. in the parity case, sampling (x, y) given x is hard).

5.2 Algorithms and Proof Overview

In this section, we describe our algorithms for data amplification for discrete and Gaussian distributions. We also give an intuitive overview of the proofs of both the upper and lower bounds.

5.2.1 Discrete Distributions with Bounded Support

We begin by providing some intuition for amplification in the discrete distribution setting, by considering the simple case where the distribution in question is a uniform distribution over an unknown support. We then describe how our more general amplification algorithm extends this intuition.

Intuition via the Uniform Distribution. Consider the problem of generating $(n+1)$ samples from a uniform distribution over k unknown elements, given a set of n samples from the distribution. Suppose $n \ll \sqrt{k}$. Then with high probability, no element appears more than once in a set of $(n+1)$ samples. Therefore, as the amplifier only knows n elements of the support with n samples, it cannot produce a set of $(n+1)$ samples such that each element only appears once in the set. Hence, no amplification is possible in this regime. Now consider the case when $n = c\sqrt{k}$ for a large constant c. By the birthday paradox, we now expect some elements to appear more than once, and the number of elements appearing twice has expectation $\approx \frac{c^2}{2}$ and standard deviation $\Theta(c)$. In light of this fact, consider an amplification procedure which takes any element that appears only once in the set X_n, adds an additional copy it to the set X_n, and then randomly shuffles these $n+1$ samples to produce the final set Z_{n+1}. It is easy to verify that the distribution of Z_{n+1} will be close in total variation distance to a set X_{n+1} of $(n+1)$ i.i.d. samples drawn from

the original uniform distribution. Since the standard deviation of the number of elements in X_{n+1} that appear twice is $\Theta(c)$, intuitively, we should be able to amplify by an additional $\Theta(c)$ samples, by taking $\Theta(c)$ elements which appear only once and repeating them, and then randomly permuting these $n + \Theta(c)$ samples. Note that with high probability, most elements only appear once in the set X_n, and hence the previous amplifier is almost equivalent to an amplifier which generates new samples by sampling from the empirical distribution of the original n samples, and then randomly shuffles them with the original samples. Our amplification procedure for general discrete distributions is based on this sample-from-empirical procedure.

Algorithm and Upper Bound. To facilitate the analysis, our general amplification procedure which applies to any discrete distribution D, deviates from the sample-from-empirical-then-shuffle scheme in two ways. The modifications avoid two sources of dependencies in the sample-from-empirical-then-shuffle schemes. First, we use the "Poissonization" trick and go from working with the multinomial distribution to the Poisson distribution—making the element counts independent for all $\leq k$ elements. And second, note that the new samples are dependent on the old samples if we generate the new samples from the empirical distribution. To leverage independence, we instead (i) divide the input samples into two sets, (ii) use the first set to estimate the empirical distribution, (iii) generate new samples using this empirical distribution, and (iv) randomly shuffle these new samples with the samples in the second set. More precisely, we simulate two sets X_{N_1} and X_{N_2}, of Poisson$(n/4)$ samples from the distribution D, using the original set X_n of n samples from D. This is straightforward to do, as a Poisson$(n/4)$ random variable is $\leq n/2$ with high probability. We then estimate the probabilities of the elements using the first set X_{N_1}, and use these estimated probabilities to generate $R \approx m - n$ more samples from a Poisson distribution, which are then randomly shuffled with the samples in X_{N_2} to produce Z_{N_2+R}. Then the set of output samples Z_m just consist of the samples in X_{N_1} concatenated with those in Z_{N_2+R}. This describes the main steps in the procedure, more technical details can be found in the full description in the supplementary. We show that this procedure achieves $(n, n + O\left(\frac{n}{\sqrt{k}}\right))$ amplification.

To prove this upper bound, first note that the counts of each element in a shuffled set Z_m are a sufficient statistics for the probability of observing Z_m, as the ordering of the elements is uniformly random. Hence we only need to show that the distribution of the counts in the set Z_m is close in total variation distance to the distribution of counts in a set X_m of m elements drawn i.i.d. from D. Since the first set X_{N_1} is independent of the second set X_{N_2}, the additional samples added to X_{N_2} are independent of the samples originally in X_{N_2}, which avoids additional dependencies in the analysis. Using this independence, we show a technical lemma that with high probability over the first set X_{N_1}, the KL-divergence between the distribution of the set Z_{N_2+R} and D^{N_2+R} of $N_2 + R$ i.i.d. samples from D is small. Then using Pinsker's inequality, it follows that the total variation distance is also small. The final result then follows by a coupling argument, and showing that the

Poissonization steps are successful with high probability.

Lower Bound. We now describe the intuition for showing our lower bound that the class of discrete distributions with support at most k does not admit an (n, m) amplification scheme for $m \geq n + \frac{cn}{\sqrt{k}}$, where c is a fixed constant. For $n \leq \frac{k}{4}$, we show this lower bound for the class of uniform distributions $D = \text{Unif}[k]$ on some unknown k elements. In this case, a verifier can distinguish between true samples from D and a set of amplified samples by counting the number of unique samples in the set. Note that as the support of D is unknown, the number of unique samples in the amplified set is at most the number of unique samples in the original set X_n, unless the amplifier includes samples that are outside the support of D, in which case the verifier will trivially reject this set. The expected number of unique samples in n and m draws from D differs by $\frac{c_1 n}{\sqrt{k}}$, for some fixed constant c_1. We use a Doob martingale and martingale concentration bounds to show that the number of unique samples in n samples from D concentrates within a $\frac{c_2 n}{\sqrt{k}}$ margin of its expectation with high probability, for some fixed constant $c_2 \ll c_1$. This implies that there will be a large gap between the number of unique samples in n and m draws from D. The verifier uses this to distinguish between true samples from D and an amplified set, which cannot have sufficiently many unique samples.

Finally, we show that for $n > \frac{k}{4}$, a $\left(n, n + \frac{c'k}{\sqrt{k}}\right)$ amplification procedure for discrete distributions on k elements implies a $(\frac{k}{4}, \frac{k}{4} + c'\sqrt{k})$ amplification procedure for the uniform distribution on $(k-1)$ elements, and for sufficiently large c' this is a contradiction to the previous part. This reduction follows by considering the distribution which has $1 - \frac{k}{4n}$ mass on one element and $\frac{k}{4n}$ mass uniformly distributed on the remaining $(k-1)$ elements. With sufficiently large probability, the number of samples in the uniform section will be $\approx \frac{k}{4}$, and hence we can apply the previous result.

5.2.2 Gaussian Distributions with Unknown Mean and Fixed Covariance

Given the success of the simple sampling-from-empirical scheme for the discrete case, it is natural to consider the analogous algorithm for d-dimensional Gaussian distributions. In this section, we first show that this analogous procedure achieves non-trivial amplification for $n = \Omega(d/\log d)$. We then describe the idea behind the lower bound that any procedure which does not modify the input samples does not work for $n = o(d/\log d)$. Inspired by the insights from this lower bound, we then discuss a more sophisticated procedure, which is optimal and achieves non-trivial amplification for $n = \Omega(\sqrt{d})$.

Upper Bound for Algorithm which Samples from the Empirical Distribution. Let $\hat{\mu}$ be the empirical mean of the original set X_n. Consider the (n, m) amplification scheme which draws $(m-n)$ new samples from $N(\hat{\mu}, \Sigma)$ and then randomly shuffles together the original samples and

the new samples. We show that for any ϵ, this procedure—with a small modification to facilitate the analysis—achieves $\left(n, n + O\left(n^{\frac{1}{2}-9\epsilon}\right)\right)$ amplification for $n = \frac{d}{\epsilon \log d}$. This is despite the empirical distribution $N(\hat{\mu}, \Sigma)$ being $1 - o(1)$ far in total variation distance from the true distribution $N(\mu, \Sigma)$, for $n = o(d)$.

We now provide the proof intuition for this result. First, note that it is sufficient to prove the result for $\Sigma = I$. This is because all the operations performed by our amplification procedure are invariant under linear transformations. The intuition for the result in the identity covariance case is as follows. Consider $n = \Theta(d/\log d)$. In this case, with high probability, the empirical mean $\hat{\mu}$ satisfies $\|\mu - \hat{\mu}\| = O(\sqrt{\log d}) \leq \sqrt{c \log n}$ for a fixed constant c. If we center and rotate the coordinate system, such that $\hat{\mu}$ has the coordinates $(\|\mu - \hat{\mu}\|, 0, \ldots, 0)$, then the distribution of samples from $N(\hat{\mu}, I)$ and $N(\mu, I)$ only differs along the first axis, and is independent across different axes. Hence, with some technical work, our problem reduces to the following univariate problem: what is the total variation distance between $(n+1)$ samples from the univariate distributions $N(0,1)$ and $\tilde{D}$, where $\tilde{D}$ is a mixture distribution where each sample is drawn from $N(0,1)$ with probability $1 - \frac{1}{n+1}$ and from $N(\sqrt{c \log n}, 1)$ with probability $\frac{1}{n+1}$? We show that the total variation distance between these distributions is small, by bounding the squared Hellinger distance between them. Intuitively, the reason for the total variation distance being small is that, even though one sample from $N(\sqrt{c \log n}, 1)$ is easy to distinguish from one sample from $N(0,1)$, for sufficiently small c it is difficult to distinguish between these two samples in the presence of n other samples from $N(0,1)$. This is because for n draws from $N(0,1)$, with high probability there are $O(n^{1-c})$ samples in a constant length interval around $\sqrt{c \log n}$, and hence it is difficult to detect the presence or absence of one extra sample in this interval.

Algorithm 7 Sample Amplification for Gaussian with Unknown Mean and Fixed Covariance

Input: $X_n = (x_1, x_2, \ldots, x_n)$, where $x_i \leftarrow N(\mu, \Sigma_{d\times d})$.
Output: $Z_m = (x'_1, x'_2, \ldots, x'_m)$, such that $D_{TV}(D^m, Z_m) \leq \frac{1}{3}$, where D is $N(\mu, \Sigma_{d\times d})$

$\hat{\mu} := \sum_{i=1}^{n} \frac{x_i}{n}$
$\epsilon_i \leftarrow N(0, \Sigma_{d\times d})$, for $i \in \{n+1, n+2, \ldots, m\}$ {Draw $m-n$ i.i.d samples from $N(0, \Sigma_{d\times d})$}
$x'_i := \hat{\mu} + \epsilon_i$, for $i \in \{n+1, n+2, \ldots, m\}$
$x'_i := x_i - \sum_{j=n+1}^{m} \frac{\epsilon_j}{n}$, for $i \in \{1, 2, \ldots, n\}$ {Remove correlations between old and new samples}
return $Z_m := (x'_1, x'_2, \ldots, x'_m)$

Lower Bound for any Procedure which Returns a Superset of the Input Samples. We show that procedures which return a superset of the input samples are inherently limited in this Gaussian setting, in the sense that they cannot achieve $(n, n+1)$ amplification for $n \leq \frac{cd}{\log d}$, where c is a fixed constant.

The idea behind the lower bound is as follows. If we consider any arbitrary direction and project a true sample from $N(\mu, I)$ along that direction, then with high probability, the projection lies close

to the projection of the mean. However, for input set X_n with mean $\hat{\mu}$, the projection of an extra sample added by any amplification procedure along the direction $\mu - \hat{\mu}$ will be far from the projection of the mean μ. This is because after seeing just $\frac{cd}{\log d}$ samples, any amplification procedure will have high uncertainty about the location of μ relative to $\hat{\mu}$. Based on this, we construct a verifier which can distinguish between a set of true samples and a set of amplified samples, for $n \leq \frac{cd}{\log d}$.

More formally, Let x'_i be the i-th sample returned by the procedure, and let $\hat{\mu}_{-i}$ be the mean of all except the i-th sample. Let "new" be the index of the additional point added by the amplifier to the original set X_n, hence the amplifier returns the set $\{x'_{\text{new}}, X_n\}$. Note that $\hat{\mu} \leftarrow N(\mu, \frac{I}{n})$, hence $||\mu - \hat{\mu}||^2 \approx \frac{d}{n}$ with high probability. Suppose the verifier evaluates the following inner product for the additional point x'_{new},

$$\langle x'_{\text{new}} - \hat{\mu}_{-\text{new}}, \mu - \hat{\mu}_{-\text{new}} \rangle. \tag{5.1}$$

Note that $\hat{\mu}_{-\text{new}} = \hat{\mu}$ as the amplifier has not modified any of the original samples in X_n. For a point x'_{new} drawn from $N(\mu, I)$, this inner product concentrates around $||\mu - \hat{\mu}||^2 \approx \frac{d}{n}$. We now argue that if the true mean μ is drawn from the distribution $N(0, \sqrt{d}I)$, then the above inner product is much smaller than $\frac{d}{n}$ with high probability over μ. The reason for this is as follows. After seeing the samples in X_n, the amplification algorithm knows that μ lies in a ball of radius $\approx \sqrt{\frac{d}{n}}$ centered at $\hat{\mu}$, but μ could lie along any direction in that ball. Formally, we can show that if μ is drawn from the distribution $N(0, \sqrt{d}I)$, then the posterior distribution of $\mu \mid X_n$ is a Gaussian $N(\bar{\mu}, \bar{\sigma}I)$ with $\bar{\mu} \approx \hat{\mu}$ and $\bar{\sigma} \approx \frac{1}{n}$. As $\mu - \hat{\mu}$ is a random direction, for any x'_{new} that the algorithm returns, the inner product in (5.1) is $\approx ||x'_{\text{new}} - \hat{\mu}||\ ||\mu - \hat{\mu}|| \left(\frac{1}{\sqrt{d}}\right)$ with high probability over the randomness in $\mu \mid X_n$. The verifier checks and ensures that $\|x'_{\text{new}} - \hat{\mu}_{-\text{new}}\| = \|x'_{\text{new}} - \hat{\mu}\| \approx \sqrt{d}$. Hence for any $(n, n+1)$ amplification scheme, the inner product in (5.1) is at most $\approx \sqrt{\frac{d}{n}}$ with high probability over $\mu \mid X_n$. In contrast, we know that this inner product is $\approx \frac{d}{n}$ for a true sample from $N(\mu, I)$.

Finally, note that the algorithm can randomly shuffle the samples, and hence the verifier does the above inner product test for every returned sample x'_i, for a total of $(n+1)$ tests. If $(n+1)$ tests are performed, then the inner product is expected to deviate by $\sqrt{\frac{d \log n}{n}}$ around its expected value of $\frac{d}{n}$, even for $(n+1)$ true samples drawn for the distribution. But if $n \ll \frac{d}{\log d}$, then $\sqrt{\frac{d}{n}} \ll \frac{d}{n} - \sqrt{\frac{d \log n}{n}}$, and hence any $(n, n+1)$ amplification scheme in this regime fails at least one of the following tests with high probability over μ: (1) $\forall\ i \in [n+1], \langle x'_i - \hat{\mu}_{-i}, \mu - \hat{\mu}_{-i} \rangle \geq \frac{d}{n} - \sqrt{\frac{d \log n}{n}}$, and (2) $\forall\ i \in [n+1], \|x'_i - \hat{\mu}_{-i}\| \approx \sqrt{d}$. As true samples pass all the tests with high probability, this shows that $(n, n+1)$ amplification without modifying the provided samples is impossible for $n \ll \frac{d}{\log d}$.

Optimal Amplification Procedure for Gaussians: Algorithm and Upper Bound. The above lower bound shows that it is necessary to modify the input samples X_n to achieve amplification for $n = o(d/\log d)$. What would be the most naive amplification scheme which does not output a superset of the input samples? One candidate could be an amplifier which first estimates the sample mean $\hat{\mu}$ of X_n, and then just outputs m samples from $N(\hat{\mu}, I)$. It is not hard to see that this scheme does not even give a valid (n, n) amplification procedure. The verifier in this case could check the distance between the true mean and the mean of the returned samples, which would be significantly more than expected, with high probability.

How should one modify the input samples then? The above lower bound also shows what such an amplification procedure must achieve—the inner product in (5.1) should be driven towards its expected value of $\frac{d}{n}$ for a true sample drawn from the distribution. Note that the inner product is too small for the algorithm which samples from the empirical distribution $N(\hat{\mu}, I)$ as the generated point x'_{new} is too correlated with the mean $\hat{\mu}_{-\text{new}} = \hat{\mu}$ of the remaining points. We can fix this by shifting the original points in X_n themselves, to hide the correlation between x'_{new} and the original mean $\hat{\mu}$ of X_n. The full procedure is quite simple to state, and is described in Algorithm 7. Note that unlike our other amplification procedures, this procedure does not involve any random shuffling of the samples. We show that this procedure achieves (n, m) amplification for all $d > 0$ and $m = n + O\left(\frac{n}{\sqrt{d}}\right)$.

We now provide a brief proof sketch for this upper bound, for the case when $m = n + 1$. Note that the returned samples in Z_m can also be thought of as a single sample from a $(m \times d)$-dimensional Gaussian distribution $N\Big(\underbrace{(\mu, \mu, \ldots, \mu)}_{m \text{ times}}, \tilde{\Sigma}_{md \times md}\Big)$, as the returned samples are linear combinations of Gaussian random variables. Hence, it is sufficient to find their mean and covariance, and use that to bound their total variation distance to true samples from the distribution (which can also be though of as a single sample from a $(d \times m)$-dimensional Gaussian distribution $N\Big((\mu, \mu, \ldots, \mu), I_{md \times md}\Big)$). The TV distance between the two distributions is proportional to $||\tilde{\Sigma}_{md \times md} - I_{md \times md}||_{\text{F}}$. Our modification procedure removes the correlations between the original samples and the generated samples to ensure that the non-diagonal entries of $\tilde{\Sigma}_{md \times md}$ are small, and hence the total variation distance is also small.

General Lower Bound for Gaussians. We show a lower bound that there is no (n, m) amplification procedure for Gaussian distibutions with unknown mean for $m \geq n + \frac{cn}{\sqrt{d}}$, where c is a fixed constant. The intuition behind the lower bound is that any such amplification procedure could be used to find the true mean μ with much smaller error than what is possible with n samples.

To show this formally, we define a verifier such that for $\mu \leftarrow N(0, \sqrt{d}I)$ and $m > n + \frac{cn}{\sqrt{d}}$, m true samples from $N(\mu, I)$ are accepted by the verifier with high probability over the randomness in the samples, but m samples generated by any (n, m) amplification scheme are rejected by the verifier

with high probability over the randomness in the samples and μ. In this case, the verifier only needs to evaluate the squared distance $||\mu - \hat{\mu}_m||^2$ of the empirical mean $\hat{\mu}_m$ of the returned samples from the true mean μ, and accept the samples if and only if this squared distance is less than $\frac{d}{m} + \frac{c_1\sqrt{d}}{m}$ for some fixed constant c_1. It is not difficult to see why this test is sufficient. Note that for m true samples drawn from $N(\mu, I)$, $||\mu - \hat{\mu}_m||^2 = \frac{d}{m} \pm O\left(\frac{\sqrt{d}}{m}\right)$. Also, the squared distance $||\mu - \hat{\mu}||^2$ of the mean $\hat{\mu}$ of the original set X_n from the true mean μ is concentrated around $\frac{d}{n} \pm O\left(\frac{\sqrt{d}}{n}\right)$. Using this, for $m > n + \frac{cn}{\sqrt{d}}$, we can show that no algorithm can find a $\hat{\mu}_m$ which satisfies $||\mu - \hat{\mu}_m||^2 \leq \frac{d}{m} \pm O\left(\frac{\sqrt{d}}{m}\right)$ with decent probability over $\mu \leftarrow N(0, \sqrt{d}I)$. This is because the algorithm only knows μ up to squared error $\frac{d}{n} \pm O\left(\frac{\sqrt{d}}{n}\right)$ based on the original set X_n.

5.3 Proofs: Gaussian with Unknown Mean and Fixed Covariance

5.3.1 Upper Bound

In this section, we prove the upper bound in Theorem 17 by showing that Algorithm 7 can be used as a $(n, n + \frac{n}{\sqrt{d}})$ amplification procedure.

First, note that it is sufficient to prove the theorem for the case when input samples come from an identity covariance Gaussian. This is because, for the purpose of analysis we can transform our samples to those coming from indentity covariance Gaussian, as our amplification procedure is invariant to linear transformations to samples. In particular, let f_Σ denote our amplification procedure for samples coming from $N(\mu, \Sigma)$, and, $Y_n = (y_1, y_2, \ldots, y_n)$ denote the random variable corresponding to n samples from $N(\mu, \Sigma)$. Let $X_n = (x_1, x_2, \ldots, x_n)$ denote n samples from $N(\mu, I)$, such that $Y_n = \Sigma^{\frac{1}{2}}(X_n - \mu) + \mu = (\Sigma^{\frac{1}{2}}(x_1 - \mu) + \mu, \Sigma^{\frac{1}{2}}(x_2 - \mu) + \mu, \ldots, \Sigma^{\frac{1}{2}}(x_n - \mu) + \mu)$. Due to invariance of our amplification procedure to linear transformations, we get that $\Sigma^{\frac{1}{2}}(f_I(X_n) - \mu) + \mu$ is equal in distribution to $f_\Sigma(\Sigma^{\frac{1}{2}}(X_n - \mu) + \mu) = f_\Sigma(Y_n)$. This gives us

$$\begin{aligned} D_{TV}(f_\Sigma(Y_n), Y_m) &= D_{TV}(f_\Sigma(\Sigma^{\frac{1}{2}}(X_n - \mu) + \mu), \Sigma^{\frac{1}{2}}(X_m - \mu) + \mu) \\ &= D_{TV}(\Sigma^{\frac{1}{2}}(f_I(X_n) - \mu) + \mu, \Sigma^{\frac{1}{2}}(X_m - \mu) + \mu) \\ &\leq D_{TV}(f_I(X_n), X_m), \end{aligned}$$

where the last inequality is true because the total variation distance between two distributions can't increase if we apply the same transformation to both the distributions. Hence, we can conclude that it is sufficient to prove our results for identity covariance case. This is true for both the amplification procedures for Gaussians that we have discussed. So in this whole section, we will work with identity covariance Gaussian distributions.

Proposition 2. *Let $\mathcal{C}$ denote the class of $d-$dimensional Gaussian distributions $N(\mu, I)$ with unknown mean μ. For all $d, n > 0$ and $m = n + O\left(\frac{n}{\sqrt{d}}\right)$, $\mathcal{C}$ admits an (n, m) amplification procedure.*

Proof. The amplification procedure consists of two parts. The first uses the provided samples to learn the empirical mean $\hat{\mu}$ and generate $m-n$ new samples from $\mathcal{N}(\hat{\mu}, I)$. The second part adjusts the first n samples to "hide" the correlations that would otherwise arise from using the empirical mean to generate additional samples.

Let $\epsilon_{n+1}, \epsilon_{n+2}, \ldots, \epsilon_m$ be $m-n$ i.i.d. samples generated from $N(0, I)$, and let $\hat{\mu} = \frac{\sum_{i=1}^{n} x_i}{n}$. The amplification procedure will return $x'_1, \ldots, x'_m$ with:

$$x'_i = \begin{cases} x_i - \frac{\sum_{j=n+1}^{m} \epsilon_j}{n}, & \text{for } i \in \{1, 2, \ldots, n\} \\ \hat{\mu} + \epsilon_i, & \text{for } i \in \{n+1, n+2, \ldots, m\}. \end{cases} \tag{5.2}$$

We will show later in this proof that subtracting $\frac{\sum_{j=n+1}^{m} \epsilon_j}{n}$ will serve to decorrelate the first n samples from the remaining samples.

Let $f_{\mathcal{C},n,m} : S^n \to S^m$ be the random function denoting the map from X_n to Z_m as described above, where $S = \mathbb{R}^d$. We need to show

$$D_{TV}\left(Z_m = f_{\mathcal{C},n,m}(X_n), X_m\right) \leq 1/3,$$

where X_n and X_m denote n and m independent samples from $N(\mu, I)$ respectively.

For ease of understanding, we first prove this result for the univariate case, and then extend it to the general setting.

So consider the setting where $d = 1$. In this case, X_m corresponds to m i.i.d. samples from a Gaussian with mean μ, and variance 1. X_m can also be thought of as a single sample from an $m-$dimensional Gaussian $N\Big(\underbrace{(\mu, \mu, \ldots, \mu)}_{m \text{ times}}, I_{m\times m}\Big)$. Now, $f_{\mathcal{C},n,m}$ is a map that takes n i.i.d samples from $N(\mu, 1)$, $m-n$ i.i.d samples (ϵ_i) from $N(0, 1)$, and outputs m samples that are a linear combination of the m input samples. So, $f_{\mathcal{C},n,m}(X_n)$ can be thought of as a $m-$dimensional random variable obtained by applying a linear transformation to a sample drawn from $N\Big(\Big(\underbrace{\mu, \mu, \ldots, \mu}_{n \text{ times}}, \underbrace{0, 0, \ldots, 0}_{m-n \text{ times}}\Big), I_{m\times m}\Big)$. As a linear transformation applied to Gaussian random variable outputs a Gaussian random variable, we get that $Z_m = (x'_1, x'_2, \ldots, x'_m)$ is distributed according to $N(\tilde{\mu}, \Sigma_{m\times m})$, where $\tilde{\mu}$ and $\Sigma_{m\times m}$ denote the mean and covariance. Note that $\tilde{\mu} = \underbrace{(\mu, \mu, \ldots, \mu)}_{m \text{ times}}$ as

$$\mathbb{E}[x'_i] = \begin{cases} \mathbb{E}[x_i] - \mathbb{E}\left[\frac{\sum_{j=n+1}^{m} \epsilon_j}{n}\right] = \mu - 0 = \mu, & \text{for } i \in \{1, 2, \ldots, n\} \\ \mathbb{E}[\hat{\mu}] + \mathbb{E}[\epsilon_i] = \mu + 0 = \mu, & \text{for } i \in \{n+1, n+2, \ldots, m\}. \end{cases} \tag{5.3}$$

Next, we compute the covariance matrix $\Sigma_{m\times m}$.

For $i = j$, and $i \in \{1, 2, \ldots, n\}$, we get

$$\begin{aligned}
\Sigma_{ii} &= \mathbb{E}[(x'_i - \mu)^2] \\
&= \mathbb{E}\left[(x_i - \mu)^2\right] + \mathbb{E}\left[\left(\frac{\sum_{j=n+1}^{m} \epsilon_j}{n}\right)^2\right] \\
&= 1 + \frac{m-n}{n^2}.
\end{aligned}$$

For $i = j$, and $i \in \{n+1, n+2, \ldots, n+m\}$, we get

$$\begin{aligned}
\Sigma_{ii} &= \mathbb{E}\left[(x'_i - \mu)^2\right] \\
&= \mathbb{E}\left[(\hat{\mu} - \mu)^2\right] + \mathbb{E}\left[\epsilon_i^2\right] \\
&= \frac{1}{n} + 1.
\end{aligned}$$

For $i \in \{1, 2, \ldots, n\}, j \in \{n+1, n+2, \ldots, n+m\}$, we get

$$\begin{aligned}
\Sigma_{ij} &= \mathbb{E}\left[(x'_i - \mu)\left(x'_j - \mu\right)\right] \\
&= \mathbb{E}\left[\left(x_i - \frac{\sum_{k=n+1}^{m} \epsilon_k}{n} - \mu\right)(\hat{\mu} + \epsilon_j - \mu)\right] \\
&= \mathbb{E}[(x_i - \mu)(\hat{\mu} - \mu)] - \mathbb{E}\left[\left(\frac{\sum_{k=n+1}^{m} \epsilon_k}{n}\right)(\epsilon_j)\right] \\
&= \frac{1}{n} - \frac{1}{n} \\
&= 0.
\end{aligned}$$

For $i, j \in \{1, 2, \ldots, n\}, i \neq j$, we get

$$\begin{aligned}
\Sigma_{ij} &= \mathbb{E}\Big[\,(x'_i - \mu)\left(x'_j - \mu\right)\Big] \\
&= \mathbb{E}\left[\left(x_i - \frac{\sum_{k=n+1}^{m} \epsilon_k}{n} - \mu\right)\left(x_j - \frac{\sum_{k=n+1}^{m} \epsilon_k}{n} - \mu\right)\right] \\
&= \mathbb{E}\left[(x_i - \mu)(x_j - \mu)\right] + \mathbb{E}\left[\left(\frac{\sum_{k=n+1}^{m} \epsilon_k}{n}\right)^2\right] \\
&= \frac{m-n}{n^2}.
\end{aligned}$$

For $i, j \in \{n+1, n+2, \ldots, m\}, i \neq j$, we get

$$\begin{aligned}\Sigma_{ij} &= \mathbb{E}[(x'_i - \mu)\,(x'_j - \mu)] \\ &= \mathbb{E}[(\hat{\mu} + \epsilon_i - \mu)\,(\hat{\mu} + \epsilon_j - \mu)] \\ &= \mathbb{E}\left[(\hat{\mu} - \mu)^2\right] \\ &= \frac{1}{n}.\end{aligned}$$

This gives us

$$\Sigma_{m\times m} = \begin{bmatrix} 1+\frac{m-n}{n^2} & \frac{m-n}{n^2} & \cdots & \frac{m-n}{n^2} & 0 & 0 & \cdots & 0 \\ \frac{m-n}{n^2} & 1+\frac{m-n}{n^2} & \cdots & \frac{m-n}{n^2} & 0 & 0 & \cdots & 0 \\ \vdots & \cdots & \cdots & \vdots & \vdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \frac{m-n}{n^2} & \vdots & \cdots & \cdots & \vdots \\ \frac{m-n}{n^2} & \cdots & \frac{m-n}{n^2} & 1+\frac{m-n}{n^2} & 0 & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & 0 & 1+\frac{1}{n} & \frac{1}{n} & \cdots & \frac{1}{n} \\ 0 & \cdots & \cdots & 0 & \frac{1}{n} & 1+\frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & \cdots & \cdots & \vdots & \vdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots & \vdots & \cdots & \cdots & \frac{1}{n} \\ 0 & \cdots & \cdots & 0 & \frac{1}{n} & \cdots & \frac{1}{n} & 1+\frac{1}{n} \end{bmatrix}.$$

Now, finding $D_{TV}\left(Z_m, X_m\right)$ reduces to computing $D_{TV}\left(N\left(\tilde{\mu}, I_{m\times m}\right), N\left(\tilde{\mu}, \Sigma_{m\times m}\right)\right)$. From [48, Theorem 1.1], we know that $D_{TV}\left(N\left(\tilde{\mu}, I_{m\times m}\right), N\left(\tilde{\mu}, \Sigma_{m\times m}\right)\right) \leq \min\left(1, \frac{3}{2}||\Sigma - I||_F\right)$. This gives us

$$\begin{aligned} D_{TV}\left(N\left(\tilde{\mu}, I_{m\times m}\right), N\left(\tilde{\mu}, \Sigma_{m\times m}\right)\right) &\leq \min\left(1, \frac{3}{2}||\Sigma - I||_F\right) \\ &\leq \sqrt{\frac{3}{2}\left(\left(\frac{m-n}{n^2}\right)^2 n^2 + \frac{1}{n^2}(m-n)^2\right)} \\ &= \frac{\sqrt{3}\,(m-n)}{n}. \end{aligned} \tag{5.4}$$

Now, for $d > 1$, by a similar argument as above, X_m can be thought of as d independent samples from the following d distributions: $N\Big(\underbrace{(\mu_1, \mu_1, \ldots, \mu_1)}_{m \text{ times}}, I_{m\times m}\Big), \ldots, N\Big(\underbrace{(\mu_d, \mu_d, \ldots, \mu_d)}_{m \text{ times}}, I_{m\times m}\Big)$. Or equivalently, as a single sample from $N\Big(\Big(\underbrace{\mu_1, \mu_1, \ldots, \mu_1}_{m \text{ times}}, \ldots, \underbrace{\mu_d, \mu_d, \ldots, \mu_d}_{m \text{ times}}\Big), I_{md\times md}\Big)$. Similarly,

Z_m can be thought of as d independent samples from $N\Big(\underbrace{(\mu_i, \mu_i, \ldots, \mu_i)}_{m \text{ times}}, \Sigma_{m\times m}\Big)$, or equivalently, a single sample from $N\Big(\Big(\underbrace{\mu_1, \mu_1, \ldots, \mu_1}_{m \text{ times}}, \ldots, \underbrace{\mu_d, \mu_d, \ldots, \mu_d}_{m \text{ times}}\Big), \tilde{\Sigma}_{md\times md}\Big)$ where $\tilde{\Sigma}_{md\times md}$ is a block diagonal matrix with block diagonal entries equal to $\Sigma_{m\times m}$ (denoted as Σ in the figure).

$$\tilde{\Sigma}_{md\times md} = \begin{bmatrix} \Sigma & 0 & \cdots & \cdots & 0 \\ 0 & \Sigma & 0 & \cdots & \vdots \\ \vdots & 0 & \ddots & 0 & \vdots \\ \vdots & \cdots & 0 & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & \Sigma \end{bmatrix}.$$

Similar to (5.4), we get

$$\begin{aligned} D_{TV}\Big(N\Big(\Big(\underbrace{\mu_1, \mu_1, \ldots, \mu_1}_{m \text{ times}}, \ldots, \underbrace{\mu_d, \mu_d, \ldots, \mu_d}_{m \text{ times}}\Big), I_{md\times md}\Big), N\Big(\Big(\underbrace{\mu_1, \mu_1, \ldots, \mu_1}_{m \text{ times}}, \ldots, \underbrace{\mu_d, \mu_d, \ldots, \mu_d}_{m \text{ times}}\Big), \tilde{\Sigma}_{md\times md}\Big)\Big) \\ &\leq \min\left(1, \frac{3}{2}||\tilde{\Sigma} - I||_F\right) \\ &\leq \sqrt{d\left(\frac{3}{2}\left(\left(\frac{m-n}{n^2}\right)^2 n^2 + \frac{1}{n^2}(m-n)^2\right)\right)} \\ &= \frac{\sqrt{3d}\,(m-n)}{n}. \end{aligned}$$

If we want the total variation distance to be less than δ, we get $m - n = O\left(\frac{n\delta}{\sqrt{d}}\right)$. Setting $\delta = \frac{1}{3}$, we get $m = n + O\left(\frac{n}{\sqrt{d}}\right)$, which completes the proof. □

5.3.2 Lower Bound

In this section we prove the lower bound from Theorem 17 and show that it is impossible to amplify beyond $O\left(\frac{n}{\sqrt{d}}\right)$ more samples. The intuition behind the lower bound is that any such amplification procedure could be used to find the true mean μ with much smaller error than what is possible with n samples.

To show this formally, we define a verifier such that for $\mu \leftarrow N(0, \sqrt{d}I)$ and $m > n + \frac{cn}{\sqrt{d}}$, m true samples from $N(\mu, I)$ are accepted by the verifier with high probability over the randomness in the samples, but m samples generated by any (n, m) amplification scheme are rejected by the verifier with high probability over the randomness in the samples and μ. In this case, the verifier only needs to evaluate the squared distance $\|\mu - \hat{\mu}_m\|^2$ of the empirical mean $\hat{\mu}_m$ of the returned samples from the true mean μ, and accept the samples if and only if this squared distance is less

than $\frac{d}{m} + \frac{c_1\sqrt{d}}{m}$ for some fixed constant c_1. It is not difficult to see why this test is sufficient. Note that for m true samples drawn from $N(\mu, I)$, $\|\mu - \hat{\mu}_m\|^2 = \frac{d}{m} \pm O\left(\frac{\sqrt{d}}{m}\right)$. Also, the squared distance $\|\mu - \hat{\mu}^2\|$ of the mean $\hat{\mu}$ of the original set X_n from the true mean μ is concentrated around $\frac{d}{n} \pm O\left(\frac{\sqrt{d}}{n}\right)$. Using this, for $m > n + \frac{cn}{\sqrt{d}}$, we can show that no algorithm can find a $\hat{\mu}_m$ which satisfies $\|\mu - \hat{\mu}_m\|^2 \leq \frac{d}{m} \pm O\left(\frac{\sqrt{d}}{m}\right)$ with decent probability over $\mu \leftarrow N(0, \sqrt{d}I)$. This is because the algorithm only knows μ up to squared error $\frac{d}{n} \pm O\left(\frac{\sqrt{d}}{n}\right)$ based on the original set X_n.

Proposition 3. *Let $\mathcal{C}$ denote the class of $d-$dimensional Gaussian distributions $N(\mu, I)$ with unknown mean μ. There is a fixed constant c such that for all sufficiently large $d, n > 0$, $\mathcal{C}$ does not admit an (n, m) amplification procedure for $m \geq n + \frac{cn}{\sqrt{d}}$.*

Proof. Note that it is sufficient to prove the theorem for $m = n + cn/\sqrt{d}$ for a fixed constant c, as an amplification procedure for $m > n + cn/\sqrt{d}$ implies an amplification procedure for $m = n + cn/\sqrt{d}$ by discarding the residual samples. To prove the theorem for $m = n + cn/\sqrt{d}$, we will define a distribution D_μ over μ and a verifier $v(Z_m)$ for the distribution $N(\mu, I)$ which takes as input a set Z_m of m samples, such that: (i) for all μ, the verifier $v(Z_m)$ will accept with probability $1-1/e^2$ when given as input a set Z_m of m i.i.d. samples from $N(\mu, I)$, (ii) but will reject any (n, m) amplification procedure for $m = n + cn/\sqrt{d}$ with probability $1 - 1/e^2$, where the probability is with respect to the randomness in $\mu \leftarrow D_\mu$, the set X_n and in any internal randomness of the amplifier. Note that by Definition 31 of an amplification procedure, this implies that there is no (n, m) amplification procedure for $m = n + cn/\sqrt{d}$.

We now define the distribution D_μ and the verifier $v(Z_m)$. We choose D_μ to be $N(0, \sqrt{d}I)$. Let $\hat{\mu}_m$ be the mean of the samples Z_m returned by the amplification procedure. The verifier $v(Z_m)$ performs the following test, accepts if $\hat{\mu}_m$ passes the test, and rejects otherwise—

$$\left| \|\hat{\mu}_m - \mu\|^2 - d/m \right| \leq 10\sqrt{d}/m. \tag{5.5}$$

We first show that m i.i.d. samples from $N(\mu, I)$ pass the above test with probability $1 - 1/e^2$. We will use the following concentration bounds for a χ^2 random variable Z with d degrees of freedom [92, 131],

$$\Pr\left[Z - d \geq 2\sqrt{dt} + 2t\right] \leq e^{-t}, \ \forall\, t > 0, \tag{5.6}$$

$$\Pr\left[|Z - d| \geq dt\right] \leq 2e^{-dt^2/8}, \ \forall\, t \in (0, 1). \tag{5.7}$$

Note that $\hat{\mu}_m \leftarrow N(\mu, \frac{I}{m})$ for m i.i.d. samples from $N(\mu, I)$. Hence by using (5.7) and setting $t = 10/\sqrt{d}$,

$$\Pr\left[\left| \|\hat{\mu}_m - \mu\|^2 - d/m \right| > 10\sqrt{d}/m\right] \leq 1/e^3.$$

Hence m i.i.d. samples from $N(\mu, I)$ pass the test with probability at least $1 - 1/e^2$.
We now show that for μ sampled from $D_\mu = N(0, \sqrt{d}I)$, the verifier rejects any (n, m) amplification procedure for $m = n + cn/\sqrt{d}$ with high probability over the randomness in μ. Let $D_{\mu|X_n}$ be the posterior distribution of μ conditioned on the set X_n. We will show that for any set X_n received by the amplifier, the amplified set Z_m is accepted by the verifier with probability at most $1/e^2$ over $\mu \leftarrow D_{\mu|X_n}$. This implies that with probability $1 - 1/e^2$ over the randomness in $\mu \leftarrow D_\mu$, the set X_n and any internal randomness in the amplifier, the amplifier cannot output a set Z_m which is accepted by the verifier, completing the proof of Proposition 3.
To show the above claim, we first find the posterior distribution $D_{\mu|X_n}$ of μ conditioned on the amplifier's set X_n. Let μ_0 be the mean of the set X_n. By standard Bayesian analysis (see, for instance, [101]), the posterior distribution $D_{\mu|X_n} = N(\bar{\mu}, \bar{\sigma}^2 I)$, where,

$$\bar{\mu} = \frac{n}{n + 1/\sqrt{d}}\mu_0, \quad \bar{\sigma}^2 = \frac{1}{n + 1/\sqrt{d}}.$$

We show that any set Z_m returned by the amplifier for $m = n + 100n/\sqrt{d}$ fails the test (5.5) with probability $1 - 1/e^2$ over the randomness in $\mu \mid X_n$. We expand $\|\hat{\mu}_m - \mu\|^2$ in the test as follows,

$$\begin{aligned}\|\hat{\mu}_m - \mu\|^2 &= \|\hat{\mu}_m - \overline{\mu} - (\mu - \overline{\mu})\|^2 \\ &= \|\hat{\mu}_m - \bar{\mu}\|^2 - 2\langle \hat{\mu}_m - \bar{\mu}, \mu - \bar{\mu}\rangle + \|\mu - \bar{\mu}\|^2.\end{aligned}$$

By using (5.7) and setting $t = 10/\sqrt{d}$, with probability $1 - 1/e^3$,

$$\begin{aligned}\|\mu - \bar{\mu}\|^2 &\geq \frac{d}{n + 1/\sqrt{d}} - \frac{10\sqrt{d}}{n + 1/\sqrt{d}} \\ &\geq \Big(\frac{d}{n}\Big)\Big(1 - \frac{1}{n\sqrt{d}}\Big) - \frac{10\sqrt{d}}{n} \\ &= d/n - \sqrt{d}/n^2 - 10\sqrt{d}/n \\ &\geq d/n - 12\sqrt{d}/n.\end{aligned}$$

As $\mu \mid X_n \leftarrow N(\bar{\mu}, \bar{\sigma}^2)$, $\langle \hat{\mu}_m - \bar{\mu}, \mu - \bar{\mu}\rangle$ is distributed as $N(0, \bar{\sigma}^2\|\hat{\mu}_m - \bar{\mu}\|^2)$. Hence with probability $1 - 1/e^3$, $\langle \hat{\mu}_m - \bar{\mu}, \mu - \bar{\mu}\rangle \leq 10\|\hat{\mu}_m - \bar{\mu}\|/\sqrt{n + 1/\sqrt{d}} \leq 10\|\hat{\mu}_m - \bar{\mu}\|/\sqrt{n}$. Therefore, with probability $1 - 2/e^3$,

$$\|\hat{\mu}_m - \mu\|^2 \geq \|\hat{\mu}_m - \bar{\mu}\|^2 - (20/\sqrt{n})\|\hat{\mu}_m - \bar{\mu}\| + d/n - 12\sqrt{d}/n.$$

We claim that $\|\hat{\mu}_m - \bar{\mu}\|^2 - 20\|\hat{\mu}_m - \bar{\mu}\|/\sqrt{n} \geq -100/n$. To verify, note that $\|\hat{\mu}_m - \bar{\mu}\|^2 - 20\|\hat{\mu}_m - \bar{\mu}\|/\sqrt{n} +$

$100/n$ is a non-negative quadratic function in $\|\hat{\mu}_m - \bar{\mu}\|$. Therefore, with probability at least $1-2/e^3$,

$$\|\hat{\mu}_m - \mu\|^2 \geq -100/n + d/n - \sqrt{d}/n^2 - 10\sqrt{d}/n \geq d/n - 20\sqrt{d}/n.$$

To pass (5.5), $\|\hat{\mu}_m - \mu\|^2 \leq d/m + 10\sqrt{d}/m$. Therefore, if an amplifier passes the test with probability greater than $1 - 2/e^3$ over the randomness in $\mu \mid X_n$ for $m = n + 100n/\sqrt{d}$, then,

$$\begin{aligned}
& d/n - 20\sqrt{d}/n \leq \|\hat{\mu}_m - \mu\|^2 \leq d/m + 10\sqrt{d}/m, \\
\implies & d/n - 20\sqrt{d}/n \leq d/m + 10\sqrt{d}/m, \\
\implies & d/n - 20\sqrt{d}/n \leq d/(n + 100n/\sqrt{d}) + 10\sqrt{d}/(n + 100n/\sqrt{d}), \\
\implies & d/n - 20\sqrt{d}/n \leq d/n(1 + 100/\sqrt{d})^{-1} + 10\sqrt{d}/n(1 + 100/\sqrt{d})^{-1}, \\
\implies & d/n - 20\sqrt{d}/n \leq d/n(1 - 50/\sqrt{d}) + 10\sqrt{d}/n(1 - 50/\sqrt{d}), \\
\implies & -20\sqrt{d}/n \leq -40\sqrt{d}/n - 1000/n, \\
\implies & -20\sqrt{d}/n \leq -30\sqrt{d}/n,
\end{aligned}$$

which is a contradiction. Hence for $m = n + 100n/\sqrt{d}$, every (n, m) amplifier is rejected by the verifier with probability greater than $1 - 1/e^2$ over the randomness in μ, the set X_n, and any internal randomness of the amplifier.

□

5.3.3 Upper Bound for Procedures which Returns a Superset of the Input Samples

In this section we prove the upper bound in Proposition 1. The algorithm itself is presented in Algorithm 8. Before we proceed with the proof we prove a brief lemma that will be useful for bounding the total variation distance.

Lemma 9. *Let X, Y_1, Y_2 be random variables such that with probability at least $1 - \epsilon$ over X, $D_{TV}(Y_1|X, Y_2|X) \leq \epsilon'$, then $D_{TV}((X, Y_1), (X, Y_2)) \leq \epsilon + \epsilon'$.*

Proof. From the definition of total variation distance, we know

$$\begin{aligned}
D_{TV}((X, Y_1), (X, Y_2)) &= \frac{1}{2} \sum_{x,y} |\Pr((X, Y_1) = (x, y)) - \Pr((X, Y_2) = (x, y)))| \\
&= \frac{1}{2} \sum_{x,y} \Pr(X = x) \left|\Pr\left(Y_1 = y \mid X = x\right) - \Pr(Y_2 = y \mid X = x)\right| \\
&= \sum_{x} \Pr(X = x) \; \frac{1}{2} \sum_{y} |\Pr(Y_1 = y \mid X = x) - Pr(Y_2 = y \mid X = x)| \\
&= \sum_{x} \Pr(X = x) \; d_{TV}(Y_1 \mid X = x, Y_2 \mid X = x).
\end{aligned}$$

Since with probability $(1-\epsilon)$ over X, $d_{TV}(Y_1 \mid X, Y_2 \mid X)$ is at most ϵ', and total variation distance is always bounded by 1, we get $\sum_x Pr(X=x)\ d_{TV}(Y_1 \mid X=x, Y_2 \mid X=x) \leq (1-\epsilon)\epsilon' + \epsilon \leq \epsilon' + \epsilon$. This same proof with summations appropriately replaced with integrals will go through when the random variables in consideration are defined over continuous domains. □

Now we prove the upper bound from Proposition 1. As in Proposition 2, it is sufficient to prove this bound only for the case of identity covariance gaussians as our algorithm in this case is also invariant to linear transformation.

Proposition 4. *Let $\mathcal{C}$ denote the class of $d-$dimensional Gaussian distributions $N(\mu, I)$ with unknown mean μ. There is a constant c' such that for any ϵ, and $n = \frac{d}{\epsilon \log d}$, and for sufficiently large d, there is an $\left(n, n + c' n^{\frac{1}{2}-9\epsilon}\right)$ amplification protocol for $\mathcal{C}$ that returns a superset of the original n samples.*

Algorithm 8 Sample Amplification for Gaussian with Unknown Mean and Fixed Covariance Without Modifying Input Samples

Input: $X_n = (x_1, x_2, \ldots, x_n)$, where $x_i \leftarrow N(\mu, \Sigma_{d\times d})$.
Output: $Z_m = (x'_1, x'_2, \ldots, x'_m)$, such that $D_{TV}(D^m, Z_m) \leq \frac{1}{3}$, where D is $N(\mu, \Sigma_{d\times d})$

$r := m - n$
$\hat{\mu} := \sum_{i=1}^{\frac{n}{2}} \frac{x_i}{n/2}$
$x'_i := x_i$, for $i \in \{1, 2, \ldots, \frac{n}{2}\}$
$X_{\text{remaining}} := (x_{\frac{n}{2}+1}, x_{\frac{n}{2}+2}, \ldots, x_n)$
for $i = \frac{n}{2} + 1$ **to** m **do**
  $T \leftarrow$ Bernoulli($\frac{2r}{r+n/2}$) {Set $T = 1$ with probability $\frac{2r}{r+n/2}$, and 0 otherwise}
  if T equals 1 **then**
    $x'_i \leftarrow N(\hat{\mu}, \Sigma_{d\times d})$
  else
    if $X_{\text{remaining}}$ is not empty **then**
      $x'_i :=$ Random Element Drawn without Replacement from $X_{\text{remaining}}$
    else
      $x'_i := x_1$ {H}appens with small probability
    end if
  end if
end for
$Z_m := (x'_1, x'_2, \ldots, x'_m)$
return Z_m

Proof. Let $m = n + r$,where $r = O\left(n^{\frac{1}{2}-9\epsilon}\right)$. We begin by describing the procedure to generate m samples $Z_m = (x'_1, x'_2, \ldots, x'_m)$, given n i.i.d. samples $X_n = (x_1, x_2, \ldots, x_n)$ drawn from $N(\mu, I)$. Let $\tilde{\mu} = \sum_{i=1}^{n/2} \frac{x_i}{n/2}$ denote the mean of first $\frac{n}{2}$ samples in X_n. For distributions P and Q, let $(1-\alpha)P + \alpha Q$ denote the mixture distribution where $(1-\alpha)$ and α are the respective mixture weights.

We first describe how to generate $Z'_m = (x''_1, x''_2, \ldots, x''_m)$, given n i.i.d samples X_n. For $i \in \{1, 2, \ldots, \frac{n}{2}\}$, we set $x''_i = x_i$. For $i \in \{\frac{n}{2}+1, \frac{n}{2}+2, \ldots, m\}$, we set x''_i to a random independent draw from the mixture distribution $\left(1 - \frac{10r}{r+\frac{n}{2}}\right) N(\mu, I_{d\times d}) + \frac{10r}{r+\frac{n}{2}} N(\tilde{\mu}, I_{d\times d})$.

Now, the construction of Z_m is very similar to Z'_m except that we don't have access to $N(\mu, I_{d\times d})$ to sample points from the mixture distribution. So, for Z_m, set $x'_i = x_i$ for $i \in \{1, 2, \ldots, \frac{n}{2}\}$. For $i \in \{\frac{n}{2}+1, \frac{n}{2}+2, \ldots, m\}$, we use samples from $(x_{\frac{n}{2}+1}, x_{\frac{n}{2}+2}, \ldots, x_n)$ instead of producing new samples from $N(\mu, I_{d\times d})$. With probability $\left(1 - \frac{10r}{r+\frac{n}{2}}\right)$, we generate a random sample without replacement from $\left(x_{\frac{n}{2}+1}, x_{\frac{n}{2}+2}, \ldots, x_n\right)$, and with probability $\frac{10r}{r+\frac{n}{2}}$ we generate a sample from $N(\tilde{\mu}, I)$, and set x'_i equal to that sample. As we sample from $(x_{\frac{n}{2}+1}, x_{\frac{n}{2}+2}, \ldots, x_n)$ without replacement, we can generate only $\frac{n}{2}$ samples this way. The expected number of samples needed is $(\frac{n}{2}+r)(1-\frac{10r}{r+\frac{n}{2}}) = \frac{n}{2} - 9r$, and with high probability, we won't need more than $\frac{n}{2}$ samples. If the total number of required samples from $\left(x_{\frac{n}{2}+1}, x_{\frac{n}{2}+2}, \ldots, x_n\right)$ turns out to be more than $\frac{n}{2}$, we set x_i to an arbitrary $d-$dimensional vector (say x_1) but this happens with low probability, leading to insignificant loss in total variation distance.

Let X_m denote the random variable corresponding to m i.i.d. samples from $N(\mu, I)$. We want to show that $D_{TV}(X_m, Z_m)$ is small. By triangle inequality, $D_{TV}(X_m, Z_m) \leq D_{TV}(X_m, Z'_m) + D_{TV}(Z'_m, Z_m)$.

We first bound $D_{TV}(Z_m, Z'_m)$. Let $Y, Y' \leftarrow \text{Binomial}\left(r + \frac{n}{2}, 1 - \frac{10r}{r+\frac{n}{2}}\right)$ be random variables that denotes the number of samples from $(1 - \frac{10r}{r+\frac{n}{2}})$ mixture component in Z_m and Z'_m respectively. Let Ω denote the sample space of Z_m and Z'_m.

$$\begin{aligned}
D_{TV}(Z_m, Z'_m) &= \max_{E\subseteq\Omega} |\Pr(Z_m \in E) - \Pr(Z'_m \in E)| \\
&= \max_{E\subseteq\Omega} |\Pr\left(Z_m \in E \mid Y \leq \frac{n}{2}\right) \Pr\left(Y \leq \frac{n}{2}\right) + \Pr\left(Z_m \in E \mid Y > \frac{n}{2}\right) \Pr\left(Y > \frac{n}{2}\right) \\
&\qquad - \Pr\left(Z'_m \in E \mid Y' \leq \frac{n}{2}\right) \Pr\left(Y' \leq \frac{n}{2}\right) - \Pr\left(Z'_m \in E \mid Y' > \frac{n}{2}\right) \Pr\left(Y' > \frac{n}{2}\right)|
\end{aligned}$$

Since Y and Y' have the same distribution, we have $\Pr\left(Y' \leq \frac{n}{2}\right) = \Pr\left(Y \leq \frac{n}{2}\right)$, and $\Pr\left(Y' > \frac{n}{2}\right) = \Pr\left(Y > \frac{n}{2}\right)$. This gives us

$$\begin{aligned}
D_{TV}(Z_m, Z'_m) &= \max_{E\subseteq\Omega} |\Pr\left(Z_m \in E \mid Y \leq \frac{n}{2}\right) \Pr\left(Y \leq \frac{n}{2}\right) + \Pr\left(Z_m \in E \mid Y > \frac{n}{2}\right) \Pr\left(Y > \frac{n}{2}\right) \\
&\qquad - \Pr\left(Z'_m \in E \mid Y' \leq \frac{n}{2}\right) \Pr\left(Y \leq \frac{n}{2}\right) - \Pr\left(Z'_m \in E \mid Y' > \frac{n}{2}\right) \Pr\left(Y > \frac{n}{2}\right)| \\
&\leq \max_{E\subseteq\Omega} \Pr\left(Y \leq \frac{n}{2}\right) | \Pr\left(Z_m \in E | Y \leq \frac{n}{2}\right) - \Pr\left(Z'_m \in E | Y' \leq \frac{n}{2}\right) | \\
&\qquad + \Pr\left(Y > \frac{n}{2}\right) | \Pr\left(Z_m \in E | Y > \frac{n}{2}\right) - \Pr\left(Z'_m \in E | Y' > \frac{n}{2}\right) | .
\end{aligned}$$

where the last inequality holds because of the triangle inequality. Now, note that $\Pr(Z_m \in E|Y \leq \frac{n}{2}) = \Pr(Z'_m \in E|Y' \leq \frac{n}{2})$ for all E, and $|\Pr(Z_m \in E|Y > \frac{n}{2}) - \Pr(Z'_m \in E|Y' > \frac{n}{2})| \leq 1$. This gives

us

$$D_{TV}(Z_m, Z'_m) \leq \Pr\left(Y > \frac{n}{2}\right).$$

We know $\mathbb{E}[Y] = \frac{n}{2} - 9r$, and $\text{Var}[Y] = \left(\frac{n}{2} + r\right)\left(1 - \frac{10r}{\frac{n}{2}+r}\right)\left(\frac{10r}{\frac{n}{2}+r}\right) \leq 10r$. Using Bernstein's inequality, we get

$$\begin{aligned}\Pr\left[Y > \frac{n}{2}\right] &= \Pr(Y - \mathbb{E}[Y] > 9r) \\ &\leq \exp\left(\frac{-(9r)^2}{2(10r + 9r/3)}\right) \\ &\leq \exp\left(\frac{-81r}{26}\right).\end{aligned}$$

So we get $D_{TV}(Z_m, Z'_m) \leq \exp\left(\frac{-81r}{26}\right)$.

Next, we calculate $D_{TV}(X_m, Z'_m)$. We write $X_m = (X_m^1, X_m^2)$ and $Z'_m = (Z_m^{1'}, Z_m^{2'})$ where X_m^1 and $Z_m^{1'}$ denote the first $\frac{n}{2}$ samples of X_m and Z'_m , and X_m^2 and $Z_m^{2'}$ denote rest of their samples. Since X_m^1 and $Z_m^{1'}$ are drawn from the same distribution, $\Pi_{i=1}^{\frac{n}{2}} N(\mu, I)$, and $Z_m^{1'}, X_m^1, X_m^2$ are independent, we get $(Z_m^{1'}, X_m^2)$ and (X_m^1, X_m^2) are equal in distribution. This gives us

$$D_{TV}(X_m, Z'_m) = D_{TV}((X_m^1, X_m^2), (Z_m^{1'}, Z_m^{2'})) = D_{TV}((Z_m^{1'}, X_m^2), (Z_m^{1'}, Z_m^{2'})).$$

From Lemma 9, we know that, if with probability at least $1-\epsilon_1$ over $Z_m^{1'}$, $D_{TV}(X_m^2|Z_m^{1'}, Z_m^{2'}|Z_m^{1'}) \leq \epsilon_2$, then $D_{TV}((Z_m^{1'}, X_m^2), (Z_m^{1'}, Z_m^{2'})) \leq \epsilon_1 + \epsilon_2$. Here, $Z_m^{1'}$ and X_m^2 are independent, and the only dependency between $Z_m^{1'}$ and $Z_m^{2'}$ is via the mean $\tilde{\mu}$ of the elements of $Z_m^{1'}$. So $D_{TV}(X_m^2|Z_m^{1'}, Z_m^{2'}|Z_m^{1'}) = D_{TV}(X_m^2, Z_m^{2'}|\tilde{\mu})$. We will show that with high probability over $\tilde{\mu}$, this total variation distance is small.

We first estimate $\|\tilde{\mu} - \mu\|$. Note that $\mathbb{E}_{Z_m^{1'}}[\|\tilde{\mu} - \mu\|^2] = \frac{2d}{n}$, and $\frac{n}{2}\|\tilde{\mu} - \mu\|^2$ is a χ^2 random variable with d degrees of freedom. To bound the deviation of $\|\tilde{\mu} - \mu\|^2$ around it's mean, we will use the following concentration bound for a χ^2 random variable R with d degrees of freedom [131, Example 2.5].

$$\Pr[|R - d| \geq dt] \leq 2e^{-dt^2/8}, \text{ for all } t \in (0, 1).$$

This gives us $\Pr(|\frac{n}{2}\|\tilde{\mu} - \mu\|^2 - d| \geq 0.5d) \leq 2e^{-d/32}$, that is, $\|\tilde{\mu} - \mu\| \leq \sqrt{\frac{3d}{n}} \leq \sqrt{3\epsilon \log d}$ with probability at least $1 - 2e^{-d/32}$.

X_m^2 is distributed as the product of $\frac{n}{2} + r$ gaussiaus $\Pi_{i=1}^{\frac{n}{2}+r} N(\mu, I_{d\times d})$ and $Z_m^{2'}|\tilde{\mu}$ is distributed as the product of $\frac{n}{2} + r$ mixture distributions $\Pi_{i=1}^{\frac{n}{2}+r}(1 - \frac{10r}{\frac{n}{2}+r})N(\mu, I_{d\times d}) + \frac{10r}{\frac{n}{2}+r}N(\tilde{\mu}, I_{d\times d})$. We evaluate the total variation distance between these two distributions by bounding their squared Hellinger distance, since squared Hellinger distance is easy to bound for product distributions and is within a quadratic factor of the total variation distance for any distribution. By the subadditivity of the

squared Hellinger distance, we get

$$\begin{aligned} &H\left(\Pi_{i=1}^{\frac{n}{2}+r}N(\mu,I_{d\times d}),\Pi_{i=1}^{\frac{n}{2}+r}\left(1-\frac{10r}{\frac{n}{2}+r}\right)N\left(\mu,I_{d\times d}\right)+\frac{10r}{\frac{n}{2}+r}N\left(\tilde{\mu},I_{d\times d}\right)\right)^2 \\ &\leq\left(\frac{n}{2}+r\right)\ H\left(N\left(\mu,I_{d\times d}\right),\left(1-\frac{10r}{\frac{n}{2}+r}\right)N\left(\mu,I_{d\times d}\right)+\frac{10r}{\frac{n}{2}+r}N\left(\tilde{\mu},I_{d\times d}\right)\right)^2. \end{aligned} \tag{5.8}$$

For sufficiently large d, r and n satisfy $r\leq\frac{n}{18}$, so we can use Lemma 10 to get

$$\begin{aligned} H\left(N(\mu,I_{d\times d}),\left(1-\frac{10r}{\frac{n}{2}+r}\right)N\left(\mu,I_{d\times d}\right)+\frac{10r}{\frac{n}{2}+r}N\left(\tilde{\mu},I_{d\times d}\right)\right)^2 &\leq\frac{576r^2}{n^2}e^{3\|\tilde{\mu}-\mu\|^2} \\ &\leq\frac{576r^2d^{9\epsilon}}{n^2}, \end{aligned} \tag{5.9}$$

with probability at least $1-2e^{-d/32}$ over $\tilde{\mu}$. From (5.8) and (5.9), we get that with probability at least $1-2e^{-d/32}$ over $\tilde{\mu}$,

$$H\left(\Pi_{i=1}^{\frac{n}{2}+r}N(\mu,I_{d\times d}),\Pi_{i=1}^{\frac{n}{2}+r}\left(1-\frac{10r}{\frac{n}{2}+r}\right)N(\mu,I_{d\times d})+\frac{10r}{\frac{n}{2}+r}N(\tilde{\mu},I_{d\times d})\right)^2\leq(\frac{n}{2}+r)\frac{576r^2d^{9\epsilon}}{n^2}\leq\frac{576r^2d^{9\epsilon}}{n},$$

where the last inequality holds because $r<\frac{n}{2}$. As the total variation distance between two distributions is upper bounded by $\sqrt{2}$ times their Hellinger distance, we get that with probability at least $1-2e^{-d/32}$ over $\tilde{\mu}$,

$$\begin{aligned} &D_{TV}\left(\Pi_{i=1}^{\frac{n}{2}+r}N(\mu,I_{d\times d}),\Pi_{i=1}^{\frac{n}{2}+r}\left(1-\frac{10r}{\frac{n}{2}+r}\right)N(\mu,I_{d\times d})+\frac{10r}{\frac{n}{2}+r}N(\tilde{\mu},I_{d\times d})\right) \\ &\leq\frac{24\sqrt{2}rd^{9\epsilon/2}}{\sqrt{n}}\leq\frac{24\sqrt{2}rn^{9\epsilon}}{\sqrt{n}}, \end{aligned}$$

where the last inequality is true because $n>\sqrt{d}$.

Now, from Lemma 9, we know that if with probability at least $1-\epsilon_1$ over $Z_m^{1'}$, $D_{TV}(X_m^2|Z_m^{1'},Z_m^{2'}|Z_m^{1'})\leq\epsilon_2$, then $D_{TV}((Z_m^{1'},X_m^2),(Z_m^{1'},Z_m^{2'}))\leq\epsilon_1+\epsilon_2$. In this case, $\epsilon_1=2e^{-d/32}$ and $\epsilon_2=\frac{24\sqrt{2}rn^{9\epsilon}}{\sqrt{n}}$, so we get $D_{TV}((Z_m^{1'},X_m^2),(Z_m^{1'},Z_m^{2'}))=D_{TV}(X_m,Z_m')\leq 2e^{-d/32}+\frac{24\sqrt{2}rn^{9\epsilon}}{\sqrt{n}}$. We also know that $D_{TV}(Z_m,Z_m')\leq e^{-81r/26}$. Using triangle inequality, we get

$$D_{TV}(X_m,Z_m)\leq 2e^{-d/32}+\frac{24\sqrt{2}rn^{9\epsilon}}{\sqrt{n}}+e^{-81r/26}.$$

For $\delta>2(2e^{-d/32}+e^{-81r/26})$, and for $r\leq\frac{n^{\frac{1}{2}-9\epsilon}\delta}{48\sqrt{2}}$, we get $D_{TV}(X_m,Z_m)\leq\delta$. For d large enough, setting $\delta=\frac{1}{3}$ and $r\leq\frac{n^{\frac{1}{2}-9\epsilon}}{144\sqrt{2}}$, we get the desired result. Note that we haven't tried to optimize the constants in this proof.

Lemma 10. *Let $P = N(0, I_{d\times d})$ and $Q = N(\hat{\mu}, I_{d\times d})$ be d-dimensional gaussian distributions. For $r \leq \frac{n}{18}$, $H\left(P, \left(1 - \frac{10r}{r+\frac{n}{2}}\right)P + \frac{10r}{r+\frac{n}{2}}Q\right) \leq \frac{24r}{n}e^{\frac{3\|\hat{\mu}\|^2}{2}}$.*

Proof. We work in the rotated basis where $Q = N((\|\hat{\mu}\|, \underbrace{0, 0, \ldots, 0}_{d-1 \text{ times}}), I_{d\times d})$ and $P = N(0, I_{d\times d})$. Let $P_1 = N(0, 1)$ and $Q_1 = N(\|\hat{\mu}\|, 1)$ denote the projection of P and Q along the first coordinate axis respectively. Note that the mixture distribution in question is the product of $\left(\left(1 - \frac{10r}{r+\frac{n}{2}}\right)P_1 + \frac{10r}{r+\frac{n}{2}}Q_1\right)$ and $N(0, I_{d-1\times d-1})$, and P is the product of P_1 and $N(0, I_{d-1\times d-1})$. Since the squared Hellinger distance is subadditive for product distributions, we get, $H\left(P, \left(1 - \frac{10r}{r+\frac{n}{2}}\right)P + \frac{10r}{r+\frac{n}{2}}Q\right)^2$

$$\leq H\left(P_1, \left(1 - \frac{10r}{r+\frac{n}{2}}\right)P_1 + \frac{10r}{r+\frac{n}{2}}Q_1\right)^2 + H(N(0, I_{d-1\times d-1}), N(0, I_{d-1\times d-1}))^2$$
$$= H\left(P_1, \left(1 - \frac{10r}{r+\frac{n}{2}}\right)P_1 + \frac{10r}{r+\frac{n}{2}}Q_1\right)^2.$$

Therefore, to bound the required Hellinger distance, we just need to bound $H\left(P_1, \left(1 - \frac{10r}{r+\frac{n}{2}}\right)P_1 + \frac{10r}{r+\frac{n}{2}}Q_1\right)$. Let p_1 and q_1 denote the probability densities of P_1 and $\left(\left(1 - \frac{10r}{r+\frac{n}{2}}\right)P_1 + \frac{10r}{r+\frac{n}{2}}Q_1\right)$ respectively. We get $H\left(P_1, \left(1 - \frac{10r}{r+\frac{n}{2}}\right)P_1 + \frac{10r}{r+\frac{n}{2}}Q_1\right)^2 = \int_{-\infty}^{\infty}\left(\sqrt{p_1} - \sqrt{q_1}\right)^2 dx$

$$= \int_{-\infty}^{\infty}\left(\sqrt{\frac{1}{\sqrt{2\pi}}e^{-x^2/2}} - \sqrt{\left(1 - \frac{10r}{r+\frac{n}{2}}\right)\frac{1}{\sqrt{2\pi}}e^{-x^2/2} + \frac{10r}{r+\frac{n}{2}}\frac{1}{\sqrt{2\pi}}e^{-(x-\|\hat{\mu}\|)^2/2}}\right)^2 dx$$
$$= \int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}e^{-x^2/2}\left(1 - \sqrt{1 - \frac{10r}{r+\frac{n}{2}} + \frac{10r}{r+\frac{n}{2}}e^{\frac{-\|\hat{\mu}\|^2+2\|\hat{\mu}\|x}{2}}}\right)^2 dx.$$

We will evaluate this integral as a sum of integral in two regions.

1. From $-\infty$ to $\|\hat{\mu}\|/2$:

$$\int_{-\infty}^{\|\hat{\mu}\|/2}\frac{1}{\sqrt{2\pi}}e^{-x^2/2}\left(1 - \sqrt{1 - \frac{10r}{r+\frac{n}{2}} + \frac{10r}{r+\frac{n}{2}}e^{\frac{-\|\hat{\mu}\|^2+2\|\hat{\mu}\|x}{2}}}\right)^2 dx \leq$$
$$\int_{-\infty}^{\|\hat{\mu}\|/2}\frac{1}{\sqrt{2\pi}}e^{-x^2/2}\left(1 - \sqrt{1 - \frac{10r}{r+\frac{n}{2}}}\right)^2 dx.$$

   Since $r \leq \frac{n}{18}$, we get $\frac{10r}{r+\frac{n}{2}} \leq 1$. Using $1 - y \leq \sqrt{1-y}$ for $0 \leq y \leq 1$, we get

$$\int_{-\infty}^{\|\hat{\mu}\|/2}\frac{1}{\sqrt{2\pi}}e^{-x^2/2}\left(1 - \sqrt{1 - \frac{10r}{r+\frac{n}{2}}}\right)^2 dx \leq \int_{-\infty}^{\|\hat{\mu}\|/2}\frac{1}{\sqrt{2\pi}}e^{-x^2/2}\left(\frac{10r}{r+\frac{n}{2}}\right)^2 dx$$
$$\leq \frac{400r^2}{n^2}.$$

2. From $\frac{\|\hat{\mu}\|}{2}$ to ∞, we get $\int_{\|\hat{\mu}\|/2}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \left(1 - \sqrt{1 - \frac{10r}{r+\frac{n}{2}} + \frac{10r}{r+\frac{n}{2}} e^{\frac{-\|\hat{\mu}\|^2+2\|\hat{\mu}\|x}{2}}}\right)^2 dx.$

$$\leq \int_{\|\hat{\mu}\|/2}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \left(\sqrt{1 + \frac{10r}{r+\frac{n}{2}} e^{\frac{-\|\hat{\mu}\|^2+2\|\hat{\mu}\|x}{2}}} - 1\right)^2 dx.$$

This is because $x \geq \|\hat{\mu}\|/2$, and therefore $\frac{10r}{r+\frac{n}{2}} e^{\frac{-\|\hat{\mu}\|^2+2\|\hat{\mu}\|x}{2}} \geq \frac{10r}{r+\frac{n}{2}}$. Now, using $\sqrt{1+y} \leq 1 + \frac{y}{2}$, we get

$$\begin{aligned}
&\int_{\|\hat{\mu}\|/2}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \left(\sqrt{1 + \frac{10r}{r+\frac{n}{2}} e^{\frac{-\|\hat{\mu}\|^2+2\|\hat{\mu}\|x}{2}}} - 1\right)^2 dx \\
&\leq \int_{\|\hat{\mu}\|/2}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \left(1 + \frac{5r}{r+\frac{n}{2}} e^{\frac{-\|\hat{\mu}\|^2+2\|\hat{\mu}\|x}{2}} - 1\right)^2 dx \\
&\leq \frac{100r^2}{n^2} \int_{\|\hat{\mu}\|/2}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\|\hat{\mu}\|^2+2\|\hat{\mu}\|x} e^{-x^2/2} dx \\
&= \frac{100r^2}{n^2} e^{-\|\hat{\mu}\|^2} \int_{\|\hat{\mu}\|/2}^{\infty} \frac{1}{\sqrt{2\pi}} e^{2\|\hat{\mu}\|x - x^2/4} e^{-x^2/4} dx.
\end{aligned}$$

Since $2\|\hat{\mu}\|x - x^2/4 \leq 4\|\hat{\mu}\|^2$, we get

$$\begin{aligned}
\frac{100r^2}{n^2} e^{-\|\hat{\mu}\|^2} \left(\int_{\|\hat{\mu}\|/2}^{\infty} e^{2\|\hat{\mu}\|x - x^2/4} \frac{1}{\sqrt{2\pi}} e^{-x^2/4}\right) dx &\leq \frac{100r^2}{n^2} e^{3\|\hat{\mu}\|^2} \left(\int_{\|\hat{\mu}\|/2}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/4}\right) dx \\
&\leq \frac{100\sqrt{2}r^2}{n^2} e^{3\|\hat{\mu}\|^2} \left(\int_{-\infty}^{\infty} \frac{1}{\sqrt{4\pi}} e^{-x^2/4}\right) dx \\
&\leq \frac{100\sqrt{2}r^2}{n^2} e^{3\|\hat{\mu}\|^2}.
\end{aligned}$$

Adding the two integrals, we get

$$\begin{aligned}
H\left(P_1, \left(1 - \frac{10r}{r+\frac{n}{2}}\right) P_1 + \frac{10r}{r+\frac{n}{2}} Q_1\right)^2 &\leq \frac{400r^2}{n^2} + \frac{100\sqrt{2}r^2}{n^2} e^{3\|\hat{\mu}\|^2} \\
&\leq \frac{576r^2}{n^2} e^{3\|\hat{\mu}\|^2}.
\end{aligned}$$

This gives us $H(P, \left(1 - \frac{10r}{r+\frac{n}{2}}\right) P + \frac{10r}{r+\frac{n}{2}} Q) \leq \frac{24r}{n} e^{3\|\hat{\mu}\|^2/2}$ which completes the proof. □

□

5.3.4 Lower Bound for Procedures which Return a Superset of the Input Samples

In this section we prove the lower bound from Proposition 1.

Proposition 5. *Let $\mathcal{C}$ denote the class of $d-$dimensional Gaussian distributions $N(\mu, I)$ with unknown mean μ. There is an absolute constant, c, such that for sufficiently large d, if $n \leq \frac{cd}{\log d}$, there is no $(n, n+1)$ amplification procedure that always returns a superset of the original n points.*

Proof. The outline of the proof is very similar to the proof of Proposition 3. As in the proof of Proposition 3, we define a verifier $v(Z_{n+1})$ for the distribution $N(\mu, I)$ which takes as input $(n+1)$ samples $\{x_i' \in \mathbb{R}^d, i \in [n+1]\}$, and a distribution D_μ over μ, such that if $n < O(d/\log(d))$; (i) for all μ, the verifier will accept with probability $1 - 1/e^2$ when given as input a set Z_{n+1} of $(n+1)$ i.i.d. samples from $N(\mu, I)$, (ii) but will reject any $(n, n+1)$ amplification procedure which does not modify the input samples with probability $1-1/e^2$, where the probability is with respect to the randomness in $\mu \leftarrow D_\mu$, the set X_n and in any internal randomness of the amplifier. Note that by Definition 31 of an amplification procedure, this implies that there is no $(n, n+1)$ amplification procedure which does not modify the input samples for $n < O(d/\log(d))$. We choose D_μ to be $N(0, \sqrt{d}I)$. Let $\hat{\mu}_{-i}$ be the mean of the all except the i-th sample returned by the amplification procedure. The verifier performs the following tests, and accepts if all tests pass, and rejects otherwise—

1. $\forall\ i \in [n+1], \|x_i' - \mu\|^2 \leq 15d$.
2. $\forall\ i \in [n+1], \langle x_i' - \hat{\mu}_{-i}, \mu - \hat{\mu}_{-i} \rangle \geq d/(4n)$.

We first show that for a sufficiently large constant C and $n < O(d/\log(d))$, $(n+1)$ i.i.d. samples from $N(\mu, I)$ pass the above tests with probability at least $1 - 1/e^2$. As $\|x_i' - \mu\|^2$ is a χ^2 random variable with d degrees of freedom, by the concentration bound for a χ^2 random variable (5.6), a true sample x_i' passes the first test with failure probability e^{-3d}. Hence by a union bound, all samples $\{x_i, i \in [n+1]\}$ pass the first test with probability at least $1 - de^{-3d} \geq 1 - 1/e^3$. Let E denote the following event,

$$\forall\ i \in [n+1], \|\hat{\mu}_n - \mu\|^2 \geq d/n - \sqrt{20d\log d}/n \geq d/(2n),$$
$$\forall\ i \in [n+1], \|\hat{\mu}_n - \mu\|^2 \leq d/n + \sqrt{20d\log d}/n \leq 2d/n.$$

Note that $\hat{\mu}_{-i} \leftarrow N(\mu, \frac{I}{n})$. Hence, by using (5.7) with $t = 20\sqrt{\frac{\log d}{d}}$, and a union bound over all $i \in [n+1]$,

$$\Pr[E] \geq 1 - 1/e^3.$$

Note that as $x_i' \leftarrow N(\mu, I)$, for a fixed $\hat{\mu}_{-i}$, $\langle x_i' - \hat{\mu}_{-i}, \mu - \hat{\mu}_{-i} \rangle \leftarrow N(\|\hat{\mu}_{-i} - \mu\|^2, \|\hat{\mu}_{-i} - \mu\|^2)$. Hence

conditioned on E, by standard Gaussian tail bounds,

$$\Pr\Big[\langle x_i' - \hat{\mu}_{-i}, \mu - \hat{\mu}_{-i}\rangle \leq d/(2n) - \sqrt{20d\log d/n}\Big] \leq 1/n^2,$$
$$\implies \Pr\Big[\langle x_i' - \hat{\mu}_{-i}, \mu - \hat{\mu}_{-i}\rangle \leq d/(4n)\Big] \leq 1/n^2,$$

where in the last step we use the fact that $n < \frac{d}{C\log d}$ for a large constant C. Therefore, conditioned on E, $\{x_i, i \in [n+1]\}$ pass the third test with probability at least $1 - 1/e^3$. Hence by a union bound, $(n+1)$ samples drawn from $N(\mu, I)$ will satisfy all 3 tests with failure probability at most $1/e^2$. Hence for any μ, the verifier accepts $n+1$ i.i.d. samples from $N(\mu, I)$ with probability at least $1 - 1/e^2$.

We now show that for $n < \frac{d}{C\log d}$ and μ sampled from $D_\mu = N(0, \sqrt{d}I)$, the verifier rejects any $(n, n+1)$ amplification procedure which does not modify the input samples with high probability over the randomness in μ and the set X_n. Let $D_{\mu|X_n}$ be the posterior distribution of μ conditioned on the set X_n. As in Proposition 3, $D_{\mu|X_n} = N(\bar{\mu}, \bar{\sigma}^2 I)$, where,

$$\bar{\mu} = \frac{n}{n + 1/\sqrt{d}}\mu_0, \quad \bar{\sigma}^2 = \frac{1}{n + 1/\sqrt{d}}.$$

We will show that with probability $1 - e^{-3d}$ over the randomness in the set X_n received by the amplifier and with probability $1 - 1/e^2$ over $\mu \leftarrow D_{\mu|X_n}$ and any internal randomness of the amplifier, the amplifier cannot output a set Z_{n+1} which contains the set X_n as a subset and which is accepted by the verifier. To show this, we first claim that $\|\mu_0\| \leq 30d^{3/4}$ with probability $1 - e^d$. Note that $\mu_0 \leftarrow N(\mu, \frac{I}{n})$, where $\mu \leftarrow N(0, \sqrt{d}I)$. By (5.6), with probability at least $1 - e^{-3d}$, $\|\mu\| \leq 15d^{3/4}$ and $\|\mu - \mu_0\| \leq 15\sqrt{d}$. Hence by the triangle inequality, $\|\mu_0\| \leq 30d^{3/4}$ with probability at least $1 - e^{-3d}$. We now show that for sets X_n such that $\|\mu_0\| \leq 30d^{3/4}$, Z_{n+1} cannot pass the verifier with probability more than $1 - e^2$ over the randomness in $\mu|X_n$. The proof consists of two cases, and the analysis of the cases is similar to the proof of Proposition 3. Without loss of generality, assume that $Z_{n+1} = \{x_1', X_n\}$, hence x_1' is the only sample not present in the set. We will show that either x_1' or $\hat{\mu}_{-1}$ fail one of the three tests performed by the verifier with high probability.

Case 1: $\|x_1' - \bar{\mu}\|^2 \geq 100d$**.**

We show that the first test is not satisfied with high probability in this case. As $\mu|X_n \leftarrow N(\bar{\mu}, \bar{\sigma}^2)$, hence by (5.6), $\|\mu - \bar{\mu}\|^2 \leq 15d/n$ with probability $1 - e^{-3d}$. Therefore, if $\|x_1' - \bar{\mu}\|^2 \geq 100d$, then with probability e^{-3d},

$$\|x_1' - \mu\|^2 \geq (\sqrt{100d} - \sqrt{15d/n})^2 > 15d,$$

in which case the first test is not satisfied. Hence in the first case, the amplifier succeeds with probability at most e^{-3d}.

Case 2: $\|x_1' - \bar{\mu}\|^2 < 100d$**.**

Note that for the sample x_1', $\mu_{-1} = \mu_0$ as the last n samples are the same as the original set X_n. We now bound $\|\hat{\mu}_{-1} - \bar{\mu}\|$ as follows,

$$\|\hat{\mu}_{-1} - \bar{\mu}\| = \Big\|\mu_0 - \frac{n}{n + 1/\sqrt{d}}\mu_0\Big\| \leq \frac{\|\mu_0\|}{n\sqrt{d}} \leq \frac{30d^{1/4}}{n}.$$

We now expand $\langle x_1' - \hat{\mu}_{-1}, \mu - \hat{\mu}_{-1}\rangle$ in the third test as follows,

$$\begin{aligned}\langle x_1' - \hat{\mu}_{-1}, \mu - \hat{\mu}_{-1}\rangle &= \langle x_1' - \bar{\mu}, \mu - \bar{\mu}\rangle - \langle \hat{\mu}_{-1} - \bar{\mu}, \mu - \bar{\mu}\rangle - \langle x_1' - \bar{\mu}, \hat{\mu}_{-1} - \bar{\mu}\rangle + \|\hat{\mu}_{-1} - \bar{\mu}\|^2, \\ &\leq \langle x_1' - \bar{\mu}, \mu - \bar{\mu}\rangle - \langle \hat{\mu}_{-1} - \bar{\mu}, \mu - \bar{\mu}\rangle + \|x_1' - \bar{\mu}\|\|\hat{\mu}_{-1} - \bar{\mu}\| + \|\hat{\mu}_{-1} - \bar{\mu}\|^2.\end{aligned}$$

Note that $\langle \hat{\mu}_{-1} - \bar{\mu}, \mu - \bar{\mu}\rangle$ is distributed as $N(0, \bar{\sigma}^2\|\hat{\mu}_{-1} - \bar{\mu}\|^2)$ and hence with probability $1 - 1/e^3$ it is at most $10\|\hat{\mu}_{-1} - \bar{\mu}\|/\sqrt{n}$. Similarly, with probability $1 - 1/e^3$, $\langle x_1' - \bar{\mu}, \mu - \bar{\mu}\rangle$ is at most $10\|x_1' - \bar{\mu}\|/\sqrt{n}$. Therefore, with probability $1 - 2/e^3$,

$$\begin{aligned}\langle x_1' - \hat{\mu}_{-1}, \mu - \hat{\mu}_{-1}\rangle &\leq 10\|x_1' - \bar{\mu}\|/\sqrt{n} + 10\|\hat{\mu}_{-1} - \bar{\mu}\|/\sqrt{n} + \|x_1' - \bar{\mu}\|\|\hat{\mu}_{-1} - \bar{\mu}\| + \|\hat{\mu}_{-1} - \bar{\mu}\|^2, \\ &\leq 100\sqrt{\frac{d}{n}} + 300\frac{d^{3/4}}{n^2} + 300\frac{d^{3/4}}{n} + 900\frac{\sqrt{d}}{n^2} \\ &\leq 100\sqrt{\frac{d}{n}} + 1500\frac{d^{3/4}}{n} \\ &= 100\sqrt{\frac{n}{d}}\Big(\frac{d}{n}\Big) + \frac{1500}{d^{1/4}}\Big(\frac{d}{n}\Big).\end{aligned}$$

Hence for a sufficiently large constant C, $n < \frac{d}{C\log d}$ and d sufficiently large, with probability $1-2/e^3$,

$$\langle x_1' - \hat{\mu}_{-1}, \mu - \hat{\mu}_{-1}\rangle \leq \frac{d}{5n},$$

which implies that the second test is not satisfied. Hence the amplifier succeeds in this case with probability at most $2/e^3$.

The overall success probability of the amplifier is the maximum success probability across the two cases, hence for sets X_n such that the $\|\mu_0\| \leq 30d^{3/4}$, the verifier accepts the amplified set Z_{n+1} with probability at most $2/e^3$. As $\Pr\Big[\|\mu_0\| \leq 30d^{3/4}\Big] \geq 1 - e^{-3d}$, the overall success probability of the amplifier over the randomness in μ, X_n and any internal randomness of the amplifier is at most $1/e^2$. □

5.4 Proofs: Discrete Distributions with Bounded Support

5.4.1 Upper Bound

In this section we prove the upper bound from Theorem 16. The algorithm itself is presented in Algorithm 9. For clarity of writing, we assume that the number of input samples is $4n$, instead of n.

Algorithm 9 Sample Amplification for Discrete Distributions

Input: $X_{4n} = (x_1, x_2, \ldots, x_{4n})$, where $x_i \leftarrow D$, for any discrete distribution D over $[k]$.
Output: $Z_m = (x'_1, x'_2, \ldots, x'_m)$, such that $D_{TV}(D^m, Z_m) \leq \frac{1}{3}$.

$N_1, N_2 \leftarrow \text{Poisson}(n)$ {Draw two i.i.d samples N_1 and N_2 from Poisson(n)}
$N := N_1 + N_2$
if $N \leq 4n$ **then**
  $X_{N_1} := (x_1, x_2, \ldots, x_{N_1})$
  $X_{N_2} := (x_{N_1+1}, x_{N_1+2}, \ldots, x_{N_1+N_2})$
else
  {Uninteresting case: happens with low probability}
  $X_{N_1} := \underbrace{(x_1, x_1, \ldots, x_1)}_{N_1 \text{ times}}$
  $X_{N_2} := \underbrace{(x_1, x_1, \ldots, x_1)}_{N_2 \text{ times}}$
end if
$r := 8(m - n)$
$(x'_1, x'_2, \ldots, x'_{N+R}) = \text{AmplifyDiscretePoissonized}(X_{N_1}, X_{N_2}, r, n)$
  ▷ Amplify first $N_1 + N_2$ samples to $N_1 + N_2 + R$ samples, for R roughly distributed as Poisson(r)
$R_1 := max(R, r/8)$
if $R < r/8$ **then**
  ▷ Uninteresting case: happens with low probability
  $(x'_1, x'_2, \ldots, x'_{N+R_1}) := (x'_1, x'_2, \ldots, x'_{N+R}, \underbrace{x_1, x_1, \ldots, x_1}_{\frac{r}{8} - R \text{ times}})$
end if
$(x'_{N+R_1+1}, x'_{N+R_1+2}, \ldots, x'_m) := (x_{N+1}, x_{N+2}, \ldots, x_{4n-(R_1-\frac{r}{8})})$
  ▷ Add the remaining samples to get $4n + r/8$ samples in total
$Z_m := (x'_1, x'_2, \ldots, x'_m)$
return Z_M

Proposition 6. *Let $\mathcal{C}$ denote the class of discrete distributions with support size at most k. For sufficiently large k, and $m = 4n + O\left(\frac{n}{\sqrt{k}}\right)$, $\mathcal{C}$ admits an $(4n, m)$ amplification procedure.*

Proof. To avoid dependencies between the count of different elements, we first prove our results in a Poissonized setting, and then in lemma 12, we describe how to use the amplifier for Poissonized setting to get an amplifier for the original multinomial setting. Let $D \in \mathcal{C}$ be an unknown probability distribution over $[k]$, and let p_i denote the probability mass associated with $i \in [k]$. Throughout

Algorithm 10 AmplifyDiscretePoissonized

Input: X_{N_1}, X_{N_2}, r, n.

Output: Generates approximately Poisson(r) more samples given $N_1 + N_2$ input samples.

$X_{N_1} = (x_1, x_2, \ldots, x_{N_1})$, $X_{N_2} = (x_{N_1+1}, x_{N_1+2}, \ldots, x_{N_1+N_2})$, and $r = 8(m-n)$

for all $j \in [k]$ **do**

count$_j := \sum_{i=1}^{N_1} \mathbb{1}(x_i = j)$, for $j \in [k]$ ▷ Find the count of each element in first N_1 samples

$\hat{p}_j := \frac{count_j}{n}$, for $j \in [k]$

$\hat{z}_j \leftarrow$ Poisson$(\hat{p}_j r)$, for $j \in [k]$

end for

$R := \sum_{j=1}^{k} \hat{z}_j$

$(x'_1, x'_2, \ldots, x'_{N_1}) := (x_1, x_2, \ldots, x_{N_1})$

$(x'_{N_1+1}, \ldots, x'_{N_1+N_2+R}) :=$ RandomPermute$((x_{N_1+1}, x_{N_1+2}, \ldots, x_{N_1+N_2}, \underbrace{1, 1, \ldots, 1}_{\hat{z}_1 \text{ times}}, \ldots, \underbrace{k, k, \ldots, k}_{\hat{z}_k \text{ times}}))$

return $(x'_1, x'_2, \ldots, x'_{N_1+N_2+R})$

the proof, we use random variable X_q to denote q independent samples from D, where q can also be a random variable. Suppose we are given $N = N_1 + N_2$ independent samples from D, denoted by X_{N_1} and X_{N_2}, where N_1 and N_2 are drawn from Poisson(n). We show how to amplify them to $\tilde{M} = N + R$ samples, denoted by $Z_{\tilde{M}}$, such that $D_{TV}(Z_{\tilde{M}}, X_M)$ is small, where $M \leftarrow$ Poisson($2n+r$). Our amplifying procedure involves estimating the probability of each element using X_{N_1}, generating R independent samples using these estimates, and randomly shuffling these samples with X_{N_2}. Let u_i be the count of element i in X_{N_1} and y_i be the count of i in X_{N_2} noting they are both distributed as Poisson(np_i). The amplification procedure proceeds through the following steps:

1. Estimate the frequency $\hat{p}_i$ of each element using u_i, that is, $\hat{p}_i = \frac{u_i}{n}$.
2. Draw $\hat{z}_i \leftarrow$ Poisson$(r\hat{p}_i)$ additional samples of element i for all $i \in [k]$.
3. Append these generated samples to X_{N_2} to get Z_{N_2+R}.
4. Randomly permute the elements of Z_{N_2+R}, and append them to X_{N_1} to get $Z_{\tilde{M}}$.

We first show that $Z_{\tilde{M}}$ is close in total variation distance, to Poisson($2n+r$) samples generated from D. We will prove this by showing that with high probability over the choice of X_{N_1}, the distribution of Z_{N_2+R} is close to Poisson($n+r$) samples generated from D. After this, we can use lemma 9 to show that appending Z_{N_2+R} to the samples in X_{N_1} results in a sequence with low total variation distance to X_M. Since our amplification procedure randomly permutes the last $N_2 + R$ elements, we can argue this using only the count of each element. Recall y_i is the count of element i in X_{N_2}, and $\hat{z}_i$ is the number of additional samples of element i added by our amplification procedure. Let $z_i \leftarrow$ Poisson(rp_i), and let $v_i = y_i + z_i$ and $\hat{v}_i = y_i + \hat{z}_i$. Here, v_i denotes the count of element i in Poisson($n+r$) samples drawn from D, and $\hat{v}_i$ denotes the corresponding count in samples generated

using our amplification procedure. We use P_v to denote the distribution associated with random variable v.

Lemma 11. *For $r \leq n\epsilon^{1.5}/(4\sqrt{k})$, with probability $1-\epsilon$ over the randomness in $\{u_i, i \in [k]\}$,*

$$d_{TV}\left(\prod_{i=1}^{k} v_i, \prod_{i=1}^{k} \hat{v}_i\right) \leq \epsilon/2.$$

where $\prod$ refers to the product distribution.

Proof. We partition the support $[k]$ into two sets. Let $S = \{i : p_i \geq \epsilon/(2nk)\}$ and $S^c = [k]\backslash S$. Let $|S| = k'$. Without loss of generality, assume that $S = \{i : 1 \leq i \leq k'\}$ and $S^c = \{i : k'+1 \leq i \leq k\}$. We will separately bound the contribution of the variables in the set S and S^c to the total variation distance. For the first set S, we will upper bound $\sum_{i=1}^{k'} D_{KL}(v_i \parallel \hat{v}_i)$, and use Pinsker's inequality to then bound the total variation distance. For the second set S^c, we will directly bound $\sum_{i=k'+1}^{k} d_{TV}(v_i, \hat{v}_i)$. All our bounds will be with high probability over the randomness in the first set $\{u_i, i \in [k]\}$.

We first bound the total variation distance for the variables in the first set S. Note that because the sum of two Poisson random variables is a Poisson random variable, v_i is distributed as Poisson$(np_i + rp_i)$ and $\hat{v}_i$ is distributed as Poisson$(np_i + ru_i/n)$. We will use the following expression for the KL divergence $D_{KL}(P \parallel Q)$ between two Poisson distributions P and Q with means λ_1 and λ_2 respectively—

$$D_{KL}(P \parallel Q) = \lambda_1 \log\left(\frac{\lambda_1}{\lambda_2}\right) + \lambda_2 - \lambda_1. \tag{5.10}$$

Using this expression, we can write the KL divergence between the distributions of v_i and $\hat{v}_i$ as follows,

$$D_{KL}(v_i \parallel \hat{v}_i) = p_i(n+r)\log\left(\frac{p_i(n+r)}{p_i n + ru_i/n}\right) + (ru_i/n - rp_i).$$

Let $\delta_i = u_i - np_i$. We can rewrite the above expression as follows,

$$\begin{aligned} D_{KL}(v_i \parallel \hat{v}_i) &= p_i(n+r)\log\left(\frac{p_i(n+r)}{p_i(n+r) + r\delta_i/n}\right) + r\delta_i/n, \\ &= p_i(n+r)\log\left(\frac{1}{1 + r\delta_i/(np_i(n+r))}\right) + r\delta_i/n. \end{aligned}$$

Note that $\log(1+x) \geq x - 2x^2$ for $x \geq 0.8$. As $\delta_i \geq -np_i$, therefore $r\delta_i/(np_i(n+r)) \geq -0.8$ for

$r \leq n$. Therefore,

$$\begin{aligned} p_i(n+r)\log\left(\frac{1}{1+r\delta_i/(np_i(n+r))}\right) &\leq -r\delta_i/n + \frac{2r^2\delta_i^2}{n^2p_i(n+r)}, \\ \implies D_{KL}(v_i \parallel \hat{v}_i) &\leq \frac{2r^2\delta_i^2}{n^2p_i(n+r)}, \\ \implies \sum_{i=1}^{k'} D_{KL}(v_i \parallel \hat{v}_i) &\leq \frac{2r^2}{n^2}\sum_{i=1}^{k'}\frac{\delta_i^2}{np_i}. \end{aligned} \tag{5.11}$$

We will now bound $\sum_{i=1}^{k'} \frac{\delta_i^2}{np_i}$. As a Poisson($\lambda$) random variable has variance λ and $\delta_i = u_i - np_i$ where $u_i \leftarrow \text{Poisson}(np_i)$, therefore,

$$\mathbb{E}\left[\sum_{i=1}^{k'}\frac{\delta_i^2}{np_i}\right] = k'.$$

Also, the fourth central moment of a Poisson(λ) random variable is $\lambda(1+3\lambda)$, hence

$$\begin{aligned} \text{Var}[\delta_i^2] &= \mathbb{E}\left[\delta_i^4\right] - \mathbb{E}\left[\delta_i^2\right]^2, \\ &= np_i(1+3np_i) - (np_i)^2 = np_i(1+2np_i), \\ \implies \text{Var}\left[\sum_{i=1}^{k'}\frac{\delta_i^2}{np_i}\right] &= \sum_{i=1}^{k'}\frac{1+2np_i}{np_i}. \end{aligned}$$

As $p_i \geq \epsilon/(2nk)$ for $i \in S$ and $k' \leq k$, therefore,

$$\text{Var}\left[\sum_{i=1}^{k'}\frac{\delta_i^2}{np_i}\right] \leq 2k^2/\epsilon + 2k \leq 4k^2/\epsilon.$$

Hence by Chebyshev's inequality,

$$\begin{aligned} \Pr\left[\sum_{i=1}^{k'}\frac{\delta_i^2}{np_i} - k' \geq 4k/\epsilon\right] &\leq \epsilon/4, \\ \implies \Pr\left[\sum_{i=1}^{k'}\frac{\delta_i^2}{np_i} \geq 4k/\epsilon\right] &\leq \epsilon/4. \end{aligned} \tag{5.12}$$

Let E_1 be the event that $\sum_{i=1}^{k'} \frac{\delta_i^2}{np_i} \leq 4k/\epsilon$. By (5.12), $\Pr(E_1) \geq 1 - \epsilon/4$. Conditioned on the event

E_1 and using (5.11), we can bound the KL divergence as follows,

$$D_{KL}\Big(\prod_{i\in S} v_i \,\Big\|\, \prod_{i\in S} \hat{v}_i\Big) = \sum_{i=1}^{k'} D_{KL}(v_i \parallel \hat{v}_i) \leq \frac{8r^2k}{n^2\epsilon}.$$

Hence for $r \leq n\epsilon^{1.5}/(4\sqrt{k})$ and conditioned on the event E_1,

$$D_{KL}\Big(\prod_{i\in S} v_i \,\Big\|\, \prod_{i\in S} \hat{v}_i\Big) \leq \epsilon^2/2.$$

Hence using Pinsker's inequality, conditioned on the event E_1,

$$d_{TV}\left(\prod_{i\in S} v_i, \prod_{i\in S} \hat{v}_i\right) \leq \epsilon/2.$$

We will now bound the total variation distance for the variables in the set S^c. Let E_2 be the event that $u_i = 0$, $\forall\ i \in S^c$. Note that as $u_i \sim \text{Poisson}(np_i)$ where $p_i < \epsilon/(2nk)$, $u_i = 0$ with probability at least $e^{-\epsilon/(2k)}$, hence $\Pr(E_2) \geq e^{-\epsilon/2} \geq 1 - \epsilon/2$. We now condition on the event E_2. Recall that $v_i = y_i + z_i$, where $z_i \sim \text{Poisson}(rp_i)$ and $\hat{v}_i = y_i + \hat{z}_i$, where $\hat{z}_i = 0$ conditioned on E_2. By a coupling argument on y_i, the total variation distance between the distributions of v_i and $\hat{v}_i$ equals the total variation distance between the distributions of z_i and $\hat{z}_i$. As $\hat{z}_i = 0$, conditioned on the event E_2,

$$\begin{aligned} d_{TV}(v_i, \hat{v}_i) = \Pr[z_i \neq 0] = 1 - e^{-rp_i} &\leq 1 - e^{-r\epsilon/(2nk)} \\ \leq \frac{r\epsilon}{2nk} \leq \frac{\epsilon}{2k}, &\quad \text{as } r \leq n. \end{aligned}$$

Hence conditioned on E_2,

$$d_{TV}\left(\prod_{i\in S^c} v_i, \prod_{i\in S^c} \hat{v}_i\right) \leq \sum_{i=k'+1}^{k} d_{TV}\left(v_i, \hat{v}_i\right) \leq \epsilon/2.$$

Hence conditioned on the events E_1 and E_2,

$$d_{TV}\left(\prod_{i=1}^{k} v_i, \prod_{i=1}^{k} \hat{v}_i\right) \leq d_{TV}\left(\prod_{i\in S} v_i, \prod_{i\in S} \hat{v}_i\right) + d_{TV}\left(\prod_{i\in S^c} v_i, \prod_{i\in S^c} \hat{v}_i\right) \leq \epsilon.$$

As $\Pr(E_1) \geq 1 - \epsilon/4$ and $\Pr(E_2) \geq 1 - \epsilon/2$, by a union bound $\Pr(E_1 \cup E_2) \geq 1 - \epsilon$. Hence with probability $1 - \epsilon$ over the randomness in $\{u_i, i \in [k]\}$,

$$d_{TV}\left(\prod_{i=1}^{k} v_i, \prod_{i=1}^{k} \hat{v}_i\right) \leq \epsilon.$$

□

Lemma 11 says that with high probability over the first N_1 samples, the $N_2 + R$ samples are close in total variation distance to Poisson$(n + r)$ samples drawn from D. Using lemma 11 and lemma 9, we can conclude that for $r \leq n\epsilon^{1.5}/(4\sqrt{k})$, $D_{TV}(X_M, Z_{\tilde{M}}) \leq \epsilon + \epsilon/2 = 3\epsilon/2$.
Next, we show how to use the above amplification procedure to amplify samples in the non-Poissonized setting. Given $N = N_1 + N_2$ samples from D, we have shown how to amplify them to get $\tilde{M} = N + R$ samples. Given such an amplifier as a black box, and $4n$ samples from D, one can use the first N samples to generate M samples. Then append these M samples with the remaining $4n - N$ samples to get an amplifier in our original non-Poissonized setting.

Lemma 12. *Let $N = N_1 + N_2$ where $N_1, N_2 \leftarrow$ Poisson(n), and let $M \leftarrow$ Poisson$(2n+r)$. Suppose we are given an (N, M) amplifier f (as described above) satisfying $D_{TV}(f(X_N), X_M) \leq \frac{3\epsilon}{2}$, for all $D \in \mathcal{C}$. Then there exists an amplifier $f' : [k]^{4n} \to [k]^{4n+\frac{r}{8}}$, such that $D_{TV}(f'(X_{4n}), X_{4n+\frac{r}{8}}) \leq \frac{5\epsilon}{2}$, for $\epsilon \geq 2e^{-\frac{n}{20}} + e^{-\frac{25r}{88}}$, and for $r \leq n\epsilon^{1.5}/(4\sqrt{k})$.*

Proof. We divide the proof into three steps:

- **Step 1:** f takes as input X_{N_1} and X_{N_2}, samples of size N_1 and N_2 drawn from D. To simulate these samples, we use the 4n samples available to us from D. We draw $N_1', N_2' \leftarrow$ Poisson(n), and let $N' = N_1' + N_2'$. If $N' \leq 4n$, we set $X_{N_1'} = (x_1, x_2, \ldots, x_{N_1'})$ and $X_{N_2'} = (x_{N_1'+1}, x_{N_1'+2}, \ldots, x_{N_2'})$. Otherwise, we set $X_{N_1'} = (\underbrace{x_1, x_1, \ldots, x_1}_{N_1' \text{ times}})$, and $X_{N_2'} = (\underbrace{x_1, x_1, \ldots, x_1}_{N_2' \text{ times}})$, but this happens with very small probability leading to small total variation distance between $f(X_{N_1}, X_{N_2})$ and $f(X_{N_1'}, X_{N_2'})$, and by triangle inequality, small TV distance between $f(X_{N_1'}, X_{N_2'})$ and X_M. We denote (X_{N_1}, X_{N_2}) by X_N and $(X_{N_1'}, X_{N_2'})$ by $X_{N'}$.

- **Step 2:** We would like to finally output $\frac{r}{8}$ more samples. Let us denote the number of samples in $f(X_{N'})$ by M'. If $M' < N' + \frac{r}{8}$, we append $N' + \frac{r}{8} - M'$ arbitrary samples to it (say x_1) so that the total sample size is equal to $N' + \frac{r}{8}$. If $M' \geq N' + \frac{r}{8}$, we don't do anything in this step. Let $t_1(f(X_{N'}))$ denote the samples outputted in this step. Since the number of new samples added by f is roughly distributed as Poisson(r), the probability that the number of new samples is less than $r/8$ is small, leading to small TV distance between $t_1(f(X_{N'}))$ and $f(X_{N'})$, and by triangle inequality, small TV distance between $t_1(f(X_{N'}))$ and X_M.

- **Step 3:** Let M_1' denote the number of samples in $t_1(f(X_{N'}))$, and let $Q_1' = 4n + \frac{r}{8} - M_1'$ denote the number of extra samples needed to output $4n + \frac{r}{8}$ samples in total. If $Q_1' \geq 0$, we append Q_1' i.i.d. samples from D to $t_1(f(X_{N'}))$, and if $Q_1' < 0$, we remove last $|Q_1'|$ samples from $t_1(f(X_{N'}))$. We use $t_2(t_1(f(X_{N'})))$ to denote the output of this step. Step 2 ensures

$M_1' \geq N' + \frac{r}{8}$, which implies $Q_1' \leq 4n - N'$. Let $X_{4n-N'} = (x_{N'+1}, x_{N'+2}, \ldots, x_{4n})$ denote the leftover samples in X_{4n} after removing the first N' samples. When $Q_1' \geq 0$, we use the first Q_1' samples from $X_{4n-N'}$ to simulate i.i.d. samples from D, that is, $t_2(t_1(f(X_{N'}))) = \text{append}(t_1(f(X_{N'})), (x_{N'+1}, x_{N'+2}, \ldots, x_{N'+Q_1'}))$. $t_2(t_1(f(X_{N'})))$ is the final output of our amplifier f'.

Similarly, let $Q_1 = 4n + \frac{r}{8} - M$ denote the number of extra samples needed to be appended to X_M to output $4n + \frac{r}{8}$ samples in total. If $Q_1 \geq 0$, $t_2(X_M)$ correspond to appending Q_1 samples from D to X_M, and otherwise, it corresponds to removing last $|Q_1|$ samples from X_M. Since applying the same transformation to two random variables can't increase their total variation distance, and from step 2, we know that $D_{TV}(t_1(f(X_{N'})), X_M)$ is small, we get $D_{TV}(t_2(t_1(f(X_{N'}))), t_2(X_M))$ is small.

As $t_2(X_M)$ corresponds to $4n + \frac{r}{8}$ i.i.d. samples from D, $D_{TV}(X_{4n+\frac{r}{8}}, t_2(X_M)) = 0$. Using triangle inequality, we get $D_{TV}(t_2(t_1(f(X_{N'}))), X_{4n+\frac{r}{8}})$ is small which is the desired result.

Next, we prove that the total variation distances involved in each of these steps are small.

- **Step 1:** We first bound $D_{TV}(f(X_N), f(X_{N'}))$.

$$\begin{aligned} D_{TV}(f(X_N), f(X_{N'})) &\leq D_{TV}(X_N, X_{N'}) \\ &= \frac{1}{2}\sum_x |\Pr(X_N = x) - \Pr(X_{N'} = x)| \\ &= \frac{1}{2}\sum_x |\Pr(X_N = x \mid N \leq 4n)\Pr(N \leq 4n) - \Pr(X_{N'} = x \mid N' \leq 4n)\Pr(N' \leq 4n) \\ &\quad + \Pr(X_N = x \mid N > 4n)\Pr(N > 4n) - \Pr(X_{N'} = x \mid N' > 4n)\Pr(N' > 4n)| \end{aligned}$$

where the first inequality holds as applying the same transformation to two random variables can't increase their total variation distance. Now, note that X_N and $X_{N'}$ have the same distribution conditioned on $N \leq 4n$ and $N' \leq 4n$. Also, $\Pr(N \leq 4n) = \Pr(N' \leq 4n)$ and $\Pr(N > 4n) = \Pr(N' > 4n)$, as both N and N' are drawn from Poisson($2n$) distribution. This gives us

$$\begin{aligned} D_{TV}(f(X_N), f(X_{N'})) &= \frac{1}{2}\sum_x \Pr(N > 4n)|\Pr(X_N = x \mid N > 4n) - \Pr(X_{N'} = x \mid N' > 4n)| \\ &\leq \Pr(N > 4n) \end{aligned}$$

Using the triangle inequality, we get $D_{TV}(X_M, f(X_N')) \leq \Pr(N > 4n) + 3\epsilon/2$. To bound $\Pr(N > 4n)$, we use the following Poisson tail bound [29]: for $X \leftarrow \text{Poisson}(\lambda)$,

$$\Pr[X \geq \lambda + x], \Pr[X \leq \lambda - x] \leq e^{\frac{-x^2}{\lambda + x}}. \tag{5.13}$$

As N is distributed as Poisson$(2n)$, we get $\Pr(N > 4n) \leq e^{-n}$, which implies $D_{TV}(X_M, f(X'_N)) \leq e^{-n} + \frac{3\epsilon}{2}$.

- **Step 2:** In this step, we need to show $D_{TV}(t_1(f(X_{N'})), X_M)$ is small. Note that $t_1(f(X_{N'}))$ is equal to $f(X_{N'})$ except when $M' < N' + \frac{r}{8}$. From step 1, we know that $D_{TV}(f(X_{N'}), X_M)$ is small. If we show $D_{TV}(f(X_{N'}), t_1(f(X_{N'})))$ is small, then by triangle inequality, we get $D_{TV}(X_M, t_1(f(X_{N'})))$ is small. Let $M' = N' + R'$ where R' denote the number of new samples added by the amplification procedure f to $X_{N'}$.

$$
\begin{aligned}
&D_{TV}\left(t_1\left(f\left(X_{N'}\right)\right), f\left(X_{N'}\right)\right) \\
&= \frac{1}{2}\sum_x \left|\Pr\left(t_1\left(f\left(X_{N'}\right)\right) = x\right) - \Pr\left(f\left(X_{N'}\right) = x\right)\right| \\
&= \frac{1}{2}\sum_x \Big|\Pr\left(R' < \frac{r}{8}\right)\left(\Pr\left(t_1\left(f\left(X_{N'}\right)\right) = x \mid R' < \frac{r}{8}\right) - \Pr\left(f\left(X_{N'}\right) = x \mid R' < \frac{r}{8}\right)\right) \\
&+ \Pr\left(R' \geq \frac{r}{8}\right)\left(\Pr\left(t_1\left(f\left(X_{N'}\right)\right) = x \mid R' \geq \frac{r}{8}\right) - \Pr\left(f\left(X_{N'}\right) = x \mid R' \geq \frac{r}{8}\right)\right)\Big|
\end{aligned}
$$

We know $\Pr\left(t_1\left(f\left(X_{N'}\right)\right) = x \mid R' \geq \frac{r}{8}\right) = \Pr\left(f\left(X_{N'}\right) = x \mid R' \geq \frac{r}{8}\right)$. This gives

$$
\begin{aligned}
&D_{TV}\left(t_1\left(f\left(X_{N'}\right)\right), f\left(X_{N'}\right)\right) \\
&= \frac{1}{2}\sum_x \Big|\Pr\left(R' < \frac{r}{8}\right)\left(\Pr\left(t_1\left(f\left(X_{N'}\right)\right) = x \mid R' < \frac{r}{8}\right) - \Pr\left(f\left(X_{N'}\right) = x \mid R' < \frac{r}{8}\right)\right)\Big| \\
&\leq \Pr\left(R' < \frac{r}{8}\right)
\end{aligned}
$$

Now, we need to bound $\Pr\left(R' < \frac{r}{8}\right)$. From the description of f, we know that the number of new copies of element i added by f is distributed as Poisson$\left(r\hat{p}_i\right)$. Here, $\hat{p}_i = \frac{u_i}{n}$ where u_i denotes the number of occurrences of element i in $X_{N'_1}$. Since the total number of samples in $X_{N'_1}$ is N'_1, we get $\sum_{i=1}^k \hat{p}_i = \frac{\sum_{i=1}^k u_i}{n} = \frac{N'_1}{n}$. Note that R' is equal to the sum of number of new copies of each element, and as the sum of Poisson random variables is Poisson, we get R' is distributed as Poisson $\left(r\frac{N'_1}{n}\right)$.

$$
\begin{aligned}
\Pr\left(R' < \frac{r}{8}\right) &= \Pr\left(R' < \frac{r}{8} \mid N'_1 \geq \frac{3n}{4}\right)\Pr\left(N'_1 \geq \frac{3n}{4}\right) + \Pr\left(R' < \frac{r}{8} \mid N'_1 < \frac{3n}{4}\right)\Pr\left(N'_1 < \frac{3n}{4}\right) \\
&\leq \Pr\left(R' < \frac{r}{8} \mid N'_1 \geq \frac{3n}{4}\right) + \Pr\left(N'_1 < \frac{3n}{4}\right)
\end{aligned}
$$

Using Poisson tail bound (5.13), we get

$$\Pr\left(R' < \frac{r}{8} \mid N_1' \geq \frac{3n}{4}\right) \leq exp\left(-\frac{(5r/8)^2}{3r/4 + 5r/8}\right) = e^{-25r/88}$$
$$\Pr\left(N_1' < \frac{3n}{4}\right) \leq exp\left(-\frac{(n/4)^2}{n + n/4}\right) = e^{-n/20}$$

This gives us $D_{TV}(f(X_{N'}), t_1(f(X_{N'}))) \leq e^{-25r/88} + e^{-n/20}$. By triangle inequality, we get $D_{TV}(X_M, t_1(f(X_{N'}))) \leq \frac{3\epsilon}{2} + e^{-n} + e^{-25r/88} + e^{-n/20}$.

- **Step 3:** For this step, we need to show $D_{TV}(t_2(t_1(f(X_{N'}))), t_2(X_M))$ is small. Since applying the same transformation to two random variables doesn't increase their TV distance, we get

$$\begin{aligned} D_{TV}(t_2(t_1(f(X_{N'}))), t_2(X_M)) &\leq D_{TV}(t_1(f(X_{N'})), X_M) \\ &\leq \frac{3\epsilon}{2} + e^{-n} + +e^{-25r/88} + e^{-n/20} \end{aligned}$$

As $D_{TV}(X_{4n+\frac{r}{8}}, t_2(X_M)) = 0$, using triangle inequality, we get

$$D_{TV}(t_2(t_1(f(X_{N'}))), X_{4n+\frac{r}{2}}) \leq \frac{3\epsilon}{2} + e^{-n} + e^{-25r/88} + e^{-n/20}$$

For $\epsilon \geq 2e^{-n/20} + e^{-25r/88}$, this gives us $D_{TV}(f'(X_{4n}), X_{4n+\frac{r}{8}}) = D_{TV}(t_2(t_1(f(X_{N'}))), X_{4n+\frac{r}{8}}) \leq \frac{5\epsilon}{2}$. □

From lemma 12, we get that for $\epsilon \geq 2e^{-n/20} + e^{-25r/88}$, and for $r \leq n\epsilon^{1.5}/(4\sqrt{k})$, $D_{TV}(f'(X_{4n}), X_{4n+\frac{r}{8}}) \leq \frac{5\epsilon}{2}$. We can assume n is at least $\sqrt{k}$, and r is at least 8, as otherwise the theorem is trivially true. So for k large enough (implying large n), we can put $\epsilon = \frac{2}{15}$, to get $D_{TV}(t_2(t_1(f(X_{N'}))), X_{4n+\frac{r}{8}}) \leq \frac{1}{3}$, which finishes the proof! □

5.4.2 Lower Bound

In this section we show that the above procedure is optimal, up to constant factors for amplifying samples from discrete distributions. We first describe the intuition for showing our lower bound that the class of discrete distributions with support at most k does not admit an (n, m) amplification scheme for $m \geq n + \frac{cn}{\sqrt{k}}$, where c is a fixed constant. For $n \leq \frac{k}{4}$, we show this lower bound for the class of uniform distributions $D = \text{Unif}[k]$ on some unknown k elements. In this case, a verifier can distinguish between true samples from D and a set of amplified samples by counting the number of unique samples in the set. Note that as the support of D is unknown, the number of unique samples in the amplified set is at most the number of unique samples in the original set X_n, unless the amplifier includes samples that are outside the support of D, in which

case the verifier will trivially reject this set. The expected number of unique samples in n and m draws from D differs by $\frac{c_1 n}{\sqrt{k}}$, for some fixed constant c_1. We use a Doob martingale and martingale concentration bounds to show that the number of unique samples in n samples from D concentrates within a $\frac{c_2 n}{\sqrt{k}}$ margin of its expectation with high probability, for some fixed constant $c_2 \ll c_1$. This implies that there will be a large gap between the number of unique samples in n and m draws from D. The verifier uses this to distinguish between true samples from D and an amplified set, which cannot have sufficiently many unique samples.

Finally, we show that for $n > \frac{k}{4}$, a $\left(n, n + \frac{c'k}{\sqrt{k}}\right)$ amplification procedure for discrete distributions on k elements implies a $(\frac{k}{4}, \frac{k}{4} + c'\sqrt{k})$ amplification procedure for the uniform distribution on $(k-1)$ elements, and for sufficiently large c' this is a contradiction to the previous part. This reduction follows by considering the distribution which has $1 - \frac{k}{4n}$ mass on one element and $\frac{k}{4n}$ mass uniformly distributed on the remaining $(k-1)$ elements. With sufficiently large probability, the number of samples in the uniform section will be $\approx \frac{k}{4}$, and hence we can apply the previous result.

Proposition 7. *There is a constant c, such that for every sufficiently large k, $\mathcal{C}$ does not admit an $\left(n, n + \frac{cn}{\sqrt{k}}\right)$ amplification procedure.*

The proposition follows by constructing a verifier and class of discrete distributions over k elements, $\mathcal{C}$ with the following property: for a universal constant c and $p \leftarrow Uniform[\mathcal{C}]$, the verifier can detect *any* $(n, n + \frac{cn}{\sqrt{d}})$ amplifier from with sufficiently high probability.

Before we prove Proposition 7, we introduce some additional notation and a basic martingale inequality. Let C^k be the set of discrete uniform distributions over k integers in $0, \ldots, 8k$. Let C^k_l be the set of discrete distributions with mass $1 - l$ on one element and uniform mass over $k-1$ remaining integers in $0, \ldots, 8k$. We also rely on some martingale inequalities which can be found in [39].

Fact 4. *Let X be the martingale associated with a filter $\mathcal{F}$ satisfying:*

1. $\mathrm{Var}[X_i \mid \mathcal{F}_{i-1}] \leq \sigma_i^2$ *for* $1 \leq i \leq n$
2. $0 \leq X_i \leq 1$ *almost surely.*

Then, we have

$$Pr(X - \mathbb{E}[x] \geq \lambda) \leq e^{-\frac{\lambda^2}{2(\sum \sigma_i^2 + \lambda/3)}}.$$

Similarly the following holds (though not simultaneously):

$$Pr(X - \mathbb{E}[x] \leq -\lambda) \leq e^{-\frac{\lambda^2}{2(\sum \sigma_i^2 + \lambda/3)}}.$$

Finally we rely on slight generalization of the birthday paradox which can be found in [19].

Fact 5. *Let n samples be drawn from a uniform distribution over k elements. Then the probability of the samples containing a duplicate is less than $\frac{n^2}{2k}$.*

The proof proceeds in two parts. First we prove a lemma that shows the desired result for $n \leq \frac{k}{2}$. We then show show a class of distributions that allows us to reduce the general case to the result shown in the lemma.

Lemma 13. *For sufficiently large k, fixed c and $m = n + 30\frac{n}{\sqrt{k}} \leq \frac{k}{4}$ the following holds: There exists a verifier that for $p \sim Uniform[C^k]$ the following holds true:*

1. *For all p, it accepts X_m with probability at least $\frac{3}{4}$ over the randomness in X_m.*
2. *It rejects $f(X_n)$ with probability at least $\frac{3}{4}$ for any amplifier f over the randomness in X_n, p and the amplifier.*

Proof. First we consider the case when $n \leq \frac{\sqrt{k}}{2}$. Consider the verifier that takes $\frac{\sqrt{k}}{2} + 1 < \sqrt{\frac{k}{2}}$ samples from the given samples uniformly at random and accepts if there are no repeats by Fact 5 and the support is correct. The probability of a duplicate with the real distribution is less than $\frac{1}{4}$ by fact 5 so the verifier will accept samples from the true distribution with at least probability $\frac{3}{4}$. An amplified set, on the other hand, must have repeats outside of the original elements it saw. This is because if the amplifier expanded the support of the set, the verifier would catch it with probability $\frac{7}{8}$. To show this, consider a sample added by the amplifier outside of the seen support. Conditioned on the at most $\frac{k}{4}$ unique samples seen so far (which implies that $\frac{3}{4}$ of the support is still unseen), the probability, over the choice of p, of said sample being in the set is at most $\frac{(3/4)k}{8k-n} \leq \frac{(3/4)k}{7.5k} \leq \frac{1}{8}$. Hence if the amplified set has any element outside the original support then it is rejected with probability $\frac{7}{8}$. Note that if the amplified set has at most $\frac{\sqrt{k}}{2}$ unique elements, then it can be immediately distinguished for having too many repeats.

We now examine the case when $n > \frac{\sqrt{k}}{2}$. Since the verifier can identify when the amplifier introduces unseen elements with probabiltiy at least $\frac{7}{8}$, we condition on the event that the verifier identifies such elements for the remainder of this proof. The proof proceeds by showing that a set the size of the amplified set must have significantly more unique elements than the original set. Before we proceed with the details of the proof we define the martingale that is central to the argument. Consider the scenario where the n samples are drawn in sequence, and let $\mathcal{F}_i$ denote the filtration corresponding to the i-th draw (i.e., information in the first i draws). Let U_i be the indicator that is the ith sample was previously unseen. Let $U^n = \sum_{i=1}^{n} U_i$. Note that $B_i = \mathbb{E}\left[\sum_{j=1}^{n} U_j \mid \mathcal{F}_i\right]$ is a Doob martingale with respect to the filtration $\mathcal{F}_i$ and $B_n = U$. Also, B_i has differences bounded by 1 as U_i is an indicator random variable. If j is the count of previously seen elements then $\text{Var}\left[B_i \mid \mathcal{F}_i\right] \leq \text{Var}[U_i \mid \mathcal{F}_i] \leq \frac{(k-j)j}{k^2}$. Since $n < \frac{k}{2}$, the variance is upper bounded by $\frac{i}{k} \leq \frac{n}{k}$.

The verifier will accept only if all elements are within the support of the distribution and the number unique elements is greater than $\mathbb{E}[U^n] + 7\frac{n}{\sqrt{k}}$ under X_n.

The remainder of the proof will show the following:

1. U^n concentrates around its expectation within a $O\left(\frac{n}{\sqrt{k}}\right)$ margin for X_n (this shows the amplifier gets too few unique samples to be accepted by the verifier).
2. The expectation $\mathbb{E}[U^m - U^n]$ increases by at least $\Omega\left(\frac{n}{\sqrt{k}}\right)$ from X_n to X_m (which shows the number of unique items is sufficiently different in expectation between X_n and X_m).
3. U^m concentrates around its expectation within a $O\left(\frac{n}{\sqrt{k}}\right)$ margin for X_m (this combined with the previous statement shows the verifier accepts real samples with sufficiently high probability).

The upper tail bound follows via Fact 4. Recall that $\frac{n}{\sqrt{k}} < 4\frac{n^2}{k}$ since $n > \frac{\sqrt{k}}{2}$.

$$\begin{aligned}
\Pr\left(U^n - \mathbb{E}[U^n] \geq 7\frac{n}{\sqrt{k}}\right) &\leq \exp\left(-\frac{7^2\frac{n^2}{k}}{2\left(\sum\sigma_i^2 + \frac{7n}{3\sqrt{k}}\right)}\right) \\
&\leq \exp\left(-\frac{7^2\frac{n^2}{k}}{2\left(\frac{n^2}{k} + \frac{7n}{3\sqrt{k}}\right)}\right) \\
&\leq \exp\left(-\frac{7^2\frac{n^2}{k}}{2\left(\frac{n^2}{k} + 7\frac{4n^2}{3k}\right)}\right) \\
&= \exp\left(-\frac{7^2\frac{n^2}{k}}{2\left(1 + \frac{4}{3}7\right)\frac{n^2}{k}}\right) \\
&\leq \frac{1}{8}.
\end{aligned}$$

Note that this suffices to show that the verifier can distinguish any amplifier with sufficiently many unique samples.

Let k be sufficiently large that the following conditions hold for both k and $k-1$:

1. $n + 30\frac{n}{\sqrt{k}} < \frac{k}{2}$
2. The samples increased by at most a factor of 2

Now we note that the $\mathbb{E}[U^n]$ and $\mathbb{E}[U^m]$ must differ by at least $\frac{15n}{\sqrt{k}}$, since $m < \frac{k}{2}$ implying that every new sample has at least a $\frac{1}{2}$ probability of being unique. Now all the remains to show that the verifier will accept X_m is to show concentration of U within $\frac{8n}{\sqrt{k}}$ of its mean.

Since the number of samples increased by at most a factor of two, the bound on the σ_i^2 increased by at most a factor of two. This suffices for the lower tail bound on U for X_m—

$$\begin{aligned}
\Pr\left(U^m - \mathbb{E}[U^m] \leq -8\frac{n}{\sqrt{k}}\right) &\leq \exp\left(-\frac{8^2\frac{n^2}{k}}{2\left(\sum \sigma_i^2 + \frac{8n}{3\sqrt{k}}\right)}\right) \\
&\leq \exp\left(-\frac{8^2\frac{n^2}{k}}{2\left(4\frac{n^2}{k} + \frac{8n}{3\sqrt{k}}\right)}\right) \\
&\leq \exp\left(-\frac{8^2\frac{n^2}{k}}{2\left(4\frac{n^2}{k} + 8\frac{4n^2}{3k}\right)}\right) \\
&= \exp\left(-\frac{8^2\frac{n^2}{k}}{2\left(4 + \frac{4}{3}8\right)\frac{n^2}{k}}\right) \\
&< \frac{1}{8}.
\end{aligned}$$

Thus X_m will have sufficiently many unique elements to be accepted by the verifier with probability at least $\frac{7}{8}$. A success probability of $\frac{3}{4}$ follows from subtracting the probabiltiy that the verifier did not properly identify unseen samples.

□

We are now ready to prove Proposition 7.

Proof. If $n \leq \frac{k}{4}$, then Lemma 13 applies directly. If not, we use the set of distributions $\mathcal{C}^k_{\frac{k}{4}}$ with the intention of applying Lemma 13 on samples that land in the uniform region.

The verifier will check that the samples are within the support of the distribution, more than $n+7\frac{n}{\sqrt{k}}$ samples are in the uniform region and the verifier from Lemma 13 accepts on the uniform region.

First note that after n samples, at most $\frac{k}{4} + \frac{\sqrt{k}}{4}$ samples will be in the uniform region with at least probability $\frac{15}{16}$ by a Chebyshev bound. Conditioned on this event, Lemma 13 shows that the amplifier cannot output more than $\frac{k}{4} + O(\sqrt{k})$ samples in the uniform region and will be rejected by our verifier.

Now we show that the verifier will accept real samples with good probability. Note that the expected number of samples to receive in the uniform region for X_m is $\frac{k}{4}+c\sqrt{k}$. The variance on this quantity is $\frac{k}{4}+c\sqrt{k}$. An application of Chebyshev's inequality shows that with probability at least $\frac{15}{16}$ sufficiently many samples will land in the uniform region.

$$\begin{aligned}
\frac{k}{4} + c\sqrt{k} - 4\sqrt{\frac{k}{4} + c\sqrt{k}} &\geq \frac{k}{4} + c\sqrt{k} - 2\sqrt{k} - 4\sqrt{c\sqrt{k}} \\
&\geq \frac{k}{4} + c\sqrt{k} - 2\sqrt{k} - 4\sqrt{ck}.
\end{aligned}$$

Since the expression above is increasing with c, we can choose a c sufficiently large so that the verifier will accept with sufficiently high probability. □

5.5 Conclusion

In this chapter we defined the concept of sample amplification and demonstrated that it is a statistically easier problem than distribution learning for discrete distributions and multivariate Gaussians of known covariance. Further work by Yanjun Han and the authors of the original sample amplification paper, extend these results to a wide variety of distributions. However, our understanding of data augmentation and synthetic data techniques remains incomplete.

In particular, the amplification factors demonstrated in this paper are rather small. While this small degree of amplification is occasionally useful, as shown in the examples, there are also empirical examples of it being useful to amplify beyond what is possible under the model used in this chapter. This motivates two directions of study. The first direction is whether larger amplification is possible when the way the data is used is restricted. For example, if we know that our data will be used to fit a particular class of models via empirical risk minimization, does this mean it is useful to amplify beyond the amounts discussed in this chapter? Should we use a different set of amplification techniques in such settings? Another direction is to develop a broader understanding of synthetic data. If we use our dataset to generate a dataset which is far from one drawn iid from the distribution our models are evaluated on, can it still be useful? What properties do we really need from a dataset in order to have guarantees that our models will perform well?

Bibliography

[1] J. Acharya, I. Diakonikolas, C. Hegde, J. Li, and L. Schmidt. Fast and near-optimal algorithms for approximating distributions by histograms. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015*, pages 249–263, 2015.

[2] J. Acharya, I. Diakonikolas, J. Li, and L. Schmidt. Sample-optimal density estimation in nearly-linear time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 1278–1289, 2017. Available at https://arxiv.org/abs/1506.00671.

[3] Jayadev Acharya, Hirakendu Das, Ashkan Jafarpour, Alon Orlitsky, Shengjun Pan, and Ananda Suresh. Competitive classification and closeness testing. In *Conference on Learning Theory*, pages 22–1, 2012.

[4] Sandip Aine, Siddharth Swaminathan, Venkatraman Narayanan, Victor Hwang, and Maxim Likhachev. Multi-heuristic a*. *The International Journal of Robotics Research*, 35(1-3):224–243, 2016.

[5] M. Y. An. Log-concave probability distributions: Theory and statistical testing. Technical Report Economics Working Paper Archive at WUSTL, Washington University at St. Louis, 1995.

[6] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

[7] Brian Williams Ashkan Jasour. Risk contours map for risk bounded motion planning under perception uncertainties. In *Robotics: Science and Systems*, 2019.

[8] Brian Axelrod. Algorithms for Safe Robot Navigation. Master's thesis, Massachusetts Institute of Technology, 2017.

[9] Brian Axelrod, Leslie Kaelbling, and Tomás Lozano-Pérez. Provably safe robot navigation with obstacle uncertainty. *Robotics Science and Systems*, 13, 2017.

[10] Brian Axelrod, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Provably safe robot navigation with obstacle uncertainty. *The International Journal of Robotics Research*, 2018.

[11] M. Bagnoli and T. Bergstrom. Log-concave probability and its applications. *Economic Theory*, 26(2):pp. 445–469, 2005.

[12] F. Balabdaoui and C. R. Doss. Inference for a two-component mixture of symmetric distributions under log-concavity. *Bernoulli*, 24(2):1053–1071, 05 2018.

[13] F. Balabdaoui, K. Rufibach, and J. A. Wellner. Limit distribution theory for maximum likelihood estimation of a log-concave density. *The Annals of Statistics*, 37(3):pp. 1299–1331, 2009.

[14] F. Balabdaoui and J. A. Wellner. Estimation of a k-monotone density: Limit distribution theory and the spline connection. *The Annals of Statistics*, 35(6):pp. 2536–2564, 2007.

[15] F. Balabdaoui and J. A. Wellner. Estimation of a k-monotone density: characterizations, consistency and minimax lower bounds. *Statistica Neerlandica*, 64(1):45–70, 2010.

[16] R.E. Barlow, D.J. Bartholomew, J.M. Bremner, and H.D. Brunk. *Statistical Inference under Order Restrictions*. Wiley, New York, 1972.

[17] T. Batu, E. Fischer, L. Fortnow, R. Kumar, R. Rubinfeld, and P. White. Testing random variables for independence and identity. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2001.

[18] Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D Smith, and Patrick White. Testing closeness of discrete distributions. *Journal of the ACM (JACM)*, 60(1):4, 2013.

[19] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining message authentication code. In *Advances in Cryptology–CRYPTO*, volume 94, pages 341–358. Citeseer, 1994.

[20] Bhaswar Bhattacharya and Gregory Valiant. Testing closeness with unequal sized samples. In *Advances in Neural Information Processing Systems*, pages 2611–2619, 2015.

[21] L. Birgé. Estimating a density under order restrictions: Nonasymptotic minimax risk. *Annals of Statistics*, 15(3):995–1012, 1987.

[22] L. Birgé. On the risk of histograms for estimating decreasing densities. *Annals of Statistics*, 15(3):1013–1022, 1987.

[23] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[24] H. D. Brunk. On the estimation of parameters restricted by inequalities. *The Annals of Mathematical Statistics*, 29(2):pp. 437–454, 1958.

[25] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 723–730. IEEE, 2011.

[26] John Canny. Some algebraic and geometric computations in pspace. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 460–467, 01 1988.

[27] John Canny and John Reif. New lower bound techniques for robot motion planning problems. In *Foundations of Computer Science*, pages 49 – 60, 11 1987.

[28] C. L. Canonne, I. Diakonikolas, T. Gouleakis, and R. Rubinfeld. Testing shape restrictions of discrete distributions. In *STACS*, pages 25:1–25:14, 2016.

[29] Clément L Canonne. A short note on poisson tail bounds, 2017.

[30] Clément L Canonne, Themis Gouleakis, and Ronitt Rubinfeld. Sampling correctors. *SIAM Journal on Computing*, 47(4):1373–1423, 2018.

[31] T. Carpenter, I. Diakonikolas, A. Sidiropoulos, and A. Stewart. Near-optimal sample complexity bounds for maximum likelihood estimation of multivariate log-concave densities. In *Conference On Learning Theory, COLT 2018*, pages 1234–1262, 2018.

[32] A.R. Cassandra, L.P. Kaelbling, and James Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *International Conference on Intelligent Robots and Systems*, volume 12, pages 963 – 972 vol.2, 12 1996.

[33] K.S. Chan and H. Tong. Testing for multimodality with dependent data. *Biometrika*, 91(1):113–123, 2004.

[34] S. Chan, I. Diakonikolas, R. Servedio, and X. Sun. Learning mixtures of structured distributions over discrete domains. In *SODA*, pages 1380–1394, 2013.

[35] S. Chan, I. Diakonikolas, R. Servedio, and X. Sun. Efficient density estimation via piecewise polynomial approximation. In *STOC*, pages 604–613, 2014.

[36] S. Chan, I. Diakonikolas, R. Servedio, and X. Sun. Near-optimal density estimation in near-linear time using variable-width histograms. In *NIPS*, pages 1844–1852, 2014.

[37] Siu-On Chan, Ilias Diakonikolas, Paul Valiant, and Gregory Valiant. Optimal algorithms for testing closeness of discrete distributions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1193–1203. SIAM, 2014.

[38] Y. Chen and R. J. Samworth. Smoothed log-concave maximum likelihood estimation with applications. *Statist. Sinica*, 23:1373–1398, 2013.

[39] Fan Chung and Linyuan Lu. Concentration inequalities and martingale inequalities: a survey. *Internet Mathematics*, 3(1):79–127, 2006.

[40] Gabe Cohn. AI art at Christie's sells for $432,500. *The New York Times*, Oct 2018.

[41] M. Cule, R. Samworth, and M. Stewart. Maximum likelihood estimation of a multi-dimensional log-concave density. *Journal of the Royal Statistical Society: Series B*, 72:545–607, 2010.

[42] Y. Dagan and G. Kur. The log-concave maximum likelihood estimator is optimal in high dimensions. *CoRR*, abs/1903.05315, 2019.

[43] C. Daskalakis, A. De, G. Kamath, and C. Tzamos. A size-free CLT for poisson multinomials and its applications. In *Proceedings of the 48th Annual ACM Symposium on the Theory of Computing*, STOC '16, 2016.

[44] C. Daskalakis, I. Diakonikolas, R. O'Donnell, R.A. Servedio, and L. Tan. Learning Sums of Independent Integer Random Variables. In *FOCS*, pages 217–226, 2013.

[45] C. Daskalakis, I. Diakonikolas, and R.A. Servedio. Learning k-modal distributions via testing. In *SODA*, pages 1371–1385, 2012.

[46] C. Daskalakis, I. Diakonikolas, and R.A. Servedio. Learning Poisson Binomial Distributions. In *STOC*, pages 709–728, 2012.

[47] A. De, P. M. Long, and R. A. Servedio. Density estimation for shift-invariant multidimensional distributions. *CoRR*, abs/1811.03744, 2018.

[48] Luc Devroye, Abbas Mehrabian, and Tommy Reddad. The total variation distance between high-dimensional gaussians. *arXiv preprint arXiv:1810.08693*, 2018.

[49] I. Diakonikolas, D. M. Kane, and A. Stewart. Efficient Robust Proper Learning of Log-concave Distributions. Arxiv report, 2016.

[50] I. Diakonikolas, D. M. Kane, and A. Stewart. The fourier transform of poisson multinomial distributions and its algorithmic applications. In *Proceedings of STOC'16*, 2016.

[51] I. Diakonikolas, D. M. Kane, and A. Stewart. Optimal learning via the fourier transform for sums of independent integer random variables. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016*, pages 831–849, 2016. Full version available at https://arxiv.org/abs/1505.00662.

[52] I. Diakonikolas, D. M. Kane, and A. Stewart. Properly learning poisson binomial distributions in almost polynomial time. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016*, pages 850–878, 2016. Full version available at https://arxiv.org/abs/1511.04066.

[53] I. Diakonikolas, D. M. Kane, and A. Stewart. Learning multivariate log-concave distributions. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017*, pages 711–727, 2017.

[54] I. Diakonikolas, J. Li, and L. Schmidt. Fast and sample near-optimal algorithms for learning multidimensional histograms. In *Conference On Learning Theory, COLT 2018*, pages 819–842, 2018.

[55] Ilias Diakonikolas and Daniel M Kane. A new approach for testing properties of discrete distributions. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 685–694. IEEE, 2016.

[56] Xu Chu Ding, Alessandro Pinto, and Amit Surana. Strategic planning under uncertainties via constrained markov decision processes. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4568–4575. IEEE, 2013.

[57] C. R. Doss and J. A. Wellner. Global rates of convergence of the mles of log-concave and s-concave densities. *Ann. Statist.*, 44(3):954–981, 06 2016.

[58] J. C. Duchi. Introductory lectures on stochastic convex optimization. *Park City Mathematics Institute, Graduate Summer School Lectures*, 2016.

[59] L. Dumbgen and K. Rufibach. Maximum likelihood estimation of a log-concave density and its distribution function: Basic properties and uniform consistency. *Bernoulli*, 15(1):40–68, 2009.

[60] Lutz Dümbgen and Kaspar Rufibach. logcondens: Computations related to univariate log-concave density estimation. *Journal of Statistical Software*, 39(6):1–28, 2011.

[61] John E. Hopcroft, Deborah Joseph, and Sue Whitesides. Movement problems for 2-dimensional linkages. In *SIAM Journal on Computing*, volume 13, pages 610–629, 08 1984.

[62] Eduard Eiben, Jonathan Gemmell, Iyad Kanj, and Andrew Youngdahl. Improved results for minimum constraint removal, 2018.

[63] Eduard Eiben, Jonathan Gemmell, Iyad Kanj, and Andrew Youngdahl. Improved results for minimum constraint removal. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[64] L. Erickson and S. LaValle. A simple, but np-hard, motion planning problem. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[65] William F. Dowling and Jean Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *The Journal of Logic Programming*, 1:267–284, 10 1984.

[66] Seyedshams Feyzabadi and Stefano Carpin. Multi-objective planning with multiple high level task specifications. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 5483–5490. IEEE, 2016.

[67] A.-L. Fougères. Estimation de densités unimodales. *Canadian Journal of Statistics*, 25:375–387, 1997.

[68] Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *Neurocomputing*, 321:321–331, 2018.

[69] F. Gao and J. A. Wellner. On the rate of convergence of the maximum likelihood estimator of a k-monotone density. *Science in China Series A: Mathematics*, 52:1525–1538, 2009.

[70] O. Goldreich and D. Ron. On testing expansion in bounded-degree graphs. Technical Report TR00-020, Electronic Colloquium on Computational Complexity, 2000.

[71] Irving J Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264, 1953.

[72] U. Grenander. On the theory of mortality measurement. *Skand. Aktuarietidskr.*, 39:125–153, 1956.

[73] P. Groeneboom. Estimating a monotone density. In *Proc. of the Berkeley Conference in Honor of Jerzy Neyman and Jack Kiefer*, pages 539–555, 1985.

[74] P. Groeneboom and G. Jongbloed. *Nonparametric Estimation under Shape Constraints: Estimators, Algorithms and Asymptotics*. Cambridge University Press, 2014.

[75] Q. Han and J. A. Wellner. Approximation and estimation of s-concave densities via renyi divergences. *Ann. Statist.*, 44(3):1332–1359, 06 2016.

[76] D. L. Hanson and G. Pledger. Consistency in concave regression. *The Annals of Statistics*, 4(6):pp. 1038–1050, 1976.

[77] Kris Hauser. The minimum constraint removal problem with three robotics applications. In *Workshop on the Algorithmic Foundations of Robotics*, 2012.

[78] Kris Hauser. The minimum constraint removal problem with three robotics applications. In *The International Journal of Robotics Research*, volume 33, 2014.

[79] John Hopcroft, Deborah Joseph, and Sue Whitesides. On the movement of robot arms in 2-dimensional bounded regions. In *SIAM Journal on Computing*, volume 14, pages 280 – 289, 12 1982.

[80] H. K. Jankowski and J. A. Wellner. Estimation of a discrete monotone density. *Electronic Journal of Statistics*, 3:1567–1605, 2009.

[81] Lucas Janson, Edward Schmerling, and Marco Pavone. Monte carlo motion planning for robot trajectory optimization under uncertainty. In *International Symposium on Robotics Research*, Sestri Levante, Italy, September 2015.

[82] Brigitte Jaumard and Bruno Simeone. On the complexity of the maximum satisfiability problem for horn formulas. *Information Processing Letters*, 26:1–4, 9 1987.

[83] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 2013.

[84] R. Kannan, L. Lovász, and M. Simonovits. Random walks and an o*(n5) volume algorithm for convex bodies. *Random Structures & Algorithms*, 11(1):1–50, 1997.

[85] Sertac Karaman and Emilio Frazzoli. Sampling-based motion planning with deterministic μ-calculus specifications. In *IEEE Conference on Decision and Control*, 2009.

[86] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[87] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[88] A. K. H. Kim and R. J. Samworth. Global rates of convergence in log-concave density estimation. *Ann. Statist.*, 44(6):2756–2779, 12 2016. Available at http://arxiv.org/abs/1404.2298.

[89] Michal Kleinbort, Kiril Solovey, Zakary Littlefield, Kostas E Bekris, and Dan Halperin. Probabilistic completeness of rrt for geometric and kinodynamic planning with forward propagation. *IEEE Robotics and Automation Letters*, 4(2):x–xvi, 2018.

[90] R. Koenker and I. Mizera. Quasi-concave density estimation. *Ann. Statist.*, 38(5):2998–3027, 2010.

[91] Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

[92] Beatrice Laurent and Pascal Massart. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pages 1302–1338, 2000.

[93] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[94] Steven M. LaValle and James J. Kuffner. Randomized kinodynamic planning. In *ICRA*, 1999.

[95] Alex Lee, Yan Duan, Sachin Patil, John Schulman, Zoe McCarthy, Jur van den Berg, Ken Goldberg, and Pieter Abbeel. Sigma hulls for gaussian belief space planning for imprecise articulated robots amid obstacles. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5660–5667. IEEE, 2013.

[96] Reut Levi, Dana Ron, and Ronitt Rubinfeld. Testing properties of collections of distributions. *Theory of Computing*, 9(1):295–347, 2013.

[97] L. Lovász and S. Vempala. Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 57–68. IEEE, 2006.

[98] L. Lovász and S. Vempala. Hit-and-run from a corner. *SIAM Journal on Computing*, 35(4):985–1005, 2006.

[99] L. Lovász and S. Vempala. Simulated annealing in convex bodies and an o*(n4) volume algorithm. *Journal of Computer and System Sciences*, 72(2):392–417, 2006.

[100] L. Lovász and S. Vempala. The geometry of logconcave functions and sampling algorithms. *Random Structures and Algorithms*, 30(3):307–358, 2007.

[101] Kevin P Murphy. Conjugate bayesian analysis of the gaussian distribution. *def*, 1(2σ2):16, 2007.

[102] Alon Orlitsky, Narayana P Santhanam, and Junan Zhang. Always good turing: Asymptotically optimal probability estimation. *Science*, 302(5644):427–431, 2003.

[103] Alon Orlitsky and Ananda Theertha Suresh. Competitive distribution estimation: Why is good-turing good. In *Advances in Neural Information Processing Systems*, pages 2143–2151, 2015.

[104] Liam Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Transactions on Information Theory*, 54(10):4750–4755, 2008.

[105] Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[106] Chonhyon Park, Jae Sung Park, and Dinesh Manocha. Fast and bounded probabilistic collision detection in dynamic environments for high-dof trajectory planning. *CoRR*, abs/1607.04788, 2016.

[107] Jae Sung Park, Chonhyon Park, and Dinesh Manocha. Efficient probabilistic collision detection for non-convex shapes. *CoRR*, abs/1610.03651, 2016.

[108] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[109] Sam Prentice and Nicholas Roy. The belief roadmap: Efficient planning in linear pomdps by factoring the covariance. In *Proceedings of the 13th International Symposium of Robotics Research (ISRR)*, Hiroshima, Japan, November 2007.

[110] B.L.S. Prakasa Rao. Estimation of a unimodal density. *Sankhya Ser. A*, 31:23–36, 1969.

[111] F. Rathke and C. Schnörr. Fast multivariate log-concave density estimation. *CoRR*, abs/1805.07272, 2018.

[112] John Reif. Complexity of the mover's problem and generalizations. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, pages 421–427, 11 1979.

[113] E. Robeva, B. Sturmfels, and C. Uhler. Geometry of Log-Concave Density Estimation. *ArXiv e-prints*, 2017. Available at https://arxiv.org/abs/1704.01910.

[114] Dorsa Sadigh and Ashish Kapoor. Safe control under uncertainty with probabilistic signal temporal logic. In *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016.

[115] Oren Salzman, Brian Hou, and Siddhartha Srinivasa. Efficient motion planning for problems lacking optimal substructure. *arXiv preprint arXiv:1703.02582*, 2017.

[116] Oren Salzman and Siddhartha Srinivasa. Open problem on risk-aware planning in the plane. *arXiv preprint arXiv:1612.05101*, 2016.

[117] R. J. Samworth. Recent progress in log-concave density estimation. *ArXiv e-prints*, 2017.

[118] A. Saumard and J. A. Wellner. Log-concavity and strong log-concavity: A review. *Statist. Surv.*, 8:45–114, 2014.

[119] Luke Shimanuki and Brian Axelrod. Hardness of 3d motion planning under obstacle uncertainty. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 852–867. Springer, 2018.

[120] Luke Shimanuki and Brian Axelrod. Hardness of motion planning with obstacle uncertainty in two dimensions. *The International Journal of Robotics Research*, 40(10-11):1151–1166, 2021.

[121] Michael Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1st edition, 1996.

[122] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. In *Advances in Neural Information Processing Systems*, volume 58, 01 2013.

[123] R. P. Stanley. Log-concave and unimodal sequences in algebra, combinatorics, and geometry. *Annals of the New York Academy of Sciences*, 576(1):500–535, 1989.

[124] Wen Sun, Luis G Torres, Jur Van Den Berg, and Ron Alterovitz. Safe motion planning for imprecise robotic manipulators by minimizing probability of collision. In *Robotics Research*, pages 685–701. Springer, 2016.

[125] Salil P Vadhan et al. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.

[126] Gregory Valiant and Paul Valiant. Instance optimal learning of discrete distributions. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 142–155. ACM, 2016.

[127] Gregory Valiant and Paul Valiant. An automatic inequality prover and instance optimal identity testing. *SIAM Journal on Computing*, 46(1):429–455, 2017.

[128] Paul Valiant. Testing symmetric properties of distributions. *SIAM Journal on Computing*, 40(6):1927–1968, 2011.

[129] Emanuele Viola. The complexity of distributions. *SIAM Journal on Computing*, 41(1):191–218, 2012.

[130] M. J. Wainwright, M. I. Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[131] Martin Wainwright. Basic tail and concentration bounds. *https://www.stat.berkeley.edu/~mjwain/stat210b/Chap2_TailBounds_Jan22_2015.pdf (visited on 12/31/2017)*, 2015.

[132] G. Walther. Inference and modeling with log-concave distributions. *Stat. Science*, 24:319–327, 2009.

[133] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7278–7286, 2018.

[134] E.J. Wegman. Maximum likelihood estimation of a unimodal density. I. and II. *Ann. Math. Statist.*, 41:457–471, 2169–2174, 1970.

[135] Hao Yi, Alon Orlitsky, Ananda Theertha Suresh, and Yihong Wu. Data amplification: A unified and competitive approach to property estimation. In *Advances in Neural Information Processing Systems*, pages 8848–8857, 2018.

[136] Xin Yi, Ekta Walia, and Paul Babyn. Generative adversarial network in medical imaging: A review. *Medical image analysis*, page 101552, 2019.