Time-Space Hardness for Learning Problems

Avishay Tal (**Stanford**)

Based on joint works with

Sumegha Garg, **Gillat Kol** & **Ran Raz**

Learning – The Streaming Model

aka Online Learning

Black Box

(0,1,1,0,1)

f

0

Learner

stream of examples

[Shamir'2014]
[Steinhardt-Valiant-Wager'2015]

Examples of Learning Problems

Parity Learning: for $a, x \in \{0,1\}^n$

$$f_x(a) = \langle a, x \rangle \ (mod\ 2)$$

DNF Learning: f is a small size DNF formula

Decision Tree Learning:

f is a small size decision tree

Junta Learning:

f depends only on $\ell \ll n$ of the input bits.

Parity Learning Problem

$$f_x(a) = \langle a, x \rangle \ (mod\ 2)$$

$x \in \{0,1\}^n$ is **unknown** to the learner

Given a stream of examples
$(a_1, b_1), (a_2, b_2), (a_3, b_3), \ldots,$
where $a_i \in_R \{0,1\}^n$ and $b_i = \langle a_i, x \rangle$,
the learner needs to learn x with high probability.

Parity Learning Problem

$$f_x(a) = \langle a, x \rangle \ (mod\ 2)$$

$x \in_R \{0,1\}^n$ is chosen uniformly at random

x is **unknown** to the learner

Given a stream of examples

$(a_1, b_1), (a_2, b_2), (a_3, b_3), \ldots,$

where $a_i \in_R \{0,1\}^n$ and $b_i = \langle a_i, x \rangle$,

the learner needs to learn x with high probability.

Algorithms for Parity Learning:

$$f_x(a) = \sum_{i=1}^{n} a_i x_i \pmod 2$$

1. Gaussian Elimination
   $O(n^2)$ **memory bits,** $O(n)$ **samples.**
2. Trying all possibilities
   $O(n)$ **memory bits,** $O(2^n \cdot n^2)$ **samples.**

Raz's Breakthrough

Theorem [Raz'16]: Any algorithm for parity learning requires either $\Omega(n^2)$ memory bits or an exponential number of samples.

Sparse Parities

$$f_x(a) = \sum_{i=1}^{n} a_i x_i \pmod 2$$

Could we learn better if we knew that $(x_1, \ldots, x_n)$ is ℓ-sparse (i.e., $\sum_{i=1}^{n} x_i = \ell$)?

Note: any $\log(n)$-sparse parity is also:

- $O(n)$ size DNF formula,
- $O(n)$ size decision-tree,
- Junta on $\log(n)$ variables.

Lower bounds for learning $\log(n)$-sparse parities
➔Lower bounds for learning all of the above

Upper Bounds

$\boxed{f_x(a) = \sum_{i=1}^{n} a_i x_i \pmod 2}$ $\boxed{\sum_{i=1}^{n} x_i = \ell}$

1. Trying all possibilities:

$O\left(\binom{n}{\ell} \cdot n^2\right) \approx n^{\ell+2}$ samples

$O(\ell \cdot \log n)$ memory bits

2. Record and Eliminate (like Gaussian Elim.)

i. Record $O(\ell \cdot \log n)$ equations in memory.

ii. Check which of all possible ℓ-sparse vectors satisfies the recorded equations.

$O(\ell \cdot \log n)$ samples

$O(n\ell \cdot \log n)$ memory bits

Algorithm #3: O(n) memory and $\ell^{O(\ell)}$ samples.

Can we learn $\log(n)$-sparse parities in O(n) memory and polynomial number of samples? **No!**

Theorem [Kol-Raz-T'17]
Any algorithm for ℓ-sparse parity learning requires either $\Omega(n \cdot \ell^{0.99})$ memory bits or $\ell^{\Omega(\ell)}$ samples.

➔ $\log(n)$-sparse parity learning requires either $\Omega(n \cdot \log^{0.99} n)$ memory or $n^{\Omega(\log \log n)}$ samples.

Motivation: Cryptography

[Raz 16, Valiant-Valiant 16]

Applications to Bounded Storage Crypto:

Encryption/Decryption scheme with:

Key's length: n

Encryption/Decryption time: n

Unconditional security, if the attacker's memory size is at most $n^2/10$

Previous works assumed that the attacker's memory size is at most linear in the time needed for encryption/decryption

Motivation: Cryptography

[Raz 16, Valiant-Valiant 16, Kol-Raz-T 16]

Applications to Bounded Storage Crypto:

Encryption/Decryption scheme with:

Key's length: ℓ

Encryption/Decryption time: n

Unconditional security, if the attacker's memory size is at most $o(n \cdot \ell)$

In the second part of the talk:

Key's length: n

Encryption/Decryption time: ℓ

Secure against memory size $o(n \cdot \ell)$

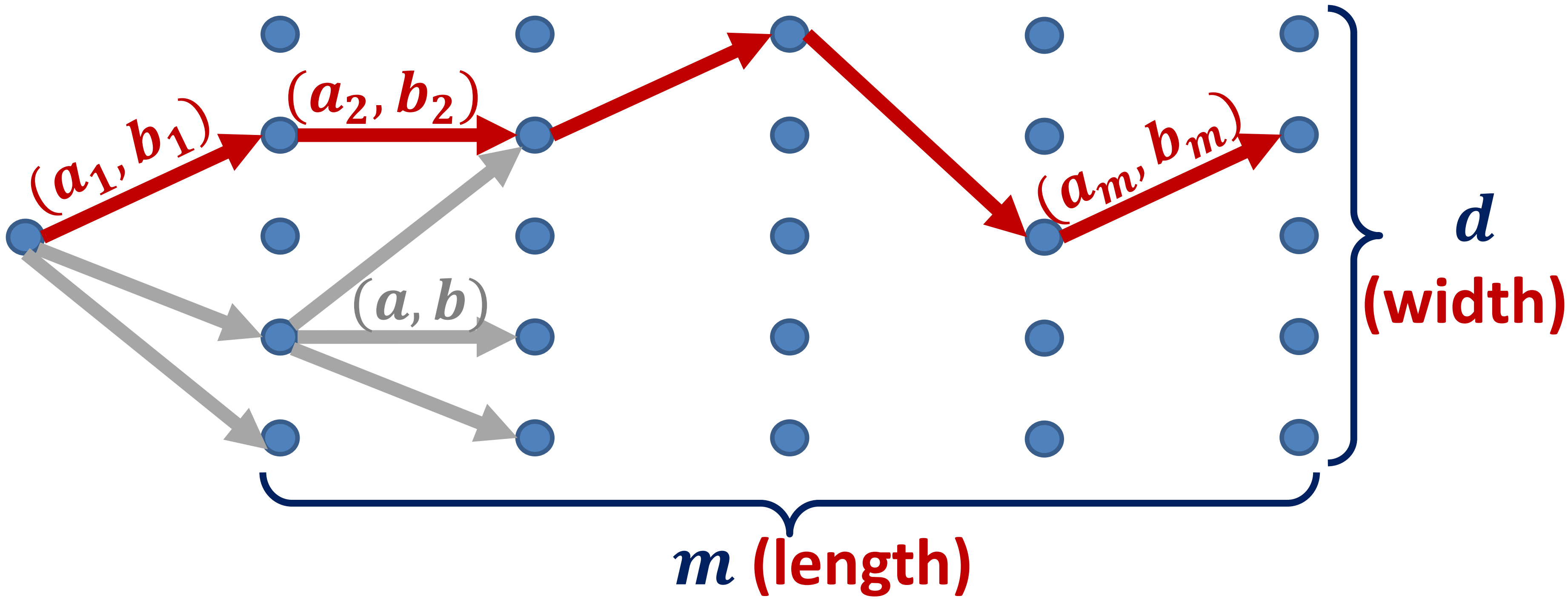
Motivation: Complexity Theory

Time-Space Lower Bounds have been studied in many models

[Beame-Jayram-Saks 98, Ajtai 99, Beame-Saks-Sun-Vee'00, Fortnow 97, Fortnow-Lipton-van Melkebeek-Viglas05, Williams'06,…]

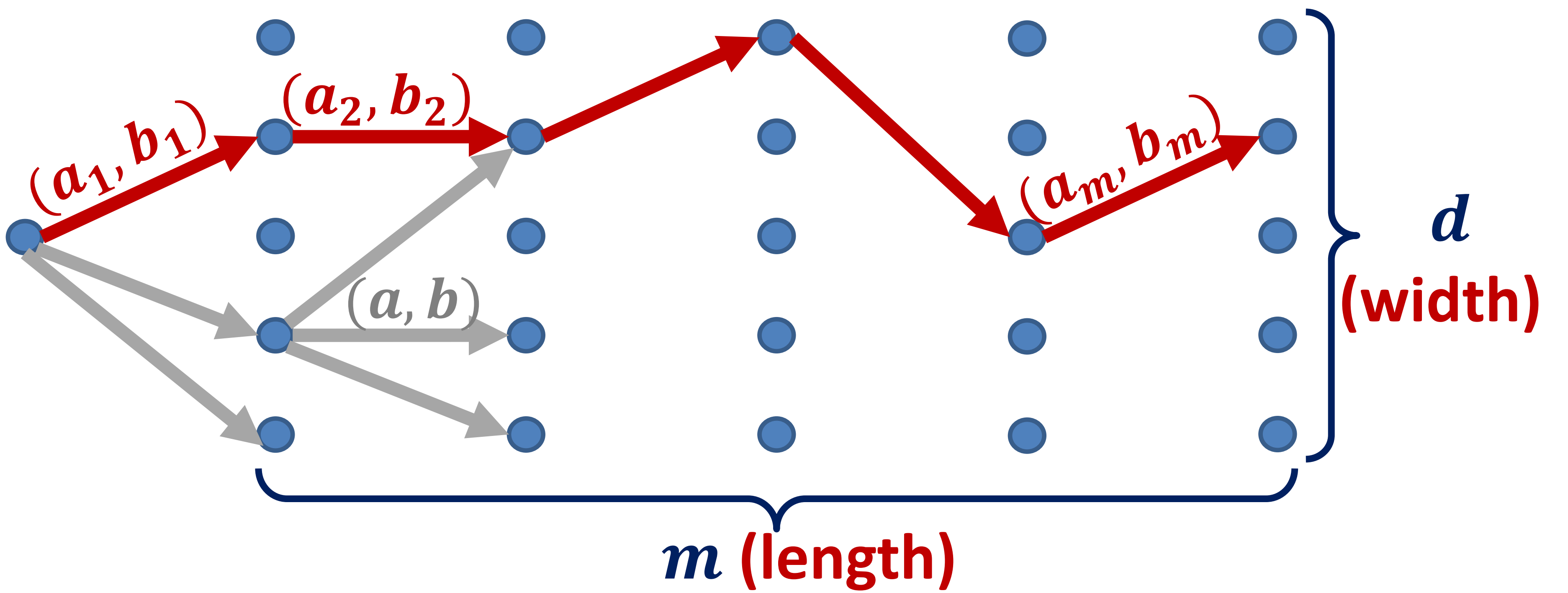
Main difference:
the online model is easier to prove lower bounds against, since the input is read only once.

The Branching Program (BP) Model



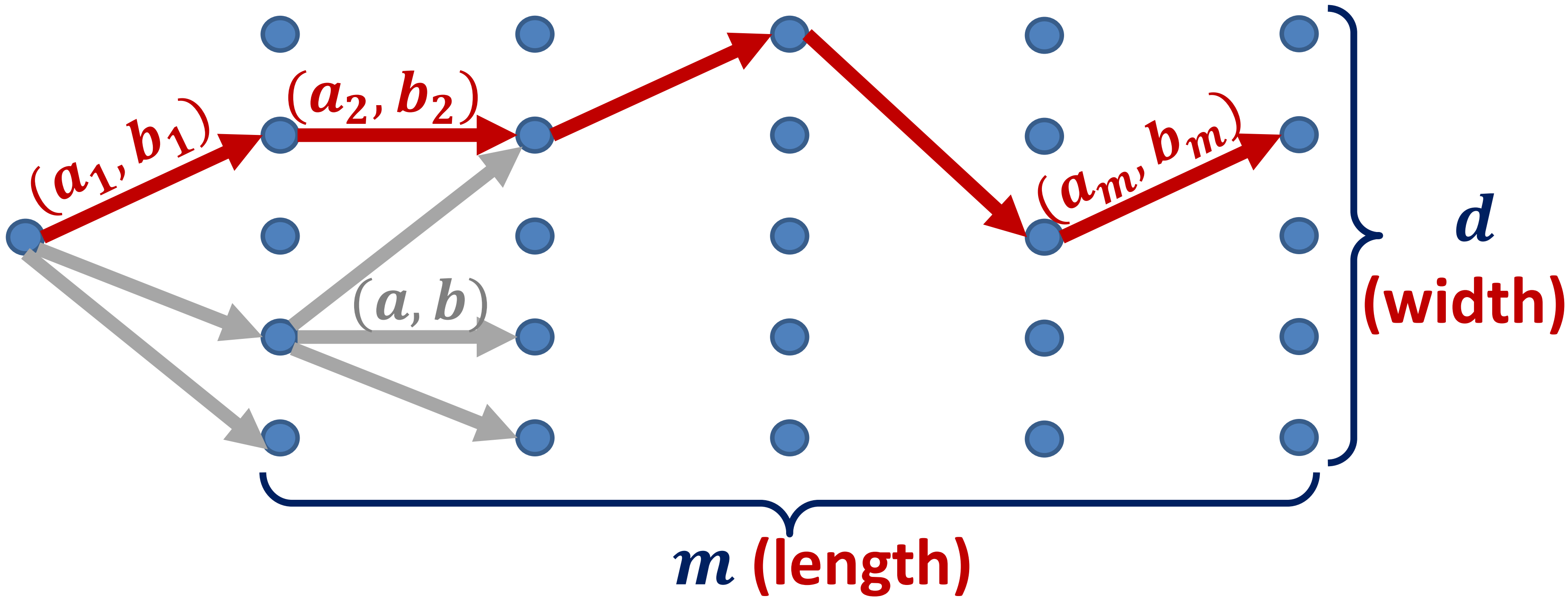
Each layer represents a time step. Each vertex represents a memory state of the learner. Each non-leaf vertex has 2^{n+1} outgoing edges, one for each $(a, b) \in \{0,1\}^n \times \{0,1\}$.

The Branching Program (BP) Model



A sequence of random examples $(a_1, b_1), (a_2, b_2)$ ... defines a computation path in the **BP**. The path finally reaches a leaf v and outputs $\tilde{x}_v$, a guess for the value of x. The program is successful if $x = \tilde{x}_v$.

Affine Branching Programs (ABP)



An **ABP** is a **BP** where each vertex v “remembers” a set of **linear equations** L_v in the variables $x_1, .., x_n$, such that, if v is reached by the computation-path then all equations in L_v are satisfied (by the true unknown x).

Accurate Affine BPs

Let V_i be the vertex reached by the computational path of the **ABP** in layer i.

V_i is a random variable that depends on $x, a_1, \dots, a_i$.

$\boldsymbol{P}_{x|V_i=v}$ = the distribution of x conditioned on reaching a specific vertex v in layer i.

Accurate ABP: for every v, $\boldsymbol{P}_{x|v}$ is close to uniform over the set of (ℓ-**sparse**) solutions to the eqs L_v.

Proof Plan

We follow Raz's two steps plan:

1. Simulate any **BP** for sparse parity learning with an accurate **ABP**.
2. Prove that **ABP** for sparse parity learning must be either **wide** or **long.**

Fix some parameter $k \approx \ell$.

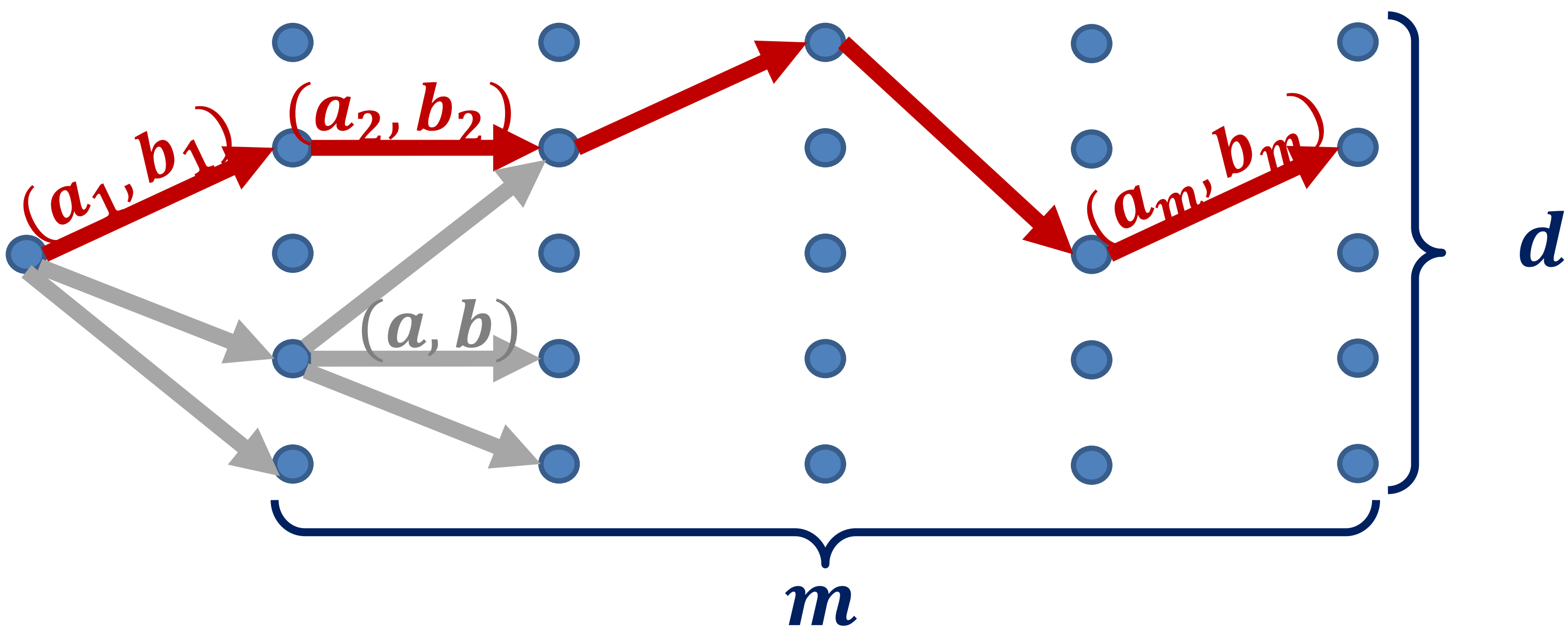
In the **ABP**, all vertices will be labeled with at most k equations. Once we reach a vertex with k equations in the **ABP** we declare **success**.

Proof Highlights – Simulation Part

Layer by layer, we convert the **BP** to an **ABP**.

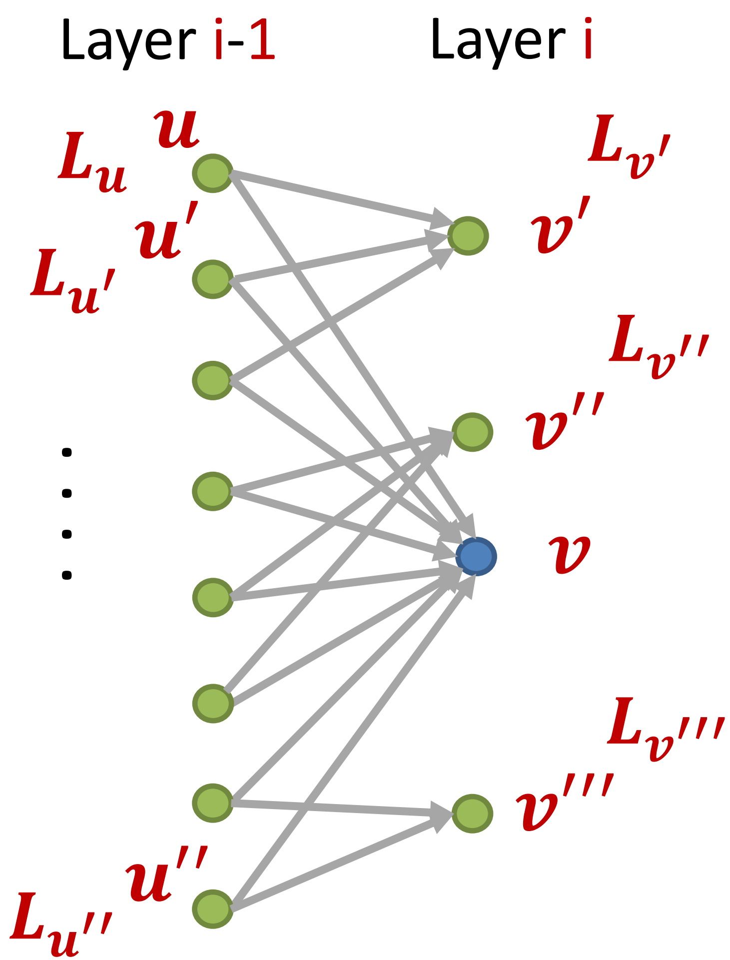
For $i = 1, \ldots, m$, we convert the i-th layer of the program.

Every vertex v in the i-th layer is **split** into many vertices by **regrouping** the edges entering v.



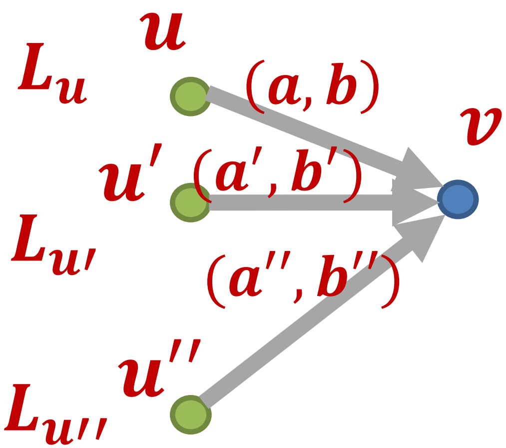
Regrouping

We **partition** the edges going into v to (not too many) groups, and associate with each group a set of accurate equations.



Main Lemma

Each edge $e = (u, v)$ going into v "remembers" a set of equations $L_e := L_u \cup \{(a_e, b_e)\}$



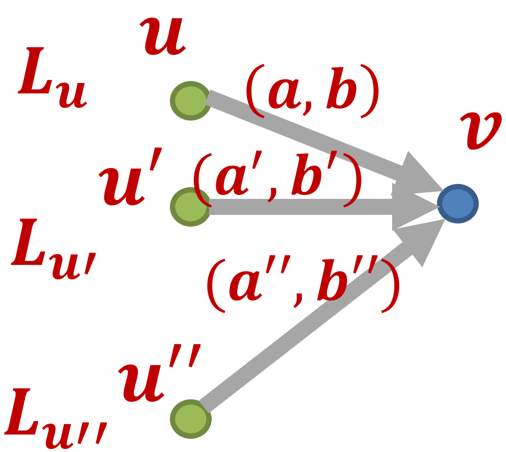
Main Lemma

Either:

1. There exists an equation $\langle a, x \rangle = b$ that is shared by many of the edges.

2. $\boldsymbol{P_{x|v}}$ is close to uniform (over **all** ℓ-sparse vectors).

Regrouping from Main Lemma



Main Lemma: Either

1. There exists an equation $\langle a, x\rangle = b$ that is shared by many of the edges.
2. $P_{x|v}$ is close to uniform (over **all** ℓ-sparse vectors).

Applying the main lemma recursively $k' \leq k$ times,
we find a large fraction of the edges with common eqs

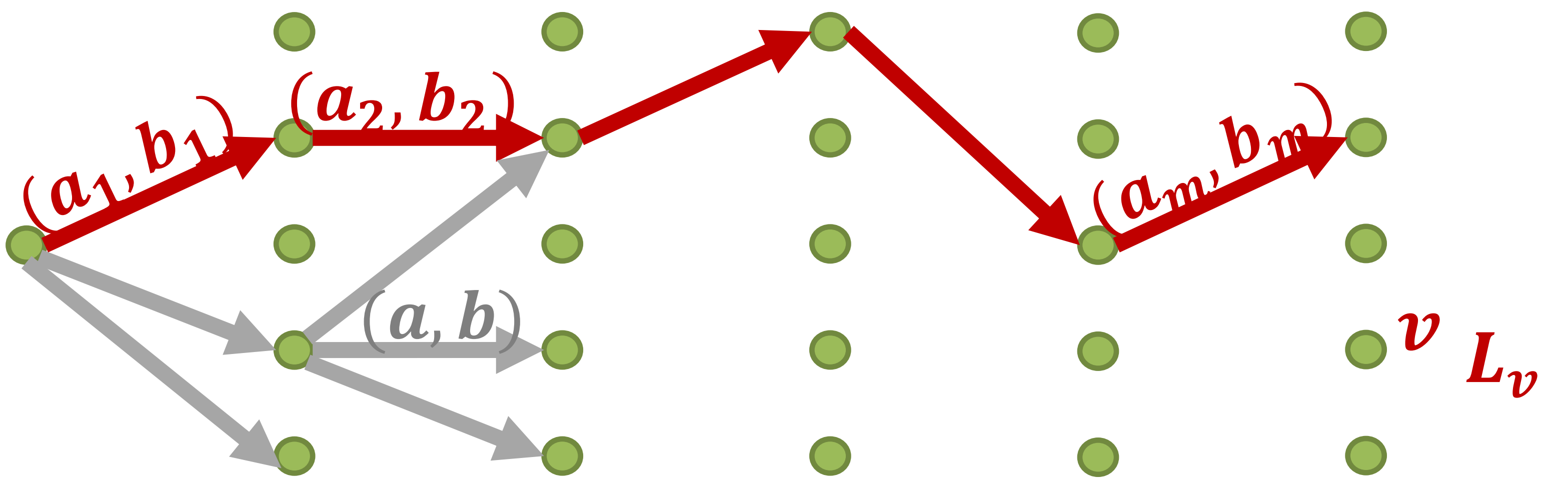
$$\langle a_1, x\rangle = b_1, \dots, \langle a_{k'}, x\rangle = b_{k'}$$

s.t. conditioned on passing through one of these edges,
x is close to uniform over all (ℓ-sparse) solutions to the eqs.

Proof on White Board

Lower Bounds on the Affine BP

Recall: all subspaces in the **Affine BP** are defined by at most k equations. **Success** = learned k equations.



Fix a node v in the **Affine BP** with k linearly independent eqs.

[Raz'16]: prob. of reaching v is at most $m^k \cdot 2^{-k(n-2k)}$

➔To succeed whp, the width should be $\Omega\left(2^{k(n-2k)}/m^{k+1}\right)$.

Proof on White Board

Conclusion – First Part

Main Theorem: Learning $\log(n)$-sparse parities requires either $\Omega(n \cdot \log^{0.99} n)$ memory bits or $n^{\Omega(\log\log n)}$ number of samples.

Implies same bounds for learning

- $O(n)$ size DNF formula
- $O(n)$ size Decision trees
- Juntas on $\log(n)$ variables

Open: proving tight samples-memory hardness for learning **DNFs**, **Decision Trees**, or **Juntas**

Lower Bounds more Generally

Q: Can we generalize the lower bounds to hold for problems not involving parities?

[Raz'17, Moshkovitz-Moshkovitz'17, Moshkovitz-Moshkovitz'18]: Yes

A new and general proof technique
(we shall focus on Raz's proof technique)

As a special case: a new proof for the memory-samples lower bound for parity learning.

[Garg-Raz-T'18, Beame-Oveis Gharan-Yang'18]:

Further generalizations of the method & more applications

A Learning Problem as a Matrix

A, X : finite sets

X : concept class

A : possible samples

$M: A \times X \rightarrow \{-1,1\}$: a matrix

$x \in_R X$ is chosen uniformly at random

A learner tries to learn x from a stream

$(a_1, b_1), (a_2, b_2) \ldots$, where $\forall t$:

$a_t \in_R A$ and $b_t = M(a_t, x)$

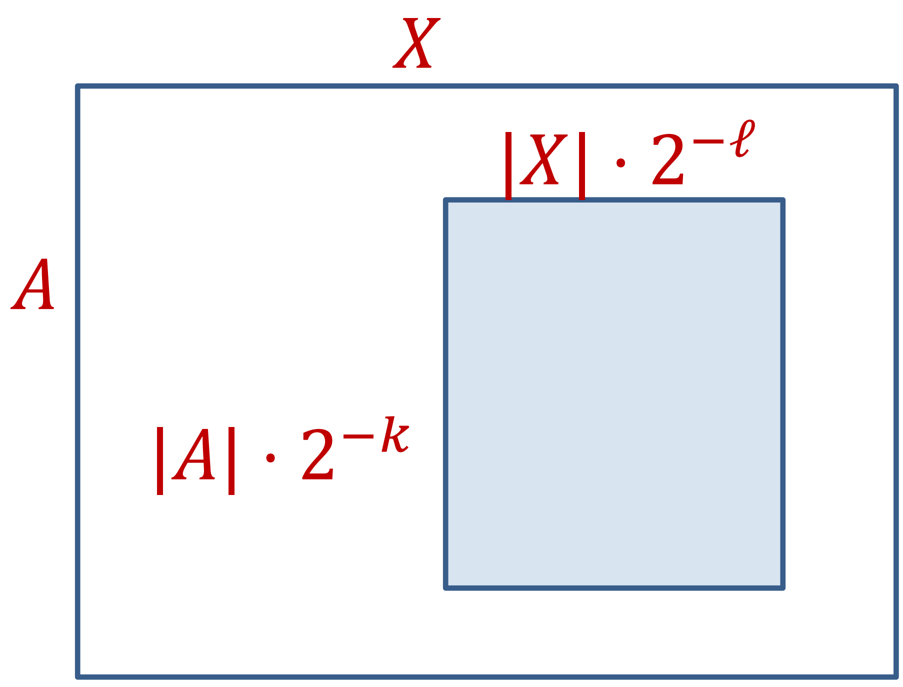
Extractor-Based Lower Bounds for Learning

Thm [Garg-Raz-T'18] Assume that any submatrix of M of fraction $2^{-k} \times 2^{-\ell}$ has bias of at most 2^{-r}.

Then, any learning algorithm for the learning problem defined by M requires either:

$\Omega(k \cdot \ell)$ **memory bits**,

or $2^{\Omega(r)}$ **samples**.



Independently, **[Beame-Oveis Gharan-Yang'18]** got a similar result

Applications of Extractor-Based Theorem

- **Learning Parities**
- **Learning Sparse Parities** and implications
- **Learning from low-degree equations**: A learner tries to learn $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, from random polynomial equations of degree at most d , over F_2.

  $\Omega(n^{d+1})$ memory or $2^{\Omega(n)}$ samples

- **Learning low-degree polynomials:** A learner tries to learn an n-variate multilinear polynomial p of degree at most d over F_2, from random evaluations of p over F_2^n.

  $\Omega(n^{d+1})$ memory or $2^{\Omega(n)}$ samples

and more…

Technique to Prove Extractor Property

$M: A\times X \rightarrow \{-1,1\}$: the learning matrix

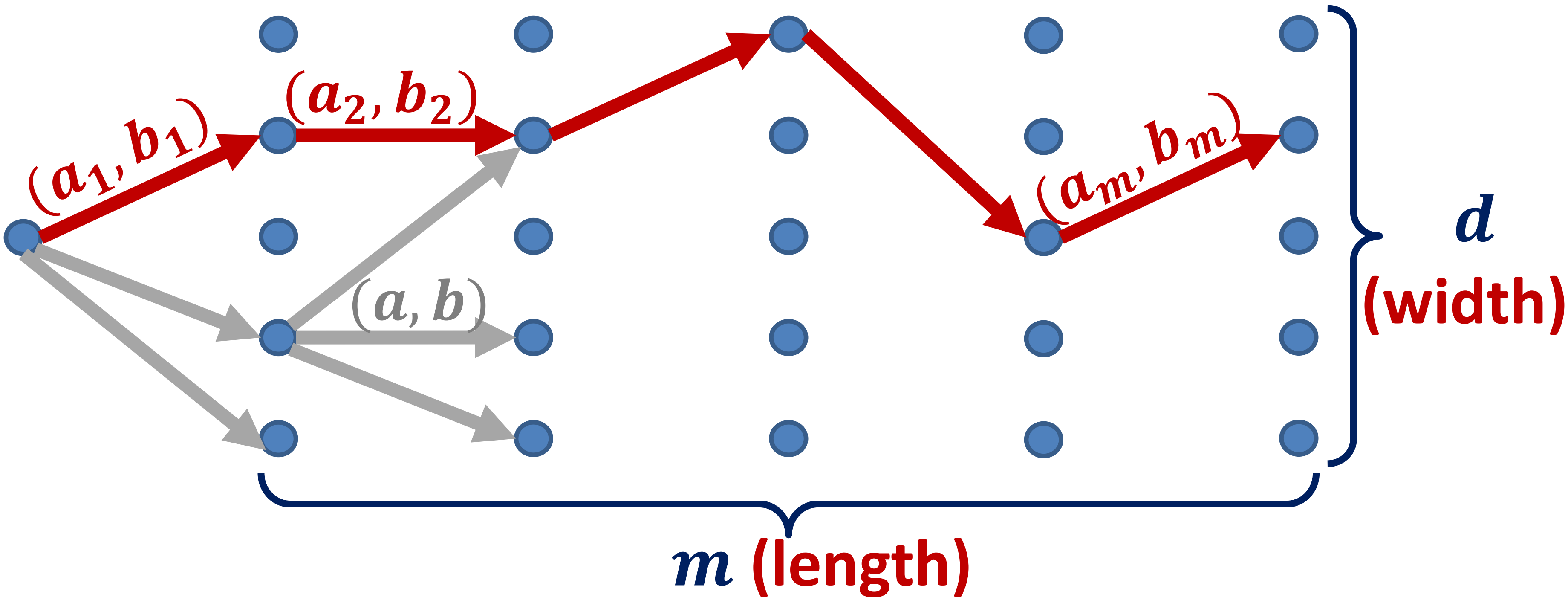
Def'n: We say that the columns of M are (ϵ, δ)-almost orthogonal if for each column x, at most $\delta \cdot |X|$ of the columns $x' \in X$ have $|\langle M_x, M_{x'} \rangle| \geq \epsilon \cdot |A|$.

Claim: Suppose the columns of M are (ϵ, δ)-almost orthogonal, for $\delta \leq \epsilon$. Then, learning requires either

$\Omega\left(\log\left(\frac{1}{\epsilon}\right) \cdot \log\left(\frac{1}{\delta}\right)\right)$ **memory bits**

or $\mathrm{poly}\left(\frac{1}{\epsilon}\right)$ **samples**

Recall: The Branching Program Model



Each layer represents a time step. Each vertex represents a memory state of the learner.

Each non-leaf vertex has $2 \cdot |A|$ outgoing edges, one for each $(a, b) \in |A| \times \{-1,1\}$.

Proof Overview

$|X| = 2^n$

$\boldsymbol{P}_{x|v}$ = the distribution of x conditioned on reaching a specific vertex v.

Significant vertices: v s.t. $||\mathrm{P}_{x|v}||_2^2 \geq 2^{\ell} \cdot 2^{-n}$

$\Pr(v)$ = probability that the path reaches v.

We prove: If v is significant, $\Pr(v) \leq 2^{-\Omega(k \cdot \ell)}$

Hence, there are at least $2^{\Omega(k \cdot \ell)}$ significant vertices.

T = same as the computational path, but stops when "atypical" things happen (stopping rules)

$\Pr(T\ stops)$ is exp small

Proof Overview

If v is significant, $\Pr(v) \leq 2^{-\Omega(k\cdot\ell)}$

Progress Function: For layer i,

$$Z_i = \mathbf{E}_{V_i}\left[\langle \mathrm{P}_{x|V_i}, \mathrm{P}_{x|v}\rangle^k\right]$$

1) $Z_0 = 2^{-nk}$
2) Z_i is very slowly growing: $Z_0 \approx Z_m$
(as long as number of steps is at most 2^r)
3) If $v \in L_m$, then $Z_m \geq \Pr(v) \cdot 2^{k\ell} \cdot 2^{-nk}$

Hence: If v is significant, $\Pr(v) \leq 2^{-\Omega(k\ell)}$

Open Problems

- Optimal tradeoffs for **DNFs**, **Juntas**, **Decision Trees**.
- What are the limits of the **Extractor-Based** lower bounds for these problems?
- **Characterize** memory-samples complexity from properties of the learning matrix M.
- **Generalize** to **Real-Valued Domains**
- **Generalize** to **k-passes** (some progress)

Open Problem: Understanding Neural Nets

Expressiveness and Learnability are empirically different in Neural Nets.

Consider the following experiment:

- Generate **(input,output)** pairs from a depth-2 NN with a fixed structure & randomly chosen weights.
- Try to learn weights from **(input,output)** pairs using **stochastic gradient descent**.
- **This usually fails.**

Can this be explained by the low-memory of the learner?

Thank You!