Electronic Colloquium on Computational Complexity, Report No. 71 (2019)

Time-Space Lower Bounds for Two-Pass Learning

Sumegha Garg[*] Ran Raz[†] Avishay Tal[‡]

Abstract

A line of recent works showed that for a large class of learning problems, any learning algorithm requires either super-linear memory size or a super-polynomial number of samples [R16, KRT17, R17, MM18, BOGY18, GRT18]. For example, any algorithm for learning parities of size n requires either a memory of size $\Omega(n^2)$ or an exponential number of samples [R16].

All these works modeled the learner as a one-pass branching program, allowing only one pass over the stream of samples. In this work, we prove the first memory-samples lower bounds (with a super-linear lower bound on the memory size and super-polynomial lower bound on the number of samples) when the learner is allowed two passes over the stream of samples. For example, we prove that any two-pass algorithm for learning parities of size n requires either a memory of size $\Omega(n^{1.5})$ or at least $2^{\Omega(\sqrt{n})}$ samples.

More generally, a matrix $M : A \times X \to \{-1, 1\}$ corresponds to the following learning problem: An unknown element $x \in X$ is chosen uniformly at random. A learner tries to learn x from a stream of samples, $(a_1, b_1), (a_2, b_2)\ldots$, where for every i, $a_i \in A$ is chosen uniformly at random and $b_i = M(a_i, x)$.

Assume that k, ℓ, r are such that any submatrix of M of at least $2^{-k} \cdot |A|$ rows and at least $2^{-\ell} \cdot |X|$ columns, has a bias of at most 2^{-r}. We show that any two-pass learning algorithm for the learning problem corresponding to M requires either a memory of size at least $\Omega\left(k \cdot \min\{k, \sqrt{\ell}\}\right)$, or at least $2^{\Omega(\min\{k,\sqrt{\ell},r\})}$ samples.

[*] Department of Computer Science, Princeton University.

[†] Department of Computer Science, Princeton University. Research supported by the Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grant No. CCF-1714779.

[‡] Department of Computer Science, Stanford University, Stanford, CA. Part of this work was done when the author was a member at the Institute for Advanced Study, Princeton, NJ. Research supported by the Simons Collaboration on Algorithms and Geometry and by the National Science Foundation grant No. CCF-1412958.

ISSN 1433-8092

1 Introduction

A large number of recent works studied the problem of proving memory-samples lower bounds for learning [S14, SVW16, R16, VV16, KRT17, MM17, R17, MM18, MT17, BOGY18, GRT18, DS18], a study that was initiated by the beautiful papers of Shamir [S14] and Steinhardt, Valiant and Wager [SVW16]. The motivation for studying this question comes from learning theory, computational complexity and cryptography (see for example the discussion and references in [S14, SVW16, R16, VV16, KRT17, MT17]).

Steinhardt, Valiant and Wager conjectured that any algorithm for learning parities of size n requires either a memory of size $\Omega(n^2)$ or an exponential number of samples. This conjecture was proven in [R16], followed by a line of works that showed that for a large number of learning problems, any learning algorithm requires either super-linear memory size or a super-polynomial number of samples [KRT17, R17, MM18, BOGY18, GRT18]. For example, such bounds were established for learning sparse parities, linear-size DNF Formulas, linear-size Decision Trees and logarithmic-size Juntas [KRT17]; learning low-degree polynomials [BOGY18, GRT18]; learning from sparse linear equations and low-degree polynomial equations [GRT18]; learning codewords from random coordinates [R17, MM18, GRT18]; etc.

All previous memory-samples lower bounds (in the regime where the lower bound on the memory size is super-linear and the lower bound on the number of samples is super-polynomial) modeled the learning algorithm by a *one-pass branching program*, allowing only one pass over the stream of samples.

In this work, we prove the first such results when two passes over the stream of samples are allowed. (We remark that we leave open the question of handling more than two passes. While some parts of the current proof naturally extend to more than two passes, others are more delicate.)

Our Results

As in [R17, BOGY18, KRT17], we represent a learning problem by a matrix. Let X, A be two finite sets of size larger than 1 (where X represents the concept-class that we are trying to learn and A represents the set of possible samples). Let $M : A \times X \to \{-1, 1\}$ be a matrix. The matrix M represents the following learning problem: An unknown element $x \in X$ was chosen uniformly at random. A learner tries to learn x from a stream of samples, $(a_1, b_1), (a_2, b_2) \ldots$, where for every i, $a_i \in A$ is chosen uniformly at random and $b_i = M(a_i, x)$.

We model the learner for the learning problem that corresponds to the matrix M, by a *two-pass ordered branching program* (Definition 2.2). Such a program reads the entire stream of samples twice, in the exact same order. Roughly speaking, the model allows a learner with infinite computational power, and bounds only the memory size of the learner and the number of samples used.

As in [GRT18], our result is stated in terms of the properties of the matrix M as a two-source extractor. Two-source extractors, first studied by Santha and Vazirani [SV84] and Chor and Goldreich [CG88], are central objects in the study of randomness and derandomization. As in [GRT18], our results hold whenever the matrix M has (even relatively weak) two-source extractor properties.

Roughly speaking, our main result can be stated as follows: Assume that k, ℓ, r are such that any submatrix of M of at least $2^{-k} \cdot |A|$ rows and at least $2^{-\ell} \cdot |X|$ columns, has a bias of at most 2^{-r}. Then, any two-pass learning algorithm for the learning problem corresponding to M requires either a memory of size at least $\Omega\left(k \cdot \min\{k, \sqrt{\ell}\}\right)$, or at least $2^{\Omega(\min\{k,\sqrt{\ell},r\})}$ samples.

Formally, our result is stated in Theorem 1 in terms of the properties of M as an L_2-Extractor (Definition 2.1), a notion that was defined in [GRT18] and (as formally proved in [GRT18]) is closely related to the notion of two-source extractor. (The two notions are equivalent up to small changes in the parameters.)

As in [GRT18], our main result can be used to prove (two-pass) memory-samples lower bounds for many of the problems that were previously studied in this context. For example, for learning parities, sparse parities, DNFs, decision trees, random matrices, error correcting codes, etc. For example, our main result implies that any two-pass algorithm for learning parities of size n requires either a memory of size $\Omega(n^{1.5})$ or at least $2^{\Omega(\sqrt{n})}$ samples.

Related Work

To the best of our knowledge, the only previous work that proved memory-samples lower bounds for more than one pass over the stream of samples, is the intriguing recent work of Dagan and Shamir [DS18]. We note however that their results apply for a very different setting and regime of parameters, where the obtained lower bound on the number of samples is at most polynomial in the dimension of the problem. (Their result is proved in a very different setting, where the samples may be noisy, and the lower bound obtained on the number of samples is at most the product of the length of one sample times one over the information given by each sample).

Motivation and Discussion

Many previous works studied the resources needed for learning, under certain information, communication or memory constraints (see in particular [S14, SVW16, R16, VV16, KRT17, MM17, R17, MM18, MT17, BOGY18, GRT18, DS18] and the many references given there). A main message of some of these works is that for some learning problems, access to a relatively large memory is crucial. In other words, in some cases, learning is infeasible, due to memory constraints.

From the point of view of human learning, such results may help to explain the importance of memory in cognitive processes. From the point of view of machine learning, these results imply that a large class of learning algorithms cannot learn certain concept classes. In addition, these works are related to computational complexity and have applications in bounded-storage cryptography.

Most of these works apply to bounded-memory learning algorithms that consider the samples one by one, with only one pass over the samples. In many practical situations, however, more than one pass over the samples is used, so it's desirable to extend these results to more than one pass over the samples.

From the point of view of computational complexity, the problem of extending these works to more than one pass over the samples is fascinating and challenging. It's a common practice in streaming-complexity to consider more than one pass over the inputs, and in computational complexity read-k-times branching programs have attracted a lot of attention.

We note that by Barrington's celebrated result, any function in NC can be computed by a polynomial-length branching program of width 5 [B89]. Hence, proving super-polynomial lower bounds on the time needed for computing a function, by a branching program of width 5, with polynomially many passes over the input, would imply super-polynomial lower bounds for formula size, and is hence a very challenging problem.

Finally, let us mention that technically, allowing more than one pass over the samples is very challenging, as all previous techniques are heavily based on the fact that in the one-pass case all the samples are independent and hence at each time step, the learning algorithm has no information about the next sample that it is going to see.

Techniques

Our proof builds on the works of [R17, GRT18] that gave a general technique for proving memory-samples lower bounds for learning problems. However, these works (as well as all other previous works that prove memory-samples lower bounds in this regime of parameters) are heavily based on the fact that in the one-pass case all the samples are independent and hence at each time step, the learning algorithm has no information about the next sample that it is going to see. Roughly speaking, the proofs of [R17, GRT18] bound the L_2-norm of the distribution of x, conditioned on reaching a given vertex v of the branching program, but they rely on the fact that the next sample is independent of x. Once one allows more than one pass over the stream of samples, the assumption that the next sample is independent of x doesn't hold, as in the second pass the vertex may remember a lot of information about the joint distribution of x and $a_1, \ldots, a_m$.

Roughly speaking, [R17, GRT18] considered the computation-path of the branching program and defined "stopping-rules". Intuitively, the computation stops if certain "bad" events occur. The proofs show that each stopping rule is only applied with negligible probability and that conditioned on the event that the computation didn't stop, the L_2-norm of the distribution of x, conditioned on reaching a vertex v of the branching program, is small (which implies that the program didn't learn x).

When more than one pass over the samples is allowed, there is a serious problem with this approach. After one pass, a vertex of the branching program has joint information on x and $a_1, \ldots, a_m$. If we only keep track of the distribution of x conditioned on that vertex, it could be the case that the next sample completely reveals x. One conceptual problem seems to be that the second part of the program (that is, the part that is doing the second pass) is not aware of what the first part did. An idea that turned out to be very important in our proof is to take the second part to be the product of the first and second part, so that, in some sense, the second part of the computation runs its own copy of the first part. In addition, we have each vertex in the second part remembering the vertex reached at the end of the first part.

As in [R17, GRT18], we define stopping rules for the computation-path and we prove that the probability that the computation stops is small. We then analyze each part separately, as a read once program. For each part separately, we prove that conditioned on the event that the program didn't stop, the L_2-norm of the distribution of x, conditioned on reaching a vertex v, is small. It turns out that since the second part of the program runs its own copy of the first part, the analysis of each part separately is sufficient.

We note, however, that the entire proof is completely different than [R17, GRT18]. The stopping rules are different and are defined differently for each part. The proof that the computation stops with low probability is much more delicate and complicated. The main challenge is that when analyzing the probability to stop on the second part, we cannot ignore the first part and we need to prove that we stop with low probability on the second part, when starting from the start vertex of the first part (that is, the start vertex of the entire program). This turns out to be very challenging and, in particular, requires a use of the results for one-pass branching programs.

A proof outline is given in Section 4.

2 Preliminaries

Denote by log the logarithm to base 2. For a random variable Z and an event E, we denote by $\mathbb{P}_Z$ the distribution of the random variables Z, and we denote by $\mathbb{P}_{Z|E}$ the distribution of the random variable Z conditioned on the event E.

We will sometimes take probabilities and expectations, conditioned on events E that may be empty. We think of these probabilities and expectations as 0, when the event E is empty.

2.1 Learning Problem

We represent a learning problem by a matrix. Let X, A be two finite sets of size larger than 1 (where X represents the concept-class that we are trying to learn and A represents the set of possible samples). Let $M : A \times X \to \{-1, 1\}$ be a matrix. The matrix M represents the following learning problem: An unknown element $x \in X$ was chosen uniformly at random. A learner tries to learn x from a stream of samples, $(a_1, b_1), (a_2, b_2) \ldots$, where for every i, $a_i \in A$ is chosen uniformly at random and $b_i = M(a_i, x)$.

Let $n = \log |X|$ and $n' = \log |A|$.

2.2 Norms and Inner Products

Let $p \geq 1$. For a function $f : X \to \mathbb{R}$, denote by $\|f\|_p$ the L_p norm of f, with respect to the uniform distribution over X, that is:

$$\|f\|_p = \left(\mathop{\mathbf{E}}_{x \in_R X} [|f(x)|^p] \right)^{1/p}.$$

For two functions $f, g : X \to \mathbb{R}$, define their inner product with respect to the uniform distribution over X as

$$\langle f, g \rangle = \mathop{\mathbf{E}}_{x \in_R X} [f(x) \cdot g(x)].$$

For a matrix $M : A \times X \to \mathbb{R}$ and a row $a \in A$, we denote by $M_a : X \to \mathbb{R}$ the function corresponding to the a-th row of M. Note that for a function $f : X \to \mathbb{R}$, we have $\langle M_a, f \rangle = \frac{(M \cdot f)_a}{|X|}$.

2.3 L_2-Extractors

Definition 2.1. **L_2-Extractor:** *Let X, A be two finite sets. A matrix $M : A \times X \to \{-1, 1\}$ is a (k, ℓ)-L_2-Extractor with error 2^{-r}, if for every non-negative $f : X \to \mathbb{R}$ with $\frac{\|f\|_2}{\|f\|_1} \leq 2^{\ell}$ there are at most $2^{-k} \cdot |A|$ rows a in A with*

$$\frac{|\langle M_a, f \rangle|}{\|f\|_1} \geq 2^{-r} \; .$$

2.4 Computational Model

In the following definition, we model the learner for the learning problem that corresponds to the matrix M, by a *branching program.* We consider a q-pass ordered branching program. Such a program reads the entire input q times, in the exact same order. That is, the program has q parts (that are sequential in time). Each part reads the same stream in the exact same order. Our main result is proved for two-pass ordered branching programs, that is, for the case $q = 2$.

Definition 2.2. **q-Pass Branching Program for a Learning Problem:** *A q-pass (ordered) branching program of length $q \cdot m$ and width d, for learning, is a directed (multi) graph with vertices arranged in $qm + 1$ layers containing at most d vertices each. In the first layer, that we think of as layer 0, there is only one vertex, called the start vertex. A vertex of outdegree 0 is called a leaf. All vertices in the last layer are leaves (but there may be additional leaves). Every non-leaf vertex in the program has $2|A|$ outgoing edges, labeled by elements $(a, b) \in A \times \{-1, 1\}$, with exactly one edge labeled by each such (a, b), and all these edges going into vertices in the next layer. Each leaf v in the program is labeled by an element $\tilde{x}(v) \in X$, that we think of as the output of the program on that leaf.*

Computation-Path: *The samples $(a_1, b_1), \ldots, (a_m, b_m) \in A \times \{-1, 1\}$ that are given as input define a computation-path in the branching program, by starting from the start vertex and following at step $(j - 1) \cdot m + i$ the edge labeled by (a_i, b_i) (where $j \in [q]$ and $i \in [m]$), until reaching a leaf. The program outputs the label $\tilde{x}(v)$ of the leaf v reached by the computation-path.*

Success Probability: *The success probability of the program is the probability that $\tilde{x} = x$, where $\tilde{x}$ is the element that the program outputs, and the probability is over $x, a_1, \ldots, a_m$ (where x is uniformly distributed over X and $a_1, \ldots, a_m$ are uniformly distributed over A, and for every i, $b_i = M(a_i, x)$).*

Remark: *We will sometimes consider branching programs in which the leaves are not labeled, and hence the program doesn't return any value. It will be convenient to refer to such objects also as branching programs. In particular, we will view a part of the branching program (e.g., the first few layers of a program) also as a branching program.*

We think of the program as composed of q parts, where for every $j \in [q]$, part-j contains layers $\{(j - 1) \cdot m + i\}_{i \in [m]}$.

For convenience, we think of each vertex u of the branching program as having a small memory S_u that contains some information about the path that led to the vertex, that the vertex "remembers" (or "records"). Formally, this means that in the actual branching program the vertex u is split into distinct vertices $u_1, \dots, u_{d(u)}$, according to the content of the memory S_u. Adding information to S_u means that the vertex u is further split into distinct vertices, according to the content of the information that was added. Thus, when we refer to a vertex u of a program, we mean, a vertex u plus content of the memory S_u.

In this paper, we will have the property that whenever we add some information to the memory of a vertex u, that information is never removed/forgotten. That is, information that was added to the memory of u, remains in the memory of all the vertices that can be reached from u.

As mentioned above, in this paper we focus on the case $q = 2$. We denote by v_0 the start vertex of the program and by v_1 the vertex reached at the end of the first part, that is, layer-m. Note that v_1 is a random variable that depends on $x, a_1, \dots, a_m$.

2.5 Product of Programs

Intuitively, the product of two branching programs is a branching program that runs both programs in parallel.

Definition 2.3. Product of One-Pass Branching Programs: *Let B, B' be two one-pass branching programs for learning, of length m and widths d, d', respectively. The product $B \times B'$ is a (one-pass) branching program of length m and width $d \cdot d'$, as follows: For every $i \in \{0, \dots, m\}$ and vertices v in layer-i of B and v' in layer-i of B', we have a vertex (v, v') in layer-i of $B \times B'$. For every two edges: (u, v) from layer-$(i-1)$ to layer-i of B and (u', v') from layer-$(i-1)$ to layer-i of B', both labeled by the same (a, b), we have in $B \times B'$ an edge $((u, u'), (v, v'))$ labeled by (a, b).*

The label of a leaf (v, v') is the label given by the second program B'. The content of the memory $S_{(v,v')}$ of a vertex (v, v') is the concatenation of the content of S_v and the content of $S_{v'}$.

Remark: *We will use this definition also in cases where the leaves of B and/or B' are not labeled (that is, where B and/or B' do not output any value; see a remark in Definition 2.2).*

3 Main Result

Fix $k, \ell, r \in \mathbb{N}$, such that $\frac{r}{k}, \frac{r}{\ell}$ are smaller than a sufficiently small constant and $k < n'$, $\ell < n$. Let $\epsilon > 0$ be a sufficiently small constant. In particular, we assume that ϵ is sufficiently smaller than all other constants that we discuss, say, $\epsilon < \frac{1}{10^{10}}$. We assume that n, n' are sufficiently large. Let

$$\tilde{\ell} = \min\left\{k, \sqrt{\ell}\right\}.$$

Let

$$\tilde{r} = \min\left\{\tfrac{r}{100}, \tfrac{\tilde{\ell}}{100}\right\}. \tag{1}$$

We assume that

$$\tilde{r} > 100 \cdot \max\left\{\log n, \log n'\right\}. \tag{2}$$

We assume that M is a $(10k, 10\ell)$-L_2-extractor with error 2^{-10r}.

Theorem 1. *Let X, A be two finite sets. Let $n = \log_2 |X|$ and $n' = \log_2 |A|$. Fix $k, \ell, r \in \mathbb{N}$, such that, $\frac{r}{k}, \frac{r}{\ell} < \frac{1}{100}$, and $k < n'$, $\ell < n$. Let $\epsilon > 0$ be a sufficiently small constant, say, $\epsilon < \frac{1}{10^{10}}$. Assume that n, n' are sufficiently large. Let*

$$\tilde{\ell} = \min\left\{k, \sqrt{\ell}\right\}.$$

Let

$$\tilde{r} = \min\left\{\tfrac{r}{100}, \tfrac{\tilde{\ell}}{100}\right\}.$$

Assume that

$$\tilde{r} > 100 \cdot \max\left\{\log n, \log n'\right\}.$$

Let $M : A \times X \to \{-1, 1\}$ be a matrix which is a $(10k, 10\ell)$-L_2-extractor with error 2^{-10r}. Let B be a two-pass ordered branching program of length $2 \cdot m$, where m is at most $2^{\epsilon \tilde{r}}$, and width at most $d = 2^{\epsilon k \tilde{\ell}/10}$, for the learning problem that corresponds to the matrix M. Then, the success probability of B is at most $\frac{1}{100} + o(1)$.

4 Overview of the Proof

One-Pass Learners

We will start with giving a short outline of the proof of [R17, GRT18] for one-pass learners. Assume that M is a $(10k, 10\ell)$-L_2-extractor with error 2^{-10r}, where $r < k, \ell$. Let B be a one-pass branching program for the learning problem that corresponds to the matrix M. Assume for a contradiction that B is of length $m = 2^{\epsilon r}$ and width $d = 2^{\epsilon k \ell}$, where ϵ is a small constant.

We define the *truncated-path*, $\mathcal{T}$, to be the same as the computation-path of B, except that it sometimes stops before reaching a leaf. Roughly speaking, $\mathcal{T}$ stops before reaching a leaf if certain "bad" events occur. Nevertheless, we show that the probability that $\mathcal{T}$ stops before reaching a leaf is negligible, so we can think of $\mathcal{T}$ as almost identical to the computation-path.

For a vertex v of B, we denote by E_v the event that $\mathcal{T}$ reaches the vertex v. We denote by $\Pr(v) = \Pr(E_v)$ the probability for E_v (where the probability is over $x, a_1, \ldots, a_m$), and we denote by $\mathbb{P}_{x|v} = \mathbb{P}_{x|E_v}$ the distribution of the random variable x conditioned on the event E_v. Similarly, for an edge e of the branching program B, let E_e be the event that $\mathcal{T}$ traverses the edge e. Denote, $\Pr(e) = \Pr(E_e)$, and $\mathbb{P}_{x|e} = \mathbb{P}_{x|E_e}$.

A vertex v of B is called *significant* if

$$\left\|\mathbb{P}_{x|v}\right\|_2 > 2^{\ell} \cdot 2^{-n}.$$

Roughly speaking, this means that conditioning on the event that $\mathcal{T}$ reaches the vertex v, a non-negligible amount of information is known about x. In order to guess x with a non-negligible success probability, $\mathcal{T}$ must reach a significant vertex. We show that the probability that $\mathcal{T}$ reaches any significant vertex is negligible, and thus the main result follows.

To prove this, we show that for every fixed significant vertex s, the probability that $\mathcal{T}$ reaches s is at most $2^{-\Omega(k\ell)}$ (which is smaller than one over the number of vertices in B). Hence, we can use a union bound to prove the bound.

The proof that the probability that $\mathcal{T}$ reaches s is extremely small is the main part of the proof. To that end, we use the following functions to measure the progress made by the branching program towards reaching s.

Let L_i be the set of vertices v in layer-i of B, such that $\Pr(v) > 0$. Let Γ_i be the set of edges e from layer-$(i-1)$ of B to layer-i of B, such that $\Pr(e) > 0$. Let

$$\mathcal{Z}_i = \sum_{v \in L_i} \Pr(v) \cdot \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k,$$

$$\mathcal{Z}'_i = \sum_{e \in \Gamma_i} \Pr(e) \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k.$$

We think of $\mathcal{Z}_i, \mathcal{Z}'_i$ as measuring the progress made by the branching program, towards reaching a state with distribution similar to $\mathbb{P}_{x|s}$.

We show that each $\mathcal{Z}_i$ may only be negligibly larger than $\mathcal{Z}_{i-1}$. Hence, since it's easy to calculate that $\mathcal{Z}_0 = 2^{-2nk}$, it follows that $\mathcal{Z}_i$ is close to 2^{-2nk}, for every i. On the other hand, if s is in layer-i then $\mathcal{Z}_i$ is at least $\Pr(s) \cdot \langle \mathbb{P}_{x|s}, \mathbb{P}_{x|s} \rangle^k$. Thus, $\Pr(s) \cdot \langle \mathbb{P}_{x|s}, \mathbb{P}_{x|s} \rangle^k$ cannot be much larger than 2^{-2nk}. Since s is significant, $\langle \mathbb{P}_{x|s}, \mathbb{P}_{x|s} \rangle^k > 2^{\ell k} \cdot 2^{-2nk}$ and hence $\Pr(s)$ is at most $2^{-\Omega(k\ell)}$.

The proof that $\mathcal{Z}_i$ may only be negligibly larger than $\mathcal{Z}_{i-1}$ is done in two steps. We show by a simple convexity argument that $\mathcal{Z}_i \leq \mathcal{Z}'_i$. The hard part is to prove that $\mathcal{Z}'_i$ may only be negligibly larger than $\mathcal{Z}_{i-1}$.

For this proof, we define for every vertex v, the set of edges $\Gamma_{out}(v)$ that are going out of v, such that $\Pr(e) > 0$ and show that for every vertex v,

$$\sum_{e \in \Gamma_{out}(v)} \Pr(e) \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k$$

may only be negligibly higher than

$$\Pr(v) \cdot \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k.$$

For this proof, we consider the function $\mathbb{P}_{x|v} \cdot \mathbb{P}_{x|s}$. We first show how to bound $\left\| \mathbb{P}_{x|v} \cdot \mathbb{P}_{x|s} \right\|_2$. We then consider two cases: If $\left\| \mathbb{P}_{x|v} \cdot \mathbb{P}_{x|s} \right\|_1$ is negligible, then $\langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k$ is negligible and doesn't contribute much, and we show that for every $e \in \Gamma_{out}(v)$, $\langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k$ is also negligible and doesn't contribute much. If $\left\| \mathbb{P}_{x|v} \cdot \mathbb{P}_{x|s} \right\|_1$ is non-negligible, we use the bound on $\left\| \mathbb{P}_{x|v} \cdot \mathbb{P}_{x|s} \right\|_2$ and the assumption that M is a $(10k, 10\ell)$-L_2-extractor to show that for almost all edges $e \in \Gamma_{out}(v)$, we have that $\langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k$ is very close to $\langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k$. Only an exponentially small (2^{-k}) fraction of edges are "bad" and give a significantly larger $\langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k$.

The reason that in the definitions of $\mathcal{Z}_i$ and $\mathcal{Z}'_i$ we raised $\langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle$ and $\langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle$ to the power of k is that this is the largest power for which the contribution of the "bad" edges is still small (as their fraction is 2^{-k}).

This outline oversimplifies many details. Let us briefly mention two of them. First, it is not so easy to bound $\left\| \mathbb{P}_{x|v} \cdot \mathbb{P}_{x|s} \right\|_2$. We do that by bounding $\left\| \mathbb{P}_{x|s} \right\|_2$ and $\left\| \mathbb{P}_{x|v} \right\|_\infty$. In order to bound $\left\| \mathbb{P}_{x|s} \right\|_2$, we force $\mathcal{T}$ to stop whenever it reaches a significant vertex (and thus we are able to bound $\left\| \mathbb{P}_{x|v} \right\|_2$ for every vertex reached by $\mathcal{T}$). In order to bound $\left\| \mathbb{P}_{x|v} \right\|_\infty$, we force $\mathcal{T}$ to stop whenever $\mathbb{P}_{x|v}(x)$ is large, which allows us to consider only the "bounded" part of $\mathbb{P}_{x|v}$. (This is related to the technique of *flattening* a distribution that was used in [KR13]). Second, some edges are so "bad" that their contribution to $\mathcal{Z}'_i$ is huge so they cannot be ignored. We force $\mathcal{T}$ to stop before traversing any such edge. (This is related to an idea that was used in [KRT17] of analyzing separately paths that traverse "bad" edges). We show that the total probability that $\mathcal{T}$ stops before reaching a leaf is negligible.

Thus, in [R17, GRT18] there are three stopping rules: We stop if we reach a **significant vertex**. We stop if we have a **bad edge** and we stop if x is a **significant-value** of $\mathbb{P}_{x|v}$, that is, if $\mathbb{P}_{x|v}(x)$ is too large.

Two-Pass Learners

Let us now give a short outline of the additional ideas in the proof for two-pass learners. Let B be a two-pass branching program for the learning problem that corresponds to the matrix M. We denote by v_0 the starting vertex of the program and by v_1 the vertex reached at the end of the first part. We assume without loss of generality that the answers are given in the last layer of the program.

We update the second part so that every vertex v in the second part "remembers" v_1. This information is stored in the memory S_v. Formally, this means that starting from every possible v_1, we have a separate copy of the entire second part of the program. We then change the second part so that it is now the **product** (see definition 2.2) of the first part and the second part. Intuitively, this means that the second part runs a copy of the first part of the computation, in parallel to its own computation.

As in [R17, GRT18], we define the *truncated-path*, $\mathcal{T}$, to be the same as the computation-path of the new branching program, except that it sometimes stops before reaching a leaf. Roughly speaking, $\mathcal{T}$ stops before reaching a leaf if certain "bad" events occur. Nevertheless, we show that the probability that $\mathcal{T}$ stops before reaching a leaf is small, so we can think of $\mathcal{T}$ as essentially identical to the computation-path. The decision of whether or not $\mathcal{T}$ stops on a given vertex v in layer-i of part-j will depend on v, S_v, x, a_{i+1}. For that reason, we are able to consider the path $\mathcal{T}$, starting from any vertex v (without knowing the history of the path that led to v, except for the information stored in S_v).

Let v be a vertex in the second part of the program (where an answer should be given). The vertex v remembers (in S_v) the vertex v_1. We denote by $v_1 \to v$ the event that the path $\mathcal{T}$ that starts from v_1 reaches v (where v_1 is the vertex at the end of the first part of the program that v remembers, and the event is over $x, a_1, \ldots, a_m$). We denote by $v_0 \to v$ the event that the path $\mathcal{T}$ that starts from the start vertex v_0 reaches v. More generally, for two vertices w_1, w_2 in the program, we denote by $w_1 \to w_2$ the event (over $x, a_1, \ldots, a_m$) that the path $\mathcal{T}$ that starts from w_1 reaches w_2.

Let v be a vertex in the last layer of the program, such that $\Pr(v_0 \to v) > 0$. Since v remembers v_1, the event $v_0 \to v$ is equivalent to $v_0 \to v_1 \to v$ (where v_1 is the vertex remembered by v). Since the second part of the program runs a copy of the first part and since v is in the last layer, the event $v_1 \to v$ implies the event $v_0 \to v_1$. Thus, the event $v_0 \to v$ is equivalent to $v_1 \to v$.

Moreover, this is true when conditioning on x, and hence,

$$\mathbb{P}_{x|v_0 \to v} = \mathbb{P}_{x|v_1 \to v}$$

and

$$\Pr[v_0 \to v] = \Pr[v_1 \to v].$$

This is a crucial point as it means that

$$\left\|\mathbb{P}_{x|v_0 \to v}\right\|_2 = \left\|\mathbb{P}_{x|v_1 \to v}\right\|_2 ,$$

that is, if we bound $\left\|\mathbb{P}_{x|v_1 \to v}\right\|_2$ we also get a bound on $\left\|\mathbb{P}_{x|v_0 \to v}\right\|_2$.

The bound on $\left\|\mathbb{P}_{x|v_0 \to v}\right\|_2$ is what we really need because if this is small then the program cannot answer correctly. On the other hand, the bound on $\left\|\mathbb{P}_{x|v_1 \to v}\right\|_2$ is easier to obtain as it is a bound for a one-pass branching program. Thus, all we need is a bound on $\left\|\mathbb{P}_{x|v_1 \to v}\right\|_2$, which is a bound for a one-pass branching program, and we already know how to obtain bounds on the conditional distribution for one-pass programs.

Things, however, are not so simple, as we need to prove that $\mathcal{T}$ stops with small probability, when starting from v_0, rather than v_1. The main problem with using the previous stopping rules (in the second part of the program) is that it's impossible to prove that we stop on a bad edge with negligible probability (as demonstrated next). Roughly speaking, we say that an edge $e = (u, v)$ is "bad" if the equation on it splits the distribution $\mathbb{P}_{x|u}$ in a biased way. That is, a good edge is one where roughly half the probability mass of $\mathbb{P}_{x|u}$ satisfies the equation on the edge e. If the program stores in memory the i-th sample from the first pass, then in the i-th step of the second pass, an edge $e = (u, v)$ will definitely be bad, since it will not split the distribution $\mathbb{P}_{x|u}$ evenly.

For that reason, we change the bad-edges stopping rule. We say that an edge (v, u), labelled by (a, b), is of *high probability* if the probability to sample a, conditioning on reaching v from v_0 (that is, reaching v from the starting vertex of the entire program) is large. The third stopping rule is changed so that $\mathcal{T}$ doesn't stop on a bad edge if it is of high probability. Instead, if $\mathcal{T}$ traverses such an edge, we "remember" the time step in which $\mathcal{T}$ traversed that edge, in all the future. That is, we enter the index i to S_u (and remember it in all the future, until the end of the program). In addition, we add a stopping rule that stops if the edge is "very-bad" and a stopping rule that stops if the number of indices in S_v is too large, that is, if the number of high-probability edges that were already traversed is too large (intuitively, S_v won't be too large because of the bounded memory size).

We analyze separately the probability to stop because of each stopping rule. The main challenge is that we need to analyze these probabilities when starting from v_0, that is, when running a two-pass program. These proofs are technically hard, but the main reason that we manage to analyze these probabilities is the following:

Recall that the second part of the program runs a copy of the first part of the program. Thus, a vertex v in layer-i of the second part has a corresponding vertex v' in layer-i of the first part, such that, if the path $\mathcal{T}$ reached v it previously reached v'. Recall also that v remembers v_1, so if the path $\mathcal{T}$ reached v it previously reached v_1. Thus, the event $v_0 \to v$ is equivalent to $v_0 \to v' \to v_1 \to v$, that is, the event

$$(v_0 \to v') \wedge (v' \to v_1) \wedge (v_1 \to v).$$

Since the second part of the program runs a copy of the first part, the event $v_1 \to v$ implies the event $v_0 \to v'$. Hence, the event $v_0 \to v$ is equivalent to the event

$$(v' \to v_1) \wedge (v_1 \to v).$$

Note that v' is in layer-i of the first part and v is in layer-i of the second part, and from layer-i of the first part to layer-i of the second part, the program is a one-pass program and is hence easier to analyze.

5 Proof of Theorem 1

Assume that we have a two-pass ordered branching program, B, for the learning problem that corresponds to the matrix M. We assume without loss of generality that the output is given in the last layer. Assume that the length of the program is $2 \cdot m$, where m is at most $2^{\epsilon\tilde{r}}$ and the width of the program is at most $d = 2^{\epsilon k \tilde{\ell}/10}$. We will show that the success probability of B is at most $\frac{1}{100} + o(1)$.

Let

$$\ell_1 = \frac{\tilde{\ell}}{100}$$

and

$$\ell_2 = \ell.$$

5.1 The Truncated Path

Below, we will make some changes in the branching program B. We will denote by $\hat{B}$ the resulting branching program. Let v_0 be the start vertex of $\hat{B}$. We will denote by v_1 the vertex reached at the end of the first part of $\hat{B}$. Note that v_1 is a random variable, that depends on $x, a_1, \ldots, a_m$.

In the resulting branching program $\hat{B}$, we will have the property that the vertex v_1 reached at the end of the first part of the program is remembered by every future vertex v. That is, every vertex v, in the second part of the program, remembers which vertex the path that led to v reached, at the end of the first part of the program. Formally, this information is stored in S_v.

Below, we will define the *truncated-path*, $\mathcal{T}$, to be the same as the computation-path of the new branching program, except that it sometimes stops before reaching a leaf. Roughly speaking, $\mathcal{T}$ stops before reaching a leaf if certain "bad" events occur. Nevertheless, we show that the probability that $\mathcal{T}$ stops before reaching a leaf is small, so we can think of $\mathcal{T}$ as essentially identical to the computation-path. The decision of whether or not $\mathcal{T}$ stops on a given vertex v in layer-i of part-j will depend on v, S_v, x, a_{i+1}. For that reason, we are able to consider the path $\mathcal{T}$, starting from any vertex v (without knowing the history of the path that led to v, except for the information stored in S_v).

Let v be a vertex in the second part of the program. The vertex v remembers (in S_v) the vertex v_1. We denote by $v_1 \to v$ the event that the path $\mathcal{T}$ that starts from v_1 reaches v (where v_1 is the vertex at the end of the first part of the program that v remembers, and the event is over $x, a_1, \ldots, a_m$).

More generally, for two vertices w_1, w_2 in the program, we denote by $w_1 \to w_2$ the event (over $x, a_1, \ldots, a_m$) that the path $\mathcal{T}$ that starts from w_1 reaches w_2. In particular, $v_0 \to v$ is the event that the path $\mathcal{T}$ that starts from the start vertex v_0 reaches v.

We change the original branching program B as follows:

First Part:

We define stopping rules for the first part as defined in [R17, GRT18] for one-pass programs, as if the first part were the entire program. Next, we describe these rules formally.

Significant Vertices

We say that a vertex v in layer-i of the first part of the program is **significant** if

$$\left\|\mathbb{P}_{x|v_0\to v}\right\|_2 > 2^{\ell_1} \cdot 2^{-n}.$$

Significant Values

Even if v is not significant, $\mathbb{P}_{x|v_0\to v}$ may have relatively large values. For a vertex v in layer-i of the first part of the program, denote by $\mathrm{Sig}(v)$ the set of all $x' \in X$, such that,

$$\mathbb{P}_{x|v_0\to v}(x') > 2^{4\ell} \cdot 2^{-n}.$$

Bad Edges

For a vertex v in layer-i of the first part of the program, denote by $\mathrm{Bad}(v)$ the set of all $a \in A$, such that,

$$\left|(M \cdot \mathbb{P}_{x|v_0\to v})(a)\right| \geq 2^{-2r}.$$

The Truncated-Path $\mathcal{T}$ on the First Part

We define $\mathcal{T}$ on the first part, by induction on the layers. Assume that we already defined $\mathcal{T}$ until it reaches a vertex v in layer-i of the first part. The path $\mathcal{T}$ stops on v if (at least) one of the following occurs:

1. v is significant.
2. $x \in \mathrm{Sig}(v)$.
3. $a_{i+1} \in \mathrm{Bad}(v)$.

Otherwise, (unless $i = m$) $\mathcal{T}$ proceeds by following the edge labeled by (a_{i+1}, b_{i+1}) (same as the computational-path).

Second Part:

We denote by v_1 the vertex in layer-m (that is, the last layer of the first part of the program) that is reached by $\mathcal{T}$. Note that v_1 is a random variable that depends on $x, a_1, \ldots, a_m$. We denote by d_1 the number of vertices in layer-m. We assume without loss of generality that each vertex in layer-m is reached with probability of at least $2^{-10\tilde{r}} \cdot d_1^{-1}$, as vertices reached with negligible probability can be ignored. Formally, if we reach a vertex in layer-m, such that, the probability to reach that vertex is smaller than $2^{-10\tilde{r}} \cdot d_1^{-1}$, the path $\mathcal{T}$ stops.

We update the second part so that every vertex v in the second part "remembers" v_1. This information is stored in the memory S_v. Formally, this means that starting from every possible v_1, we have a separate copy of the entire second part of the program.

We then change the second part so that it is now the **product** (see definition 2.2) of the first part (after it was changed as described above) and the second part. Intuitively, this means that the second part runs a copy of the first part of the computation, in parallel to its own computation.

Next, we define stopping rules for the second part, by induction over the layers, and at the same time (by the same induction), we also define for each vertex v, a list L_v of indices $i_1, \dots, i_{d(v)} \in [m]$ that the vertex v remembers (that is, the list L_v is stored in the memory S_v). Once an index was added to L_v, it is remembered in all the future, that is, for every vertex u reached from v (in the second part of the program), we have $L_v \subseteq L_u$. Note that the stopping rules are defined for the updated second part (as described above).

The stopping rules for the second part extend the stopping rules in the case of one-pass programs, as defined in [R17, GRT18], as if the second part were the entire program, with starting vertex v_1. However, the third stopping rule (*bad* edges) is now different. We say that an edge (v, u), labelled by (a, b), is of *high probability* if the probability to sample a, conditioning on reaching v from v_0 (that is, reaching v from the starting vertex of the entire program) is larger than $2^k \cdot 2^{-n'}$. That is, if v is in layer-i of the second part, (v, u) is of high probability if $\Pr[a_{i+1} = a \mid v_0 \to v] \geq 2^k \cdot 2^{-n'}$. The third stopping rule is changed so that $\mathcal{T}$ doesn't stop on a bad edge if it is of high probability. Instead, if $\mathcal{T}$ traverses such an edge, we "remember" the time step in which $\mathcal{T}$ traversed that edge, in all the future. That is, we enter the index i to L_u (and remember it in all the future, until the end of the program). In addition, we add a stopping rule that stops if the edge is "very-bad" and a stopping rule that stops if the number of indices in L_v is too large, that is, if the number of high-probability edges that were already traversed is too large.

Next, we describe these rules formally. We initiate $L_{v_1} = \emptyset$.

Significant Vertices

We say that a vertex v in layer-i of the second part of the program is **significant** if

$$\left\| \mathbb{P}_{x|v_1 \to v} \right\|_2 > 2^{\ell_2} \cdot 2^{-n}.$$

Significant Values

For a vertex v in layer-i of the second part of the program, denote by $\mathrm{Sig}(v)$ the set of all $x' \in X$, such that,

$$\mathbb{P}_{x|v_1 \to v}(x') > 2^{4\ell} \cdot 2^{-n}.$$

Bad Edges

For a vertex v in layer-i of the second part of the program, denote by $\mathrm{Bad}(v)$ the set of all $a \in A$, such that,

$$\left| (M \cdot \mathbb{P}_{x|v_1 \to v})(a) \right| \geq 2^{-2r}.$$

Very-Bad Edges

For a vertex v in layer-i of the second part of the program, denote by VeryBad(v) the set of all $(a,b) \in A \times \{-1,1\}$, such that,

$$\Pr_x[M(a,x) = b \mid v_1 \to v] \leq 2^{-4\tilde{\ell}}.$$

High-Probability Edges

For a vertex v in layer-i of the second part of the program, denote by High(v) the set of all $a \in A$, such that,

$$\Pr[a_{i+1} = a \mid v_0 \to v] \geq 2^k \cdot 2^{-n'}.$$

The Truncated-Path $\mathcal{T}$ on the Second Part

We define $\mathcal{T}$ on the second part, by induction on the layers. Assume that we already defined $\mathcal{T}$ until it reaches a vertex v in layer-i of the second part (and we already defined L_v). The path $\mathcal{T}$ stops on v if (at least) one of the following occurs:

1. v is significant.
2. $x \in \mathrm{Sig}(v)$.
3. $a_{i+1} \in \mathrm{Bad}(v) \setminus \mathrm{High}(v)$.
4. $(a_{i+1}, b_{i+1}) \in \mathrm{VeryBad}(v)$.
5. $|L_v| \geq 200\epsilon\tilde{\ell}$.
6. Recall that we changed the second part of the program so that it is the product of the first part and the (original) second part. This means that the second part of the program runs its own copy of the first part of the program. If the path $\mathcal{T}$, that was defined for the first part, stops on the copy of the first part that the second part runs, the path $\mathcal{T}$ stops on the vertex v too.

   Remark 5.1. *We note that if $\mathcal{T}$ stopped on the first part, it couldn't have reached v_1 in the first place. Thus, conditioned on the event $v_0 \to v_1$, the path $\mathcal{T}$ didn't stop on the first part. Therefore, conditioned on the event $v_0 \to v_1$, the path $\mathcal{T}$ never stops because of stopping rule 6. Thus, this stopping rule is not necessary. Nevertheless, we add this stopping rule for completeness, so that it would be possible to consider the path $\mathcal{T}$ starting from any vertex (even in the middle of the program), without conditioning on the event of reaching that vertex.*

Otherwise, unless $\mathcal{T}$ already reached the end of the second part, $\mathcal{T}$ proceeds by following the edge labeled by (a_{i+1}, b_{i+1}) (same as the computational-path). Let (v,u) be the edge labeled by (a_{i+1}, b_{i+1}). It remains to define the list L_u.

Updating L_u

Let (v,u) be the traversed edge, labeled by (a_{i+1},b_{i+1}). If the traversed edge (v,u) is not a high-probability edge, that is, if $a_{i+1} \notin \mathrm{High}(v)$, we define $L_u = L_v$.

If the traversed edge (v,u) is a high-probability edge, that is, if $a_{i+1} \in \mathrm{High}(v)$, we define $L_u = L_v \cup \{i+1\}$, and hence, by induction, L_u is the list of all indices corresponding to the high-probability edges that $\mathcal{T}$ traversed, in the second part of the program, until reaching u.

5.2 Bounding the Width of the Branching Program $\hat{B}$

From now on, we will only consider the final branching program, $\hat{B}$.

The final branching program, $\hat{B}$, has a larger width than the original one. The main contributions to the larger width is that we changed the second part to be the product of the first and second parts of the original program and that each vertex in the second part of $\hat{B}$ remembers the vertex v_1 (reached at the end of the first part). This multiplies the memory needed (that is, the logarithm of the width of the program) by a factor of at most 3. In addition, each vertex v has to remember L_v, but by Equation (1) and since $\mathcal{T}$ stops when $|L_v| \geq 200\epsilon\tilde{\ell}$, this adds memory of at most $\frac{\epsilon\tilde{\ell}r}{100}$. Thus, the final width of $\hat{B}$ is at most $2^{\epsilon k\tilde{\ell}/2}$.

5.3 The Probability that $\mathcal{T}$ Stops is Small

We will now prove that the probability that $\mathcal{T}$ stops before reaching a leaf is at most $\frac{1}{100}+o(1)$.

Lemma 5.2. *The probability that $\mathcal{T}$ stops before reaching a leaf is at most $\frac{1}{100}+o(1)$.*

Proof. First, recall that if $\mathcal{T}$ reaches a vertex in layer-m, such that, the probability to reach that vertex is smaller than $2^{-10\tilde{r}} \cdot d_1^{-1}$, then $\mathcal{T}$ stops. By the union bound, the probability that $\mathcal{T}$ stops because of this rule is at most $2^{-10\tilde{r}} = o(1)$.

We will now bound the probability that $\mathcal{T}$ stops because of each of the other stopping rules.

Recall that for two vertices w_1, w_2 in the program, we denote by $w_1 \to w_2$ the event (over $x, a_1, \ldots, a_m$) that the path $\mathcal{T}$ that starts from w_1 reaches w_2.

5.3.1 Stopping Rule 1: Significant-Vertices

Lemma 5.3. *The probability that $\mathcal{T}$ reaches a significant vertex is at most $o(1)$.*

Lemma 5.3 is proved in Section 6.

Next, we will bound the probability that $\mathcal{T}$ stops because of each of the other stopping rules. By Lemma 5.3, it's sufficient to bound these probabilities, under the assumption that $\mathcal{T}$ doesn't reach any significant vertex (as otherwise, $\mathcal{T}$ would have stopped because of stopping rule 1).

5.3.2 Stopping Rule 2: Significant-Values

We will now bound the probability that $\mathcal{T}$ stops because of stopping rule 2. We will first prove the following claim.

Claim 5.4. *If v is a non-significant vertex in layer-i of part-j (where $j \in \{1,2\}$), then*

$$\Pr_x[x \in \mathrm{Sig}(v) \mid v_{j-1} \to v] \leq 2^{-2\ell}.$$

Proof. Since v is not significant,

$$\mathop{\mathbf{E}}_{x' \sim \mathbb{P}_{x|v_{j-1}\to v}} \left[\mathbb{P}_{x|v_{j-1}\to v}(x')\right] = \sum_{x' \in X} \left[\mathbb{P}_{x|v_{j-1}\to v}(x')^2\right] = 2^n \cdot \mathop{\mathbf{E}}_{x' \in_R X} \left[\mathbb{P}_{x|v_{j-1}\to v}(x')^2\right] \leq 2^{2\ell_j} \cdot 2^{-n}.$$

Hence, by Markov's inequality,

$$\Pr_{x' \sim \mathbb{P}_{x|v_{j-1}\to v}} \left[\mathbb{P}_{x|v_{j-1}\to v}(x') > 2^{4\ell} \cdot 2^{-n}\right] \leq 2^{2\ell_j - 4\ell} \leq 2^{-2\ell}.$$

Since conditioned on the event $v_{j-1} \to v$, the distribution of x is $\mathbb{P}_{x|v_{j-1}\to v}$, we obtain

$$\Pr_x\left[x \in \mathrm{Sig}(v) \;\middle|\; v_{j-1} \to v\right] = \Pr_x\left[\left(\mathbb{P}_{x|v_{j-1}\to v}(x) > 2^{4\ell} \cdot 2^{-n}\right) \;\middle|\; v_{j-1} \to v\;\right] \leq 2^{-2\ell}. \qquad \square$$

By Claim 5.4, if v is a non-significant vertex in layer-i of part-j then

$$\Pr_x[x \in \mathrm{Sig}(v) \mid v_{j-1} \to v] \leq 2^{-2\ell} \leq 2^{-4\tilde{\ell}}. \tag{3}$$

We need to bound from above

$$\mathop{\mathbf{E}}_v\left[\Pr_x[x \in \mathrm{Sig}(v) \mid v_0 \to v]\right], \tag{4}$$

where the expectation is over the non-significant vertices v in layer-i of part-j, reached by the path $\mathcal{T}$. (If $\mathcal{T}$ stops before reaching layer-i of part-j, or if it reaches a significant vertex, we think of v as undefined and think of the inner probability as 0). If $j = 1$, we are done by Claim 5.4. We will proceed with the case $j = 2$. Recall that $\ell_2 = \ell$.

We will use the following lemma, whose proof is deferred to the next subsection. We shall instantiate the lemma by setting $\mathcal{S}_v = \mathrm{Sig}(v)$.

Lemma 5.5. *Assume that for every non-significant vertex v in layer-i of part-2, we have some subset of values $\mathcal{S}_v \subseteq X$ that depends only on v. Assume that for every such v (with positive probability for the event $v_1 \to v$, where v_1 is the vertex recorded by v), we have*

$$\Pr_x[x \in \mathcal{S}_v \mid v_1 \to v] \leq 2^{-4\tilde{\ell}}.$$

Then,

$$\mathop{\mathbf{E}}_v[\Pr_x[x \in \mathcal{S}_v \mid v_0 \to v]] < 2^{-\Omega(\tilde{\ell})} \tag{5}$$

where the expectation is over the non-significant vertices v in layer-i of part-2, reached by the path $\mathcal{T}$. (If $\mathcal{T}$ stops before reaching layer-i of part-2, or if it reaches a significant vertex, we think of v as undefined and think of the inner probability as 0).

By Expression (3), the assumption of the lemma is satisfied by the choice $\mathcal{S}_v = \mathrm{Sig}(v)$. Thus, the conclusion of the lemma implies that

$$\mathop{\mathbf{E}}_v\left[\Pr_x[x \in \mathrm{Sig}(v) \mid v_0 \to v]\right] \leq 2^{-\Omega(\tilde{\ell})}.$$

Thus, the probability that $\mathcal{T}$ stops because of stopping rule 2 is at most $2^{-\Omega(\tilde{\ell})}$, in each step, and taking a union bound over the length of the program, the probability that $\mathcal{T}$ stops because of stopping rule 2 is at most $2^{-\Omega(\tilde{\ell})}$.

5.3.3 Proof of Lemma 5.5

Proof. We could also write $\mathbf{E}_v[\Pr_x[x \in \mathcal{S}_v \mid v_0 \to v]]$ as

$$\sum_{v \in \mathcal{L}_{i,2}} \Pr[v_0 \to v] \cdot \Pr_x[x \in \mathcal{S}_v \mid v_0 \to v] = \sum_{v \in \mathcal{L}_{i,2}} \Pr[(x \in \mathcal{S}_v) \ \wedge \ (v_0 \to v)]$$

where $\mathcal{L}_{i,2}$ denotes the **non-significant** vertices v in layer-i of part-2, **that are reachable (with probability larger than 0) from the start vertex**.

Later on, we will define for every $v \in \mathcal{L}_{i,2}$, an event G_v that will occur with high probability. We will denote by $\bar{G_v}$, the complement of G_v. We will bound

$$\sum_{v \in \mathcal{L}_{i,2}} \Pr[(x \in \mathcal{S}_v) \ \wedge \ (v_0 \to v)],$$

by bounding separately

$$\sum_{v \in \mathcal{L}_{i,2}} \Pr[G_v \ \wedge \ (x \in \mathcal{S}_v) \ \wedge \ (v_0 \to v)] \tag{6}$$

and

$$\sum_{v \in \mathcal{L}_{i,2}} \Pr[\bar{G_v} \ \wedge \ (x \in \mathcal{S}_v) \ \wedge \ (v_0 \to v)] \tag{7}$$

The second expression will be bounded by

$$\sum_{v \in \mathcal{L}_{i,2}} \Pr[\bar{G_v} \ \wedge \ (v_0 \to v)], \tag{8}$$

that will be at most $2^{-\Omega(\tilde{\ell})}$ (see Claim 5.6). Thus, we will focus first on bounding Expression (6), which is equal to

$$\sum_{v \in \mathcal{L}_{i,2}} \sum_{x' \in \mathcal{S}_v} \Pr[G_v \ \wedge \ (x = x') \ \wedge \ (v_0 \to v)] \tag{9}$$

$$= \sum_{v \in \mathcal{L}_{i,2}} \sum_{x' \in \mathcal{S}_v} \Pr[G_v \ \wedge \ (v_0 \to v) \mid (x = x')] \cdot \Pr[x = x']. \tag{10}$$

Recall that for two vertices w_1, w_2 in the program, we denote by $w_1 \to w_2$ the event (over $x, a_1, \ldots, a_m$) that the path $\mathcal{T}$ that starts from w_1 reaches w_2.

Recall that by the construction of the branching-program $\hat{B}$, part-2 runs a copy of part-1 of the computation. Thus, the vertex v has a corresponding vertex v' in layer-i of part-1, such that, if the path $\mathcal{T}$ reached v it previously reached v'. Recall also that v remembers v_1, so if the path $\mathcal{T}$ reached v it previously reached v_1.

Thus, the event $v_0 \to v$ is equivalent to $v_0 \to v' \to v_1 \to v$, that is, the event

$$(v_0 \to v') \wedge (v' \to v_1) \wedge (v_1 \to v).$$

Since the second part of the program runs a copy of the first part, the event $v_1 \to v$ implies the event $v_0 \to v'$. Hence, the event $v_0 \to v$ is equivalent to the event

$$(v' \to v_1) \wedge (v_1 \to v).$$

Note also that if we fix x, that is, if we condition on $x = x'$, and we fix v (which also fixes v', v_1) the events $(v' \to v_1)$ and $(v_1 \to v)$ are independent (as the first one depends only on $a_{i+1}, \dots, a_m$ and the second depends only on $a_1, \dots, a_i$). We will also have the property that the event G_v is a function of v' rather than v, and hence will also be denoted by $G_{v'} = G_v$ (recall that v determines v'). Moreover, if we fix x and v', we will have the property that the event $G_{v'}$ depends only on $a_{i+1}, \dots, a_m$, and hence the events $G_{v'}$ and $(v' \to v_1)$ are independent of $(v_1 \to v)$.

Thus, for a fixed v (which also fixes v', v_1) and any $x' \in X$,

$$\begin{aligned}
&\Pr[G_v \wedge (v_0 \to v) \mid x = x'] \\
&\quad = \Pr[G_{v'} \wedge (v' \to v_1) \wedge (v_1 \to v) \mid x = x'] \\
&\quad = \Pr[G_{v'} \wedge (v' \to v_1) \mid x = x'] \cdot \Pr[v_1 \to v \mid x = x'].
\end{aligned}$$

We introduce the event $(v' \widetilde{\rightarrow} v_1)$ to indicate that the computational path from v' reached v_1 (as opposed to the usual notation that denotes the truncated path). Since $(v' \to v_1)$ implies $(v' \widetilde{\rightarrow} v_1)$ we have

$$\begin{gathered}
\Pr[G_{v'} \wedge (v' \to v_1) \mid x = x'] \cdot \Pr[v_1 \to v \mid x = x'] \\
\leq \Pr[G_{v'} \wedge (v' \widetilde{\rightarrow} v_1) \mid x = x'] \cdot \Pr[v_1 \to v \mid x = x'].
\end{gathered}$$

By Bayes' rule, the last expression is at most

$$\begin{aligned}
&\Pr[x = x' \mid G_{v'} \wedge (v' \widetilde{\rightarrow} v_1)] \cdot \Pr[x = x' \mid v_1 \to v] \cdot \frac{\Pr[v' \widetilde{\rightarrow} v_1] \cdot \Pr[v_1 \to v]}{\Pr[x = x']^2} \\
&\qquad = \mathbb{P}_{x \mid G_{v'} \wedge (v' \widetilde{\rightarrow} v_1)}(x') \cdot \mathbb{P}_{x \mid v_1 \to v}(x') \cdot \frac{\Pr[v' \widetilde{\rightarrow} v_1] \cdot \Pr[v_1 \to v]}{\Pr[x = x']^2}.
\end{aligned}$$

Thus, Expression (10) is at most

$$\sum_{v \in \mathcal{L}_{i,2}} \left(\Pr[v' \widetilde{\rightarrow} v_1] \cdot \Pr[v_1 \to v] \cdot \sum_{x' \in \mathcal{S}_v} \frac{\mathbb{P}_{x \mid G_{v'} \wedge (v' \widetilde{\rightarrow} v_1)}(x')}{\Pr[x = x']} \cdot \mathbb{P}_{x \mid v_1 \to v}(x') \right). \tag{11}$$

Note that from layer-i of part-1 to layer-m of part-1, the branching program is one-pass. Denote by $R_{v'}$ the one-pass branching program, from layer-i of part-1 to layer-m of part-1, with starting vertex v'. Thus, we can use what we already know about one-pass branching programs. We will apply a slight modification of the main theorem of [GRT18] (Proposition 7.1 from Appendix), for one-pass branching programs, with parameters $k' = k, \ell' = \tilde{\ell}, r' = \tilde{r}/4$.

As $m \leq 2^{\epsilon \tilde{r}}$ and $R_{v'}$ has width at most $2^{\epsilon k \tilde{\ell}/2} \leq 2^{k' \cdot \ell'/100}$ (ϵ is small enough), by Proposition 7.1, we know that for any fixed v', there exists an event $G_{v'}$ that depends only on $x, a_{i+1}, \dots, a_m$, such that, $\Pr(G_{v'}) \geq 1 - 2^{-\tilde{\ell}/8}$ ($\tilde{\ell} \leq k$), and for every $x' \in X$, and every v_1 such that $\Pr[G_{v'} \wedge (v' \widetilde{\rightarrow} v_1)] > 0$ it holds that

$$\mathbb{P}_{x \mid G_{v'} \wedge (v' \widetilde{\rightarrow} v_1)}(x') \leq 2^{2\tilde{\ell}} \cdot 2^{-n}.$$

Namely, the event $G_{v'}$ is the event G from Proposition 7.1 corresponding to the branching program $R_{v'}$ (that is, the event $G_{v'}$ is the event that the truncated-path as defined for one-pass branching programs in [GRT18] with slight modification, didn't stop because of one of the stopping rules, until the last layer, and didn't violate the significant vertices and significant values stopping rules in the last layer, that is, layer-m of part-1).

Substituting this in Expression (11), we get that the expression is at most

$$2^{2\tilde{\ell}} \cdot \sum_{v \in \mathcal{L}_{i,2}} \left(\Pr[v' \widetilde{\rightarrow} v_1] \cdot \Pr[v_1 \rightarrow v] \cdot \sum_{x' \in \mathcal{S}_v} \mathbb{P}_{x|v_1 \rightarrow v}(x') \right). \tag{12}$$

By the assumption of the lemma, for any $v \in \mathcal{L}_{i,2}$ we have $\sum_{x' \in \mathcal{S}_v} \mathbb{P}_{x|v_1 \rightarrow v}(x') \leq 2^{-4\tilde{\ell}}$, thus Expression (12) is at most

$$2^{-2\tilde{\ell}} \cdot \sum_{v \in \mathcal{L}_{i,2}} \Pr[v' \widetilde{\rightarrow} v_1] \cdot \Pr[v_1 \rightarrow v].$$

Recall that $\mathcal{L}_{i,2}$ denotes only the vertices v in layer-i of part-2, that are reachable (with probability larger than 0) from the start vertex, v_0. Recall that the event $(v_1 \rightarrow v)$ is equivalent to the event $(v_0 \rightarrow v') \wedge (v_1 \rightarrow v)$.

Thus,

$$\begin{aligned}
\sum_{v \in \mathcal{L}_{i,2}} \Pr[v' \widetilde{\rightarrow} v_1] \cdot \Pr[v_1 \rightarrow v] &\leq \sum_{v', v_1, v} \Pr\left[v' \widetilde{\rightarrow} v_1\right] \cdot \Pr\left[(v_0 \rightarrow v') \wedge (v_1 \rightarrow v)\right] \\
&= \sum_{v', v_1} \Pr\left[v' \widetilde{\rightarrow} v_1\right] \cdot \left(\sum_{v} \Pr\left[(v_0 \rightarrow v') \wedge (v_1 \rightarrow v)\right] \right) \\
&\leq \sum_{v', v_1} \Pr\left[v' \widetilde{\rightarrow} v_1\right] \cdot \Pr\left[v_0 \rightarrow v'\right] \\
&= \sum_{v'} \Pr\left[v_0 \rightarrow v'\right] \cdot \left(\sum_{v_1} \Pr\left[v' \widetilde{\rightarrow} v_1\right] \right) \\
&\leq \sum_{v'} \Pr\left[v_0 \rightarrow v'\right] \leq 1
\end{aligned}$$

(where the possible inequality in the first line is because the first sum is on all the paths $v_0 \rightarrow v' \rightarrow v_1 \rightarrow v$, obtained with positive probabilities, whereas the second sum is on all possible vertices v_0, v', v_1, v in the corresponding layers of the branching program).

Thus, we conclude that Expression (6) is at most $2^{-2\tilde{\ell}}$. It remains to bound Expression (8).

Claim 5.6.

$$\sum_{v \in \mathcal{L}_{i,2}} \Pr[\bar{G}_v \wedge (v_0 \rightarrow v)] \leq 2^{-\Omega(\tilde{\ell})}.$$

Proof.

$$\sum_{v \in \mathcal{L}_{i,2}} \Pr[\bar{G}_v \wedge (v_0 \rightarrow v)] = \sum_{v \in \mathcal{L}_{i,2}} \Pr[\bar{G}_v \wedge (v_0 \rightarrow v') \wedge (v' \rightarrow v_1) \wedge (v_1 \rightarrow v)]$$

$$\leq \sum_{v', v_1, v} \Pr[\bar{G}_v \wedge (v_0 \rightarrow v') \wedge (v' \rightarrow v_1) \wedge (v_1 \rightarrow v)]$$

$$= \sum_{v',v_1,v} \Pr[\bar{G_{v'}} \wedge (v_0 \to v') \wedge (v' \to v_1) \wedge (v_1 \to v)]$$

$$= \sum_{v' \in \mathcal{L}_{i,1}} \Pr[\bar{G_{v'}} \wedge (v_0 \to v')] \cdot \sum_{v_1,v} \Pr[(v' \to v_1) \wedge (v_1 \to v) | \bar{G_{v'}} \wedge (v_0 \to v')]$$

$$\leq \sum_{v' \in \mathcal{L}_{i,1}} \Pr[\bar{G_{v'}} \wedge (v_0 \to v')]. \tag{13}$$

For every non-significant $v' \in \mathcal{L}_{i,1}$, denote by

$$\mathcal{X}_{v'} = \{x' : \mathbb{P}_{x|v_0 \to v'}(x') \geq 2^{\tilde{\ell}/16} \cdot 2^{-n}\},$$

and split the expression $\Pr[\bar{G_{v'}} \wedge (v_0 \to v')]$ according to whether or not $(x \in \mathcal{X}_{v'})$.

$$\Pr[\bar{G_{v'}} \wedge (v_0 \to v')]$$

$$\leq \Pr[(v_0 \to v') \wedge (x \in \mathcal{X}_{v'})] + \Pr[\bar{G_{v'}} \wedge (v_0 \to v') \wedge (x \notin \mathcal{X}_{v'})] \tag{14}$$

We begin by bounding the first summand in Expression (14):

$$\Pr[(v_0 \to v') \wedge (x \in \mathcal{X}_{v'})] = \Pr(v_0 \to v') \cdot \Pr[(x \in \mathcal{X}_{v'}) | v_0 \to v']$$

We bound $\Pr[x \in \mathcal{X}_{v'} | v_0 \to v']$ very similarly to the proof of Claim 5.4, but with a different threshold. Since v' is not significant,

$$\mathop{\mathbf{E}}_{x' \sim \mathbb{P}_{x|v_0 \to v'}} \left[\mathbb{P}_{x|v_0 \to v'}(x')\right] = \sum_{x' \in X} \left[\mathbb{P}_{x|v_0 \to v'}(x')^2\right] = 2^n \cdot \mathop{\mathbf{E}}_{x' \in_R X} \left[\mathbb{P}_{x|v_0 \to v'}(x')^2\right] \leq 2^{2\ell_1} \cdot 2^{-n}.$$

Hence, by Markov's inequality,

$$\Pr[x \in \mathcal{X}_{v'} | v_0 \to v'] = \Pr_{x' \sim \mathbb{P}_{x|v_0 \to v'}} \left[\mathbb{P}_{x|v_0 \to v'}(x') \geq 2^{\tilde{\ell}/16} \cdot 2^{-n}\right] \leq 2^{2\ell_1 - \tilde{\ell}/16} \leq 2^{-\tilde{\ell}/32}$$

(recall that $\ell_1 = \tilde{\ell}/100$). Overall, we bounded the first summand in Expression (14) by $\Pr(v_0 \to v') \cdot 2^{-\tilde{\ell}/32}$.

Next, we bound the second summand in Expression (14).

$$\Pr\left[\bar{G_{v'}} \wedge (v_0 \to v') \wedge (x \notin \mathcal{X}_{v'})\right]$$

$$= \sum_{x' \in X \setminus \mathcal{X}_{v'}} \Pr(x = x') \cdot \Pr\left[\bar{G_{v'}} \wedge (v_0 \to v') \mid x = x'\right].$$

Since if we fix x and v', the event $G_{v'}$ depends only on $a_{i+1}, \ldots, a_m$ and hence is independent of $(v_0 \to v')$, we have

$$\sum_{x' \in X \setminus \mathcal{X}_{v'}} \Pr(x = x') \cdot \Pr\left[\bar{G_{v'}} \wedge (v_0 \to v') \mid x = x'\right]$$

$$= \sum_{x' \in X \setminus \mathcal{X}_{v'}} \Pr(x = x') \cdot \Pr\left[\bar{G_{v'}} \mid x = x'\right] \cdot \Pr\left[v_0 \to v' \mid x = x'\right]$$

$$= \Pr\left(v_0 \to v'\right) \cdot \sum_{x' \in X \setminus \mathcal{X}_{v'}} \Pr\left[\bar{G_{v'}} \mid x = x'\right] \cdot \Pr\left[x = x' \mid v_0 \to v'\right]$$

(by Bayes' rule)

$$= \Pr\left(v_0 \to v'\right) \cdot \sum_{x' \in X \setminus \mathcal{X}_{v'}} \Pr\left[\bar{G}_{v'} \mid x = x'\right] \cdot \mathbb{P}_{x|v_0 \to v'}(x')$$

$$\leq \Pr\left(v_0 \to v'\right) \cdot \sum_{x' \in X \setminus \mathcal{X}_{v'}} \Pr\left[\bar{G}_{v'} \mid x = x'\right] \cdot 2^{\tilde{\ell}/16} \cdot 2^{-n}$$

(by the definition of $\mathcal{X}_{v'}$)

$$\leq \Pr\left(v_0 \to v'\right) \cdot \Pr\left(\bar{G}_{v'}\right) \cdot 2^{\tilde{\ell}/16}$$

$$\leq \Pr\left(v_0 \to v'\right) \cdot 2^{-\tilde{\ell}/8} \cdot 2^{\tilde{\ell}/16}$$

$$\leq \Pr\left(v_0 \to v'\right) \cdot 2^{-\tilde{\ell}/16}.$$

Substituting in Expression (14), we have

$$\Pr[\bar{G}_{v'} \wedge (v_0 \to v')] \leq \Pr\left(v_0 \to v'\right) \cdot 2^{-\tilde{\ell}/32} + \Pr\left(v_0 \to v'\right) \cdot 2^{-\tilde{\ell}/16}.$$

Substituting in Expression (13), we have

$$\sum_{v \in \mathcal{L}_{i,2}} \Pr[\bar{G}_v \wedge (v_0 \to v)] \leq (2^{-\tilde{\ell}/32} + 2^{-\tilde{\ell}/16}) \cdot \sum_{v' \in \mathcal{L}_{i,1}} \Pr\left(v_0 \to v'\right) \leq 2 \cdot 2^{-\tilde{\ell}/32}. \qquad \square$$

This finishes the proof of Lemma 5.5. □

5.3.4 Stopping Rule 3: Bad-Edges

We will now bound the probability that $\mathcal{T}$ stops because of stopping rule 3. We will first prove the following claim.

Claim 5.7. *If v is a non-significant vertex in layer-i of part-j (where $j \in \{1, 2\}$), then*

$$\Pr_{a_{i+1}}[a_{i+1} \in \mathrm{Bad}(v)] \leq 2^{-4k}.$$

Proof. Since v is not significant, $\left\|\mathbb{P}_{x|v_{j-1}\to v}\right\|_2 \leq 2^{\ell_j} \cdot 2^{-n} \leq 2^{\ell} \cdot 2^{-n}$. Since $\mathbb{P}_{x|v_{j-1}\to v}$ is a distribution, $\left\|\mathbb{P}_{x|v_{j-1}\to v}\right\|_1 = 2^{-n}$. Thus,

$$\frac{\left\|\mathbb{P}_{x|v_{j-1}\to v}\right\|_2}{\left\|\mathbb{P}_{x|v_{j-1}\to v}\right\|_1} \leq 2^{\ell}.$$

Since M is a $(10k, 10\ell)$-L_2-extractor with error 2^{-10r}, there are at most $2^{-10k} \cdot |A|$ elements $a \in A$ with

$$\left|\langle M_a, \mathbb{P}_{x|v_{j-1}\to v}\rangle\right| \geq 2^{-10r} \cdot \left\|\mathbb{P}_{x|v_{j-1}\to v}\right\|_1 = 2^{-10r} \cdot 2^{-n}$$

The claim follows since a_{i+1} is uniformly distributed over A. □

By Claim 5.7, if v is a non-significant vertex then

$$\Pr_{a_{i+1}}\left[a_{i+1} \in \mathrm{Bad}(v)\right] \le 2^{-4k}.$$

We need to bound

$$\Pr_{a_{i+1}}\left[a_{i+1} \in \mathrm{Bad}(v) \setminus \mathrm{High}(v) \mid v_0 \to v\right].$$

We bound

$$\Pr_{a_{i+1}}\left[a_{i+1} \in \mathrm{Bad}(v) \setminus \mathrm{High}(v) \mid v_0 \to v\right] = \sum_{a \in \mathrm{Bad}(v) \setminus \mathrm{High}(v)} \Pr[a_{i+1} = a \mid v_0 \to v]$$

$$\le \sum_{a \in \mathrm{Bad}(v) \setminus \mathrm{High}(v)} 2^k \cdot 2^{-n'} \le 2^k \cdot \sum_{a \in \mathrm{Bad}(v)} \Pr[a_{i+1} = a]$$

$$= 2^k \cdot \Pr_{a_{i+1}}\left[a_{i+1} \in \mathrm{Bad}(v)\right] \le 2^k \cdot 2^{-4k} = 2^{-3k}.$$

Thus, the probability that $\mathcal{T}$ stops because of stopping rule 3 is at most 2^{-3k}, in each step, and taking a union bound over the length of the program, the probability that $\mathcal{T}$ stops because of stopping rule 3 is at most 2^{-2k}.

5.3.5 Stopping Rule 4: Very-Bad Edges

We will now bound the probability that $\mathcal{T}$ stops because of stopping rule 4.

Recall that for a vertex v in layer-i of part-2 of the program, $\mathrm{VeryBad}(v)$ is the set of all $(a, b) \in A \times \{-1, 1\}$, such that,

$$\Pr_x[M(a, x) = b \mid v_1 \to v] \le 2^{-4\tilde{\ell}}.$$

Note that for every $a \in A$, there is at most one $b \in \{-1, 1\}$, denoted $b_v(a)$, such that

$$\Pr_x[M(a, x) = b \mid v_1 \to v] \le 2^{-4\tilde{\ell}}.$$

If such a b doesn't exist we let $b_v(a) = *$, and think of it as undefined. Thus, for every v, and every $(a, b) \in A \times \{-1, 1\}$,

$$\big((a, b) \in \mathrm{VeryBad}(v)\big) \iff \big(b = b_v(a)\big), \tag{15}$$

and

$$\Pr_x[M(a, x) = b_v(a) \mid v_1 \to v] \le 2^{-4\tilde{\ell}}. \tag{16}$$

Let $a_v \in A$ be an $a \in A$, such that $\Pr_x[M(a, x) = b_v(a) \mid v_0 \to v]$ is maximal and let $b_v = b_v(a_v)$. We need to bound from above

$$\mathop{\mathbf{E}}_v \left[\Pr[(a_{i+1}, b_{i+1}) \in \mathrm{VeryBad}(v) \mid v_0 \to v] \right], \tag{17}$$

where the expectation is over the vertex v in layer-i of part-2, reached by the path $\mathcal{T}$. (If $\mathcal{T}$ stops before reaching layer-i of part-2, we think of v as undefined and think of the inner probability as 0). That is, we could also write Expression (17) as

$$\sum_{v \in \mathcal{L}_{i,2}} \Pr[v_0 \to v] \cdot \Pr[(a_{i+1}, b_{i+1}) \in \mathrm{VeryBad}(v) \mid v_0 \to v],$$

where $\mathcal{L}_{i,2}$ denotes the vertices v in layer-i of part-2, **that are reachable (with probability larger than 0) from the start vertex**. By Equation (15), Expression (17) is equal to

$$\mathop{\mathbf{E}}_{v}\left[\Pr[b_{i+1} = b_v(a_{i+1}) \mid v_0 \to v]\right],$$

which, by the definition of b_{i+1}, is equal to

$$\mathop{\mathbf{E}}_{v}\left[\Pr[M(a_{i+1}, x) = b_v(a_{i+1}) \mid v_0 \to v]\right],$$

which, by the definitions of a_v, b_v, is at most

$$\mathop{\mathbf{E}}_{v}\left[\Pr[M(a_v, x) = b_v \mid v_0 \to v]\right]. \tag{18}$$

In what follows, we assume for simplicity and without loss of generality that for every v, $b_v \in \{-1, 1\}$ is defined (as otherwise $\Pr[M(a_v, x) = b_v \mid v_0 \to v] = 0$ and can be omitted from the expectation).

For any fixed v, denote by $\mathcal{S}_v = \{x : M(a_v, x) = b_v\}$. We can apply Lemma 5.5, since from Expression (16) for any non-significant v

$$\Pr[x \in \mathcal{S}_v \mid v_1 \to v] \leq 2^{-4\tilde{\ell}}.$$

Thus, we get

$$\mathop{\mathbf{E}}_{v}[\Pr[x \in \mathcal{S}_v \mid v_0 \to v]] \leq 2^{-\Omega(\tilde{\ell})} ,$$

and since $(x \in \mathcal{S}_v) \iff (M(a_v, x) = b_v)$, we have

$$\mathop{\mathbf{E}}_{v}[\Pr[M(a_v, x) = b_v \mid v_0 \to v]] \leq 2^{-\Omega(\tilde{\ell})} .$$

Finally, by the definitions of a_v and b_v we have

$$\mathop{\mathbf{E}}_{v}[\Pr[(a_{i+1}, b_{i+1}) \in \text{VeryBad}(v) \mid v_0 \to v]] \leq \mathop{\mathbf{E}}_{v}[\Pr[M(a_v, x) = b_v \mid v_0 \to v]] \leq 2^{-\Omega(\tilde{\ell})} .$$

Thus, the probability that $\mathcal{T}$ stops because of stopping rule 4 is at most $2^{-\Omega(\tilde{\ell})}$, in each step, and taking a union bound over the length of the program, the probability that $\mathcal{T}$ stops because of stopping rule 4 is at most $2^{-\Omega(\tilde{\ell})}$.

5.3.6 Stopping Rule 5: Large L_v

Recall that for two vertices w_1, w_2 in the program, we denote by $w_1 \to w_2$ the event (over $x, a_1, \ldots, a_m$) that the path $\mathcal{T}$ that starts from w_1 reaches w_2.

Recall that v_1 is the vertex reached by the path at the end of part-1. Fix v_1 and denote by E the event $v_0 \to v_1$. Let $u_0, u_1, \ldots, u_m$ be the vertices reached by the path in part-2, where $u_0 = v_1$. (If the path stops before reaching layer-i of part-2, we define u_i to be a special *stop* vertex in that layer). Note that conditioned on the event E, the random variable u_i is a function of $x, a_1, \ldots, a_i$ and for $i \geq 1$ it can also be viewed as a function of x, u_{i-1}, a_i.

Denote by T the number of high-probability edges that the path traverses in part-2. For every $i \in [m]$, let $T_i \in \{0, 1\}$ be an indicator random variable that indicates whether the path traverses a high-probability edge at step-i of part-2. Thus,

$$T = \sum_{i=1}^{m} T_i.$$

For every $i \in [m]$, we have that $T_i = 1$ only if $a_i \in \mathrm{High}(u_{i-1})$, that is, only if $\Pr(a_i|u_{i-1}, E) \geq 2^k \cdot 2^{-n'}$, or equivalently

$$\frac{\log\left(2^{n'} \cdot \Pr(a_i|u_{i-1}, E)\right)}{k} \geq 1.$$

Claim 5.8. *Let $Z \in \{0,1\}^{n'}$ be any random variable. Let $k \geq 4$. Let $T(Z) \in \{0,1\}$ be an indicator random variable for the event $\Pr(Z) \geq 2^k \cdot 2^{-n'}$. Then,*

$$2 \cdot \mathop{\mathbf{E}}_{Z}\left[\frac{\log\left(2^{n'} \cdot \Pr(Z)\right)}{k}\right] \geq \mathop{\mathbf{E}}_{Z}[T(Z)].$$

Proof. Let $\alpha = \Pr_Z(T(Z) = 1)$. That is, we have $\Pr(Z) \geq 2^k \cdot 2^{-n'}$ with probability α. Thus,

$$\mathop{\mathbf{E}}_{Z}\left[\frac{\log\left(2^{n'} \cdot \Pr(Z)\right)}{k}\right] =$$

$$\alpha \cdot \mathop{\mathbf{E}}_{Z}\left[\frac{\log\left(2^{n'} \cdot \Pr(Z)\right)}{k} \;\middle|\; T(Z) = 1\right] + (1-\alpha) \cdot \mathop{\mathbf{E}}_{Z}\left[\frac{\log\left(2^{n'} \cdot \Pr(Z)\right)}{k} \;\middle|\; T(Z) = 0\right].$$

By the monotonicity of the logarithm function, we have,

$$\alpha \cdot \mathop{\mathbf{E}}_{Z}\left[\frac{\log\left(2^{n'} \cdot \Pr(Z)\right)}{k} \;\middle|\; T(Z) = 1\right] \geq$$

$$\alpha \cdot \mathop{\mathbf{E}}_{Z}\left[\frac{\log\left(2^{n'} \cdot 2^k \cdot 2^{-n'}\right)}{k} \;\middle|\; T(Z) = 1\right] = \alpha$$

By the monotonicity of the logarithm function and the concavity of the entropy function, we have,

$$(1-\alpha) \cdot \mathop{\mathbf{E}}_{Z}\left[\frac{\log\left(2^{n'} \cdot \Pr(Z)\right)}{k} \;\middle|\; T(Z) = 0\right] \geq$$

$$(1-\alpha) \cdot \mathop{\mathbf{E}}_{Z}\left[\frac{\log\left(2^{n'} \cdot (1-\alpha) \cdot 2^{-n'}\right)}{k} \;\middle|\; T(Z) = 0\right] =$$

$$\frac{(1-\alpha)\log(1-\alpha)}{k}$$

(as, by the concavity of the entropy function, the expression is minimized when the random variable $Z|(T(Z) = 0)$ is uniformly distributed).

Thus, the left hand side of the claim is at least

$$2\alpha + \frac{2(1-\alpha)\log(1-\alpha)}{k} \geq 2\alpha - \frac{4\alpha}{k} \geq 2\alpha - \frac{4\alpha}{4} = \alpha.$$

The claim follows Since $\mathbf{E}_Z[T(Z)] = \alpha$. □

By Claim 5.8,

$$\mathop{\mathbf{E}}_{x,a_1,\ldots,a_m}[T|E] = \sum_{i=1}^{m} \mathop{\mathbf{E}}_{x,a_1,\ldots,a_m}[T_i|E] \leq 2 \cdot \sum_{i=1}^{m} \mathop{\mathbf{E}}_{x,a_1,\ldots,a_m}\left[\frac{\log\left(2^{n'} \cdot \Pr(a_i|u_{i-1},E)\right)}{k}\right]$$

$$= \frac{2}{k} \cdot \left(mn' - \sum_{i=1}^{m} \mathbf{H}(a_i|u_{i-1},E)\right),$$

where $\mathbf{H}$ denotes the entropy function. Since conditioning may only decrease the entropy, the last expression is at most

$$\leq \frac{2}{k} \cdot \left(mn' - \sum_{i=1}^{m} \mathbf{H}(a_i|x,u_{i-1},E)\right).$$

Since, conditioned on E, the random variable u_{i-1} is a function of $x, a_1, \ldots, a_{i-1}$, by the data-processing inequality, $\mathbf{H}(a_i|x,u_{i-1},E) \geq \mathbf{H}(a_i|x,a_1,\ldots,a_{i-1},E)$, and hence the last expression is at most

$$\leq \frac{2}{k} \cdot \left(mn' - \sum_{i=1}^{m} \mathbf{H}(a_i|x,a_1,\ldots,a_{i-1},E)\right).$$

By the chain rule, the last expression is equal to

$$= \frac{2}{k} \cdot \left(mn' - \mathbf{H}(a_1,\ldots,a_m|x,E)\right)$$

$$= \frac{2}{k} \cdot \left(mn' - \mathbf{H}(x,a_1,\ldots,a_m|E) + \mathbf{H}(x|E)\right)$$

$$\leq \frac{2}{k} \cdot \left(mn' + n - \mathbf{H}(x,a_1,\ldots,a_m|E)\right)$$

$$\leq \frac{2}{k} \cdot \log\left(\tfrac{1}{\Pr(E)}\right).$$

Thus,

$$\mathop{\mathbf{E}}_{x,a_1,\ldots,a_m}[T|E] \leq \frac{2}{k} \cdot \log\left(\tfrac{1}{\Pr(E)}\right).$$

By Markov inequality

$$\Pr_{x,a_1,\ldots,a_m}\left[T \geq \frac{200}{k} \cdot \log\left(\tfrac{1}{\Pr(E)}\right) \,\middle|\, E\right] \leq \frac{1}{100}.$$

Since we assumed that $\Pr(E) \geq 2^{-10\tilde{r}} \cdot d_1^{-1}$ and since the width of $\hat{B}$ is at most $2^{\epsilon k\tilde{\ell}/2}$ and since by Equation (1) and Equation (2), $\tilde{r}$ is negligible compared to $\epsilon k\tilde{\ell}/2$, we have that $\log\left(\tfrac{1}{\Pr(E)}\right) \leq \epsilon k\tilde{\ell}$. Hence,

$$\Pr_{x,a_1,\ldots,a_m}\left[T \geq 200\epsilon\tilde{\ell} \,\middle|\, E\right] \leq \frac{1}{100}.$$

Thus, the probability to stop on part-2 because of stopping rule 5 is at most $\frac{1}{100}$.

5.3.7 Stopping Rule 6: Consistency-Stop

We will now show that the probability that $\mathcal{T}$ stops on a vertex v, in layer-i of part-2, because of stopping rule 6, conditioned on the event $v_0 \to v$, is 0.

Recall that by the construction of the branching-program $\hat{B}$, part-2 runs a copy of part-1 of the computation. Thus, the vertex v has a corresponding vertex v' in layer-i of part-1, such that, if the path $\mathcal{T}$ reached v it previously reached v'.

If $\mathcal{T}$ needs to stop on v, because of stopping rule 6, because $\mathcal{T}$ stopped on the vertex v', it couldn't have reached v in the first place (as it would have stopped on v'). Thus, conditioned on the event $v_0 \to v$, the path $\mathcal{T}$ didn't stop on v' and doesn't need to stop on v because of stopping rule 6.

Thus, the probability that $\mathcal{T}$ stops because of stopping rule 6 is 0.

This completes the proof of Lemma 5.2. □

5.4 The Final Success Probability is Small

Let v be a vertex in the last layer of the program. Assume that the probability for the event $v_0 \to v$ is larger than 0. Since v is in the last layer, the event $v_0 \to v$ is equivalent to $v_1 \to v$ (since the second part of the program runs a copy of the first part). Hence,

$$\mathbb{P}_{x|v_0 \to v} = \mathbb{P}_{x|v_1 \to v}$$

and

$$\Pr[v_0 \to v] = \Pr[v_1 \to v].$$

In particular, if v is not significant, $\mathbb{P}_{x|v_0 \to v}$ has small L_2-norm.

$$\mathop{\mathbf{E}}_{x' \in_R X} \left[\mathbb{P}_{x|v_0 \to v}(x')^2\right] \leq 2^{2\ell} \cdot 2^{-2n}.$$

Hence, for every $x' \in X$,

$$\Pr[x = x' \mid v_0 \to v] = \mathbb{P}_{x|v_0 \to v}(x') \leq 2^{\ell} \cdot 2^{-n/2} \leq 2^{-n/4}$$

In particular,

$$\Pr[\tilde{x}(v) = x \mid v_0 \to v] \leq 2^{-n/4}.$$

Thus, either the computation path stops before reaching v which happens with probability at most $\frac{1}{100} + o(1)$ or it reaches a non-significant vertex where the probability of guessing correctly is $o(1)$. Thus, the final success probability is bounded by $\frac{1}{100} + o(1)$. This completes the proof of Theorem 1.

6 Proof of Lemma 5.3

Proof Overview. Let s be a significant vertex in part-j (that remembers the vertices visited at the end of parts $1, \dots, j-1$, denoted by $s_1, \dots, s_{j-1}$). Assume that the probability for the event $v_0 \to s$ is larger than 0. We need to bound from above the probability for the event $v_0 \to s$. Since the event $v_0 \to s$ is equivalent to $(v_0 \to s_{j-1}) \wedge (s_{j-1} \to s)$, it suffices to bound from above the probability for $(s_{j-1} \to s)$. Note that to analyze this probability we can ignore all parts of the program, except for part-j, which is a one-pass branching program.

We would like to reprove Lemma 4.1 of [GRT18], with the updated stopping rules. In the definition of the progress function $\mathcal{Z}_i$, we will take the sum only on vertices $u \in \mathcal{L}_{i,j}$, such that s can be reached from u (and in the same way for edges in the definition of $\mathcal{Z}'_i$). In particular, this implies that every index in L_u is contained in L_s (as otherwise s cannot be reached from u).

The progress function is still small at the beginning and large at the end, so as before the main thing to do is to prove that it grows slowly. This was done in Claim 4.10 of [GRT18].

The main difference here is that the progress function doesn't grow slowly for every edge, as some edges are now bad, and we have to take the bad edges into account. We separate to time steps that are in L_s and time steps that are not in L_s. For time steps that are not in L_s, we don't need to count the bad edges at all, as they are not recorded by L_s and hence s is not reachable from these edges.

As for steps in L_s, we know that the edges are not very-bad, and we show that the progress function may increase by a factor of at most $2^{5\tilde{\ell}k}$. Since $|L_s| \leq 200\epsilon\tilde{\ell}$ (as otherwise $\mathcal{T}$ would have stopped by stopping rule 5), the total effect of the bad edges on the progress function is a factor of at most $2^{5\tilde{\ell}k\cdot 200\epsilon\tilde{\ell}} \leq 2^{1000\epsilon k\ell}$, which we can afford.

6.1 Proof of Lemma 5.3

Proof. We need to prove that the probability that $\mathcal{T}$ reaches any significant vertex is $o(1)$. Let s be a significant vertex in part-j. Assume that the probability that $\mathcal{T}$ reaches s is larger than 0. We will bound from above the probability that $\mathcal{T}$ reaches s, and then use a union bound over all significant vertices of $\hat{B}$. Since the event $v_0 \to s$ is equivalent to $(v_0 \to s_{j-1}) \wedge (s_{j-1} \to s)$, it suffices to bound from above the probability for $(s_{j-1} \to s)$. Note that to analyze this probability we can ignore all parts other than j of the program, which leaves us with a one-pass branching program. Furthermore, since s determines s_{j-1}, we can only consider the subprogram that starts at s_{j-1} and analyze the probability that the restriction of $\mathcal{T}$ to this subprogram reaches s. We denote by B' the subprogram of $\hat{B}$ restricted to the j-part with s_{j-1} as the starting node.

The Distributions $\mathbb{P}_{x|v}$ and $\mathbb{P}_{x|e}$

For a vertex v in B', we denote by E_v the event that $\mathcal{T}$ starting from s_{j-1} reaches the vertex v. For simplicity, we denote by $\Pr(v) = \Pr(E_v)$ the probability for E_v (where the probability is over $x, a_1, \ldots, a_m$), and we denote by $\mathbb{P}_{x|v} = \mathbb{P}_{x|E_v}$ the distribution of the random variable x conditioned on the event E_v.

Similarly, for an edge e of the branching program B', let E_e be the event that $\mathcal{T}$ starting from s_{j-1} traverses the edge e. Denote, $\Pr(e) = \Pr(E_e)$ (where the probability is over $x, a_1, \ldots, a_m$), and $\mathbb{P}_{x|e} = \mathbb{P}_{x|E_e}$.

Notation: B' inherits the definitions of significant vertices, $\mathrm{Sig}(v)$, $\mathrm{Bad}(v)$, $\mathrm{VeryBad}(v)$ and $\mathrm{High}(v)$ from $\hat{B}$. Note that significant vertices, $\mathrm{Sig}(v)$, $\mathrm{Bad}(v)$ and $\mathrm{VeryBad}(v)$ are defined conditioned on the event $v_{j-1} \to v$, which is equivalent to the event E_v. Recall that the walk $\mathcal{T}$ does not stop on an edge (v, u) marked (a, b) if $a \in \mathrm{High}(v)$, as long as $(a, b) \notin \mathrm{VeryBad}(v)$. We will use the following fact on $\mathcal{T}$: if $i \notin L_s$ and $\mathcal{T}$ takes a bad-edge (v, u) on the i-th step, then $L_u \not\subseteq L_s$ and s is not reachable from u.

For $i \in \{0, \ldots, m\}$, let $\mathcal{L}'_i$ be the set of vertices v in layer-i of B' such that $\Pr(v) > 0$ and **it is possible to reach** s **from** v (in particular, the set of high-probability equations stored in v is also stored in s). For $i \in \{1, \ldots, m\}$, let Γ_i be the set of edges e from $\mathcal{L}'_{i-1}$ to $\mathcal{L}'_i$ of B', such that $\Pr(e) > 0$.

Recall that by the construction of the branching-program $\hat{B}$, part-j runs a copy of all previous parts of the computation. Thus, a vertex v in B' or equivalently a vertex v in part-j of $\hat{B}$ has corresponding vertices $v'_1, \ldots, v'_{j-1}$ in layer-i of parts $1, \ldots, j-1$, respectively, such that, if the path $\mathcal{T}$ reached v it previously reached $v'_1, \ldots, v'_{j-1}$. We denote by $v'_j = v$. We denote by

$$\widetilde{\mathrm{Sig}}(v) \triangleq \bigcup_{j'=1}^{j} \mathrm{Sig}(v'_{j'}).$$

Recall that by stopping rules 2 and 6, the path $\mathcal{T}$ stops if $x \in \widetilde{\mathrm{Sig}}(v)$.

The next claim bounds the probability of stopping on a vertex v in part-2 due to stopping rule 2 of part-1 on the vertex v' that v remembers.

Claim 6.1. *If v is a non-significant vertex in layer-i of part-2 that remembers v', and v' is a non-significant vertex in layer-i of part-1, then*

$$\Pr_x[x \in \mathrm{Sig}(v') \mid v_1 \to v] \leq 2^{-2\ell}.$$

Proof. Since v is not significant,

$$\mathop{\mathbf{E}}_{x' \sim \mathbb{P}_{x|v_1 \to v}} \left[\mathbb{P}_{x|v_0 \to v'}(x')\right] = \sum_{x' \in X} \left[\mathbb{P}_{x|v_0 \to v'}(x') \cdot \mathbb{P}_{x|v_1 \to v}(x')\right]$$

(using Cauchy-Schwarz)

$$\leq \sqrt{\sum_{x' \in X} \mathbb{P}_{x|v_0 \to v'}(x')^2 \cdot \sum_{x' \in X} \mathbb{P}_{x|v_1 \to v}(x')^2}$$

$$= 2^n \cdot \sqrt{\mathop{\mathbf{E}}_{x' \in_R X} \left[\mathbb{P}_{x|v_0 \to v'}(x')^2\right] \mathop{\mathbf{E}}_{x' \in_R X} \left[\mathbb{P}_{x|v_1 \to v}(x')^2\right]}$$

(since both v' and v are non-significant)

$$\leq 2^{\ell_1 + \ell_2} \cdot 2^{-n}.$$

Hence, by Markov's inequality,

$$\Pr_{x' \sim \mathbb{P}_{x|v_1 \to v}} \left[\mathbb{P}_{x|v_0 \to v'}(x') > 2^{4\ell} \cdot 2^{-n}\right] \leq 2^{\ell_1 + \ell_2 - 4\ell} \leq 2^{-2\ell}.$$

Since conditioned on the event $v_1 \to v$, the distribution of x is $\mathbb{P}_{x|v_1 \to v}$, we obtain

$$\Pr_x \left[x \in \mathrm{Sig}(v') \;\middle|\; v_1 \to v\right] = \Pr_x \left[\left(\mathbb{P}_{x|v_0 \to v}(x) > 2^{4\ell} \cdot 2^{-n}\right) \;\middle|\; v_1 \to v \;\right] \leq 2^{-2\ell}. \qquad \square$$

Claim 6.2. *Let $i \in \{1, \dots, m\}$. For any edge $e = (v, u) \in \Gamma_i$, labeled by (a, b), such that $\Pr(e) > 0$, for any $x' \in X$,*

$$\mathbb{P}_{x|e}(x') = \begin{cases} 0 & \textit{if} \quad x' \in \widetilde{\mathrm{Sig}}(v) \quad \textit{or} \quad M(a, x') \neq b \\ \mathbb{P}_{x|v}(x') \cdot c_e^{-1} & \textit{if} \quad x' \notin \widetilde{\mathrm{Sig}}(v) \quad \textit{and} \quad M(a, x') = b \end{cases}$$

where c_e is a normalization factor that satisfies

- $c_e \geq \frac{1}{2} - 2 \cdot 2^{-2r}$, *if $i \notin L_s$.*
- $c_e \geq 2^{-4\tilde{\ell}} - 2 \cdot 2^{-2\ell} \geq 2^{-5\tilde{\ell}}$, *if $i \in L_s$.*

Proof. Let $v'_1, \dots, v'_j$ be the vertices in the branching program $\hat{B}$ that v remembers. Let $e = (v, u)$ be an edge of B', labeled by (a, b), and such that $\Pr(e) > 0$. Since $\Pr(e) > 0$, the vertices $v'_1, \dots, v'_j$ are not significant (as otherwise $\mathcal{T}$ always stops on v and hence $\Pr(e) = 0$). Also, since $\Pr(e) > 0$, we know that (a, b) is not very-bad (as otherwise $\mathcal{T}$ never traverses e and hence $\Pr(e) = 0$).

If $\mathcal{T}$ reaches v, it traverses the edge e if and only if: $x \notin \widetilde{\mathrm{Sig}}(v)$ (as otherwise $\mathcal{T}$ stops on v) and $M(a, x) = b$ and $a_{i+1} = a$. Therefore, for any $x' \in X$,

$$\mathbb{P}_{x|e}(x') = \begin{cases} 0 & \text{if} \quad x' \in \widetilde{\mathrm{Sig}}(v) \quad \text{or} \quad M(a, x') \neq b \\ \mathbb{P}_{x|v}(x') \cdot c_e^{-1} & \text{if} \quad x' \notin \widetilde{\mathrm{Sig}}(v) \quad \text{and} \quad M(a, x') = b \end{cases}$$

where c_e is a normalization factor, given by

$$c_e = \sum_{\left\{x' \,:\, x' \notin \widetilde{\mathrm{Sig}}(v) \,\wedge\, M(a, x') = b\right\}} \mathbb{P}_{x|v}(x') \;=\; \Pr_x[(x \notin \widetilde{\mathrm{Sig}}(v)) \wedge (M(a, x) = b) \mid E_v].$$

Since $v'_1, \dots, v'_j$ are not significant, by Claim 5.4 and Claim 6.1:

$$\Pr_x[x \in \widetilde{\mathrm{Sig}}(v) \mid E_v] \leq \sum_{j'=1}^{j} 2^{-2\ell} \leq 2 \cdot 2^{-2\ell} \leq 2^{-2r}.$$

If $i \notin L_s$, then $a \notin \mathrm{Bad}(v)$, as otherwise $L_u \not\subseteq L_s$ and s is not reachable from u. Thus

$$\left| \Pr_x[M(a, x) = 1 \mid E_v] - \Pr_x[M(a, x) = -1 \mid E_v] \right| = \left| (M \cdot \mathbb{P}_{x|v})(a) \right| \leq 2^{-2r},$$

and hence

$$\Pr_x[M(a, x) \neq b \mid E_v] \leq \tfrac{1}{2} + 2^{-2r}.$$

Hence, by the union bound,

$$c_e = \Pr_x[(x \notin \widetilde{\mathrm{Sig}}(v)) \wedge (M(a, x) = b) \mid E_v] \geq \tfrac{1}{2} - 2 \cdot 2^{-2r}.$$

If $i \in L_s$, then $(a, b) \notin \mathrm{VeryBad}(v)$, and we have $\Pr_x[M(a, x) = b \mid E_v] \geq 2^{-4\tilde{\ell}}$. Thus,

$$c_e = \Pr_x[(x \notin \widetilde{\mathrm{Sig}}(v)) \wedge (M(a, x) = b) \mid E_v] \geq 2^{-4\tilde{\ell}} - 2 \cdot 2^{-2\ell}\ . \qquad \square$$

Bounding the Norm of $\mathbb{P}_{x|s}$

We will show that $\left\|\mathbb{P}_{x|s}\right\|_2$ cannot be too large. Towards this, we will first prove that for every edge e of B' that is traversed by $\mathcal{T}$ starting from s_{j-1} with probability larger than zero, $\left\|\mathbb{P}_{x|e}\right\|_2$ cannot be too large.

Claim 6.3. *For any edge e of B', such that* $\Pr(e) > 0$*,*

$$\left\|\mathbb{P}_{x|e}\right\|_2 \leq 2^{5\tilde{\ell}} \cdot 2^{\ell_j} \cdot 2^{-n}.$$

Proof. Let $e = (v, u)$ be an edge of B', labeled by (a, b), and such that $\Pr(e) > 0$. Since $\Pr(e) > 0$, the vertex v is not significant (as otherwise $\mathcal{T}$ always stops on v and hence $\Pr(e) = 0$). Thus,

$$\left\|\mathbb{P}_{x|v}\right\|_2 \leq 2^{\ell_j} \cdot 2^{-n}.$$

By Claim 6.2, for any $x' \in X$,

$$\mathbb{P}_{x|e}(x') = \begin{cases} 0 & \text{if} \quad x' \in \widetilde{\mathrm{Sig}}(v) \quad \text{or} \quad M(a, x') \neq b \\ \mathbb{P}_{x|v}(x') \cdot c_e^{-1} & \text{if} \quad x' \notin \widetilde{\mathrm{Sig}}(v) \quad \text{and} \quad M(a, x') = b \end{cases}$$

where c_e satisfies $c_e \geq 2^{-5\tilde{\ell}}$. Thus,

$$\left\|\mathbb{P}_{x|e}\right\|_2 \leq c_e^{-1} \cdot \left\|\mathbb{P}_{x|v}\right\|_2 \leq 2^{5\tilde{\ell}} \cdot 2^{\ell_j} \cdot 2^{-n}$$

□

Claim 6.4.

$$\left\|\mathbb{P}_{x|s}\right\|_2 \leq 2^{5\tilde{\ell}} \cdot 2^{\ell_j} \cdot 2^{-n}.$$

Proof. Let $\Gamma_{in}(s)$ be the set of all edges e of B', that are going into s, such that $\Pr(e) > 0$. Note that

$$\sum_{e \in \Gamma_{in}(s)} \Pr(e) = \Pr(s).$$

By the law of total probability, for every $x' \in X$,

$$\mathbb{P}_{x|s}(x') = \sum_{e \in \Gamma_{in}(s)} \tfrac{\Pr(e)}{\Pr(s)} \cdot \mathbb{P}_{x|e}(x'),$$

and hence by Jensen's inequality,

$$\mathbb{P}_{x|s}(x')^2 \leq \sum_{e \in \Gamma_{in}(s)} \tfrac{\Pr(e)}{\Pr(s)} \cdot \mathbb{P}_{x|e}(x')^2.$$

Summing over $x' \in X$, we obtain,

$$\left\|\mathbb{P}_{x|s}\right\|_2^2 \leq \sum_{e \in \Gamma_{in}(s)} \tfrac{\Pr(e)}{\Pr(s)} \cdot \left\|\mathbb{P}_{x|e}\right\|_2^2.$$

By Claim 6.3, for any $e \in \Gamma_{in}(s)$,

$$\left\|\mathbb{P}_{x|e}\right\|_2^2 \leq \left(2^{5\tilde{\ell}} \cdot 2^{\ell_j} \cdot 2^{-n}\right)^2.$$

Hence,

$$\left\|\mathbb{P}_{x|s}\right\|_2^2 \leq \left(2^{5\tilde{\ell}} \cdot 2^{\ell_j} \cdot 2^{-n}\right)^2.$$

□

Similarity to a Target Distribution

Recall that for two functions $f, g : X \to \mathbb{R}^+$, we defined

$$\langle f, g \rangle = \mathop{\mathbf{E}}_{z \in_R X}[f(z) \cdot g(z)].$$

We think of $\langle f, g \rangle$ as a measure for the similarity between a function f and a target function g. Typically f, g will be distributions.

Claim 6.5.

$$\langle \mathbb{P}_{x|s}, \mathbb{P}_{x|s} \rangle > 2^{2\ell_j} \cdot 2^{-2n}.$$

Proof. Since s is significant,

$$\langle \mathbb{P}_{x|s}, \mathbb{P}_{x|s} \rangle = \left\| \mathbb{P}_{x|s} \right\|_2^2 > 2^{2\ell_j} \cdot 2^{-2n}. \qquad \square$$

Claim 6.6.

$$\langle \mathcal{U}_X, \mathbb{P}_{x|s} \rangle = 2^{-2n},$$

where $\mathcal{U}_X$ is the uniform distribution over X.

Proof. Since $\mathbb{P}_{x|s}$ is a distribution,

$$\langle \mathcal{U}_X, \mathbb{P}_{x|s} \rangle = 2^{-2n} \cdot \sum_{z \in X} \mathbb{P}_{x|s}(z) = 2^{-2n}. \qquad \square$$

Measuring the Progress

For $i \in \{0, \ldots, m\}$, let

$$\mathcal{Z}_i = \sum_{v \in \mathcal{L}'_i} \Pr(v) \cdot \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k.$$

For $i \in \{1, \ldots, m\}$, let

$$\mathcal{Z}'_i = \sum_{e \in \Gamma_i} \Pr(e) \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k.$$

We think of $\mathcal{Z}_i, \mathcal{Z}'_i$ as measuring the progress made by the branching program, towards reaching a state with distribution similar to $\mathbb{P}_{x|s}$.

For a vertex $v \in \mathcal{L}'_i$ of B', let $\Gamma_{out}(v)$ be the set of all edges e of B', that are going out of v to $\mathcal{L}'_{i+1}$, such that $\Pr(e) > 0$. Note that

$$\sum_{e \in \Gamma_{out}(v)} \Pr(e) \le \Pr(v).$$

(We don't always have an equality here, since sometimes $\mathcal{T}$ stops on v, or goes to a vertex from which s is not reachable).

Recall that L_s stores a (not too long) list of indices to layers on which the path might choose to go over bad edges. The next four claims show that the progress made by the branching program is slow on every layer $i \notin L_s$. On layers $i \in L_s$ the progress might be significant but we will still have meaningful bounds on it.

Claim 6.7. *For every vertex $v \in \mathcal{L}'_{i-1}$, such that* $\Pr(v) > 0$,

$$\sum_{e \in \Gamma_{out}(v)} \tfrac{\Pr(e)}{\Pr(v)} \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k \leq \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k \cdot c_i^k + 2^{-2nk+k} \cdot c_i^k,$$

where c_i is defined as

- $c_i = 1 + 2^{-r}$, *if $i \notin L_s$.*
- $c_i = 2^{5\tilde{\ell}}$, *if $i \in L_s$.*

Proof. If v is significant or v is a leaf, then $\mathcal{T}$ always stops on v and hence $\Gamma_{out}(v)$ is empty and thus the left hand side is equal to zero and the right hand side is positive, so the claim follows trivially. Thus, we can assume that v is not significant and is not a leaf.

Define $P : X \to \mathbb{R}^+$ as follows. For any $x' \in X$,

$$P(x') = \begin{cases} 0 & \text{if} \quad x' \in \widetilde{\mathrm{Sig}}(v) \\ \mathbb{P}_{x|v}(x') & \text{if} \quad x' \notin \widetilde{\mathrm{Sig}}(v) \end{cases}$$

Note that by the definition of $\mathrm{Sig}(v)$ and since $\mathrm{Sig}(v) \subseteq \widetilde{\mathrm{Sig}}(v)$, for any $x' \in X$,

$$P(x') \leq 2^{4\ell} \cdot 2^{-n}. \tag{19}$$

Define $f : X \to \mathbb{R}^+$ as follows. For any $x' \in X$,

$$f(x') = P(x') \cdot \mathbb{P}_{x|s}(x').$$

By Claim 6.4 and Equation (19),

$$\|f\|_2 \leq 2^{4\ell} \cdot 2^{-n} \cdot \left\|\mathbb{P}_{x|s}\right\|_2 \leq 2^{4\ell} \cdot 2^{-n} \cdot 2^{5\tilde{\ell}} \cdot 2^{\ell_j} \cdot 2^{-n} \leq 2^{10\ell} \cdot 2^{-2n}. \tag{20}$$

By Claim 6.2, for any edge $e \in \Gamma_{out}(v)$, labeled by (a, b), for any $x' \in X$,

$$\mathbb{P}_{x|e}(x') = \begin{cases} 0 & \text{if} \quad M(a, x') \neq b \\ P(x') \cdot c_e^{-1} & \text{if} \quad M(a, x') = b \end{cases}$$

where c_e satisfies $c_e \geq \frac{1}{2} - 2 \cdot 2^{-2r}$ if $i \notin L_s$ and $c_e \geq 2^{-5\tilde{\ell}}$ if $i \in L_s$. Denote by c_v the minimal value that c_e can get for $e \in \Gamma_{out}(v)$. By the above, $c_v \geq 2^{-5\tilde{\ell}}$ and $c_v \geq \frac{1}{2} - 2 \cdot 2^{-2r}$ if $i \notin L_s$. Note that $c_v^{-1} \leq 2c_i$ in both cases (recall that $c_i = 2^{5\tilde{\ell}}$ for $i \in L_s$ and $c_i = 1 + 2^{-r}$ if $i \notin L_s$). Therefore, for any edge $e \in \Gamma_{out}(v)$, labeled by (a, b), for any $x' \in X$,

$$\mathbb{P}_{x|e}(x') \cdot \mathbb{P}_{x|s}(x') = \begin{cases} 0 & \text{if} \quad M(a, x') \neq b \\ f(x') \cdot c_e^{-1} & \text{if} \quad M(a, x') = b \end{cases}$$

and hence, we have

$$\begin{aligned} \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle &= \mathop{\mathbf{E}}_{x' \in_R X}[\mathbb{P}_{x|e}(x') \cdot \mathbb{P}_{x|s}(x')] = \mathop{\mathbf{E}}_{x' \in_R X}[f(x') \cdot c_e^{-1} \cdot \mathbf{1}_{\{x' \in X \,:\, M(a,x') = b\}}] \\ &= \mathop{\mathbf{E}}_{x' \in_R X}\left[f(x') \cdot c_e^{-1} \cdot \tfrac{(1 + b \cdot M(a,x'))}{2}\right] \leq (\|f\|_1 + b \cdot \langle M_a, f \rangle) \cdot (2c_v)^{-1}. \end{aligned} \tag{21}$$

We will now consider two cases:

Case I: $\|f\|_1 < 2^{-2n}$

In this case, we bound $|\langle M_a, f\rangle| \leq \|f\|_1$ (since f is non-negative and the entries of M are in $\{-1, 1\}$) and obtain for any edge $e \in \Gamma_{out}(v)$,

$$\langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s}\rangle < c_v^{-1} \cdot 2^{-2n} \leq 2c_i \cdot 2^{-2n}.$$

Since $\sum_{e\in\Gamma_{out}(v)} \frac{\Pr(e)}{\Pr(v)} \leq 1$, Claim 6.7 follows, as the left hand side of the claim is smaller than the second term on the right hand side.

Case II: $\|f\|_1 \geq 2^{-2n}$

For every $a \in A$, define

$$t(a) = \frac{|\langle M_a, f\rangle|}{\|f\|_1}.$$

By Equation (21),

$$\langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s}\rangle^k < \|f\|_1^k \cdot (1 + t(a))^k \cdot (2c_v)^{-k} \tag{22}$$

Note that by the definitions of P and f,

$$\|f\|_1 = \mathop{\mathbf{E}}_{x' \in_R X}[f(x')] = \langle P, \mathbb{P}_{x|s}\rangle \leq \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s}\rangle.$$

Note also that for every $a \in A$, there is at most one edge $e_{(a,1)} \in \Gamma_{out}(v)$, labeled by $(a, 1)$, and at most one edge $e_{(a,-1)} \in \Gamma_{out}(v)$, labeled by $(a, -1)$, and we have

$$\tfrac{\Pr(e_{(a,1)})}{\Pr(v)} + \tfrac{\Pr(e_{(a,-1)})}{\Pr(v)} \leq \tfrac{1}{|A|},$$

since $\frac{1}{|A|}$ is the probability that the next sample read by the program is a. Thus, summing over all $e \in \Gamma_{out}(v)$, by Equation (22),

$$\sum_{e\in\Gamma_{out}(v)} \tfrac{\Pr(e)}{\Pr(v)} \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s}\rangle^k < \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s}\rangle^k \cdot \mathop{\mathbf{E}}_{a\in_R A}\left[(1 + t(a))^k\right] \cdot (2c_v)^{-k}. \tag{23}$$

It remains to bound

$$\mathop{\mathbf{E}}_{a\in_R A}\left[(1 + t(a))^k\right], \tag{24}$$

using the properties of the matrix M and the bounds on the L_2 versus L_1 norms of f.

By Equation (20) and the assumption that $\|f\|_1 \geq 2^{-2n}$ we get

$$\frac{\|f\|_2}{\|f\|_1} \leq 2^{10\ell}.$$

Since M is a $(10k, 10\ell)$-L_2-extractor with error 2^{-10r}, there are at most $2^{-10k} \cdot |A|$ rows $a \in A$ with $t(a) = \frac{|\langle M_a, f\rangle|}{\|f\|_1} \geq 2^{-10r}$. We bound the expectation in Equation (24), by splitting the expectation into two sums

$$\mathop{\mathbf{E}}_{a\in_R A}\left[(1 + t(a))^k\right] = \tfrac{1}{|A|} \cdot \sum_{a\,:\,t(a)\leq 2^{-10r}} (1 + t(a))^k + \tfrac{1}{|A|} \cdot \sum_{a\,:\,t(a)>2^{-10r}} (1 + t(a))^k. \tag{25}$$

We bound the first sum in Equation (25) by $(1+2^{-10r})^k$. As for the second sum in Equation (25), we know that it is a sum of at most $2^{-10k} \cdot |A|$ elements, and since for every $a \in A$, we have $t(a) \leq 1$, we have

$$\tfrac{1}{|A|} \cdot \sum_{a \,:\, t(a) > 2^{-10r}} (1+t(a))^k \leq 2^{-10k} \cdot 2^k \leq 2^{-2r}$$

(where in the last inequality we used the fact that $r \leq k$). Overall, we get

$$\mathop{\mathbf{E}}_{a \in_R A}\left[(1+t(a))^k\right] \leq (1+2^{-10r})^k + 2^{-2r} \leq (1+2^{-2r})^{k+1}. \tag{26}$$

Substituting Equation (26) into Equation (23), we obtain

$$\sum_{e \in \Gamma_{out}(v)} \tfrac{\Pr(e)}{\Pr(v)} \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k < \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k \cdot \left(1+2^{-2r}\right)^{k+1} \cdot (2c_v)^{-k}.$$

If $i \notin L_s$, then $(2c_v)^{-1} \leq (1+2^{-2r+3})$ and thus $(1+2^{-2r})^{k+1} \cdot (2c_v)^{-k} \leq (1+2^{-r})^k$ (where the inequality uses the assumption that r is sufficiently large).

If $i \in L_s$, then $(2c_v)^{-1} \leq \frac{1}{2} \cdot 2^{5\tilde{\ell}}$ and thus $(1+2^{-2r})^{k+1} \cdot (2c_v)^{-k} \leq 2^{5\tilde{\ell}k}$. This completes the proof of Claim 6.7. □

Claim 6.8. *Recall the definition of c_i from Claim 6.7. For every $i \in \{1, \ldots, m\}$,*

$$\mathcal{Z}'_i \leq (\mathcal{Z}_{i-1} + 2^{-2nk+k}) \cdot c_i^k$$

Proof. By Claim 6.7,

$$\begin{aligned}
\mathcal{Z}'_i = \sum_{e \in \Gamma_i} \Pr(e) \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k &= \sum_{v \in \mathcal{L}'_{i-1}} \Pr(v) \cdot \sum_{e \in \Gamma_{out}(v)} \tfrac{\Pr(e)}{\Pr(v)} \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k \\
&\leq \sum_{v \in \mathcal{L}'_{i-1}} \Pr(v) \cdot \left(\langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k + 2^{-2nk+k}\right) \cdot c_i^k \\
&= c_i^k \cdot \left(\mathcal{Z}_{i-1} + \sum_{v \in \mathcal{L}'_{i-1}} \Pr(v) \cdot 2^{-2nk+k}\right) \\
&\leq c_i^k \cdot \left(\mathcal{Z}_{i-1} + 2^{-2nk+k}\right)
\end{aligned}$$

□

Claim 6.9. *For every $i \in \{1, \ldots, m\}$,*

$$\mathcal{Z}_i \leq \mathcal{Z}'_i.$$

Proof. For any $v \in \mathcal{L}'_i$, let $\Gamma_{in}(v)$ be the set of all edges $e \in \Gamma_i$, that are going into v. Note that

$$\sum_{e \in \Gamma_{in}(v)} \Pr(e) = \Pr(v).$$

By the law of total probability, for every $v \in \mathcal{L}'_i$ and every $x' \in X$,

$$\mathbb{P}_{x|v}(x') = \sum_{e \in \Gamma_{in}(v)} \tfrac{\Pr(e)}{\Pr(v)} \cdot \mathbb{P}_{x|e}(x'),$$

and hence

$$\langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle = \sum_{e \in \Gamma_{in}(v)} \tfrac{\Pr(e)}{\Pr(v)} \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle.$$

Thus, by Jensen's inequality,

$$\langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k \le \sum_{e \in \Gamma_{in}(v)} \tfrac{\Pr(e)}{\Pr(v)} \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k.$$

Summing over all $v \in \mathcal{L}'_i$, we get

$$\begin{aligned}\mathcal{Z}_i = \sum_{v \in \mathcal{L}'_i} \Pr(v) \cdot \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k &\le \sum_{v \in \mathcal{L}'_i} \Pr(v) \cdot \sum_{e \in \Gamma_{in}(v)} \tfrac{\Pr(e)}{\Pr(v)} \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k \\ &= \sum_{e \in \Gamma_i} \Pr(e) \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k = \mathcal{Z}'_i. \qquad \square\end{aligned}$$

Claim 6.10. *For every $i \in \{1, \dots, m\}$,*

$$\mathcal{Z}_i \le 2^{r+3k+5\tilde{\ell}k \cdot |L_s|} \cdot 2^{-2k \cdot n}.$$

Proof. By Claim 6.8 and Claim 6.9, for every $i \in \{1, \dots, m\}$,

$$\mathcal{Z}_i \le (\mathcal{Z}_{i-1} + 2^{-2nk+k}) \cdot c_i^k$$

where $c_i = (1 + 2^{-r})$ if $i \notin L_s$ and $c_i = 2^{5\tilde{\ell}}$ if $i \in L_s$. Thus, we can show by induction on $i \in \{1, \dots, m\}$ that

$$\mathcal{Z}_i \le 2^{-2nk+k} \cdot (i+1) \cdot \prod_{i'=1}^{i} c_{i'}^k$$

Hence, for any $i \in \{1, \dots, m\}$ it holds that

$$\mathcal{Z}_i \le 2^{-2nk+k} \cdot (m+1) \cdot (1 + 2^{-r})^{mk} \cdot 2^{5\tilde{\ell}k|L_s|}.$$

Since $m \le 2^{\epsilon\tilde{r}} \le 2^r - 1$,

$$\mathcal{Z}_i \le 2^{-2k \cdot n + k} \cdot 2^r \cdot e^k \cdot 2^{5\tilde{\ell}k|L_s|}. \qquad \square$$

Proof of Lemma 5.3

We can now complete the proof of Lemma 5.3. Assume that s is in layer-i of B'. By Claim 6.5,

$$\mathcal{Z}_i \ge \Pr(s) \cdot \langle \mathbb{P}_{x|s}, \mathbb{P}_{x|s} \rangle^k > \Pr(s) \cdot \left(2^{2\ell_j} \cdot 2^{-2n}\right)^k = \Pr(s) \cdot 2^{2\ell_j \cdot k} \cdot 2^{-2k \cdot n}.$$

On the other hand, by Claim 6.10,

$$\mathcal{Z}_i \le 2^{r+3k+5\tilde{\ell}k \cdot |L_s|} \cdot 2^{-2k \cdot n}.$$

Thus, we get

$$\Pr(s) \le 2^{r+3k+5\tilde{\ell}k \cdot |L_s|} \cdot 2^{-2\ell_j \cdot k}$$

We treat differently the case $j = 1$ and $j = 2$ as follows. For $j = 1$, the set L_s is empty, and we have

$$\Pr(s) \leq 2^{r+3k} \cdot 2^{-2\ell_1 \cdot k} \leq 2^{-\ell_1 \cdot k}.$$

For $j = 2$ we have

$$\begin{aligned} \Pr(s) &\leq 2^{r+3k+5\tilde{\ell}k \cdot |L_s|} \cdot 2^{-2\ell \cdot k} \\ &\leq 2^{r+3k+1000\epsilon\tilde{\ell}^2 k} \cdot 2^{-2\ell \cdot k} && (|L_s| \leq 200\epsilon\tilde{\ell}) \\ &\leq 2^{4k+1000\epsilon\ell k} \cdot 2^{-2\ell \cdot k} && (r \leq k,\ \tilde{\ell} \leq \sqrt{\ell}) \\ &\leq 2^{-\ell k} \leq 2^{-\ell_1 \cdot k}\ . && (\epsilon < 1/10^{10}) \end{aligned}$$

Thus, in both cases we showed $\Pr(s) \leq 2^{-\ell_1 k}$.

Recall that we showed that the width of $\hat{B}$ is at most $2^{\epsilon k \tilde{\ell}/2}$, and note that the length of $\hat{B}$ is at most $2 \cdot 2^{\epsilon \tilde{r}}$. Taking a union bound over at most $2^{\epsilon k \tilde{\ell}/2} \cdot 2 \cdot 2^{\epsilon \tilde{r}} \leq 2^{k\ell_1/2}$ significant vertices of $\hat{B}$, we conclude that the probability that $\mathcal{T}$ reaches any significant vertex is at most $2^{-k\ell_1/2} = o(1)$. □

References

[B89] David A. Mix Barrington: Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC1. J. Comput. Syst. Sci. 38(1): 150-164 (1989)

[BOGY18] Paul Beame, Shayan Oveis Gharan, Xin Yang: Time-Space Tradeoffs for Learning Finite Functions from Random Evaluations, with Applications to Polynomials. COLT 2018: 843-856

[CG88] Benny Chor, Oded Goldreich: Unbiased Bits from Sources of Weak Randomness and Probabilistic Communication Complexity. SIAM J. Comput. 17(2): 230-261 (1988)

[DS18] Yuval Dagan, Ohad Shamir: Detecting Correlations with Little Memory and Communication. COLT 2018: 1145-1198

[GRT18] Sumegha Garg, Ran Raz, Avishay Tal: Extractor-Based Time-Space Lower Bounds for Learning. STOC 2018: 990-1002

[KR13] Gillat Kol, Ran Raz: Interactive channel capacity. STOC 2013: 715-724

[KRT17] Gillat Kol, Ran Raz, Avishay Tal: Time-Space Hardness of Learning Sparse Parities. STOC 2017: 1067-1080

[MM17] Dana Moshkovitz, Michal Moshkovitz: Mixing Implies Lower Bounds for Space Bounded Learning. COLT 2017: 1516-1566

[MM18] Dana Moshkovitz, Michal Moshkovitz: Entropy Samplers and Strong Generic Lower Bounds For Space Bounded Learning. ITCS 2018: 28:1-28:20

[MT17] Michal Moshkovitz, Naftali Tishby: Mixing Complexity and its Applications to Neural Networks. CoRR abs/1703.00729 (2017)

[R16] Ran Raz: Fast Learning Requires Good Memory: A Time-Space Lower Bound for Parity Learning. FOCS 2016: 266-275

[R17] Ran Raz: A Time-Space Lower Bound for a Large Class of Learning Problems. FOCS 2017: 732-742

[S14] Ohad Shamir: Fundamental Limits of Online and Distributed Algorithms for Statistical Learning and Estimation. NIPS 2014: 163-171

[SV84] Miklos Santha, Umesh V. Vazirani: Generating Quasi-Random Sequences from Slightly-Random Sources. FOCS 1984: 434-440

[SVW16] Jacob Steinhardt, Gregory Valiant, Stefan Wager: Memory, Communication, and Statistical Queries. COLT 2016: 1490-1516

[VV16] Gregory Valiant, Paul Valiant: Information Theoretically Secure Databases. Electronic Colloquium on Computational Complexity (ECCC) 23: 78 (2016)

7 Appendix

We first state the main theorem of [GRT18] and then the modified proposition used in the proof of Lemma 5.5.

Theorem 2 (Theorem 1, [GRT18]). *Let $\frac{1}{100} < c < \frac{2}{3}$. Fix γ to be such that $\frac{3c}{2} < \gamma^2 < 1$. Let X, A be two finite sets. Let $n = \log_2 |X|$. Let $M : A \times X \to \{-1, 1\}$ be a matrix which is a (k', ℓ')-L_2-extractor with error $2^{-r'}$, for sufficiently large[1] k', ℓ' and r', where $\ell' \leq n$. Let*

$$r := \min\left\{\tfrac{r'}{2}, \tfrac{(1-\gamma)k'}{2}, \tfrac{(1-\gamma)\ell'}{2} - 1\right\}.$$

Let B be a branching program of length at most 2^r and width at most $2^{c \cdot k' \cdot \ell'}$ for the learning problem that corresponds to the matrix M. Then, the success probability of B is at most $O(2^{-r})$.

The authors prove the above theorem by first defining a truncated path that stops on a significant vertex, a significant value or a bad edge, such that, if the path doesn't stop before reaching a leaf, then the probability of guessing the correct x is small (at most $O(2^{-r})$ to be precise). Then, the authors prove that the probability that the truncated path stops is at most $O(2^{-r})$. Through slight modifications to the proof of the above theorem (with weaker bounds on the memory and length of B, in terms of constants), we can prove that the probability that a slightly modified truncated path stops is at most $2^{-\Omega(\min\{k',\ell'\})}$. As the modified proof is very similar to that of Theorem 2 and the original proof is lengthy, we just highlight the changes to the proof to get the following proposition.

[1] k', ℓ', r' are larger than some constant that depends on γ.

Proposition 7.1. *Let X, A be two finite sets. Let $n = \log_2 |X|$. Let $M : A \times X \to \{-1, 1\}$ be a matrix which is a (k', ℓ')-L_2-extractor with error $2^{-r'}$, for sufficiently large k', ℓ' and r', where $\ell' \leq n$. Let*

$$r := \frac{\min\{r', k', \ell'\}}{100}.$$

Let B be a branching program of length at most 2^r and width at most $2^{\frac{k' \cdot \ell'}{100}}$ for the learning problem that corresponds to the matrix M. Then, there exists an event G such that

$$\Pr[G] \geq 1 - 2^{-\frac{\min\{k',\ell'\}}{8}}$$

and for every $x' \in X$ and every leaf z of the branching program B (with starting vertex z_0),

$$\Pr\left[x = x' \mid G \wedge (z_0 \widetilde{\rightarrow} z)\right] \leq 2^{2\ell'} \cdot 2^{-n},$$

whenever the event $G \wedge (z_0 \widetilde{\rightarrow} z)$ is non-empty, where $z_0 \widetilde{\rightarrow} z$ denotes the event that the computational path (as opposed to the truncated path) from z_0 reaches z.

Proof. The proof of Theorem 1 of [GRT18] defines the *truncated-path*, $\mathcal{T}$, to be the same as the computation-path of B, except that it sometimes stops before reaching a leaf. Roughly speaking, $\mathcal{T}$ stops before reaching a leaf if certain "bad" events occur. Nevertheless, the proof shows that the probability that $\mathcal{T}$ stops before reaching a leaf is negligible, so we can think of $\mathcal{T}$ as almost identical to the computation-path.

For a vertex v of B, we denote by E_v the event that $\mathcal{T}$ reaches the vertex v. We denote by $\Pr(v) = \Pr(E_v)$ the probability for E_v, and we denote by $\mathbb{P}_{x|v} = \mathbb{P}_{x|E_v}$ the distribution of the random variable x conditioned on the event E_v.

We first look at the definition of the truncated-path from the proof of Theorem 1 of [GRT18]. We modify the stopping rules for a path as follows:

Let $\hat{l} = \frac{\ell'}{6}$.

Significant Vertices: A vertex v in layer-i of B is significant if

$$\left\|\mathbb{P}_{x|v}\right\|_2 > 2^{\hat{l}} \cdot 2^{-n}.$$

Significant Values: Even if v is not significant, $\mathbb{P}_{x|v}$ may have relatively large values. For a vertex v in layer-i of B, denote by $\mathrm{Sig}(v)$ the set of all $x' \in X$, such that,

$$\mathbb{P}_{x|v}(x') > 2^{3\hat{l}} \cdot 2^{-n}.$$

Bad Edges: For a vertex v in layer-i of B, denote by $\mathrm{Bad}(v)$ the set of all $\alpha \in A$, such that,

$$\left|(M \cdot \mathbb{P}_{x|v})(\alpha)\right| \geq 2^{-r'}.$$

Recall, that the truncated path is defined by induction on the layers of the branching program B:

The Truncated-Path $\mathcal{T}$

Assume that we already defined $\mathcal{T}$ until it reaches a vertex v in layer-i of B. The path $\mathcal{T}$ stops on v if (at least) one of the following occurs:

1. v is significant.

2. $x \in \mathrm{Sig}(v)$.

3. $a_{i+1} \in \mathrm{Bad}(v)$.

4. v is a leaf.

Otherwise, $\mathcal{T}$ proceeds by following the edge labeled by (a_{i+1}, b_{i+1}) (same as the computational-path).

The Event G

We define G to be the event that the truncated-path $\mathcal{T}$ didn't stop because of one of the first three stopping rules: That is, $\mathcal{T}$ didn't stop before reaching a leaf and didn't violate the significant vertices and significant values stopping rules (that is, the first two stopping rules) on the leaf that it reached.

We can upper bound the probability for $\bar{G}$ similarly to the way that it's done in [GRT18].

Lemma 7.2. *The probability that $\mathcal{T}$ reaches a significant vertex is at most $2^{-k'}$.*

The proof of the above lemma is very similar to the analogous lemma in the proof of Theorem 2. The only change is in the definition of significant value - we define the significant values to be the set of all $x' \in X$, such that, $\mathbb{P}_{x|v}(x') > 2^{3\hat{l}} \cdot 2^{-n}$ instead of the set of all $x' \in X$, such that, $\mathbb{P}_{x|v}(x') > 2^{2\hat{l}+2r} \cdot 2^{-n}$. With the above (worse in terms of constants) bounds on the memory and the length of the branching program, the proof works in the same way.

Lemma 7.2 shows that the probability that $\mathcal{T}$ stops on a vertex, because of the first reason (i.e., that the vertex is significant), is small. The next two claims imply that the probabilities that $\mathcal{T}$ stops on a vertex, because of the second and third reasons, are also small.

Claim 7.3. *If v is a non-significant vertex of B then*

$$\Pr_x[x \in \mathrm{Sig}(v) \mid E_v] \le 2^{-\hat{l}}.$$

Proof. Since v is not significant,

$$\mathop{\mathbf{E}}_{x' \sim \mathbb{P}_{x|v}} \left[\mathbb{P}_{x|v}(x')\right] = \sum_{x' \in X} \left[\mathbb{P}_{x|v}(x')^2\right] = 2^n \cdot \mathop{\mathbf{E}}_{x' \in_R X} \left[\mathbb{P}_{x|v}(x')^2\right] \le 2^{2\hat{l}} \cdot 2^{-n}.$$

Hence, by Markov's inequality,

$$\Pr_{x' \sim \mathbb{P}_{x|v}} \left[\mathbb{P}_{x|v}(x') > 2^{\hat{l}} \cdot 2^{2\hat{l}} \cdot 2^{-n}\right] \le 2^{-\hat{l}}.$$

Since conditioned on E_v, the distribution of x is $\mathbb{P}_{x|v}$, we obtain

$$\Pr_x \left[x \in \mathrm{Sig}(v) \mid E_v\right] = \Pr_x \left[\left(\mathbb{P}_{x|v}(x) > 2^{\hat{l}} \cdot 2^{2\hat{l}} \cdot 2^{-n}\right) \mid E_v \right] \le 2^{-\hat{l}}. \qquad \square$$

Claim 7.4. *If v is a non-significant vertex of B then*

$$\Pr_{a_{i+1}}\left[a_{i+1} \in \mathrm{Bad}(v)\right] \leq 2^{-k'}.$$

Proof. Since v is not significant, $\left\|\mathbb{P}_{x|v}\right\|_2 \leq 2^{\hat{l}} \cdot 2^{-n}$. Since $\mathbb{P}_{x|v}$ is a distribution, $\left\|\mathbb{P}_{x|v}\right\|_1 = 2^{-n}$. Thus,

$$\frac{\left\|\mathbb{P}_{x|v}\right\|_2}{\left\|\mathbb{P}_{x|v}\right\|_1} \leq 2^{\hat{l}} \leq 2^{\ell'}.$$

Since M is a (k', ℓ')-L_2-extractor with error $2^{-r'}$, there are at most $2^{-k'} \cdot |A|$ elements $\alpha \in A$ with

$$\left|\langle M_\alpha, \mathbb{P}_{x|v}\rangle\right| \geq 2^{-r'} \cdot \left\|\mathbb{P}_{x|v}\right\|_1 = 2^{-r'} \cdot 2^{-n}.$$

The claim follows since a_{i+1} is uniformly distributed over A. □

We can now use Lemma 7.2, Claim 7.3 and Claim 7.4 to prove that the probability that $\mathcal{T}$ stops because of the first three stopping rules is at most $2^{-\frac{\min\{k',\ell'\}}{8}}$. Lemma 7.2 shows that the probability that $\mathcal{T}$ reaches a significant vertex and hence stops because of the first stopping rule, is at most $2^{-k'}$. Assuming that $\mathcal{T}$ doesn't reach any significant vertex (in which case it would have stopped because of the first stopping rule), Claim 7.3 shows that in each step, the probability that $\mathcal{T}$ stops because of the second stopping rule, is at most $2^{-\frac{\ell'}{6}}$. Taking a union bound over the 2^r steps, the total probability that $\mathcal{T}$ stops because of the second stopping rule, is at most $2^{-\frac{\ell'}{7}}$ (for sufficiently large ℓ'). In the same way, assuming that $\mathcal{T}$ doesn't reach any significant vertex (in which case it would have stopped because of the first stopping rule), Claim 7.4 shows that in each step, the probability that $\mathcal{T}$ stops because of the third stopping rule, is at most $2^{-k'}$. Again, taking a union bound over the 2^r steps, the total probability that $\mathcal{T}$ stops because of the third stopping rule, is at most $2^{-\frac{k'}{7}}$. Thus, the total probability that $\mathcal{T}$ stops (for any reason) before reaching a leaf (or violated the significant vertices or significant values stopping rules (that is, the first two stopping rules) on the leaf that it reached) is at most $2^{-\frac{\min\{k',\ell'\}}{8}}$. (Summing over the three probabilities and using the fact that k', ℓ' are sufficiently large).

Thus, $\Pr[\bar{G}] \leq 2^{-\frac{\min\{k',\ell'\}}{8}}$, as required.

Bounding $\Pr\left[\mathbf{x} = \mathbf{x}' \mid \mathbf{G} \wedge (\mathbf{z_0} \widetilde{\rightarrow} \mathbf{z})\right]$:

It remains to prove that for every $x' \in X$ and every leaf z of the branching program B (with starting vertex z_0),

$$\Pr\left[x = x' \mid G \wedge (z_0 \widetilde{\rightarrow} z)\right] \leq 2^{2\ell'} \cdot 2^{-n},$$

whenever the event $G \wedge (z_0 \widetilde{\rightarrow} z)$ is non-empty, where $z_0 \widetilde{\rightarrow} z$ denotes the event that the computational path (as opposed to the truncated path) from z_0 reaches z.

Recall that E_z is the event that $\mathcal{T}$ reaches the vertex z.

Claim 7.5. *The event $G \wedge (z_0 \widetilde{\rightarrow} z)$ is equivalent to $E_z \wedge (z$ is not significant$) \wedge (x \notin \mathrm{Sig}(z))$.*

Proof. If $G \wedge (z_0 \widetilde{\rightarrow} z)$ occurs then the truncated-path $\mathcal{T}$ didn't stop before reaching a leaf (since G occurs) and the computational path from z_0 reaches z (since $(z_0 \widetilde{\rightarrow} z)$ occurs). Thus, E_z occurs. Also, since the first stopping rule is not violated on z, we have that z is not significant and since the second stopping rule is not violated on z, we have $x \notin \mathrm{Sig}(z)$.

On the other direction, if $E_z \wedge (z \text{ is not significant}) \wedge (x \notin \mathrm{Sig}(z))$ occurs, then $(z_0 \widetilde{\rightarrow} z)$ occurs (since E_z occurs), the truncated-path $\mathcal{T}$ didn't stop before reaching a leaf (since E_z occurs) and none of the first two stopping rules are violated on z, since z is not significant and $x \notin \mathrm{Sig}(z)$. □

By Claim 7.5, it remains to prove that for every leaf z and every $x' \in X$, if the event $E_z \wedge (z \text{ is not significant}) \wedge (x \notin \mathrm{Sig}(z))$ is non-empty then

$$\Pr\left[x = x' \mid E_z \wedge (z \text{ is not significant}) \wedge (x \notin \mathrm{Sig}(z))\right] \le 2^{2\ell'} \cdot 2^{-n}.$$

Equivalently, we need to prove that for every non-significant leaf z and every $x' \in X$, if the event $E_z \wedge (x \notin \mathrm{Sig}(z))$ is non-empty then

$$\mathbb{P}_{x|E_z \wedge (x \notin \mathrm{Sig}(z))}(x') = \Pr\left[x = x' \mid E_z \wedge (x \notin \mathrm{Sig}(z))\right] \le 2^{2\ell'} \cdot 2^{-n}.$$

By the definition of conditional distribution,

$$\mathbb{P}_{x|E_z \wedge (x \notin \mathrm{Sig}(z))}(x') = \begin{cases} 0 & \text{if} \quad x' \in \mathrm{Sig}(z) \\ \mathbb{P}_{x|E_z}(x') \cdot c^{-1} & \text{if} \quad x' \notin \mathrm{Sig}(z) \end{cases}$$

where $c = \sum_{x' \notin \mathrm{Sig}(z)} \mathbb{P}_{x|E_z}(x')$ is the normalization factor. As z is not significant, by Claim 7.3,

$$\Pr_x[x \in \mathrm{Sig}(z) \mid E_z] \le 2^{-\hat{l}}.$$

Therefore, $c \ge 1 - 2^{-\hat{l}}$. Since by the definition of $\mathrm{Sig}(z)$, for $x' \notin \mathrm{Sig}(z)$, we have $\mathbb{P}_{x|z}(x') \le 2^{3\hat{l}} \cdot 2^{-n}$, we can bound

$$\mathbb{P}_{x|E_z \wedge (x \notin \mathrm{Sig}(z))}(x') \le 2^{3\hat{l}} \cdot 2^{-n} \cdot c^{-1} \le 2^{3\hat{l}+1} \cdot 2^{-n} \le 2^{2\ell'} \cdot 2^{-n}.$$

□