Computational Concentration of Measure: Optimal Bounds, Reductions, and More

Omid Etesami[*]   Saeed Mahloujifar[†]   Mohammad Mahmoody[‡]

July 12, 2019

Abstract

Product measures of dimension n are known to be "concentrated" under Hamming distance. More precisely, for any set $\mathcal{S}$ in the product space of probability $\Pr[\mathcal{S}] \geq \varepsilon$, a random point in the space, with probability $1 - \delta$, has a neighbor in $\mathcal{S}$ that is different from the original point in only $O(\sqrt{n \cdot \ln(1/\varepsilon\delta)})$ coordinates (and this is optimal). In this work, we obtain the tight *computational* (algorithmic) version of this result, showing how given a random point and access to an $\mathcal{S}$-membership query oracle, we can find such a close point of Hamming distance $O(\sqrt{n \cdot \ln(1/\varepsilon\delta)})$ in time $\mathrm{poly}(n, 1/\varepsilon, 1/\delta)$. This resolves an open question of [MM19] who proved a weaker result (that works only for $\varepsilon \gg 1/\sqrt{n}$). As corollaries, we obtain polynomial-time poisoning and (in certain settings) evasion attacks against learning algorithms when the original vulnerabilities have any cryptographically non-negligible probability.

We call our algorithm MUCIO (short for "MUltiplicative Conditional Influence Optimizer") since proceeding through the coordinates of the product space, it decides to change each coordinate of the given point based on a multiplicative version of the influence of a variable, where the influence is computed conditioned on the value of all previously updated coordinates. MUCIO is an online algorithm in that it decides on the i'th coordinate of the output given only the first i coordinates of the input. It also does not make any convexity assumption about the set $\mathcal{S}$.

Motivated by obtaining algorithmic variants of measure concentration in other metric probability spaces, we define a new notion of algorithmic reduction between computational concentration of measure in different probability metric spaces. This notion, whose definition has some subtlety, requires *two* (inverse) algorithmic mappings one of which is an algorithmic Lipschitz mapping and the other one is an algorithmic coupling connecting the two distributions. As an application, we apply this notion of reduction to obtain computational concentration of measure for high-dimensional Gaussian distributions under the ℓ_1 distance.

We further prove several extensions to the results above as follows. (1) Generalizing in another dimension, our computational concentration result is also true when the Hamming distance is weighted. (2) As measure concentration is usually proved for concentration around mean, we show how to use our results above to obtain algorithmic concentration for that setting as well. In particular, we prove a computational variant of McDiarmid's inequality, when properly defined. (3) Our result generalizes to discrete random processes (instead of just product distributions), and this generalization leads to new tampering algorithms for collective coin tossing protocols. (4) Finally, we prove exponential lower bounds on the average running time of *non-adaptive* query algorithms for proving computational concentration for the case of product spaces. Perhaps surprisingly, such lower bound shows any efficient algorithm must query about $\mathcal{S}$-membership of points that are *not* close to the original point even though we are only interested in finding a close point in $\mathcal{S}$.

Contents

1 Introduction

Let $(\mathcal{X}, \mathsf{d}, \boldsymbol{\mu})$ be a metric probability space in which d is a metric over $\mathcal{X}$, and $\boldsymbol{\mu}$ is a probability measure over $\mathcal{X}$. The concentration of measure phenomenon [Led01, MS86] states that many natural metric probability spaces of high dimension are concentrated in the following sense. Any set $\mathcal{S} \subseteq \mathcal{X}$ of "not too small" probability $\boldsymbol{\mu}(\mathcal{S}) \geq \varepsilon$ is "close" (according to d) to "almost all" points ($1 - \delta$ measure according to $\boldsymbol{\mu}$).

A well-studied class of concentrated spaces is the set of product spaces in which the measure $\boldsymbol{\mu} = \boldsymbol{\mu}_1 \times \ldots \boldsymbol{\mu}_n$ is a product measure of dimension n, and the metric d is Hamming distance of dimension n; namely, $\mathsf{HD}(\overline{u}, \overline{v}) = |\{i : u_i \neq v_i\}|$ for vectors $\overline{u} = (u_1, \ldots, u_n), \overline{v} = (v_1, \ldots, v_n)$. More specifically, it is known, e.g., by results implicit in [AM80, MS86] and explicit in [McD89, Tal95], and weaker versions known as blowing-up lemma proved in [AGK76, Mar74, Mar86], that any such metric probability space is a so-called Normal Lévy family [Lév51, AM85]. Namely, for any $\mathcal{S}$ of probability $\boldsymbol{\mu}(\mathcal{S}) \geq \varepsilon$, at least $1 - \delta$ fraction of the points (under the product measure $\boldsymbol{\mu}$) are $O(\sqrt{n \cdot \ln(1/\varepsilon\delta)})$-close in Hamming distance to $\mathcal{S}$. Previous proofs of measure concentration, and in particular those proofs for product spaces are *information theoretic*, and only show the *existence* of a "close" such point $\overline{y} \in \mathcal{S}$ to most of $\overline{x} \leftarrow \boldsymbol{\mu}$ sampled according to $\boldsymbol{\mu}$. Naive sampling of points around $\overline{x}$ will likely *not* fall into $\mathcal{S}$ (see Section 6).

Motivated by finding polynomial-time attacks on the "robustness" of machine learning algorithms, recently Mahloujifar and Mahmoody [MM19] studied a *computational* variant of the measure concentration in which the mapping from a given point $\overline{x} \leftarrow \boldsymbol{\mu}$ to its close neighbor $\overline{y} \in \mathcal{S}$ is supposed to be computed by an efficient polynomial-time algorithm $A^{\mathcal{S}, \boldsymbol{\mu}}(\overline{x}) = \overline{y}$ that has oracle access to test membership in $\mathcal{S}$ and a sampling oracle from the measure $\boldsymbol{\mu}$.[1] It was shown in [MM19] that if $\mathcal{S}$ is large enough, then the measure computationally concentrates around $\mathcal{S}$. In particular, it was shown that if $\Pr[\mathcal{S}] \geq 1/\operatorname{polylog}(n)$, then $A^{\mathcal{S}, \boldsymbol{\mu}}(\overline{x})$ finds $\overline{y}$ with Hamming distance $\widetilde{O}(\sqrt{n})$ from $\overline{x}$, and instead if $\mathcal{S}$ is at least $\Pr[\mathcal{S}] \geq \omega(1/\sqrt{n})$, then A finds $\overline{y}$ with Hamming distance $o(n)$. Consequently, it was left open to prove computational concentration of measure around any smaller sets of "non-negligible" $1/\operatorname{poly}(n)$ probability, e.g., of measure $1/n$.

1.1 Our Results

In this work, we resolve the open question about the computational concentration of measure in product spaces under Hamming distance and prove (tight up to constant) computational concentration for all range of initial probabilities $\Pr[\mathcal{S}]$ for the target set $\mathcal{S}$. Namely, we prove the following result matching what information theoretic concentration of product spaces guarantees up to a constant factor, while the mapping is done algorithmically. As we deal with algorithms, without loss of generality, we focus on discrete distributions.[2]

Theorem 1.1 (Main result). *There is an algorithm $A^{\mathcal{S}, \boldsymbol{\mu}}_{\varepsilon, \delta}(\cdot)$ called MUCIO (short for "MUltiplicative Conditional Influence Optimizer") that given access to a membership oracle for any set $\mathcal{S}$ and a sampling oracle from any product measure $\boldsymbol{\mu}$ of dimension n, it achieves the following. If $\Pr[\mathcal{S}] \geq \varepsilon$, given ε and δ, the algorithm $A^{\mathcal{S}, \boldsymbol{\mu}}_{\varepsilon, \delta}(\cdot)$ runs in time $\operatorname{poly}(n/\varepsilon\delta)$, and with probability $\geq 1 - \delta$ given a random point $\overline{x} \leftarrow \boldsymbol{\mu}$, it maps $\overline{x}$ to a point $\overline{y} \in \mathcal{S}$ of bounded Hamming distance $\mathsf{HD}(\overline{x}, \overline{y}) \leq O(\sqrt{n \cdot \ln(1/\varepsilon\delta)})$.*

See Theorem 3.2 for a more general version of Theorem 1.1.

For the special case that $\varepsilon, \delta = 1/\operatorname{poly}(n)$ (implying $\mathcal{S}$ has a non-negligible measure) the algorithm MUCIO of Theorem 1.1 achieves its goal in $\operatorname{poly}(n)$ time, while it changes only $\widetilde{O}(\sqrt{n})$ of the coordinates.

---

[1] In case of product measure, oracle access to a sampler from $\boldsymbol{\mu} = \boldsymbol{\mu}_1 \times \ldots \boldsymbol{\mu}_n$ is equivalent to having such samplers for all $\boldsymbol{\mu}_i$.

[2] Note that even seemingly non-discrete distributions like Gaussian, when used as input to efficient algorithms, are necessarily rounded to limited precision and thus end up being discrete.

Our work can be seen as another example of works in computer science that make previously existential proofs algorithmic. A good example of a similar successful effort is the active line of work started from [Mos09, MT10] that presented algorithmic proofs of Lovász's local lemma, leading to algorithms that efficiently find objects that previously where only shown to exist using Lovász's local lemma. The work of [IK10] also approaches measure concentration from an algorithmic perspective, but their goal is to algorithmically find witness for *lack* of concentration.

1.1.1 Extensions

In this work we also prove several extensions to our main result in different directions expanding a direct study of computational concentration as an independent direction.

Extension to random processes and coin-tossing attacks. We prove a more general result than Theorem 1.1 in which the perturbed object is a random process. Namely, suppose $\overline{\mathbf{w}} \equiv (\mathbf{w}_1, \ldots, \mathbf{w}_n)$ is a discrete (non-product) random process in which, given the history of blocks $w_1, \ldots, w_{i-1}$, the ith block w_i is sampled from its corresponding random variable $(\mathbf{w}_i \mid w_1, \ldots, w_{i-1})$. Suppose $\Pr_{\overline{w} \leftarrow \mathbf{w}}[\overline{w} \in \mathcal{S}] \geq \varepsilon$ for an arbitrary set $\mathcal{S}$. A natural question is: how much can an adversary increase the probability of falling into $\mathcal{S}$, if it is allowed to partially tamper with the online process of sampling $w_1, \ldots, w_n$ up to $K < n$ times? In other words, the adversary has a limited budget of K, and in the ith step, it can use one of its budget, and in exchange it gets to override the originally (honestly) sampled value $w_i \leftarrow (\mathbf{w}_i \mid w_1, \ldots, w_{i-1})$ by a new value. Note that if the adversary does a tampering, the changed value will *substitute* w_i and will affect the way the future blocks of the random process are sampled, e.g., in the next sampling of $w_{i+1} \leftarrow (\mathbf{w}_{i+1} \mid w_1, \ldots, w_i)$.

Our generalized version of Theorem 1.1 (stated in Theorem 3.2) shows that in the above setting of tampering with random processes, an adversary with budget $O(\sqrt{n \cdot \ln(1/\varepsilon\delta)})$ can indeed change the distribution of the random process and make the resulting tampered sequence end up in $\mathcal{S}$ with probability at least $1 - \delta$, while the adversary also runs in time $\text{poly}(n/\varepsilon\delta)$. Previously, [MM19] also showed a similar less tight result for random processes, but their result was limited to the setting that $\mathcal{S}$ is sufficiently large $\Pr[\mathcal{S}] \geq \omega(1/\sqrt{n})$.

The variant of Theorem 1.1 for random processes allows us to attack cryptographic coin-tossing protocols [BOL89, CI93, MPS10, BHT14, HO14, KKR18] in which n parties $P_1, \ldots, P_n$ each send a single message during a total of n rounds, and the full transcript $M = (m_1, \ldots, m_n)$ determines a bit b. The goal of an attacker is to corrupt up to K of the parties and bias the bit b towards its favor. Our results show that even if the original bit b had a small probability of being 1, $\Pr_{\text{no-attack}}[b = 1] \geq \varepsilon = 1/\text{poly}(n)$, then a $\text{poly}(n)$-time attacker who can corrupt up to $\widetilde{O}(\sqrt{n})$ parties and change their messages can bias the output bit b all the way up to make it $\Pr_{\text{attack}}[b = 1] \geq 1 - 1/\text{poly}(n)$. The corruption model here was first introduced by Goldwasser, Kalai and Park [GKP15] and is called *strong* adaptive corruption, because the adversary has the option to first see the message m_i before deciding to corrupt (or not corrupt) P_i to change its message m_i (or not). [3]

Weighted Hamming distance. In another extension to our Theorem 1.1 (see Theorem 3.2) we allow the Hamming distance to have different costs α_i when changing the ith coordinate for any vector $\overline{\alpha} = (\alpha_1, \ldots, \alpha_n)$ of ℓ_2 norm $\sum_i \alpha_i^2 = n$. In Talagrand's inequality [Tal95], it is proved that even if $\overline{\alpha}_{\overline{x}}$ can completely depend on the original point $\overline{x}$, we still can conclude that most points are "close" to any sufficiently large set $\mathcal{S}$, when the distance from $\overline{x}$ to $\mathcal{S}$ is measured by the $\overline{\alpha}_{\overline{x}}$-weighted Hamming distance. An algorithmic version of Talagrand's inequality, then, shall find a close point $\overline{y} \in \mathcal{S}$ to $\overline{x}$ measured by

---

[3] If each message m_i is a bit, it turns out that our attack can be modified to an attack that is not strong.

$\overline{\alpha}_{\overline{x}}$-weighed Hamming distance. Interestingly, our proof allows the coordinate α_i to completely depend on $(x_1, \ldots, x_{i-1})$, but falls short of proving an algorithmic version of Talagrand's inequality, if possible at all.

Reductions and other metric probability spaces. Motivated by proving computational concentration of measure in other metric probability spaces, as well as designing a machinery for this goal, we define a new model of *algorithmic reductions* between computational concentration of measure in different metric probability spaces. This notion, whose definition has some subtle algorithmic aspects, requires *two* (inverse) polynomial-time mappings one of which is an algorithmic Lipschitz mapping and the other one is an algorithmic coupling connecting the two distributions. As an application, we apply this notion of reduction to obtain computational concentration of measure for high-dimensional Gaussian distributions under the ℓ_1 distance. We prove this exemplary case by revisiting the proof of [B$^+$97] who proved the *information theoretic* reduction from the concentration of Gaussian distributions under the ℓ_1 distance to that of Hamming cube. We show how the core ideas of [B$^+$97] could be extended to obtain all the algorithmic components that are needed for a computational variant. Although there are known results on concentration of Gaussian distribution ℓ_1 in information theoretic regime, this is the first time (to the best of our knowledge) that a computational variant of concentration is proved for Gaussian spaces. We envision the same machinery can be applied to more information theoretic results for obtaining new computational variants; we leave doing so for future work. See Theorem 4.2 for the formal statement.

Computational concentration around mean. As measure concentration is usually proved for concentration around mean of a function $f(\cdot)$ when the inputs come from certain distributions, we show how to use our main result of Theorem 3.2 to obtain algorithmic concentration results for that setting as well. Namely, at a high level, we show that in certain settings (where concentration is known to follow from those settings) one can algorithmically find the right minimal perturbations to sampled points $\overline{x}$ so that the new perturbed point $\overline{x}'$ gives us the average of the concentrated function: $f(\overline{x}') \approx \mathbb{E}_{x \leftarrow \mu}[f(\overline{x})]$. Sometimes doing so is trivial (e.g., in case of Chernoff bound, when f is simply the addition of i.i.d. sampled Boolean values, as one can greedily change Boolean variables to decrease their summation) but sometimes doing so is not straightforward. In particular, we prove a computational variant of McDiarmid's inequality. Namely, we show how to modify $\sqrt{n}$ coordinates of a vector $\overline{x} \leftarrow \mu$ sampled from a product distribution μ of dimension n, such that $f(\overline{x}')$ gets arbitrary (i.e., $1/\operatorname{poly}(n)$) close to the average $\mu = \mathbb{E}_{\overline{x} \leftarrow \mu}[f(\overline{x})]$ for a function f that is Lipschitz under Hamming distance. (Note that the Lipschitz property is needed for the McDiarmid inequality as well). See Theorem 5.1 for the formal statement.

Lower bounds for simple methods. We also prove exponential lower bounds on the query complexity of natural, yet restricted, classes of algorithms. Two such classes stand out: One is non-adaptive algorithms where the queries made do not depend on the answer of previous queries. Another, natural class of algorithms are algorithms where all the queried points are at the distance where an acceptable final output may be at that distance. These lower bounds shed light on why perhaps some of the ideas behind our algorithm MUCIO are necessary, and that some simpler more straightforward algorithms are not as efficient.

Polynomial-time biasing attacks against extractors. At a high level, our biasing attacks on random processes are also related to impossibility results on extracting randomness from blockwise Santha-Vazirani sources [SV86, CG88, BEG17, RVW04, DOPS04] and specifically the p-tampering and p-resetting attacks of [BGZ16, MM17, MDM18]. In those attacks, an attacker might get to tamper each incoming block with an *independent* probability p, and they can achieve a bias of magnitude $O(p)$ (in polynomial time). However, our attackers *can choose* which blocks are the target of their tampering substitutions, but then achieve much stronger bias and almost fixing the output with much smaller $o(n)$ number of tamperings.

1.1.2   Polynomial-time Attacks on Robust Learning

Our results also have implications on (limits) of robust learning, which is also the focus of the work of [MM19] where computational concentration of measure was also studied. We refer the reader to [MM19] for a more in-depth treatment of the literature and settings for (attacks on) robust learning. For sake of completeness, below we describe the basic setting of such attacks and briefly discuss the implication of our computational concentration results to robust learning attacks.

Suppose L is a (deterministic) learning algorithm, taking as input a training set T consisting of m iid sampled and labeled examples $T = \{x_i, c(x_i)\}_{i \in [m]}$ where $x_i \leftarrow \mu$ for $i \in [m]$, and that $c(\cdot)$ is a concept function to be learned. Let $h = L(T)$ be the hypothesis that the learner produces based on the training set T. Main attacks against robustness of learners are studied during the training phase or the testing phase of a learning process. We describe the settings and previous work before explaining the implication of our new computational concentration results to those settings.

Poisoning attacks.   In a so-called data poisoning attack [BNS$^+$06, BNL12], which is tightly related to Valiant's malicious noise model [Val85, KL93, BEK02], the adversary only tampers with the training phase and substitutes a small $p < 1$ fraction of the examples in T with other arbitrary examples, leading to a poisoned data set $\widetilde{T}$. The goal of the adversary, in general, is to make $L(\widetilde{T})$ produce a "bad" hypothesis $h \in \widetilde{H}$ (e.g., bad might mean having large risk or making a mistake on a particular test x during the test time) where $\widetilde{H} \subseteq H$ includes the set of all undesired hypothesis. It was shown by [MDM19] that the concentration of measure in product spaces (under Hamming distance) implies that in any such learning process, so long as $\Pr_T[L(T) \in \widetilde{H}] \geq \varepsilon$, then an adversary A who changes $O(\sqrt{m \cdot \ln(1/\varepsilon\delta)})$ of the training examples (and substitute them with still correctly labeled data) can increase the probability of producing a bad hypothesis in $\widetilde{H}$ to $\Pr_{\widetilde{T} \leftarrow A(T)}[L(\widetilde{T}) \in \widetilde{H}] \geq \delta$. It was left open whether such attack can be made polynomial time, or that perhaps computational intractability can be leveraged to prevent such attacks. The work of [MM19] showed how to make such attacks polynomial time, only for the setting where the probability of falling into $\widetilde{H}$ was already not too small, and in particular at least $\omega(1/\sqrt{n})$, and also with looser bounds. Our Theorem 1.1 shows how to get such polynomial time evasion attacks for *any non-negligible* probability $\varepsilon \geq 1/\operatorname{poly}(n)$. In fact, as stated in Theorem 1.1, our attack's complexity can gracefully adapt to ε.

The previous attacks of [MDM19, MM19] and our newer attacks of this work do not contradict recent exciting works in defending against poisoning attacks [DKK$^+$16, LRV16, DKK$^+$18, PSBR18], as those defenses either focus on learning parameters of distributions or, even in the classification setting, they aim to bound the *risk* of the hypothesis, while we increase the probability of a bad Boolean property.[4]

Evasion attacks.   In another active line of work, other types of attacks on learners are studied in which the adversary enters the game during the test time. In such so-called *evasion* attacks [BFR14, CW17, SZS$^+$14, GMP18] that find "adversarial examples", the goal of the adversary is to perturb the test input x into a "close" input $\widetilde{x}$ under some metric d (perhaps because this small perturbation is imperceptible to humans) in such a way that this tampering makes the hypothesis h make a mistake. In [MDM19], it was also shown that the concentration of measure can potentially lead to inherent evasion attacks, as long as the input metric probability space $(\mathcal{X}, d, \mu)$ is concentrated. This holds e.g., if the space is a Normal Lévy family [Lév51, AM85]. The work of [MM19] showed the existence of polynomial time evasion attacks with sublinear perturbations for classification tasks in which the input distribution is a n-dimensional product space (e.g.,

---

[4]In fact, the challenge in those works is to obtain polynomial-time *learners* in settings where inefficient robust methods were perhaps known in the robust statistics literature. The focus here, however, is to obtain polynomial-time *attacks*.

the uniform distribution over the hypercube) under Hamming distance. But their attacks could be applied only when the original risk ε of the hypothesis h is at least $\varepsilon = \omega(1/\sqrt{n})$. However, standard PAC learners (e.g., based on empirical risk minimization) can indeed achieve polynomially small risk $\varepsilon = 1/\operatorname{poly}(m)$ where m is the sample complexity. Our Theorem 1.1 shows how to obtain polynomial-time attacks even in the low-risk regime $\varepsilon = 1/\operatorname{poly}(n)$[5] and perturb given samples $x \leftarrow \boldsymbol{\mu}$ in $\widetilde{O}(\sqrt{n})$ coordinates and make the perturbed adversarial instance $\widetilde{x}$ misclassified with high probability.

Our results of Section 4 show that one can also obtain polynomial time evasion attacks for classifiers whose inputs come from metric probability spaces that use metrics other than Hamming distance (e.g., Gaussian under ℓ_1). Using the reductionist approach of Section 4 one can perhaps obtain more such results. Our attacks, however, do not rule out the possibility of robust classifiers for specific input distributions such as images or voice that is the subject of recent intense research [SZS+14, CW17, MFF16], but they shed light on barriers for robustness in theoretically natural settings. See [BPR18, DV19] for more discussion on other possible barriers for robust learning.

1.2 Technical Overview

In this subsection, we describe the challenges and key ideas behind the proof of Theorem 1.1 and some of its extensions. The extension for the concentration around mean (see Section 5) follows directly from the main result about concentration around noticeably large sets. Thus, we only focus on explaining ideas behind some other extensions to our result; namely how to obtain new results through carefully defined algorithmic reductions, and proving limits for the power of simple methods for proving computational concentration.

Setting. (The reader might find the explanations for our notation at the beginning of Section 2 useful.) Suppose $\overline{\mathbf{w}} \equiv (\mathbf{w}_1 \times \cdots \times \mathbf{w}_n)$ is a random variable with a product distribution of dimension n.[6] Also, suppose the set $\mathcal{S} \subseteq \operatorname{Supp}(\mathbf{w})$ is denoted by its characteristic function f, where $f(\overline{w}) = 1$ iff $\overline{w} \in \mathcal{S}$. The goal of the tampering algorithm Tam is to change as few as possible of the sampled blocks $\overline{w} = (w_1, \ldots, w_n) \leftarrow \overline{\mathbf{w}}$ making the new vector $\overline{v} = (v_1, \ldots, v_n)$ such that $f(\overline{v}) = 1$ with high probability (over the both steps of sampling $\overline{w}$ and obtaining $\overline{v}$ from it).

Our starting point is the previous attack of [MM19] that only proved computational concentration around large sets of measure $\Pr[\mathcal{S}] \geq \omega(1/\sqrt{n})$. The result of [MM19], in turn, was built upon techniques developed in the work of Komargodski, Raz, and Kalai [KKR18] that presented an alternative simpler proof for a previously known result of Lichtenstein et al. [LLS89]. Below, we first describe the high level ideas behind the approach of [MM19, KKR18], and then we describe why that approach breaks down when $\mathcal{S}$ gets smaller than $1/\sqrt{n}$, and thus fails to obtain the optimal information theoretic bounds for concentration. We then describe our new techniques to bypass this challenge and obtain computational concentration with optimal bounds.

The high-level approach of [MM19]. As it turns out, the tampering algorithm of [MM19], as well as ours, do not need to know $w_{i+1}, \ldots, w_n$ when deciding to change w_i (into a different $v_i \neq w_i$) or leaving it as is (i.e., $w_i = v_i$). So, a useful notation to use is the partial expected values, capturing the chance of

---

[5]Note that in the "high dimensional" setting where input dimension n is huge, we can see the sample complexity m bounded, which implies $\varepsilon \geq 1/\operatorname{poly}(m)$ if $\varepsilon = 1/\operatorname{poly}(n)$.

[6]As discussed above, our results extend to random processes as well, when formalized carefully, but for simplicity we focus on the interesting special case of product distributions.

falling into $\mathcal{S}$ (i.e., $f(\overline{w}) = 1$) over the randomness of the remaining blocks.

$$\hat{f}(w_1, \ldots, w_i) = \underset{(w_{i+1}, \ldots, w_n) \leftarrow (\mathbf{w}_{i+1}, \ldots, \mathbf{w}_n)}{\mathbb{E}} [f(w_1, \ldots, w_n)].$$

One obvious reason for working with $\hat{f}(\cdot)$ quantities is that they can be *approximated* with arbitrary small $\pm 1/\operatorname{poly}(n)$ additive error. This can be done using the sampling oracle of the distribution of $\overline{\mathbf{w}} \equiv \mathbf{w}_1 \times \cdots \times \mathbf{w}_n$ and the oracle $f(\cdot)$ determining membership in $\mathcal{S}$.

At a high level, the idea behind the attack of [MM19] is to change w_i only if this change allows us to increase $\hat{f}(\cdot)$ *additively* by $+\lambda$ for a parameter $\lambda \approx 1/\sqrt{n}$. We first describe this attack, and then explain its challenges against obtaining optimal bounds and how we resolve them.

At a high level, the attack of [MM19] tampers with the ith block (i.e., w_i), if just before or just after looking at w_i, we conclude that we can increase $\hat{f}(\cdot)$ by λ.

Construction 1.2 (Attack of [MM19] oracle $\hat{f}(\cdot)$)**.** Suppose that we are given a prefix $v_{\leq i-1}$ that is finalized, and we are also given a candidate value w_i for the i'th block (supposedly sampled from $\mathbf{w}_i$) and we want to decide to keep it $v_i = w_i$ or change it $v_i \neq w_i$. Let $\lambda > 0$ be a parameter of the attack to be chosen later, $v_i^* = \operatorname{argmax}_{y_i} \hat{f}(v_{\leq i-1}, y_i)$ be the choice for i'th block that maximizes $\hat{f}(v_{\leq i})$, and let $f^* = \hat{f}(v_{\leq i-1}, v_i^*)$.

1. (Case 1) If $f^* \geq \hat{f}(v_{\leq i-1}) + \lambda$, then output $v_i = v_i^*$ (regardless of w_i).

2. (Case 2) Otherwise, if (by looking at w_i) $\hat{f}(v_{\leq i-1}, w_i) \leq \hat{f}(v_{\leq i-1}) - \lambda$, then again output $v_i = v_i^*$.

3. (Case 3) Otherwise, keep the value w_i and output $v_i = w_i$.

Why this attack biases $f(\cdot)$ towards 1? For simplicity, support $\Pr[\mathcal{S}] = 1/2$. Suppose we "color" different $i \in [n]$ depending on whether the tampering algorithm changes the ith block w_i or not. If $v_i \neq w_i$ (tampering happened), color i green, denoted by $i \in G$, and otherwise color i red, denoted as $i \in R = [n] \setminus G$. A simple yet extremely useful observation is that we can write $f(\overline{v})$ as the sum of the *changes* in $\hat{f}(v_{\leq i})$ between consecutive i. Namely, if we let $\hat{g}(v_{\leq i}) = \hat{f}(v_{\leq i}) - \hat{f}(v_{\leq i-1})$, then

$$\hat{f}(v_{\leq n}) - \hat{f}(\varnothing) = f(\overline{v}) - 1/2 = \sum_{i \in [n]} \hat{g}(v_{\leq i}).$$

This means that we have to study the affect of the green and red coordinates i on how $\hat{g}(v_{\leq i})$ behaves, because that will tell us how the final output bit is determined and distributed.

Construction 1.2 is designed so that, whenever i is green, the partial expectation oracle $\hat{f}(v_{\leq i})$ jumps up at least by λ (i.e., $\hat{g}(v_{\leq i}) \geq \lambda$). So, the only damage (leading to falling outside $\mathcal{S}$) could come from the red coordinates and how they change $\hat{f}(v_{\leq i})$ downwards. Let us now focus on the red coordinates $i \in R$. A simple inspection of Construction 1.2 shows that, the change in $\hat{f}(\cdot)$ captured by $\hat{g}(v_{\leq i})$ is bounded in absolute value by λ, and that is the result of no-tampering for a block. Therefore, the summation of $\hat{g}(v_{\leq i})$ for red coordinates i would cancel out each other and, by the Azuma inequality, the probability that this summation is more than 1 is at most $\exp(-1/(n \cdot \lambda^2))$. So, by choosing $\lambda \ll 1/\sqrt{n}$, the red coordinates cannot control the final bit, as with high probability this summation is less than one. This means that the outcome (whenever the red coordinates do not fix the function) should be 1, because the green coordinates only increase the $\hat{f}(\cdot)$ function.

Why the attack is efficient? The efficiency of the attack follows form its effectiveness and the same argument described above. Namely, whenever the green coordinates are determining the output, it means that their total sum of of $\hat{g}(v_{\leq i})$ is going from a specific number in $[0, +1]$ to 1, and each time they jump up by at least λ, so they cannot be more than n/λ green steps. Since we chose $\lambda = 1/\sqrt{n}$, the efficiency follows as well.

The challenge when $\Pr[\mathcal{S}] = \mathbb{E}[\overline{w}] = \varepsilon$ **is too small.** The issue with the above approach is that whenever ε is too small (not around $1/2$) we need to pick λ much smaller, so that the summation (i.e., the effect of the red coordinates does not make the function reach zero). Simple calculation shows that after the threshold $\varepsilon \approx 1/\sqrt{n}$, the number of tampered (green) blocks would grow too much and eventually become *more* than n. However, note that when we reach n tamperings, it means the attack's efficiency is meaningless.

1.2.1 Our Approach (MUCIO: MUltiplicative Conditional Influence Optimizer)

Main step 1: tampering with *multiplicatively* influential blocks. Our first key idea is to judge whether a block is influential (and thus tamper it) based how much it can change the partial expectations in a *multiplicative* way. (This is related to the notion of a *log-likelihood ratio* in statistics and information theory.) Construction 1.3 below describes this simple change. However, as we will see, doing this simple change will have big advantages as well as new challenges to be resolved. We will describe both the advantages and thew new challenges after the construction.

Construction 1.3 (*Multiplicative* online tampering using oracle $\hat{f}(\cdot)$)**.** The key difference between this attack and that of Construction 1.2 is that here, in order to judge whether tampering with the current i^{th} block is worth it or not, we make the decision based on the *multiplicative* gain (in how $\hat{f}(\cdot)$ changes) that this would give us. Namely, for the same setting of Construction 1.2, we do as follows.

1. (Case 1) If $f^* \geq e^{\lambda} \cdot \hat{f}(v_{\leq i-1})$, then output $v_i = v_i^*$ (regardless of w_i).

2. (Case 2) Otherwise, if $\hat{f}(v_{\leq i-1}, w_i) \leq e^{-\lambda} \cdot \hat{f}(v_{\leq i-1})$, then output $v_i = v_i^*$.

3. (Case 3) Otherwise, keep the value w_i and output $v_i = w_i$.

Main advantage: the output is fully biased. We first describe what advantages the above change gives us, and then will discuss the remaining challenges. The key insight into why this is a better approach is that the tampering algorithm of Construction 1.3 will *always* lead to obtaining $f(\overline{v}) = 1$ at the end (i.e., we always end up in $\mathcal{S}$). In order to see why this is a big difference, notice that if $\hat{f}(w_{\leq 0}) = \varepsilon$ is very small at the beginning and we tamper only based on additive differences (as is done in Construction 1.2), there is a possibility that we do *not* tamper with the first block and end up at $\hat{f}(w_{\leq 1}) = 0$. Such a problem does not happen when we decide on tampering based on multiplicative improvement, and every tiny chance of falling into $\mathcal{S}$ is taken advantage of.

Only few tamperings happen. To analyze the number of tamperings that occur in the "idealized" attack of Construction 1.3 we keep track of $\ln\left(\hat{f}(v_{\leq i})/\hat{f}(v_{\leq i-1})\right)$ as we go. We know that the output of function under the attack is always 1 which means:

$$\sum_{i=1}^{n} \ln\left(\frac{\hat{f}(v_{\leq i})}{\hat{f}(v_{\leq i-1})}\right) = \ln\left(\frac{\hat{f}(v_{\leq n})}{\hat{f}(\varnothing)}\right) = \ln\left(\frac{1}{\hat{f}(\varnothing)}\right).$$

We again categorize the indices i to red and green. Green set indicates the locations that the algorithm tampers with w_i and red is the set of locations that tampering has not happened and $v_i = w_i$. For the red locations, we prove the following inequality that plays a key role in our analysis of the attack. One interpretation of this inequality is that we will now use $\ln(1/\hat{f}(v_{\leq i-1}))$ as a potential function that allows us keep track of, and control, the number of tamperings.

$$\ln(1/\hat{f}(v_{\leq i-1})) - \mathbb{E}_{v_i \leftarrow \mathbf{v}[v_{\leq i-1}]}[\ln(1/\hat{f}(v_{\leq i}))] \geq -\frac{\lambda^2}{2}.$$

7

This inequality follows from a Jensen Gap inequality on the natural logarithm function. For green locations, we increase $\ln(\hat{f}(v_{\leq i}))$ whenever we tamper by at least λ. Therefore, the overall effect of green locations on $\sum_{i=1}^{n} \ln(\hat{f}(v_{\leq i})/\hat{f}(v_{\leq i-1}))$ will be

$$\lambda \cdot \mathbb{E}[\# \text{ of tampering}].$$

Combining these together we get the following:

$$\lambda \cdot \mathbb{E}[\# \text{ of tampering}] - \frac{n \cdot \lambda^2}{2} \leq \ln(1/\varepsilon).$$

Now we can optimize λ to get the best inequality on the expected number of tampering.

New challenge: obtaining good multiplicative approximations when $\hat{f}(\cdot)$ gets too small. Construction 1.3 increases the average to 1 (i.e., we always end up in $\mathcal{S}$) with small number of tampering. However we cannot implement that construction in polynomial time. The problem is that it is hard to instantiate the oracle $\hat{f}(v_{\leq i})$ polynomial time when the partial average gets close to 0. To solve this issue, we add a step to the construction that makes the algorithm abort if the partial average goes below some threshold.

Construction 1.4 (Online tampering *with abort* TamAb using partial-expectations oracle)**.** This construction is identical to Construction 1.3, except that whenever the fixed prefix has a too small partial expectation $\hat{f}(v_{\leq i-1}, w_i)$ (based on a new parameter τ) we will abort. Also, in that case the tampering algorithm does not tamper with any future v_i block either. Namely, we add the following "Case 0" to the previous steps:

- (Case 0) If $\hat{f}(v_{\leq i-1}, w_i) \leq e^{-\tau} \cdot \varepsilon$ abort. If had aborted before, do nothing.

Main step 2: showing that reaching low expectations is unlikely under the attack. To argue that the new construction does not hurt the performance of our algorithm by much, we show that the probability of getting a low $\hat{f}(v_{\leq i})$ is small because of the way our algorithm works. The idea is that, our algorithm always guarantees that

$$-\lambda \leq \ln\left(\hat{f}(v_{\leq i})/\hat{f}(v_{\leq i-1})\right) \leq \lambda.$$

We also show that

$$\mathbb{E}[\ln(\hat{f}(v_{\leq i})/\hat{f}(v_{\leq i-1}))] \geq -\frac{\lambda^2}{2}.$$

This means that the sequence of $\ln\left(\frac{\hat{f}(v_{\leq i})}{\hat{f}(v_{\leq i-1})}\right)$ forms an "approximate" sub-martingale difference sequence. We can use Azuma inequality to show that sum of this sequence will remain bigger than some small threshold, with high probability. After all, we can bound the probability of getting into Case 0 to be very small.

1.2.2 More Computational Concentration Results through Algorithmic Reductions

Here we explain a technical overview of our generic reduction technique. Let $S_1 = (\mathcal{X}_1, \mathsf{d}_1, \mu_1)$ and $S_2 = (\mathcal{X}_2, \mathsf{d}_2, \mu_2)$ be two metric probability spaces. In addition, assume we already know some level of computational concentration proved for S_2, and that we want to prove (some level of) computational concentration for S_1 through a reduction. In Section 4, we formalize a generic framework to prove such reductions. The main ingredients of such algorithmic reduction are two polynomial time mappings $\mathbf{f} \colon \mathcal{X}_1 \to \mathcal{X}_2$ and $\mathbf{g} \colon \mathcal{X}_2 \to \mathcal{X}_1$ with 3 properties. The first property (roughly speaking) requires that $\mathbf{f}(\mu_1) \approx \mu_2$ and $\mathbf{g}(\mu_2) \approx \mu_1$. This property guarantees that if we sample a point from one space and use the mapping and

go to the other space, we get a distribution close to the probability measure of the second space. (This can be interpreted as an algorithmic coupling.) The second property requires that the mapping $\mathbf{g}$ is Lipschitz. The third property requires that $\mathbf{g}(\mathbf{f}(x))$ is close to x. The idea behind why such reduction (as a collection of these mappings) work is as follows. We are given a point x_1 in S_1 and we want to find a close x_2 such that x_2 falls inside a subset $\mathcal{S}$. To do that we first map x_1 to a point x_1' in S_2 using $\mathbf{f}$. We know that S_2 is computationally concentrated and we can efficiently find a close x_2' such that x_2' falls into an specific subset $\mathcal{S}'$. Then we use $\mathbf{g}$ to go back to a point x_2 in S_1. The second and third properties together guarantee that x_1 and x_2 are close, because x_1' and x_2' are close. At the same time, the first condition guarantees that x_2 will hit $\mathcal{S}$ if we select $\mathcal{S}'$ in a careful way. See Theorem 4.2 for more details.

We use this general framework to prove computational concentration bounds for Gaussian spaces under ℓ_1 norm. We reduce the computational concentration of Gaussian distribution under ℓ_1 to the computational concentration of the Boolean Hamming cube. For this goal, we show how to build two mappings $\mathbf{f}$ and $\mathbf{g}$ from an n-dimensional Gaussian space to a n^2-dimensional Hamming cube and vice versa, following the footsteps of a reduction by [B$^+$97] who proved an information theoretic variant of this result. Here we show that the algorithmic ingredients that are necessary, in addition to the ideas already in [B$^+$97], could indeed be obtained. The main idea behind this mappings is the fact that the number of 1's in a sample from n-dimensional hamming cube approximately forms a Gaussian distribution centered around $\frac{n}{2}$. Therefore, we can map each dimension of the Gauss space to a n-dimensional hamming cube and vice versa. Here we observe that we can use the same idea and build the mappings in a way that achieves the three properties mentioned above. See Section 4 for more details.

1.2.3 Lower Bounds for Simple Methods

To prove exponential lower bounds on the query complexity of too-simple algorithms, we consider the half-space $\mathcal{S}$ in the Hamming cube consisting of those points with below-average Hamming weight.

A uniformly random point $\overline{x}$ in the cube, with high probability has Hamming distance $\Omega(\sqrt{n})$ from the set $\mathcal{S}$. Now, if for such a point $\overline{x}$, we hope to find a close point in $\mathcal{S}$ simply by sampling uniformly at random among points close to x, we fail except with exponentially small probability. For only random points with distance $n^{1-o(1)}$ have a significant chance of changing the weight of point $\overline{x}$ by $\Omega(\sqrt{n})$, whereas the information-theoretic bound says there exists a point of distance $O(\sqrt{n})$ that changes the weight by $\Omega(\sqrt{n})$.

To achieve lower bounds for more general classes of algorithms, we use a random half-space instead of a fixed half-space. This gives us exponential lower bounds for non-adaptive attacks as well as attacks that query about $\mathcal{S}$-membership of points outside a ball of size $d = O(\sqrt{n \cdot \ln(1/\varepsilon\delta)})$ even when we are interested in finding a point in the intersection of $\mathcal{S}$ and this ball. Notice that MUCIO avoids this last restriction by surveying the influence of the first coordinate on the totality of points, while it ends up changing only a small fraction of the coordinates.

2 Preliminaries

General notation.   We use calligraphic letters (e.g., $\mathcal{X}$) for sets. By default, all distributions and random variables in this work are discrete. We use bold letters (e.g., $\mathbf{w}$) to denote random variables that return a sample from a corresponding discrete distribution. By $w \leftarrow \mathbf{w}$ we denote sampling w from the random variable $\mathbf{w}$. By $\mathrm{Supp}(\mathbf{w})$ we denote the support set of $\mathbf{u}$. For an event $\mathcal{S} \subseteq \mathrm{Supp}(\mathbf{w})$, the probability function of $\mathbf{w}$ for $\mathcal{S}$ is denoted as $\Pr[\mathbf{w} \in \mathcal{S}] = \Pr_{w \leftarrow \mathbf{w}}[w \in \mathcal{S}]$. For a randomized algorithm $R(\cdot)$, by

$y \leftarrow R(x)$ we denote the randomized execution of R on input x outputting y. By $\mathbf{u} \equiv \mathbf{v}$ we denote that the random variables $\mathbf{u}$ and $\mathbf{v}$ have the same marginal distributions. Unless stated otherwise, we denote vectors by using a bar over a variable. By $\overline{\mathbf{w}} \equiv (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n)$ we refer to a sequence of n *jointly sampled* random variables. For a vector $\overline{w} = (w_1 \ldots w_n)$, we use $w_{\leq i}$ to denote the prefix $(w_1, \ldots, w_i)$, and we use the same notation $\mathbf{w}_{\leq i}$ for jointly distributed random variables. For a jointly distributed random variables $(\mathbf{u}, \mathbf{v})$, by $(\mathbf{u} \mid v)$ we denote the conditional distribution $(\mathbf{u} \mid \mathbf{v} = v)$. For a random variable $\mathbf{u}$, by $T^{\mathbf{u}}(\cdot)$ we denote an oracle-aided algorithm $T^{(\cdot)}(\cdot)$ that can query fresh sample from $\mathbf{u}$. By $\mathbf{u} \times \mathbf{v}$ we refer to the product distribution in which $\mathbf{u}$ and $\mathbf{v}$ are sampled independently. For a real-valued random variable $\mathbf{x}$, by $\mathbb{E}[\mathbf{x}]$ we refer to the expected value of $\mathbf{x}$, and by $\mathbb{V}[\mathbf{x}]$ we denote its variance.

Notation on random processes and online samplers. Let $\overline{\mathbf{w}} \equiv (\mathbf{w}_1, \ldots, \mathbf{w}_n)$ be a sequence of jointly distributed random variables. We can interpret the distribution of $\overline{\mathbf{w}}$ as a random process in which the ith block w_i is sampled from the marginal distribution $(\mathbf{w}_i \mid w_{\leq i-1})$. For simplicity, we use notation $\mathbf{w}[w_{\leq i-1}] \equiv (\mathbf{w}_i \mid w_{\leq i-1})$ to refer to this marginal conditional distribution. (Note that i is dropped from the distribution's name, relying on the input $w_{\leq i-1}$ that uniquely determines i.) We can interpret $w_{\leq i-1}$ as a "node" in a tree of depth i, and the sampling $w_i \leftarrow \mathbf{w}[w_{\leq i-1}]$ can be seen as the process of sampling the next child according to the distribution of $\mathbf{w}[w_{\leq i-1}]$. Alternatively, describing the distributions of the random variables $\mathbf{w}[w_{\leq i-1}]$ defines the distribution of $\overline{\mathbf{w}}$. For random variable $\overline{\mathbf{w}} \equiv (\mathbf{w}_1, \ldots, \mathbf{w}_n)$ we sometimes refer to the random variable $\mathbf{w}[w_{\leq i-1}]$ as the *online* sampler for $\overline{\mathbf{w}}$, because it returns fresh samples form the next block, given the previously fixed prefix $w_{\leq i-1}$.

Definition 2.1 (Online tampering). Let $\overline{\mathbf{w}} \equiv (\mathbf{w}_1, \ldots, \mathbf{w}_n)$ be a sequence of jointly distributed random variables, and let $\mathbf{w}[w_{\leq i-1}]$ be the online sampler for $\overline{\mathbf{w}}$ for all $i \in [n]$ and all $w_{\leq i-1} \in \mathrm{Supp}(\mathbf{w}_{\leq i-1})$. Online tampering algorithms for $\overline{\mathbf{w}}$ and their properties are defined as follows.

- **Online tampering.** We call a (potentially randomized and computationally unbounded) algorithm Tam an *online tampering* algorithm for $\overline{\mathbf{w}}$, if for all $i \in [n]$ and $w_{\leq i} \in \mathrm{Supp}(\mathbf{w}_{\leq i})$, it holds that

$$\Pr[\mathsf{Tam}(w_{\leq i}) \in \mathrm{Supp}(\mathbf{w}[w_{\leq i-1}])] = 1 .$$

  Namely, $\mathsf{Tam}(w_{\leq i})$ always outputs a candidate i^{th} block that still falls into $\mathrm{Supp}(\mathbf{w}[w_{\leq i-1}])$.

- **Resulting tampered distribution.** For an online tampering algorithm Tam for $\overline{\mathbf{w}}$, by $(\overline{\mathbf{u}}, \overline{\mathbf{v}}) \equiv \langle \overline{\mathbf{w}} \parallel \mathsf{Tam} \rangle$ we refer to the jointly distributed sequence of random varaibles defined as follows. For $i = 1, 2, \ldots, n$, we first sample $u_i \leftarrow \mathbf{w}[v_{\leq i-1}]$, and then we obtain $v_i \leftarrow \mathsf{Tam}(v_{\leq i-1}, u_i)$ as the (possibly different than u_i) choice of the tampering algorithm Tam for the ith block (that will override u_i). At the end, we output the pair of sequences $(\overline{u} = u_{\leq n}, \overline{v} = v_{\leq n})$ as the sample from $(\overline{\mathbf{u}}, \overline{\mathbf{v}})$.

  Notation. For simplicity, we use $\mathbf{v}[v_{\leq i-1}]$ to denote $(\mathbf{v}_i \mid v_{\leq i-1})$ and use $(\mathbf{w}, \mathbf{v})[v_{\leq i-1}]$ to denote the *jointly* distributed random variables from which (u_i, v_i) are sampled conditioned on the prefix $v_{\leq i-1}$. The notation allows us to use $\mathbf{v}[v_{\leq i-1}], (\mathbf{w}, \mathbf{v})[v_{\leq i-1}]$ similarly to how we use online samplers.[7]

- **Budget of tampering attacks.** Let d be a metric defined over $\mathrm{Supp}(\overline{\mathbf{u}})$ as vectors of dimension n. We say a tampering algorithm Tam has *budget* (at most) b, if

$$\Pr_{(\overline{u}, \overline{v}) \leftarrow \langle \overline{\mathbf{w}} \parallel \mathsf{Tam} \rangle}[\mathsf{d}(\overline{u}, \overline{v}) \leq b] = 1.$$

---

[7]Note that are *not* defining a similar notation of the form $\mathbf{u}[v_{\leq i-1}]$ for $\overline{\mathbf{u}}$. Firstly, this is not needed as $\mathbf{w}[v_{\leq i-1}]$ already provides a sampler for u_i. Moreover, such notation would be inconsistent with our notation for online samplers for random processes based on joint distributions, because the notation would implicitly interpret $v_{\leq i-1}$ as previous samples from $\mathbf{u}_{\leq i-1}$.

We say that Tam has *average budget* (at most) b, if the following weaker condition holds

$$\mathop{\mathbb{E}}_{(\overline{u},\overline{v})\leftarrow\langle\overline{\mathbf{w}}\,\|\,\mathsf{Tam}\rangle}[\mathsf{d}(\overline{u},\overline{v})] \le b.$$

- **Algorithmic efficiency of attacks.** If $\overline{\mathbf{w}} = \overline{\mathbf{w}}_n$ is a member from a *family* defined for all $n \in \mathbb{N}$, we call an online or offline tampering algorithm *efficient*, if its running time is $\mathrm{poly}(N)$ where N is the total bit-length representation of any $\overline{w} \in \mathrm{Supp}(\overline{\mathbf{w}}_n)$.

Definition 2.2 (Partial expectations)**.** Suppose $f\colon \mathrm{Supp}(\overline{\mathbf{w}}) \mapsto \mathbb{R}$ for $\overline{\mathbf{w}} \equiv (\mathbf{w}_1,\ldots,\mathbf{w}_n)$, $i \in [n]$, and $w_{\le i} \in \mathrm{Supp}(\mathbf{w}_{\le i})$. Then (using a small hat) we define the notation $\hat{f}(w_{\le i}) = \mathbb{E}_{\overline{w}\leftarrow(\overline{\mathbf{w}}|w_{\le i})}[f(\overline{w})]$ to define the expected value of f for a sample from $\overline{\mathbf{w}}$ given the prefix $w_{\le i}$. In particular, for $\overline{w} = w_{\le n}$, we have $\hat{f}(\overline{w}) = f(\overline{w})$, and also $\hat{f}(\varnothing) = \mathbb{E}[f(\overline{\mathbf{w}})]$.

Lemma 2.3 (Hoeffding's lemma)**.** *Let $\mathbf{x}$ be a random variable such that* $\Pr[a \le \mathbf{x} \le b] = 1$ *and* $\mathbb{E}[\mathbf{x}] = 0$. *Then, it holds that* $\mathbb{E}[e^{\mathbf{x}}] \le e^{(b-a)^2/8}$.

Lemma 2.4. *Let $\mathbf{x}$ be a random variable where* $\Pr\left[e^{-\lambda} \le \mathbf{x}\right] = 1$ *and* $\Pr\left[\mathbf{x} \le e^{\lambda}\right] \ge 1 - \delta$ *and* $\Pr\left[\mathbf{x} \le c\right] = 1$. *Then,* $\mathbb{E}\left[\ln(\mathbf{x})\right] \ge \ln(\mathbb{E}\left[\mathbf{x}\right] - \delta \cdot c) - \lambda^2/2$.

Proof. Let $\mathbb{E}\left[\min(\ln(\mathbf{x}),\lambda)\right] = s$. Consider a random variable $\mathbf{y} \equiv \min(\ln(\mathbf{x}),\lambda) - s$. We have $\mathbb{E}[\mathbf{y}] = 0$ and $-\lambda - s \le \mathbf{y} \le \lambda - s$. Therefore, by Lemma 2.3 we have

$$\mathbb{E}\left[e^{\mathbf{y}}\right] \le e^{\lambda^2/2}.$$

On the other hand, we have $\mathbb{E}\left[e^{\mathbf{y}}\right] = \mathbb{E}\left[e^{\min(\ln(\mathbf{x}),\lambda)-s}\right] = \mathbb{E}\left[\min\left(\mathbf{x},e^{\lambda}\right)\right] \cdot e^{-s}$. Thus, we have $\mathbb{E}\left[\min\left(\mathbf{x},e^{\lambda}\right)\right] \cdot e^{-s} \le e^{\lambda^2/2}$ which implies $e^{-s} \le e^{\lambda^2/2 - \ln(\mathbb{E}\left[\min\left(\mathbf{x},e^{\lambda}\right)\right])}$, and so $s \ge \ln\left(\mathbb{E}\left[\min\left(\mathbf{x},e^{\lambda}\right)\right]\right) - \lambda^2/2$. Therefore we have, $s \ge \ln\left(\mathbb{E}\left[\mathbf{x}\right] - \delta \cdot c\right) - \lambda^2/2$. $\qquad\square$

Lemma 2.5. *Let $\mathbf{x}$ be a random variable where* $\Pr[e^{-\lambda} \le \mathbf{x}] = 1$ *and* $\Pr[\mathbf{x} \le e^{\lambda}] \ge 1 - \delta$ *and* $\Pr[\mathbf{x} \le c] = 1$. *Then,* $\mathbb{E}[1/\mathbf{x}] \le \frac{e^{\lambda^2}}{\mathbb{E}[\mathbf{x}]-\delta\cdot c}$.

Proof. Let $\mathbb{E}[\min(\ln(\mathbf{x}),\lambda)] = s$. Consider a random variable $\mathbf{y} = \min(\ln(\mathbf{x}),\lambda) - s$. Similar to proof of Lemma 2.4 we have $s \ge \ln(\mathbb{E}[\mathbf{x}] - \delta \cdot c) - \lambda^2/2$. Now consider another random variable $\mathbf{y}' \equiv -\mathbf{y}$. Again by using Hoeffding Lemma we have $\mathbb{E}[e^{\mathbf{y}'}] \le e^{\lambda^2/2}$ which means

$$\mathbb{E}[e^{-\min(\ln(\mathbf{x}),\lambda)}] \cdot e^s \le e^{\lambda^2/2}$$

which implies

$$\mathbb{E}[\max(1/\mathbf{x},e^{-\lambda})] \le e^{\lambda^2/2} \cdot e^{-s} \le \frac{e^{\lambda^2}}{\mathbb{E}[\mathbf{x}]-\delta\cdot c}.$$

$\qquad\square$

The following lemma is implied by Theorem 3.13 from [MHRAR98].

Lemma 2.6 (Azuma's inequality for sub-martingales)**.** *Let $\overline{\mathbf{t}} \equiv (\mathbf{t}_1,\ldots,\mathbf{t}_n)$ be a sequence of n jointly distributed random variables such that for all $i \in [n]$, $\Pr[|\mathbf{t}_i| \le c_i] \ge 1 - \xi$, for all $t_{\le i-1} \leftarrow \mathbf{t}_{\le i-1}$, and that $\mathbb{E}[\mathbf{t}_i \mid t_{\le i-1}] \ge -\gamma_i$. If $\gamma = \sum_{i=1}^{n} \gamma_i$, then we have*

$$\Pr\left[\sum_{i=1}^{n} \mathbf{t}_i \le -s\right] \le e^{\frac{-(s-\gamma)^2}{2\sum_{i=1}^{n} c_i^2}} + n \cdot \xi.$$

11

3 Optimal Computational Concentration for Hamming Distance

In this section, we formally state and prove our main result, which is the computational concentration of measure in any product space under Hamming distance.

Definition 3.1 (Weighted Hamming Distance). For $\overline{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{R}_+^n$, the $\overline{\alpha}$-weighted Hamming distance between vectors of dimension n is denoted by $\mathsf{HD}_{\overline{\alpha}}(\cdot, \cdot)$ and is defined as

$$\mathsf{HD}_{\overline{\alpha}}(\overline{u}, \overline{v}) = \sum_{i \in [n], u_i \neq v_i} \alpha_i.$$

Theorem 3.2. *Let $(\alpha_1, \ldots, \alpha_n) \in \mathbb{R}^n$ be such that $\sum_{i=1}^n \alpha_i^2 = n$. Then, there is a (uniform) oracle-aided randomized algorithm Tam such that the following holds. Suppose $f \colon \mathrm{Supp}(\overline{\mathbf{w}}) \mapsto \{0, 1\}$ is a Boolean function for random variable $\overline{\mathbf{w}} \equiv (\mathbf{w}_1, \ldots, \mathbf{w}_n)$, and that $\Pr[f(\overline{\mathbf{w}}) = 1] = \varepsilon$. Then, the oracle-aided algorithm $\mathsf{Tam}^{\mathbf{w}[\cdot], f(\cdot)}(\varepsilon, \delta, \cdot)$ (also denoted by Tam for simplicity) with access to the online sampler $\mathbf{w}[\cdot]$ for $\overline{\mathbf{w}}$ and $f(\cdot)$ as oracles is an online tampering algorithm for $\overline{\mathbf{w}}$ and has the following features:*

1. *$\Pr[f(\overline{\mathbf{v}}) = 1] \geq 1 - \delta$ where $\overline{\mathbf{v}}$ is the tampered sequence, i.e., $\Pr_{(\overline{u}, \overline{v}) \leftarrow \langle \overline{\mathbf{w}} \, \| \, \mathsf{Tam} \rangle}[f(\overline{v}) = 1] \geq 1 - \delta$.*

2. *Tam's tampering budget in α-weighed Hamming distance $\mathsf{HD}_{\overline{\alpha}}$ is $O(\sqrt{n \cdot \ln(1/\varepsilon\delta)})$.*

3. *Tam runs in time $\mathrm{poly}(N/\varepsilon\delta)$ where N is the total bit representation of any $\overline{w} \leftarrow \overline{\mathbf{w}}$.*

Remark 3.3 (Corollary for product distributions). If the original random variable $\overline{\mathbf{w}} = (\mathbf{w}_1, \ldots, \mathbf{w}_n)$ in Theorem 3.2 is a product, $\overline{\mathbf{w}} = (\mathbf{w}_1 \times \cdots \times \mathbf{w}_n)$, then the distribution of the samples $\mathbf{u}$ obtained through $(\overline{u}, \overline{v}) \leftarrow \langle \overline{\mathbf{w}} \, \| \, \mathsf{Tam} \rangle$ would be identical to that of $\overline{\mathbf{w}}$. Namely, we can simply think of the samples $\overline{u}$ as the original *untampered* vector sampled from $\overline{\mathbf{w}}$, and $\overline{v}$ would be the perturbed vector.

In the rest of this section, we prove Theorem 3.2.

3.1 Proof Using Promised Approximate Partial Expectation Oracles

The following result works in the model where the approximate partial-expectations oracle $\tilde{f}(\cdot)$ is available to the online tampering algorithm AppTam.

Consider three oracles $\tilde{f}(v_{\leq i})$, $m(v_{\leq i})$ and $\tilde{f}^*(v_{\leq i}) = \tilde{f}(v_{\leq i}, m(v_{\leq i}))$ with the guarantee that for all $v_{\leq i} \in \mathrm{Supp}(\mathbf{w}_{\leq i})$ we have 5 conditions:

1. $\left| \ln \tilde{f}(v_{\leq i}) - \ln \hat{f}(v_{\leq i}) \right| \leq \gamma$,

2. $\tilde{f}^*(v_{\leq i}) = \tilde{f}(v_{\leq i}, m(v_{\leq i})) \geq \tilde{f}(v_{\leq i})$,

3. $\Pr\left[\tilde{f}(v_{\leq i}, \mathbf{w}[v_{\leq i}]) \geq \tilde{f}^*(v_{\leq i}) \right] \leq \gamma \cdot \tilde{f}(v_{\leq i})$,

4. $0 \leq \tilde{f}(v_{\leq i}) \leq 1$,

5. $\tilde{f}(v_{\leq n}) = f(v_{\leq n})$.

The first condition states that the approximate partial expectation oracle has a small multiplicative error. The second and third conditions state that $m(v_{\leq i-1})$ is a good approximation of some $v*$ that maximized $\tilde{f}(v_{\leq i-1}, v^*)$. Now we construct an algorithm using these oracles.

Construction 3.4 (Online tampering using promised *approximate* partial-expectations oracle). Recall that we are given a prefix $v_{\leq i-1}$ that is finalized, and we are also given a candidate value u_i for the i'th block (supposedly sampled from $\mathbf{w}[v_{\leq i-1}]$) and we want to decide to keep $v_i = u_i$ or change it. Let $\lambda > 0$ be a parameter of the attack to be chosen later, $v_i^* = \tilde{f}^*(v_{\leq i-1})$ and let $\tilde{f}^* = \tilde{f}^*(v_{\leq i-1})$ be that maximum.

1. (Case 1) If $\tilde{f}^* \geq e^{\lambda \alpha_i} \cdot \tilde{f}(v_{\leq i-1})$, then output $v_i = v_i^*$ (regardless of u_i).

2. (Case 2) Otherwise, if $\tilde{f}(v_{\leq i-1}, u_i) \leq e^{-\lambda \alpha_i} \cdot \tilde{f}(v_{\leq i-1})$, then output $v_i = v_i^*$.

3. (Case 3) Otherwise keep the value u_i and output $v_i = u_i$.

Claim 3.5 (Average case analysis of Construction 3.4). *Let $\mathbf{k}_i$ be the Boolean random variable that $\mathbf{k}_i = 1$ iff the tampering over the i'th block happens, and let $\mathbf{K}_{\overline{\alpha}} = \sum_{i \in [n]} \alpha_i \cdot \mathbf{k}_i$ capture the resulting $\mathsf{HD}_{\overline{\alpha}}$ distance between the jointly sampled $\overline{u}$ and $\overline{v}$. Also let $\tilde{\varepsilon} = \tilde{f}(\varnothing)$. Then, it holds that*

$$\ln(1/\tilde{\varepsilon}) \geq \mathbb{E}[\mathbf{K}_{\overline{\alpha}}] \cdot \lambda - \lambda^2 n / 2 + n \cdot \ln(1 - 3\gamma).$$

Corollary of Claim 3.5. By choosing $\lambda = \sqrt{2\ln(1/\varepsilon)/n}$, we obtain $\mathbb{E}[\mathbf{K}_{\overline{\alpha}}] \leq \sqrt{2n\ln(1/\varepsilon)}$.

We prove the following stronger statement that implies Claim 3.5.

Claim 3.6. *Let $v_{\leq i-1}$ be fixed. Then,*

$$\ln(1/\tilde{f}(v_{\leq i-1})) - \underset{v_i \leftarrow \mathbf{v}[v_{\leq i-1}]}{\mathbb{E}}\left[\ln\left(1/\tilde{f}(v_{\leq i})\right)\right] \geq \Pr[\mathbf{k}_i] \cdot (\alpha_i \lambda) - \frac{\alpha_i^2 \lambda^2}{2} + \ln(1 - 3\gamma).$$

Proof of Claim 3.5 using Claim 3.6. A key property of Construction 3.4 is that, because the tampering algorithm does not allow the function reach 0, the final sequence $\overline{v}$ always makes the function 1, namely

$$\Pr[f(\mathbf{v}_{\leq n}) = 1] = 1. \tag{1}$$

Using the above equation, Claim 3.5 follows from Claim 3.6 and linearity of expectation as follows.

$$
\begin{aligned}
\ln(1/\tilde{\varepsilon}) &= \ln(1/\tilde{f}(\varnothing)) - \mathbb{E}[\ln(1)] \\
\text{(by Equation 1)} \quad &= \ln(1/\tilde{f}(\varnothing)) - \mathbb{E}[\ln(1/\tilde{f}(\mathbf{v}_{\leq n}))] \\
\text{(by linearity of expectation)} \quad &= \sum_{i \in [n]}\left[\mathbb{E}[\ln(1/\tilde{f}(\mathbf{v}_{\leq i-1}))] - \mathbb{E}[\ln(1/\tilde{f}(\mathbf{v}_{\leq i}))]\right] \\
\text{(by Claim 3.6)} \quad &\geq \sum_{i \in [n]}\left[\mathbb{E}[\alpha_i \cdot \mathbf{k}_i] \cdot \lambda - \frac{\alpha_i^2 \cdot \lambda^2}{2} + \ln(1 - 3\gamma)\right] \\
\text{(by linearity of expectation)} \quad &= \mathbb{E}[\mathbf{K}_{\overline{\alpha}}] \cdot \lambda - \frac{n \cdot \lambda^2}{2} + \ln(1 - 3\gamma) \cdot n.
\end{aligned}
$$

$\square$

Now we prove Claim 3.6.

Proof of Claim 3.6. There are two cases:

13

- If tampering of Case 1 happens, then we have $\Pr[\mathbf{k}_i = 1] = 1$, and

$$\ln(1/\tilde{f}(v_{\leq i-1})) - \mathop{\mathbb{E}}_{v_i \leftarrow (\mathbf{v}[v_{\leq i-1}]}[\ln(1/\tilde{f}(v_{\leq i}))] \geq \ln(1/\tilde{f}(v_{\leq i-1})) - \ln(1/\tilde{f}^*) \geq \lambda\alpha_i$$

Thus, in this case Claim 3.6 follows trivially.

- If tampering of Case 1 does not happen, it means that $\tilde{f}^*$ is bounded from above. In the following, we focus on this case and all the probabilities and expectations are conditioned on Case 1 not happening; namely, we have $\tilde{f}^* \leq \tilde{f}(v_{\leq i-1}) \cdot e^\lambda$.

Let $I(v_{\leq i})$ be the indicator function for the set $\left\{ v_{\leq i} \colon \tilde{f}(v_{\leq i}) \leq e^{-\lambda\alpha_i} \cdot \tilde{f}(v_{\leq i-1}) \right\}$. We have

$$\mathop{\Pr}_{(u_i,v_i) \leftarrow (\mathbf{w},\mathbf{v})[v_{\leq i-1}]} \left[\tilde{f}(v_{\leq i}) \geq \max\left(e^{-\lambda\alpha_i} \cdot \tilde{f}(v_{\leq i-1}), \tilde{f}(v_{\leq i-1}, u_i) \right) \cdot e^{\lambda\alpha_i \cdot I(v_{\leq i-1}, u_i)} \right] = 1.$$

This is correct because we are either in Case 2, which means $I(v_{\leq i}) = 1$ and

$$\tilde{f}(v_{\leq i}) = \tilde{f}^* \geq \tilde{f}(v_{\leq i-1}) \geq \tilde{f}(v_{\leq i-1}, u_i) \cdot e^{\lambda \cdot \alpha_i}$$

or we are in Case 3 which means $I(v_{\leq i}) = 0$ and

$$\tilde{f}(v_{\leq i}) = \tilde{f}(v_{\leq i-1}, u_i).$$

Note that the two terms on each side of the inequality inside the probability above depend only on either of u_i or v_i (not both). Therefore, by linearity of expectation we have

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{v_i \leftarrow \mathbf{v}[v_{\leq i-1}]}[\ln(\tilde{f}(v_{\leq i}))] \\
&\geq \mathop{\mathbb{E}}_{u_i \leftarrow \mathbf{w}[v_{\leq i-1}]} \left[\ln\left(\max\left(e^{-\lambda\alpha_i} \cdot \tilde{f}(v_{\leq i-1}), \tilde{f}(v_{\leq i-1}, u_i) \right) \cdot e^{\lambda \cdot \alpha_i \cdot I(v_{\leq i-1}, u_i)} \right) \right] \\
&= \mathop{\mathbb{E}}_{u_i \leftarrow \mathbf{w}[v_{\leq i-1}]} \left[\ln\left(\max\left(e^{-\lambda\alpha_i} \cdot \tilde{f}(v_{\leq i-1}), \tilde{f}(v_{\leq i-1}, u_i) \right) \right) \right] + \lambda \cdot \alpha_i \cdot \mathop{\mathbb{E}}_{u_i \leftarrow \mathbf{w}[v_{\leq i-1}]}[I(v_{\leq i-1}, u_i)] \\
&= \mathop{\mathbb{E}}_{u_i \leftarrow \mathbf{w}[v_{\leq i-1}]} \left[\ln\left(\max\left(e^{-\lambda\alpha_i} \cdot \tilde{f}(v_{\leq i-1}), \tilde{f}(v_{\leq i-1}, u_i) \right) \right) \right] + \lambda \cdot \alpha_i \cdot \mathbb{E}[\mathbf{k}_i]. \qquad (2)
\end{aligned}
$$

Now consider the random variable $\mathbf{t}$ for a fixed $v_{\leq i-1}$ as follows

$$\mathbf{t} \equiv \frac{\max\left(e^{-\lambda} \cdot \tilde{f}(v_{\leq i-1}), \tilde{f}(v_{\leq i-1}, \mathbf{w}[v_{\leq i-1}]) \right)}{\tilde{f}(v_{\leq i-1})}.$$

It holds that

$$\Pr\left[e^{-\lambda\alpha_i} \leq \mathbf{t} \right] = 1. \qquad (3)$$

We also know by condition 3 of the $\tilde{f}^*(\cdot)$ oracle that

$$\Pr[\tilde{f}^*(v_{\leq i-1}) \geq \tilde{f}(v_{\leq i-1}, \mathbf{w}[v_{\leq i-1}])] \geq 1 - \gamma \cdot \tilde{f}(v_{\leq i-1})$$

which together with $\tilde{f}^*(v_{\leq i-1}) \leq \tilde{f}(v_{\leq i-1}) \cdot e^{\lambda \alpha_i}$ implies

$$\Pr[\mathbf{t} \leq e^{\lambda \cdot \alpha_i}] \leq 1 - \gamma \cdot \tilde{f}(v_{\leq i-1}). \tag{4}$$

We also know that

$$\Pr\left[\mathbf{t} \leq \frac{1}{\tilde{f}(v_{\leq i-1})}\right] = 1. \tag{5}$$

We also have

$$\begin{aligned}
\mathbb{E}\left[\mathbf{t} \cdot \tilde{f}(v_{\leq i-1})\right] &= \mathbb{E}\left[\max\left(e^{-\lambda} \cdot \tilde{f}(v_{\leq i-1}), \tilde{f}(v_{\leq i-1}, \mathbf{w}[v_{\leq i-1}])\right)\right] \\
&\geq \mathbb{E}\left[\tilde{f}(v_{\leq i-1}, \mathbf{w}[v_{\leq i-1}])\right] \\
&\geq \mathbb{E}\left[\hat{f}(v_{\leq i-1}, \mathbf{w}[v_{\leq i-1}])\right] \cdot e^{-\gamma} \\
&= \hat{f}(v_{\leq i-1}) \cdot e^{-\gamma} \\
&\geq \tilde{f}(v_{\leq i-1}) \cdot e^{-2\gamma}.
\end{aligned}$$

which implies

$$\mathbb{E}[\mathbf{t}] \geq e^{-2\gamma} \geq 1 - 2\gamma. \tag{6}$$

Therefore using 3, 4, 5 and 6 and applying Lemma 2.4 we get,

$$\mathbb{E}[\ln(\mathbf{t})] \geq \ln\left(\mathbb{E}[\mathbf{t}] - \gamma \cdot \tilde{f}(v_{\leq i-1}) \cdot \frac{1}{\tilde{f}(v_{\leq i-1})}\right) - \frac{\alpha_i^2 \cdot \lambda^2}{2} \geq \ln(1 - 3\gamma) - \frac{\alpha_i^2 \cdot \lambda^2}{2}. \tag{7}$$

Combining Equations (2) and (7), we get

$$\mathbb{E}_{v_i \leftarrow \mathbf{v}[v_{\leq i-1}]}\left[\ln(\tilde{f}(v_{\leq i}))\right] \geq \ln(\tilde{f}(v_{\leq i-1})) + \lambda \cdot \alpha_i \cdot \mathbb{E}[\mathbf{k}_i] - \frac{\lambda^2 \cdot \alpha_i^2}{2} + \ln(1 - 3\gamma)$$

which finishes the proof. $\qquad\square$

Claim 3.7 (Worst case analysis of Construction 3.4). *Let $\mathbf{k}_i$ be the Boolean random variable that $\mathbf{k}_i = 1$ iff the tampering over the i'th block happens, and let $\mathbf{K}_{\overline{\alpha}} = \sum_{i \in [n]} \alpha_i \cdot \mathbf{k}_i$ capture the resulting $\mathsf{HD}_{\overline{\alpha}}$ distance between the jointly sampled $\overline{u}$ and $\overline{v}$. Also let $\tilde{\varepsilon} = \tilde{f}(\varnothing)$. Then, it holds that*

$$\Pr[\mathbf{K} \geq k] \leq \frac{e^{(\sum_{i=1}^{n} \alpha_i^2)\lambda^2 - k\lambda}}{\tilde{\varepsilon} \cdot (1 - 2\gamma)^n}.$$

Proof. We prove this claim by induction on n. Let $A(n, k, \tilde{\varepsilon})$ be a function that indicates the maximum probability of using more than k budget, over all random processes with boolean outcome of length n, and average $\tilde{\varepsilon}$. We want to inductively show that

$$A(n, k, \tilde{\varepsilon}) \leq \frac{e^{(\sum_{i=1}^{n} \alpha_i^2) \cdot \lambda^2 - k\lambda}}{\tilde{\varepsilon} \cdot (1 - 2\gamma)^n}.$$

Consider different cases that might happen during the tampering of first block. If we tamper on first block through Case I, we have

$$\Pr[\mathbf{K} \geq k] \leq A(n - 1, k - \alpha_1, \tilde{f}^*(\varnothing))$$

15

And by induction hypothesis we have

$$A(n-1, k-\alpha_1, \tilde{f}^*(\varnothing)) \leq \frac{e^{(\sum_{i=2}^n \alpha_i^2)\lambda^2 - k\lambda + \lambda \cdot \alpha_i}}{\tilde{f}^*(\varnothing) \cdot (1-2\gamma)^n} \leq \frac{e^{(\sum_{i=2}^n \alpha_i^2)\lambda^2 - k\lambda + \lambda \cdot \alpha_i}}{e^{\lambda \alpha_1} \cdot \tilde{\varepsilon} \cdot (1-2\gamma)^n} \leq \frac{e^{(\sum_{i=1}^n \alpha_i^2)\cdot \lambda^2 - k\lambda}}{\tilde{\varepsilon} \cdot (1-2\gamma)^n}.$$

So the induction goes through for Case 1. If we are not in Case 1, then we have,

$$
\begin{aligned}
\Pr\left[\mathbf{K} \geq k\right] &= \Pr\left[\mathbf{K} \geq k \mid \text{Case 3}\right] \cdot \Pr\left[\text{Case 3}\right] + \Pr\left[\mathbf{K} \geq k \mid \text{Case 2}\right] \cdot \Pr\left[\text{Case 2}\right] \\
&\leq \mathbb{E}\left[A\left(n-1, k, \tilde{f}(u_{\leq 1})\right) \mid \text{Case 3}\right] \cdot \Pr\left[\text{Case 3}\right] \\
&\quad + \mathbb{E}\left[A\left(n-1, k-\alpha_1, \tilde{f}^*(\varnothing)\right) \mid \text{Case 2}\right] \cdot \Pr\left[\text{Case 2}\right] \\
&\leq \mathbb{E}\left[\frac{e^{(\sum_{i=2}^n \alpha_i^2)\cdot\lambda^2 - k\lambda}}{\tilde{f}(u_{\leq 1}) - 2(n-1)\gamma} \mid \text{Case 3}\right] \cdot \Pr\left[\text{Case 3}\right] \\
&\quad + \mathbb{E}\left[\frac{e^{(\sum_{i=2}^n \alpha_i^2)\cdot\lambda^2 - k\lambda + \lambda \cdot \alpha_1}}{\tilde{f}^*(\varnothing) \cdot (1-2\gamma)^{n-1}} \mid \text{Case 2}\right] \cdot \Pr\left[\text{Case 2}\right] \\
&\leq \mathbb{E}\left[\frac{e^{(\sum_{i=2}^n \alpha_i^2)\cdot\lambda^2 - k\lambda}}{\tilde{f}(u_{\leq 1}) \cdot (1-2\gamma)^{n-1}} \mid \text{Case 3}\right] \cdot \Pr\left[\text{Case 3}\right] \\
&\quad + \mathbb{E}\left[\frac{e^{(\sum_{i=2}^n \alpha_i^2)\cdot\lambda^2 - k\lambda + \lambda \cdot \alpha_1}}{\max\left(e^{-\lambda \cdot \alpha_i} \cdot \tilde{\varepsilon}, \tilde{f}(u_{\leq 1})\right) \cdot e^{\lambda \cdot \alpha_i} \cdot (1-2\gamma)^{n-1}} \mid \text{Case 2}\right] \cdot \Pr\left[\text{Case 2}\right] \\
&\leq e^{(\sum_{i=2}^n \alpha_i^2)\cdot\lambda^2 - k\lambda} \cdot \mathbb{E}\left[\frac{1}{\max(e^{-\lambda \cdot \alpha_i} \cdot \tilde{\varepsilon}, \tilde{f}(u_{\leq 1})) \cdot (1-2\gamma)^{n-1}}\right] \quad (8)
\end{aligned}
$$

We know that $\mathbb{E}[\max(e^{-\lambda \cdot \alpha_i} \cdot \tilde{\varepsilon}, \tilde{f}(u_{\leq 1})))] \geq \mathbb{E}[\tilde{f}(u_{\leq 1})] \geq \tilde{\varepsilon} \cdot e^{-\gamma}$. Now we can use Lemma 2.5 and get

$$\mathbb{E}[\frac{1}{\max(e^{-\lambda \cdot \alpha_1} \cdot \tilde{\varepsilon}, \tilde{f}(u_{\leq 1})))}] \leq \frac{e^{\alpha_1^2 \cdot \lambda^2}}{\tilde{\varepsilon} \cdot (e^{-\gamma} - \gamma)} \leq \frac{e^{\alpha_1^2 \cdot \lambda^2}}{\tilde{\varepsilon} \cdot (1-2\gamma)} \quad (9)$$

Combining Equations 8 and 9 we get,

$$\Pr[\mathbf{K} \geq k] \leq \frac{e^{(\sum_{i=1}^n \alpha_i^2)\lambda^2 - k\lambda}}{\tilde{\varepsilon}(1-2\gamma)^n}$$

which finishes the proof. $\square$

3.1.1 Tampering with Abort

The Construction 3.4 achieves average close to 1 with small number of tampering. However we cannot implement that construction it in polynomial time. The problem is that it is hard to instantiate the oracle $\tilde{f}(\cdot)$ and $\tilde{f}^*(\cdot)$ in polynomial time when the partial average gets close to 0. Following we add a step to our construction to address this issue. Then we will show that this additional step will not hurt the performance of the algorithm by much.

Construction 3.8 (Online tampering *with abort* AppTamAb using promised approximate partial-expectations oracle)**.** This construction is identical to Construction 3.4, except that whenever the fixed prefix has a too small approximate partial expectation $\tilde{f}(v_{\leq i-1}, u_i)$ (based on a parameter τ) we will abort. Also, in that case the tampering algorithm does not tamper with any future v_i block either. Namely, we add the following "Case 0" to the previous steps:

- (Case 0) If $\tilde{f}(v_{\leq i-1}, u_i) \leq e^{-\tau} \cdot \tilde{\varepsilon}$ abort ($\tilde{\varepsilon} = \tilde{f}(\varnothing)$). If had aborted before, do nothing.

Average and worst case analysis of Construction 3.8. The average number of tampering of Construction 3.8 is trivially less than average number of tampering of Construction 3.4. Therefore, the same bound of Claim 3.5 still applies to Construction 3.8 as well. Also, the probability of number of tampering going beyond some threshold does not increase compared to Construction 3.4 which means the same bound of Claim 3.7 hold here.

Claim 3.9. *The probability of ever aborting during sampling* $(\overline{u}, \overline{v}) \leftarrow \langle \overline{\mathbf{w}} \parallel \mathsf{TamAb} \rangle$ *is at most* $n \cdot e^{-\frac{(\tau - n \cdot \lambda^2/2)^2}{2 \cdot n \cdot \lambda^2}}$*. As a result, we also have*

$$\mathbb{E}_{(\overline{u}, \overline{v}) \leftarrow \langle \overline{\mathbf{w}} \parallel \mathsf{TamAb} \rangle}[f(\overline{v})] \geq 1 - n \cdot e^{-\frac{(\tau - n \cdot \lambda^2/2)^2}{2 \cdot n \cdot \lambda^2}} - n^2 \gamma.$$

Proof. Define Boolean indicator functions I_0, I_1, I_2 and I_3, as well as a real-valued vector $\overline{y}$ as follows. The first function I_0 indicates that we have not aborted yet, and the others define a condition for their corresponding cases in Construction 3.4.

$$I_0(v_{\leq i-1}) = \begin{cases} 0 & \text{if } \forall j \leq i; \ \tilde{f}(v_{\leq j}) \geq \tilde{\varepsilon} \cdot e^{-\tau}, \\ 1 & \text{otherwise.} \end{cases}$$

$$I_1(v_{\leq i-1}) = \begin{cases} 1 & \text{if } \tilde{f}^*(v_{\leq i-1}) \geq e^{\lambda \cdot \alpha_i} \cdot \tilde{f}(v_{\leq i-1}) \text{ and } \neg I_0(v_{\leq i-1}), \\ 0 & \text{otherwise;} \end{cases}$$

$$I_2(v_{\leq i}) = \begin{cases} 1 & \text{if } \tilde{f}(v_{\leq i}) \leq e^{-\lambda \cdot \alpha_i} \cdot \tilde{f}(v_{\leq i-1}) \text{ and } \neg I_1(v_{\leq i-1}) \text{ and } \neg I_0(v_{\leq i-1}), \\ 0 & \text{otherwise;} \end{cases}$$

The last function indicates that the above conditions are *not* happening.

$$I_3(v_{\leq i}) = \begin{cases} 1 & \text{if } \neg I_0(v_{\leq i}) \text{ and } \neg I_1(v_{\leq i-1}) \text{ and } \neg I_2(v_{\leq i}), \\ 0 & \text{otherwise.} \end{cases}$$

Finally, we define a real-valued function y as follows that captures the change in the potential function for the cases where none of I_0, I_1, I_2 are happening.

$$y(v_{\leq i}) = \left(\ln(\tilde{f}(v_{\leq i})) - \ln(\tilde{f}(v_{\leq i-1})) \right) \cdot I_3(v_{\leq i}).$$

Now consider a sequence of random variables $\overline{\mathbf{y}} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ sampled as follows. We first sample $(\overline{u}, \overline{v}) \leftarrow (\overline{\mathbf{u}}, \overline{\mathbf{v}})$ then set $y_i = y(v_{\leq i-1}, u_i) = y(v_{\leq i})$ for $i \in [n]$. Note that $y(v_{\leq i-1}, u_i) = y(v_{\leq i})$ because if $I_3(v_{\leq i-1}, u_i) = 1$ it means that $u_i = v_i$.

Claim 3.10. *We have* $\mathbb{E}[e^{\mathbf{y}_i} \mid y_{\leq i-1}] \geq e^{-2\gamma}$.

Notation. Since $I_j(\cdot)$'s are Boolean, we can use the notation $(I_i \vee I_j)(v_{\leq i})$ or $(1 - (I_i \vee I_j))(v_{\leq i})$ based on logical operators to construct more Boolean indicators.

Proof of Claim 3.10. The high level idea is that $e^{\mathbf{y}_i}$ is approximately equal to $\hat{f}(v_{\leq i-1}, u_i)/\hat{f}(v_{\leq i})$ when we are in Case 3. The average of $\hat{f}(v_{\leq i-1}, u_i)/\hat{f}(v_{\leq i})$ conditioned on Case 2 and Case 3 is exactly 1. We know that in Case 2 the average is less than one, therefore the average in Case 3 should be at least 1. Following, we formalize this idea.

$$\mathbb{E}[e^{\mathbf{y}_i} \mid y_{\leq i-1}] = \mathop{\mathbb{E}}_{v_{\leq i-1} \leftarrow (\mathbf{v}_{\leq i-1}|y_{\leq i-1})} \left[\mathop{\mathbb{E}}_{u_i \leftarrow \mathbf{w}[v_{\leq i-1}]} \left[e^{\left(\ln(\tilde{f}(v_{\leq i-1}, u_i)) - \ln(\tilde{f}(v_{\leq i-1})) \right) \cdot I_3(v_{\leq i-1}, u_i)} \right] \right]$$

$$\geq \mathop{\mathbb{E}}_{v_{\leq i-1} \leftarrow (\mathbf{v}_{\leq i-1}|y_{\leq i-1})} \left[\mathop{\mathbb{E}}_{u_i \leftarrow \mathbf{w}[v_{\leq i-1}]} \left[e^{\left(\ln(\tilde{f}(v_{\leq i-1}, u_i)) - \ln(\tilde{f}(v_{\leq i-1})) \right) \cdot \left((I_3 \vee I_2)(v_{\leq i-1}, u_i) \right)} \right] \right]$$

$$= \mathop{\mathbb{E}}_{v_{\leq i-1} \leftarrow (\mathbf{v}_{\leq i-1}|y_{\leq i-1})} \left[\mathop{\mathbb{E}}_{u_i \leftarrow \mathbf{w}[v_{\leq i-1}]} \left[e^{\left(\ln(\tilde{f}(v_{\leq i-1}, u_i)) - \ln(\tilde{f}(v_{\leq i-1})) \right) \cdot \left(1 - (I_1 \vee I_0)(v_{\leq i-1}) \right)} \right] \right]$$

$$\geq \mathop{\mathbb{E}}_{v_{\leq i-1} \leftarrow (\mathbf{v}_{\leq i-1}|y_{\leq i-1})} \left[\min \left(\mathop{\mathbb{E}}_{u_i \leftarrow \mathbf{w}[v_{\leq i-1}]} \left[e^{\left(\ln(\tilde{f}(v_{\leq i-1}, u_i)) - \ln(\tilde{f}(v_{\leq i-1})) \right)} \right], 1 \right) \right]$$

$$= \mathop{\mathbb{E}}_{v_{\leq i-1} \leftarrow (\mathbf{v}_{\leq i-1}|y_{\leq i-1})} \left[\min \left(\mathop{\mathbb{E}}_{u_i \leftarrow \mathbf{w}[v_{\leq i-1}]} \left[\tilde{f}(v_{\leq i-1}, u_i)/\tilde{f}(v_{\leq i-1}) \right], 1 \right) \right]$$

$$\geq \mathop{\mathbb{E}}_{v_{\leq i-1} \leftarrow (\mathbf{v}_{\leq i-1}|y_{\leq i-1})} \left[\min \left(e^{-2\gamma} \cdot \mathop{\mathbb{E}}_{u_i \leftarrow \mathbf{w}[v_{\leq i-1}]} \left[\hat{f}(v_{\leq i-1}, u_i)/\hat{f}(v_{\leq i-1}) \right], 1 \right) \right]$$

$$= e^{-2\gamma}.$$

$\square$

Claim 3.11. *We have* $\Pr[\mathbf{y}_i \geq -\lambda \cdot \alpha_i] = 1$ *and* $\Pr[\mathbf{y}_i \leq \lambda \cdot \alpha_i] \geq 1 - \gamma \cdot \tilde{f}(v_{\leq i-1})$.

Proof. If $I_3(v_{\leq i}) = 0$ then $y(v_{\leq i}) = 0$ and both inequalities hold. On the other hand, If $I_3(v_{\leq i}) = 1$ it means that $e^{-\lambda \cdot \alpha_i} \cdot \tilde{f}(v_{\leq i-1}) \leq \tilde{f}(v_{\leq i})$. Also $\Pr[\mathbf{y}_i \leq e^{\lambda \cdot \alpha_i} \cdot \tilde{f}(v_{\leq i-1})] \geq 1 - \gamma \cdot \tilde{f}(v_{\leq i-1})$ holds because of gaurantee of the oracle $f^*(\cdot)$. $\square$

Claim 3.12. *We have* $\mathbb{E}[\mathbf{y}_i \mid y_{\leq i-1}] \geq \ln(1 - 3\gamma) - \frac{\lambda^2 \cdot \alpha_i^2}{2}$.

Proof. The proof follows by using Lemma 2.4 and Claims 3.10 and 3.11. $\square$

Claim 3.13. *The probability of aborting is at most* $n \cdot e^{\frac{(\tau - \lambda n \cdot \lambda^2/2)^2}{2 \cdot n \cdot \lambda^2}}$.

Proof. By Claims 3.12 and 3.11, the sequence $\overline{\mathbf{y}} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ forms an (approximate) submartingale and by Azuma inequality of Lemma 2.6 we have,

$$\Pr \left[\sum_{i=1}^{n} \mathbf{y}_i \leq -\tau \right] \leq e^{-\frac{\left(\tau - n \cdot \lambda^2/2 \right)^2}{2 \cdot n \cdot \lambda^2}} + n \cdot \gamma \ .$$

On the other hand, for every $v_{\leq i} \in \mathrm{Supp}(\mathbf{v}_{\leq i})$ we have $I_2(v_{\leq i}) = 0$. Therefore, for every $v_{\leq j} \in \mathrm{Supp}(\mathbf{v}_{\leq j})$,

$$
\begin{aligned}
\ln(\tilde{f}(v_{\leq j})) &= \ln(\tilde{\varepsilon}) + \sum_{i=1}^{j}(\ln(\tilde{f}(v_{\leq i})) - \ln(\tilde{f}(v_{\leq i-1}))) \\
&= \ln(\tilde{\varepsilon}) + \sum_{i=1}^{j}\left(\ln(\tilde{f}(v_{\leq i})) - \ln(\tilde{f}(v_{\leq i-1}))\right) \cdot ((I_0 \vee I_1)(v_{\leq i-1}) + I_3(v_{\leq i})) \\
&\geq \ln(\tilde{\varepsilon}) + \sum_{i=1}^{j}\left(\ln(\tilde{f}(v_{\leq i})) - \ln(\tilde{f}(v_{\leq i-1}))\right) \cdot (I_0(v_{\leq i-1}) + I_3(v_{\leq i})) \\
&= \ln(\tilde{\varepsilon}) + \sum_{i=1}^{j} y(v_{\leq i}) + \sum_{i=1}^{j}\left(\ln(\tilde{f}(v_{\leq i})) - \ln(\tilde{f}(v_{\leq i-1}))\right) \cdot I_0(v_{\leq i-1}).
\end{aligned}
$$

We now calculate probability of the event A_j that the partial average goes bellow $e^{-\tau} \cdot \tilde{\varepsilon}$ (i.e., abort happens) at the jth block for the *first time*.

$$
\begin{aligned}
&\Pr_{v_{\leq j} \leftarrow \mathbf{v}_{\leq j}}[A_j] \\
&= \Pr_{v_{\leq j} \leftarrow \mathbf{v}_{\leq j}}\left[\tilde{f}(v_{\leq j}) \leq e^{-\tau} \cdot \tilde{\varepsilon} \wedge \neg I_0(v_{\leq j-1})\right] \\
&= \Pr_{v_{\leq j} \leftarrow \mathbf{v}_{\leq j}}[\ln(\tilde{f}(v_{\leq j})) \leq -\tau + \ln(\tilde{\varepsilon}) \wedge \neg I_0(v_{\leq j-1})] \\
&\leq \Pr_{v_{\leq j} \leftarrow \mathbf{v}_{\leq j}}\left[\sum_{i=1}^{j} y(v_{\leq i}) + \sum_{i=1}^{j}\left(\ln(\tilde{f}(v_{\leq i})) - \ln(\tilde{f}(v_{\leq i-1}))\right) \cdot I_0(v_{\leq i-1}) \leq -\tau \wedge \neg I_0(v_{\leq j-1})\right] \\
&\leq \Pr_{v_{\leq j} \leftarrow \mathbf{v}_{\leq j}}\left[\sum_{i=1}^{j} y(v_{\leq i}) \leq -\tau\right] \\
&\leq e^{-\frac{(\tau - n \cdot \lambda^2/2)^2}{2 \cdot n \cdot \lambda^2}} + n \cdot \gamma.
\end{aligned}
$$

The above means that the probability that the tampering algorithm of Construction 3.8 enters the abort state is less than $n \cdot e^{-\frac{(\tau - n \cdot \lambda^2/2)^2}{2 \cdot n \cdot \lambda^2}} + n^2 \cdot \gamma$. $\qquad\square$

We already know that if abort does not happen then the output will always be 1. Therefore, we have

$$
\mathbb{E}_{\overline{v} \leftarrow \mathbf{v}}[f(\overline{v})] \geq 1 - n \cdot e^{-\frac{(\tau - n \cdot \lambda^2/2)^2}{2 \cdot n \cdot \lambda^2}} - n^2 \gamma.
$$

$$
\mathbb{E}_{\overline{v} \leftarrow \mathbf{v}}[f(\overline{v})] \geq 1 - \delta.
$$

$\qquad\square$

3.2 Putting Things Together

In this subsection we show how to instantiate parameters of Construction 3.8 so that we can get polynomial time attack. We first show how to instantiate the oracles. To compute oracle $\tilde{f}(v_{\leq i})$, we sample $\frac{8}{\gamma^3 \cdot e^{-\tau} \cdot \tilde{\varepsilon}}$

random continuation and take the average over all of them. By Hoeffding inequality, if $\hat{f}(v_{\leq i}) \geq e^{-\tau} \cdot \tilde{\varepsilon}$ we get the following:

$$\Pr[|\ln(\tilde{f}(v_{\leq i})) - \ln(\hat{f}(v_{\leq i}))| \geq \gamma] \leq \gamma.$$

For $m(v_{\leq i})$ and $\tilde{f}^*(v_{\leq i})$ oracle, sample $\frac{1}{\gamma^2 \cdot e^{-\tau \cdot \tilde{\varepsilon}}}$ number of v_{i+1} and take the maximum over $\tilde{f}(v_{\leq i+1})$. This way, we can easily bound the probability of Conditions 2 or 3 not happening by γ for all $v_{\leq i}$ that $\tilde{f}(v_{\leq i}) \geq e^{-\tau} \cdot \tilde{\varepsilon}$. Note that in both of these oracle, we are ignoring the case where $\tilde{f}(v_{\leq i})$ is smaller than the threshold that causes the construction to abort. This enables us to achieve high confidence on our oracles. Using these oracles, we can bound the average of function, average budget and worst case budget of construction 3.8 as follows. Based on Claim 3.9 we have

$$\mathbb{E}_{(\overline{u},\overline{v}) \leftarrow \langle \overline{\mathbf{w}} \,\|\, \mathsf{TamAb}\rangle}[f(\overline{v})] \geq 1 - n \cdot e^{-\frac{(\tau - n \cdot \lambda^2/2)^2}{2 \cdot n \cdot \lambda^2}} - n^2 \gamma - 2n \cdot \gamma.$$

The last $-2n \cdot \gamma$ is added to the right hand side to capture the probability of any of the algorithm's oracle calls failing. For the average budget, following Claim 3.5 we have,

$$\mathbb{E}[\mathbf{K}_{\overline{\alpha}}] \leq \frac{\ln(1/\tilde{\varepsilon}) + \lambda^2 n/2 - n \cdot \ln(1 - 3\gamma)}{\lambda} + 2 \cdot n \cdot \gamma.$$

And for the worst case budget, following Claim 3.7 we have

$$\Pr[\mathbf{K} \geq k] \leq \frac{e^{n\lambda^2 - k\lambda}}{\tilde{\varepsilon} - 2\gamma} + 2n \cdot \gamma.$$

Instantiating the Average Case Algorithm: Now if we set $\lambda = \sqrt{-2\ln(\varepsilon)/n}$, $\tau = \ln(1/\tilde{\varepsilon}) + \sqrt{4\ln(\delta/2n) \cdot \ln(\tilde{\varepsilon})}$ and $\gamma = \frac{\delta}{24n^2}$ then we can provide the oracles in time $poly(n/\varepsilon \cdot \delta)$ and we get:

$$\mathbb{E}_{(\overline{u},\overline{v}) \leftarrow \langle \overline{\mathbf{w}} \,\|\, \mathsf{TamAb}\rangle}[f(\overline{v})] \geq 1 - \delta$$

and

$$\mathbb{E}[\mathbf{K}_{\overline{\alpha}}] \leq \sqrt{-2n\ln(\varepsilon)} + \delta.$$

Instantiating the Worst Case Algorithm: Also, for the worst case attacks. If we select the tampering budget $k = \sqrt{2n \cdot \ln(\delta/8) \cdot \ln(\varepsilon/2)}$ and then let $\lambda = k/2n$. For $\tau = \ln(1/\tilde{\varepsilon}) + \sqrt{4\ln(\delta/2n) \cdot \ln(\tilde{\varepsilon})}$ and $\gamma = \min(\delta/24n^2, \varepsilon/4n)$ we get an algorithm that runs in time $poly(n/\varepsilon \cdot \delta)$, uses at most k tamperings and increases the average as follows

$$\mathbb{E}_{(\overline{u},\overline{v}) \leftarrow \langle \overline{\mathbf{w}} \,\|\, \mathsf{TamAb}\rangle}[f(\overline{v})] \geq 1 - \delta.$$

4 Algorithmic Reductions for Computational Concentration

In this section, we show a generic framework to prove computational concentration for a metric probability space by reducing its computational concentration to that of another metric probability space. We first define an embedding with some properties.

Definition 4.1. Let $S_1 = (\mathcal{X}_1, \mathsf{d}_1, \boldsymbol{\mu}_1)$ and $S_2 = (\mathcal{X}_2, \mathsf{d}_2, \boldsymbol{\mu}_2)$ be two metric probability spaces. We call a pair of mappings $(\mathbf{f}, \mathbf{g})$ (where $\mathbf{f}$ and $\mathbf{g}$ are potentially randomized) an (α, b, w) computational concentration (CC) reduction from S_1 to S_2 if the following hold:

- **Probability embedding.** The distribution $\mathbf{f}(\boldsymbol{\mu}_1)$ is α-close (in statistical distance) to $\boldsymbol{\mu}_2$ and $\mathbf{g}(\boldsymbol{\mu}_2)$ is α-close to $\boldsymbol{\mu}_1$.

- **Almost Lipschitz property of g.** With probability 1 over all $x, x' \leftarrow \boldsymbol{\mu}_2$, $\mathsf{d}_1(\mathbf{g}(x), \mathbf{g}(x')) \leq w \cdot \mathsf{d}_2(x, x') + b$.

- **Almost inverse mappings.** For every $x_1 \in \mathcal{X}_1$, and all $x_2 \leftarrow \mathbf{f}(x_1)$, it holds that $\mathsf{d}_1(x_1, \mathbf{g}(x_2)) \leq b$.

Now we have the following lemma which how to reduce computational concentration on a metric probability space by reducing it to computational concentration on another metric probability space using the embedding between them.

Theorem 4.2. *Let $S_2 = (\mathcal{X}_2, \mathsf{d}_2, \boldsymbol{\mu}_2)$ be a metric probability space and let $A_2^{\mathcal{S}(\cdot)} : \mathcal{X}_2 \to \mathcal{X}_2$ be an oracle algorithm such that for any subset $\mathcal{S} \subseteq \mathcal{X}_2$ we have $\mathsf{d}_2(A_2^{\mathcal{S}(\cdot)}(x), x) \leq k$ and*

$$\Pr_{x \leftarrow \boldsymbol{\mu}_2} [A_2^{\mathcal{S}(\cdot)}(x) \in \mathcal{S}] \geq c(\boldsymbol{\mu}_2(\mathcal{S}))$$

for a function $c \colon [0,1] \to [0,1]$. If $(\mathbf{f}, \mathbf{g})$ is an (α, b, w) CC reduction from $S_1 = (\mathcal{X}_1, \mathsf{d}_1, \boldsymbol{\mu}_1)$ to $S_2 = (\mathcal{X}_2, \mathsf{d}_2, \boldsymbol{\mu}_2)$, then there is an oracle algorithm $A_1^{\mathcal{S}(\cdot)} \colon \mathcal{X}_1 \to \mathcal{X}_1$ such that for any subset $\mathcal{S} \subseteq \mathcal{X}_1$ we have $\mathsf{d}_1(A_1^{\mathcal{S}(\cdot)}(x), x) \leq w \cdot k + 2b$ and

$$\Pr_{x \leftarrow \boldsymbol{\mu}_1} [A_1^{\mathcal{S}(\cdot)}(x) \in \mathcal{S}] \geq c(\boldsymbol{\mu}_1(\mathcal{S})/2 - \alpha) - \alpha - \mathrm{negl}(n).$$

Furthermore, if A_2, $\mathbf{f}$ and $\mathbf{g}$ run in time $\mathrm{poly}(\frac{n}{\varepsilon})$, then A_1 also runs in time $\mathrm{poly}(\frac{n}{\varepsilon})$.

Proof. We define algorithm $A_1^{\mathcal{S}(\cdot)}$ on input x as follows: A_1 first computes $f(x_1)$ to get x_1'. Then it creates a set $\mathcal{S}' = \{x \in \mathcal{X}_2 \colon \Pr[g(x) \in \mathcal{S}] \geq 1/2\}$ and runs $A_2^{\mathcal{S}'(\cdot)}$ on x_1' to get x_2'. Then, it computes $g(x_2')$ for at most n times until it gets some $x_2 \in \mathcal{S}$ and outputs x_2, otherwise it outputs a fresh $g(x_2')$. We have

$$\begin{aligned}
\Pr_{x_1 \leftarrow \boldsymbol{\mu}_1} [A_1^{\mathcal{S}(\cdot)}(x_1) \in \mathcal{S}] &\geq \Pr_{x_1 \leftarrow \boldsymbol{\mu}_1} [A_2^{\mathcal{S}'(\cdot)}(f(x_1)) \in \mathcal{S}'] - 2^{-n} \\
&\geq \Pr_{x_1' \leftarrow \boldsymbol{\mu}_2} [A_2^{\mathcal{S}'(\cdot)}(x_1') \in \mathcal{S}'] - \alpha - 2^{-n} \\
&\geq c(\boldsymbol{\mu}_2(\mathcal{S}')) - 2^{-n} - \alpha \\
&\geq c(\boldsymbol{\mu}_1(\mathcal{S})/2 - \alpha) - 2^{-n} - \alpha.
\end{aligned}$$

Note that the oracle $\mathcal{S}'(\cdot)$ cannot be implemented in polynomial time, but it could be approximated with negligible error in polynomial time. On the other hand, we have

$$\begin{aligned}
\mathsf{d}_1(A_1(x_1), x_1) &= \mathsf{d}(x_2, x_1) \\
&\leq \mathsf{d}_1(x_2, g(x_1')) + \mathsf{d}_1(g(x_1'), x_1) \\
&\leq \mathsf{d}_1(x_2, g(x_1')) + b \\
&\leq w \cdot \mathsf{d}_2(x_2', x_1')) + 2b \\
&\leq w \cdot k + 2b.
\end{aligned}$$

$\square$

The following construction shows an embedding from Gaussian distribution to hamming cube. Using this embedding and Lemma 4.2 we get computational concentration for the Gaussian distribution. The following embedding uses ideas similar to [B$^+$97].

Construction 4.3 (CC reduction from (Gaussian, ℓ_1) to Hamming cube)**.** We construct f and g as follows.

f : Let n be an even number. Given a point $x = (x_1, \ldots, x_n)$ sampled from Gaussian space of dimension n, do the following:

    1. If $\exists i; |x_i| \geq \sqrt{n}/2$, output 0^{n^2}.

    2. Otherwise, for each $i \in n$ compute $a_i = \lceil \frac{x_i}{\sqrt{n}} + \frac{n}{2} \rceil$ then uniformly sample some $y_i \in \{0,1\}^n$ such that y_i has exactly a_i number of 1s. Then append y_i s to get $y = (y_1 | \ldots | y_n)$.

g : Let $y = (y_1 | \ldots | y_n)$ be a Boolean vector of size n^2 (each y_i has size n). Let a_i be the number of 1s in y_i. Then sample $x = (x_1, \ldots, x_n)$ from Gaussian space conditioned on $\frac{2a_i - n}{2\sqrt{n}} \leq x_i < \frac{2a_i - n + 1}{2\sqrt{n}}$

Claim 4.4. *The embedding of Construction 4.3 is an* $(\mathrm{negl}(n), 1/\sqrt{n}, 1/\sqrt{n})$ *CC reduction from Gaussian space under ℓ_1 to Hamming cube (i.e., Boolean hypercube under Hamming distance).*

Proof. The embedding property of these mappings is proved in [B$^+$97]. The mappings f and g are clearly polynomial time in n and the Almost Lipschitz and Inverse Mappings properties are straightforward. $\square$

The following Corollary follows from Lemma 4.2, Claim 4.4 and Theorem 3.2.

Corollary 4.5 (Computational concentration of Gaussian under ℓ_1)**.** *There is an algorithm $A_{\varepsilon,\delta}^{\mathcal{S},\mu}(\cdot)$ that given access to a membership oracle for any set $\mathcal{S}$ and a sampling oracle from an isotropic Gaussian measure μ of dimension n, it achieves the following. If $\Pr[\mathcal{S}] \geq \varepsilon$, given ε and δ, the algorithm $A_{\varepsilon,\delta}^{\mathcal{S},\mu}(\cdot)$ runs in time $\mathrm{poly}(n/\varepsilon\delta)$, and with probability $\geq 1 - \delta$ given a random point $\overline{x} \leftarrow \mu$, it maps $\overline{x}$ to a point $\overline{y} \in \mathcal{S}$ of bounded ℓ_1 distance $\ell_1(\overline{x}, \overline{y}) \leq O(\sqrt{n \cdot \ln(1/\varepsilon\delta)})$.*

4.1 Case of Gaussian or Sphere under ℓ_2

A reduction may also be used to obtain a (non-optimal) computational concentration of measure for the multi-dimensional Gaussian distribution under the ℓ_2 metric.

Theorem 4.6. *There is an algorithm $A_{\varepsilon,\delta}^{\mathcal{S},\mu}(\cdot)$ that given access to a membership oracle for any set $\mathcal{S}$ and a sampling oracle from an isotropic Gaussian measure μ of dimension n, where each coordinate has variance 1, it achieves the following. If $\Pr[\mathcal{S}] \geq \varepsilon$, given $\varepsilon, \delta \geq 1/n^{O(1)}$, the algorithm $A_{\varepsilon,\delta}^{\mathcal{S},\mu}(\cdot)$ runs in time $\mathrm{poly}(n)$, and with probability $\geq 1 - \delta$ given a random point $\overline{x} \leftarrow \mu$, it maps $\overline{x}$ to a point $\overline{y} \in \mathcal{S}$ of bounded ℓ_2 distance $\ell_2(\overline{x}, \overline{y}) \leq O(n^{1/4} \log^{O(1)} n)$.*

Proof. Since $\epsilon \geq 1/n^{O(1)}$, at most $\epsilon/2$ and $\delta/2$ fraction of the points have a coordinate of size $\geq O(\sqrt{\log n})$. So ignoring points having such large coordinates, we may assume $\Pr[\mathcal{S}] \geq \epsilon/2$ while every point of $\mathcal{S}$ has coordinates as small as $O(\sqrt{\log n})$, and we may assume the point we are mapping also has small coordinates (except our algorithm should now work for $1 - \delta/2$ fraction of the points instead of for $1 - \delta$ fraction.)

Now, when each coordinate is $O(\sqrt{\log n})$, the l_2 distance between two points is at most $O(\sqrt{d_H \log n})$, where d_H is the Hamming distance of the two points. Now, the theorem follows from our main theorem for Hamming distance. $\square$

We should note that the above computational bound is not information-theoretically tight, since for the Gaussian ℓ_2 metric probability space, where each coordinate has variance 1, the right bound is $O(\sqrt{\ln(1/(\epsilon\delta))})$. (This follows e.g. from the Gaussian isoperimetric inequality proved in [ST78,Bor75], which shows the half-space is isopermetrically optimal for the Gaussian distribution.)

Finally, the following shows that our results are not limited to product spaces, and may for example be applied to computational concentration of measure for the high-dimensional sphere.

Theorem 4.7. *There is an algorithm $A_{\varepsilon,\delta}^{\mathcal{S},\mu}(\cdot)$ that given access to a membership oracle for any set $\mathcal{S}$ and a sampling oracle from the uniform measure μ on the unit sphere of dimension n, it achieves the following. If $\Pr[\mathcal{S}] \geq \varepsilon$, given $\varepsilon, \delta \geq 1/n^{O(1)}$, the algorithm $A_{\varepsilon,\delta}^{\mathcal{S},\mu}(\cdot)$ runs in time $\mathrm{poly}(n)$, and with probability $\geq 1 - \delta$ given a random point $\overline{x} \leftarrow \mu$, it maps $\overline{x}$ to a point $\overline{y} \in \mathcal{S}$ of bounded ℓ_2 distance $\ell_2(\overline{x}, \overline{y}) \leq O(n^{-1/4} \log^{O(1)} n)$.*

Proof. First, we note that a random Gaussian vector, where each coordinate has variance 1, has ℓ_2 norm $\sqrt{n} + O(n^{1/4})$ except for arbitrary inverse polynomial probability.

So given $\overline{x}$, we can map it to a new vector $\overline{x}'$ with the same direction as $\overline{x}$ but with a random length of distribution square root of chi square, so that the new vector has the Gaussian distribution. We also map the set $\mathcal{S}$ to the set $\mathcal{S}' = \{r \cdot s : r \in n^{1/2} + O(n^{1/4}), s \in \mathcal{S}\}$, where the new set still has probability $\geq \epsilon/2$ under the Gaussian distribution. By the computational concentration of measure for the Gaussian, we know that we can map, with probability $1 - \delta/2$, $\overline{x}'$ to a point $\overline{y}' \in \mathcal{S}'$ of distance $n^{1/4} \log^{O(1)} n$ from $\overline{x}'$ in ℓ_2. Let $\overline{y}$ be the projection of $\overline{y}'$ onto the unit sphere. Therefore

$$d_{\ell_2}(\overline{x}, \overline{y}) \leq d_{\ell_2}(\overline{x}, \overline{x}'/\sqrt{n}) + d_{\ell_2}(\overline{x}'/\sqrt{n}, \overline{y}'\sqrt{n}) + d_{\ell_2}(\overline{y}'/\sqrt{n}, \overline{y}) = O(n^{1/4} \log^{O(1)} n).$$

$\square$

These types of relations between concentration of measure of Gaussian and uniform sphere measures has been well-known information-theoretically, e.g. see [Led01, page 2] where concentration for Gaussian is derived from concentration for sphere. In the above we showed a similar relation for *computational* concentration of measure, this time deriving for the sphere from the Gaussian.

5 Computational Concentration around Mean

Let $(\mathcal{X}, \mathsf{d}, \mu)$ be a metric probability space and $f: \mathcal{X} \mapsto \mathbb{R}$ a measurable function (with respect to μ). For any Borel set $\mathcal{T} \subseteq \mathbb{R}$, an parameters $k, \delta \in \mathbb{R}_+$, one can define a computational problem as follows. Given oracle access to a sampler from μ, d and function $f(\cdot)$, map a given input $x \in \mathcal{X}$ algorithmically to $y \in \mathcal{Y}$, such that: (1) $\mathsf{d}(x, y) \leq k$, and (2) $f(y) \in \mathcal{T}$ for $1 - \delta$ fraction of $x \in \mathcal{X}$ according to μ. If we already know that $(\mathcal{X}, \mathsf{d}, \mu)$ is (ε, δ, k) (computationally) concentrates, and if $\Pr_{x \leftarrow \mu}[f(x) \in \mathcal{T}] \geq \varepsilon$, then it implies that by changing x by at most distance k into a new point y, we can (algorithmically) get $f(y) \in \mathcal{T}$, by defining $\mathcal{S} = f^{-1}(\mathcal{T})$ and noting that $\Pr_\mu[\mathcal{S}] \geq \varepsilon$. This algorithm needs oracle access to $\mathcal{S}$

Computational concentration around mean. Again, let $(\mathcal{X}, \mathsf{d}, \mu)$ be a metric probability space and let $f: \mathcal{X} \mapsto \mathbb{R}$ be measurable. Now suppose $\eta = \mathbb{E}_{x \leftarrow \mu}[f(x)]$. If we already know, by information theoretic concentration bounds, that $\Pr_{x \leftarrow \mu}[|f(x) - \eta| \leq T] \geq 1 - \delta$, then it means that a trivial algorithm that does not even change given $x \leftarrow \mu$, finds a point where $f(x)$ is T-close to the average η. However, this becomes nontrivial, if the goal of the algorithm is to find y that is close to x, and that $f(y)$ is much *closer* to the mean η than what x achieves. In particular, suppose we somehow know that $\Pr_{x \leftarrow \mu}[|f(x) - \eta| \leq t] \geq \varepsilon$

for $t \ll T, \varepsilon \ll 1 - \delta$. (Such results usually follow from the same concentration inequalities proving $\Pr_{x \leftarrow \mu}[|f(x) - \eta| \leq T] \approx 1$.) The smaller t is, the "higher quality" the point x has in terms of $f(x)$ being closer to the mean. This means the set $\mathcal{S} = \{x \colon |f(x) - \eta| \leq t\}$ has μ measure at least ε. Therefore, if the space $(\mathcal{X}, \mathsf{d}, \mu)$ is (ε, δ, k) computationally concentrated, then we can conclude that there is an efficient algorithm (whose running time can polynomially depend on $1/\varepsilon\delta$ and) that maps $1 - \delta$ fraction of $x \leftarrow \mu$ to a point $y \in \mathcal{S}$. Different, but similar, statements about one-sided concentration can be made as well, if we start from weaker conditions of the form $\Pr_{x \leftarrow \mu}[f(x) > \eta + t] \leq \varepsilon$ (or $\Pr_{x \leftarrow \mu}[f(x) < \eta - t] \leq 1 - \varepsilon$) leading to a weaker conclusion: we can map x to a point y satisfies $f(x) \geq \eta - t$ (or $f(x) \leq \eta + t$).

Finally, we note that even if the mean η is *not* known to the mapping algorithm A, good approximations of it can be obtained by repeated sampling and taking their average. So for simplicity, and without loss of generality, the reader can assume that η is known to the mapping algorithm A.

Special case of Lipschitz functions: algorithmic proofs of concentration. When $f \colon \mathcal{X} \mapsto \mathbb{R}$ is Lipschitz, i.e., $|f(x) - f(y)| \leq \mathsf{d}(x, y)$, computational concentration around a set like $\mathcal{S} = \{x \colon |f(x) - \eta| \leq t\}$ (or similar one-sided variants) means something stronger than before. We now have an algorithm that *indirectly proves* the concentration around η by efficiently finding points that are almost at the border defined by η. Namely, the Lipschitz now implies that $|f(x) - f(y)| \leq k$, whenever $|x - y| \leq k$. Therefore, the algorithm A mapping x to y is also proving that $1 - \delta$ measure of the space $(\mathcal{X}, \mu)$ is mapped under f to a point that is $k + t$ close to average η.

All the above arguments are general and apply to any metric probability space. Below, we discuss an special case of a "McDiarmid type" inequality in more detail to demonstrate the power of this argument.

Theorem 5.1 (An algorithmic variant of McDiarmid inequality). *Suppose $\mu \equiv \mu_1 \times \cdots \times \mu_n$ is a product measure on a product space $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, and let $f \colon \mathcal{X} \mapsto \mathbb{R}$ be such that $|f(\overline{x}) - f(\overline{x}')| \leq \alpha_i$ whenever $\overline{x}$ and $\overline{x}'$ only differ in the ith coordinate. Let $a = \|\overline{\alpha}\|_2$ for $\overline{\alpha} = (\alpha_1, \ldots, \alpha_n)$. Let $\eta = \mathbb{E}_{x \leftarrow \mu}[f(\overline{x})]$ and $\mathcal{S} = \{x \colon f(\overline{x}) \leq \eta + \varepsilon \cdot a\}$. Then there is an algorithm $A_{\varepsilon,\delta}^{\mu,f(\overline{x})}(\cdot)$ running in time $\mathrm{poly}(n/\varepsilon\delta)$ that uses oracle access to f and a sampler from μ, and it holds that*

$$\Pr_{x \leftarrow \mu, y \leftarrow A_{\varepsilon,\delta}^{\mu,f}(\overline{x})} \left[\overline{y} \in \mathcal{S} \quad \text{and} \quad |f(\overline{x}) - f(\overline{y})| \leq O\left(\sqrt{m \cdot \log(1/\varepsilon\delta)} \right) \right] \geq 1 - \delta.$$

Corollaries for special cases. Theorem 5.1 implies a similar result when the quality of the destination region is base on the ℓ_1 norm; namely, $\mathcal{S} = \{x \colon f(\overline{x}) \leq \eta + \varepsilon \cdot \|\alpha\|_1\}$, but this follows from the same statement since $\|\alpha\|_2 \leq \|\alpha\|_1$. In addition, for the special case where $\alpha_i = 1$ for all i,[8] and let $\gamma, \delta = 1/\mathrm{poly}(n)$ be arbitrarily small inverse polynomials. In that case, Theorem 5.1, shows that for $1 - \delta$ fraction of $\overline{x} \leftarrow \mu$, we can map $\overline{x}$ to $\overline{y}$ in $\mathrm{poly}(n)$ time in such a way that $f(\overline{y}) \leq \mathbb{E}[f(\mu)] + \gamma$ and $|f(\overline{x}) - f(\overline{y})| \leq \widetilde{O}(\sqrt{n})$. If we choose $\gamma < 1/2$, due to the Lipschitz condition, we can also find some $\overline{y}$ for which $f(\overline{y}) \in \mathbb{E}[f(\mu)] \pm 1$. This is possible by first finding some $\overline{y}$ where $f(\overline{y}) \leq \mathbb{E}[f(\mu)] + \gamma$, and then go back over the coordinates in which $\overline{x}$ and $\overline{y}$ differ and only changing some of them to get $\overline{y}'$ where $f(\overline{y}') \in \mathbb{E}[f(\mu)] \pm 1$, and output $\overline{y}'$ instead. We note that, however, that whenever we want to choose $\gamma < 1/2$, we need to also choose $\varepsilon < 1/(2n)$. For this range of small ε, we *cannot* use the computational concentration results of [MM19], but we can indeed use the stronger computational concentration results of this work that prove computational concentration around any non-negligible event.

---

[8]For example, this could be the setting of Hoeffding's inequality in which each coordinate μ_i is arbitrarily distributed over $[0, 1]$, and $f(\overline{x}) = \sum_{i \in [n]} x_i$, where $\overline{x} = (x_1, \ldots, x_n)$

Proof of Theorem 5.1. For starters, suppose η is given. In that case, we first observe that $\Pr_{\mu}[\mathcal{S}] \geq 1 - e^{-2\varepsilon^2} = \Theta(\varepsilon^2)$ by McDiarmid's inequality itself. We can then apply Theorem 3.2.

When η is not given, we can find a sufficiently good approximation of it, such that $\eta' \in \eta \pm \|\alpha\|_2 \cdot \varepsilon/10$ (in time $\mathrm{poly}(n/\varepsilon\delta)$ and error probability $\delta/10$) and use it instead of η. Obtaining such η' can be done because any x, x' satisfy $|f(x) - f(x')| \leq \|\alpha\|_1$. Therefore, we can obtain $\eta' \in \eta \pm \lambda \cdot \|\alpha\|_1$ in time by sampling $\ell = \mathrm{poly}(n/\lambda\delta)$ (for sufficiently large ℓ) many points $\overline{x}_1, \ldots, \overline{x}_\ell \leftarrow \mu$ and letting $\eta' = \mathbb{E}_{i \leftarrow \ell} f(\overline{x}_i)$. The only catch is that we want $\eta' \in \eta \pm \varepsilon \cdot \|\alpha\|_2$. However, since it holds that $\|\alpha\|_2 \leq \|\alpha\|_1 \cdot \sqrt{n}$, we can choose $\lambda = \varepsilon/\sqrt{n}$, and use the same procedure to obtain $\eta' \in \eta \pm \lambda \cdot \|\alpha\|_1$ with probability $1 - \delta/10$ in time $\mathrm{poly}(n/\varepsilon\delta)$. $\square$

6 Limits of Nonadaptive Methods for Proving Computational Concentration

In this section, we consider three restricted types of attacks and prove exponential lower bounds on their running time. The attacks are

- **I.i.d. queries:** An attack where given $\overline{x}$, we query i.i.d. points whose distribution may depend on $\overline{x}$, until one of these points lies in $\mathcal{S}$. The analysis of this attack boils down to analysis of a single-query attack where we want to maximize the probability of $\mathcal{S}$-membership of the queried point.

- **Non-adaptive queries:** An attack where given $\overline{x}$, we output a list of points, and query all the points in this list. Since the points in the list are determined before the querying, this attack is non-adaptive. It is easy to see (and we give a proof below) how lower bounding this type of attack reduces to the previous type of attack.

- **Querying only points close enough to have a chance to be output:** If we are interested in finding a point at distance $\leq d$ from $\overline{x}$, one may be tempted to limit the queried points to points at distance $\leq d$ from $\overline{x}$. We show how lower bounding this type of attack reduces to the previous type of attack.

Theorem 6.1 (Lower bound for non-adaptive algorithms)**.** *Let μ be the uniform probability distribution on $\{1, -1\}^n$, and let $\varepsilon = 1/2$ and $\delta < 1/2$ be constants. There does not exist any non-adaptive algorithm A that given $\overline{x} \leftarrow \mu$, the algorithm outputs $m = n^{O(1)}$ (random) points $\overline{y}^1, \ldots, \overline{y}^m$, all within Hamming distance $n^{1-\Omega(1)}$ of $\overline{x}$, such that given any set $\mathcal{S}$ with $\Pr[\mathcal{S}] \geq \varepsilon$, one of these m points lies in $\mathcal{S}$ with probability $1 - \delta$ over the randomness of x and randomness of $\overline{y}^1, \ldots, \overline{y}^m$.*

Proof. Assume for the sake of contradiction that such an algorithm A exists. Consider the following modified algorithm: given $\overline{x}$, run A to produce $\overline{y}^1, \ldots, \overline{y}^m$, and then let $\overline{z}^1$ be one of those m vectors uniformly at random. To produce $\overline{z}^2$, run A independently afresh, and let $\overline{z}^2$ be one of the m freshly produced vectors. We can continue in this way, and produce the vectors $\overline{z}^1, \ldots, \overline{z}^{m'}$ as the output of the modified algorithm. By the assumption, for any constant $\delta' \in (\delta, 1/2)$, with probability $1 - \delta'$ over the randomness of $\overline{x}$, algorithm A has success probability at least $1/n$, hence each $\overline{z}^i$ lies in $\mathcal{S}$ with probability $\geq 1/mn$. Hence for these $\overline{x}$, if we choose $m' = mn^2$, with probability $1 - (1 - 1/mn)^{m'} = 1 - o(1)$, the modified algorithm succeeds. Therefore, the average success probability of the algorithm is $\geq 1 - \delta' - o(1) \geq 1/2 + \Omega(1)$.

The above argument shows that we only need to look at algorithms where $\overline{y}^1, \ldots, \overline{y}^m$ are independent given $\overline{x}$. Thus, it is enough to show that there does not exist a random mapping from $\overline{x}$ to a vector $\overline{y}$ in such a way that with probability $1 - \delta$ over the randomness of $\overline{x}$, the probability $\Pr[\overline{y} \in \mathcal{S}]$ is non-negligible (since m is polynomial in n).

For the sake of contradiction, assume such a mapping from $\overline{x}$ to $\overline{y}$ exists. Let $\mathcal{S}$ be a random half-space, i.e. $\mathcal{S} = \{\overline{z} : \sum_{i=1}^{n} a_i z_i \leq 0\}$ for a uniformly random vector $a = (a_1, \ldots, a_n) \in \{-1, 1\}^n$. We will show that for every $\overline{x}$, with probability δ over the randomness of a, the probability $\Pr[\overline{y} \in \mathcal{S}]$ is negligible. By an averaging argument, this shows that there exists a half-space $\mathcal{S}$ such that with probability δ over the randomness of $\overline{x}$, $\Pr[\overline{y} \in \mathcal{S}]$ is negligible, completing the proof.

As mentioned above, we want to show that for every $\overline{x}$, a random half-space is troublesome for the algorithm. By symmetry, without loss of generality, we may assume $\overline{x} = (1, 1, \ldots, 1)$. Let $\eta = (\eta_1, \ldots, \eta_n) = (\overline{x} - \overline{y})/2$ be the characteristic vector for the coordinates for which $\overline{y}$ is different from $\overline{x}$. We note that $\overline{y} \in \mathcal{S}$ iff $\sum_i a_i - 2\sum_i a_i \eta_i \leq 0$. We know that with probability $\delta + \Omega(1)$ over the randomness of a, we have $\sum_i a_i \geq \Omega(\sqrt{n})$. (This easily follows from the central limit theorem.) Now, conditioned on η, the sum $\sum_i \eta_i a_i$ is actually a sum of $n^{1-\Omega(1)}$-many ± 1 independent random variables of mean zero, so $\Pr[\sum_i \eta_i a_i \geq \Omega(\sqrt{n})]$ is a negligible, actually exponentially small, probability. This implies over the randomness of a and η, $\Pr[\sum_i \eta_i a_i \geq \Omega(\sqrt{n})]$ is negligible. Thus, except for an $o(1)$ fraction of random half-spaces, $\Pr[\sum_i \eta_i a_i \geq \Omega(\sqrt{n})]$ is negligible over the randomness of $\overline{y}$. Thus, with probability at least $\delta + \Omega(1) - o(1) \geq \delta$ over the randomness of a, we have both

- $\sum_i a_i \geq \Omega(\sqrt{n})$, and

- $\Pr[\sum_i a_i \eta_i = \Omega(\sqrt{n})]$ is negligible over the randomness of $\overline{y}$.

In this case, $\overline{y}$ does not lie in $\mathcal{S}$ except with non-negligible. $\qquad\square$

Remark 6.2. It can be seen that the above theorem holds whenever ε and δ are positive constants such that $\varepsilon + \delta < 1$. It can be seen that the above theorem does not hold when $\varepsilon + \delta > 1$ since when we set $\overline{y} = \overline{x}$, our failure probability δ is exactly $1 - \varepsilon$.

Lemma 6.3. *Given a radius r, assume an adaptive algorithm A, given $\overline{x}$, wants to find a vector $\overline{y} \in \mathcal{S}$ in the ball of radius r around $\overline{x}$. Furthermore, assume that the algorithm does not make any $\mathcal{S}$-membership oracle queries regarding points outside the ball. Then, we can transform the algorithm into a non-adaptive algorithm with the same performance.*

Proof. When the algorithm ever queries about a point $\overline{y}$ (and by assumption $\overline{y}$ is in the ball), if the oracle says that $\overline{y} \in S$, then we are done (since we have found our desired point.) So the algorithm may always pretend that the result of each membership query about each queried point is that the point is not in $\mathcal{S}$. This equivalent algorithm is non-adaptive. $\qquad\square$

Corollary 6.4. *In the $\{0, 1\}^n$ uniform product space, when we want to find a point $\overline{y} \in \mathcal{S}$ at distance $n^{1/2+\varepsilon}$ from a random $\overline{x}$ (for some $\varepsilon \in (0, 1/2)$), to be query-efficient, we need to query about $\mathcal{S}$-membership of points having distance more than $n^{1/2+\varepsilon}$.*

The above corollary says that even though we are interested in points in a ball of certain radius around $\overline{x}$, we have to query about points outside that ball. When we notice that we are not assuming any structure on the set $\mathcal{S}$ other than it should have some minimum mass, the above corollary becomes all the more surprising!

7 Acknowledgement

References

[AGK76]   Rudolf Ahlswede, Peter Gács, and János Körner. Bounds on conditional probabilities with applications in multi-user communication. *Probability Theory and Related Fields*, 34(2):157–177, 1976. 1

[AM80]   D Amir and VD Milman. Unconditional and symmetric sets in n-dimensional normed spaces. *Israel Journal of Mathematics*, 37(1-2):3–20, 1980. 1

[AM85]   Noga Alon and Vitali D Milman. λ1, isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, Series B*, 38(1):73–88, 1985. 1, 4

[B$^+$97]   Sergey G Bobkov et al. An isoperimetric inequality on the discrete cube, and an elementary proof of the isoperimetric inequality in gauss space. *The Annals of Probability*, 25(1):206–214, 1997. 3, 9, 22

[BEG17]   Salman Beigi, Omid Etesami, and Amin Gohari. Deterministic randomness extraction from generalized and distributed santha–vazirani sources. *SIAM Journal on Computing*, 46(1):1–36, 2017. 3

[BEK02]   Nader H. Bshouty, Nadav Eiron, and Eyal Kushilevitz. PAC learning with nasty noise. *Theoretical Computer Science*, 288(2):255–275, 2002. 4

[BFR14]   Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering*, 26(4):984–996, 2014. 4

[BGZ16]   Iddo Bentov, Ariel Gabizon, and David Zuckerman. Bitcoin beacon. *arXiv preprint arXiv:1605.04559*, 2016. 3

[BHT14]   Itay Berman, Iftach Haitner, and Aris Tentes. Coin flipping of any constant bias implies one-way functions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 398–407. ACM, 2014. 2

[BNL12]   Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1467–1474. Omnipress, 2012. 4

[BNS$^+$06]   Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25. ACM, 2006. 4

[BOL89]   M. Ben-Or and N. Linial. Collective coin flipping. *Randomness and Computation*, 5:91–115, 1989. 2

[Bor75]   Christer Borell. The brunn-minkowski inequality in gauss space. *Inventiones mathematicae*, 30(2):207–216, 1975. 23

[BPR18]   Sébastien Bubeck, Eric Price, and Ilya Razenshteyn. Adversarial examples from computational constraints. *arXiv preprint arXiv:1805.10204*, 2018. 5

[CG88]   Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988. 3

[CI93]   Richard Cleve and Russell Impagliazzo. Martingales, collective coin flipping and discrete control processes. *Manuscript*, 1993. 2

[CW17]   Nicholas Carlini and David A. Wagner. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 39–57, 2017. 4, 5

[DKK+16]   Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high dimensions without the computational intractability. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 655–664. IEEE, 2016. 4

[DKK+18]   Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. *arXiv preprint arXiv:1803.02815*, 2018. 4

[DOPS04]   Yevgeniy Dodis, Shien Jin Ong, Manoj Prabhakaran, and Amit Sahai. On the (Im)possibility of Cryptography with Imperfect Randomness. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 2004. 3

[DV19]   Akshay Degwekar and Vinod Vaikuntanathan. Computational limitations in robust classification and win-win results. *arXiv preprint arXiv:1902.01086*, 2019. 5

[GKP15]   Shafi Goldwasser, Yael Tauman Kalai, and Sunoo Park. Adaptively secure coin-flipping, revisited. In *International Colloquium on Automata, Languages, and Programming*, pages 663–674. Springer, 2015. 2

[GMP18]   Ian J. Goodfellow, Patrick D. McDaniel, and Nicolas Papernot. Making machine learning robust against adversarial inputs. *Communications of the ACM*, 61(7):56–66, 2018. 4

[HO14]   Iftach Haitner and Eran Omri. Coin flipping with constant bias implies one-way functions. *SIAM Journal on Computing*, 43(2):389–409, 2014. 2

[IK10]   Russell Impagliazzo and Valentine Kabanets. Constructive proofs of concentration bounds. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 617–631. Springer, 2010. 2

[KKR18]   Yael Tauman Kalai, Ilan Komargodski, and Ran Raz. A lower bound for adaptively-secure collective coin-flipping protocols. In *32nd International Symposium on Distributed Computing*, 2018. 2, 5

[KL93]   Michael J. Kearns and Ming Li. Learning in the Presence of Malicious Errors. *SIAM Journal on Computing*, 22(4):807–837, 1993. 4

[Led01]   Michel Ledoux. *The Concentration of Measure Phenomenon*. Number 89 in Mathematical Surveys and Monographs. American Mathematical Society, 2001. 1, 23

[Lév51]   Paul Lévy. *Problèmes concrets d'analyse fonctionnelle*, volume 6. Gauthier-Villars Paris, 1951. 1, 4

[LLS89]   David Lichtenstein, Nathan Linial, and Michael Saks. Some extremal problems arising from discrete control processes. *Combinatorica*, 9(3):269–287, 1989. 5

[LRV16]   Kevin A Lai, Anup B Rao, and Santosh Vempala. Agnostic estimation of mean and covariance. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 665–674. IEEE, 2016. 4

[Mar74]   Grigorii Aleksandrovich Margulis. Probabilistic characteristics of graphs with large connectivity. *Problemy peredachi informatsii*, 10(2):101–108, 1974. 1

[Mar86]   Katalin Marton. A simple proof of the blowing-up lemma (corresp.). *IEEE Transactions on Information Theory*, 32(3):445–446, 1986. 1

[McD89]   Colin McDiarmid. On the method of bounded differences. *Surveys in combinatorics*, 141(1):148–188, 1989. 1

[MDM18]   Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. Learning under p-Tampering Attacks. In *ALT*, pages 572–596, 2018. 3

[MDM19]   Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. *AAAI Conference on Artificial Intelligence*, 2019. 4

[MFF16]   Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *CVPR*, pages 2574–2582, 2016. 5

[MHRAR98] Colin Mcdiarmid, M Habib, J Ramirez-Alfonsin, and B Reed. Probabilistic methods for algorithmic discrete mathematics. *Algorithms and Combinatorics Series*, 16:1–46, 1998. 11

[MM17]   Saeed Mahloujifar and Mohammad Mahmoody. Blockwise p-tampering attacks on cryptographic primitives, extractors, and learners. In *Theory of Cryptography Conference*, pages 245–279. Springer, 2017. 3

[MM19]   Saeed Mahloujifar and Mohammad Mahmoody. Can adversarially robust learning leverage computational hardness? In Aurélien Garivier and Satyen Kale, editors, *Proceedings of the 30th International Conference on Algorithmic Learning Theory*, volume 98 of *Proceedings of Machine Learning Research*, pages 581–609, Chicago, Illinois, 22–24 Mar 2019. PMLR. 1, 2, 4, 5, 6, 24

[Mos09]   Robin A Moser. A constructive proof of the lovász local lemma. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 343–350. ACM, 2009. 2

[MPS10]   Hemanta K Maji, Manoj Prabhakaran, and Amit Sahai. On the computational complexity of coin flipping. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 613–622. IEEE, 2010. 2

[MS86]   Vitali D Milman and Gideon Schechtman. *Asymptotic theory of finite dimensional normed spaces*, volume 1200. Springer Verlag, 1986. 1

[MT10]   Robin A Moser and Gábor Tardos. A constructive proof of the general lovász local lemma. *Journal of the ACM (JACM)*, 57(2):11, 2010. 2

[PSBR18]   Adarsh Prasad, Arun Sai Suggala, Sivaraman Balakrishnan, and Pradeep Ravikumar. Robust estimation via robust gradient estimation. *arXiv preprint arXiv:1802.06485*, 2018. 4

[RVW04]   Omer Reingold, Salil Vadhan, and Avi Wigderson. A note on extracting randomness from santha-vazirani sources. *Unpublished manuscript*, 2004. 3

[ST78]   Vladimir N Sudakov and Boris S Tsirel'son. Extremal properties of half-spaces for spherically invariant measures. *Journal of Mathematical Sciences*, 9(1):9–18, 1978. 23

[SV86]   Miklos Santha and Umesh V. Vazirani. Generating quasi-random sequences from semi-random sources. *J. Comput. Syst. Sci.*, 33(1):75–87, 1986. 3

[SZS+14]   Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014. 4, 5

[Tal95]   Michel Talagrand. Concentration of measure and isoperimetric inequalities in product spaces. *Publications Mathématiques de l'Institut des Hautes Etudes Scientifiques*, 81(1):73–205, 1995. 1, 2

[Val85]   Leslie G. Valiant. Learning disjunctions of conjunctions. In *IJCAI*, pages 560–566, 1985. 4