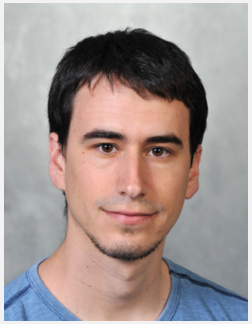
Extractor-Based Time-Space Lower Bounds for Learning

Sumegha Garg (Princeton)

Joint work with Ran Raz and Avishay Tal



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

Initiated a study of memory-samples lower bounds for learning

Can one prove unconditional lower bounds on the number of samples needed for learning, under memory constraints?

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

Initiated a study of memory-samples lower bounds for learning

Can one prove unconditional lower bounds on the number of samples needed for learning, under memory constraints?

(when the samples are viewed one by one)

(also known as online learning)

Example: Parity Learning

$x \in_R \{0,1\}^n$ is unknown

A learner tries to learn x from a stream

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

$a_{\mathrm{t}} \in_R \{0,1\}^n$ and

$b_t = a_t \cdot x$ (inner product mod 2)

Example: Parity Learning

$x \in_R \{0,1\}^n$ is unknown

A learner tries to learn x from a stream

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

$a_t \in_R \{0,1\}^n$ and

$b_t = a_t \cdot x$ (inner product mod 2)

In other words:

We get random linear equations in $x_1, \ldots, x_n$, one by one, and need to solve them

(no noise)

Example: Parity Learning

$x \in_R \{0,1\}^n$ is unknown

A learner tries to learn x from a stream

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

$a_t \in_R \{0,1\}^n$ and

$b_t = a_t \cdot x$ (inner product mod 2)

By solving linear equations:

$O(n)$ samples, $O(n^2)$ memory bits

By trying all possibilities:

$O(n)$ memory bits, exponential number of samples

Raz’s Breakthrough [2016]

Any algorithm for parity learning requires either $\Omega(n^2)$ memory bits or an exponential number of samples

Conjectured by:

Steinhardt, Valiant and Wager [2015]

Raz's Breakthrough [2016]

Any algorithm for parity learning requires either $\Omega(n^2)$ memory bits or an exponential number of samples

Conjectured by:

Steinhardt, Valiant and Wager [2015]

[Kol-Raz-Tal 2017]:

Any algorithm for learning sparse parities (hence also: DNF, CNF, decision tress, Juntas) requires either super-linear memory size or a super-polynomial number of samples

[Raz 2017]

For a large class of learning problems, any learning algorithm requires either quadratic memory size or an exponential number of samples

A new and general proof technique

[Raz 2017]

For a large class of learning problems, any learning algorithm requires either quadratic memory size or an exponential number of samples

A new and general proof technique

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

Our result uses a similar proof technique

Other Related Work

Independently, [Moshkovitz,Moshkovitz 2017a]: Any algorithm requires either $\sim 1.25n$ memory bits or an exponential number of samples for large class of learning problems

Subsequently, [Moshkovitz,Moshkovitz 2017b]: Similar results as [Raz 2017]

Other Related Work

Independently, [Moshkovitz,Moshkovitz 2017a]: Any algorithm requires either $\sim 1.25n$ memory bits or an exponential number of samples for large class of learning problems

Subsequently, [Moshkovitz,Moshkovitz 2017b]: Similar results as [Raz 2017]

Independently of our result, [Beame,Oveis-Gharan,Yang 2018] proved related lower bounds

Motivation

Learning Theory: [S 14, SVW 15,…] In some cases, learning is infeasible, due to memory constraints

Motivation

Learning Theory: [S 14, SVW 15,…] In some cases, learning is infeasible, due to memory constraints

Cryptography: [R 16, VV 16, KRT 16] Bounded Storage Crypto -

Key's length: n Encryption/Decryption time: n

Unconditional security, if the attacker's memory size is at most $\frac{n^2}{25}$

Motivation

Learning Theory: [S 14, SVW 15,…] In some cases, learning is infeasible, due to memory constraints

Cryptography: [R 16, VV 16, KRT 16] Bounded Storage Crypto -

Key's length: n Encryption/Decryption time: n

Unconditional security, if the attacker's memory size is at most $\frac{n^2}{25}$

Complexity Theory: Different Time-Space Tradeoffs have been studied in many models [BJS 98, Ajt 99, BSSV 00, For 97, FLvMV 05, Wil 06,…]

A Learning Problem as a Matrix

A, X : finite sets

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

A Learning Problem as a Matrix

A, X : finite sets

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

$x \in_R X$ is unknown. 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

$b_t = M(a_t, x)$

A Learning Problem as a Matrix

A, X : finite sets

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

$x \in_R X$ is unknown. 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

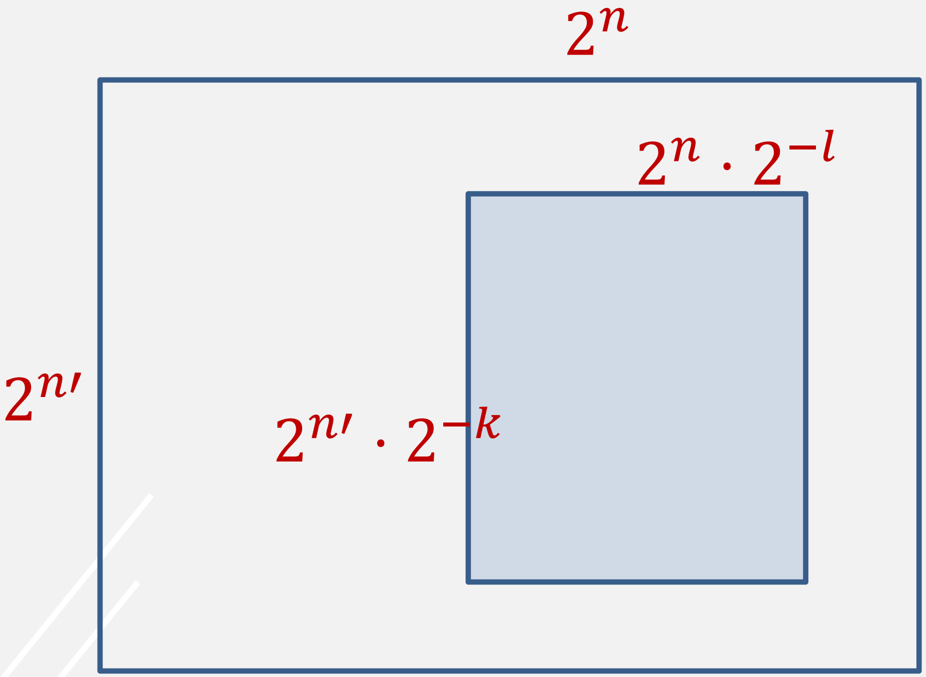
$b_t = M(a_t, x)$

X : concept class $= \{0,1\}^n$

A : possible samples $= \{0,1\}^{n'}$

Our Result

Assume that any submatrix of M of fraction $2^{-k}\times 2^{-\ell}$ has bias of at most 2^{-r}. Then:



Our Result

Assume that any submatrix of M of fraction $2^{-k} \times 2^{-\ell}$ has bias of at most 2^{-r}. Then:

Any algorithm requires either $\Omega(k \cdot \ell)$ memory bits or $2^{\Omega(r)}$ samples

(Implies all previous results)

Our Result

Assume that any submatrix of M of fraction $2^{-k} \times 2^{-\ell}$ has bias of at most 2^{-r}. Then:

Any algorithm requires either $\Omega(k \cdot \ell)$ memory bits or $2^{\Omega(r)}$ samples

(Implies all previous results)

[R 17] looked only at largest singular value of M

An independent related result by [Beame,Oveis-Gharan,Yang 2018]

Our Result and [Beame,Oveis-Gharan,Yang 2018]

For large classes of learning problems, any learning algorithm requires either memory of size $\Omega((\log|A|) \cdot (\log|X|))$ or an exponential number of samples

[R 17]: bound on memory of at most $\min\left((\log|A|)^2, (\log|X|)^2\right)$

Applications

Parity Learning: A learner tries to learn $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, from random linear equations over F_2.

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

Sparse Parities: A learner tries to learn $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ of sparsity l, from random linear equations over F_2.

$\Omega(n \cdot l)$ memory or $2^{\Omega(l)}$ samples

Applications

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

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

Applications

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

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

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

Applications

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

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

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..

Techniques to Prove Extractor Property

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

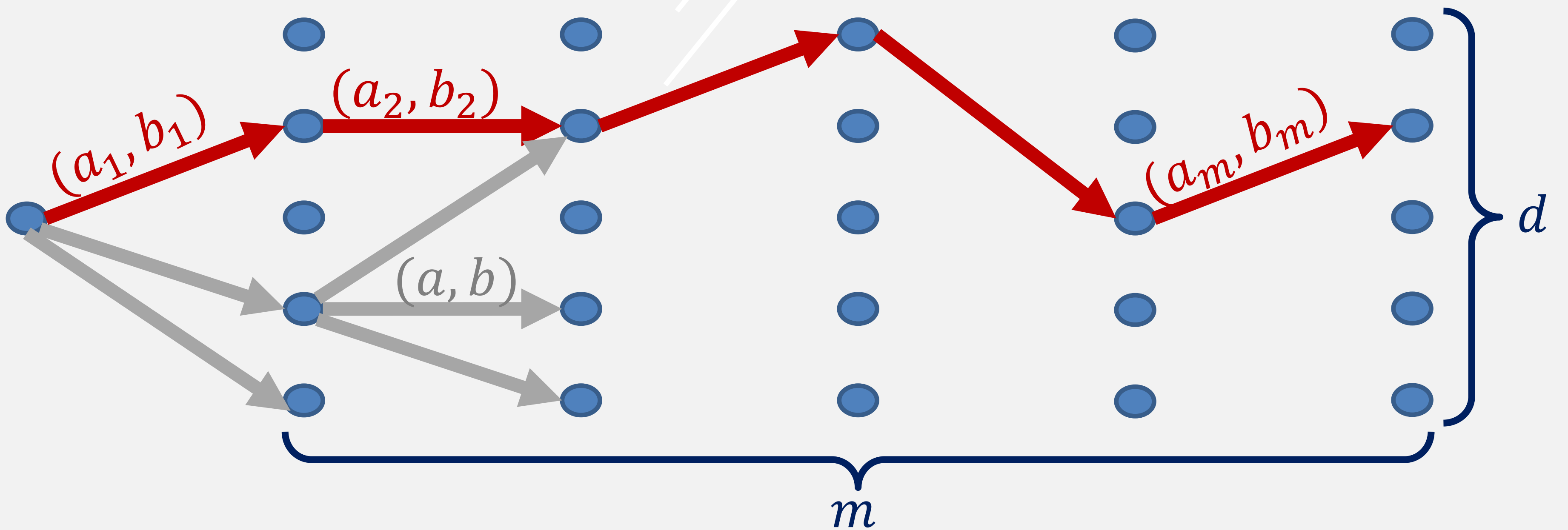
M_a are (ϵ, δ)-almost orthogonal

For each row a_i, at most δ fraction of the rows $a \in A$ have

$$| < M_a, M_{a_i} > | \geq \epsilon$$

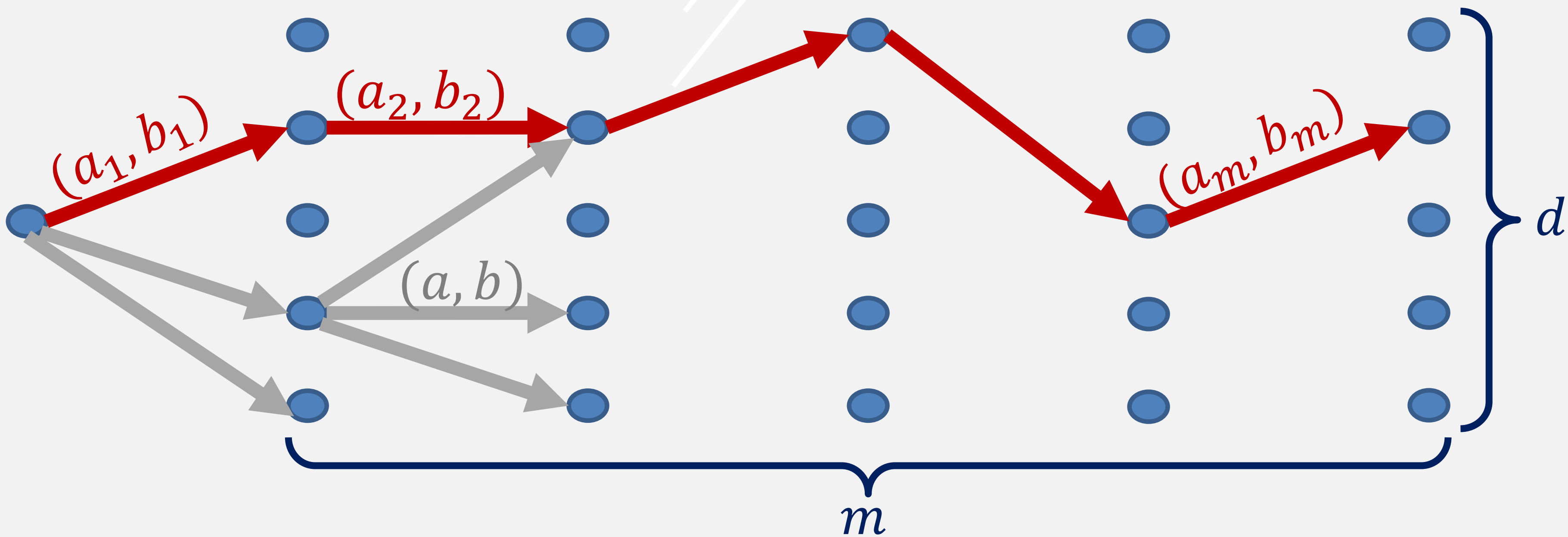
Then, learning requires either $\Omega(\log\frac{1}{\delta} \cdot \log(\min(\frac{1}{\epsilon}, \frac{1}{\delta})))$ memory or $\Omega(\min(\frac{1}{\epsilon}, \frac{1}{\delta}))$ samples

Branching Program (length m, width d)



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 $(a, b) \in \{0,1\}^{n'} \times \{-1,1\}$

Branching Program (length m, width d)



The samples $(a_1, b_1), \ldots, (a_m, b_m)$ define a computation-path. Each vertex v in the last layer is labeled by $\hat{x}_v \in \{0,1\}^n$. The output is the label $\hat{x}_v$ of the vertex reached by the path

Proof Outline

$\mathrm{P}_{x|v}$ = distribution of x conditioned on the event that the computation-path reaches v

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

Proof Outline

$\mathrm{P}_{x|v}$ = distribution of x conditioned on the event that the computation-path reaches v

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

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

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

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

Proof Outline

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

Progress Function: For layer L_i,

$$Z_i = \sum_{v \in L_i} Pr(v) \cdot \langle \mathrm{P}_{x|v}, \mathrm{P}_{x|s} \rangle^k$$

1) $Z_0 = 2^{-2n \cdot k}$

2) Z_i is very slowly growing: $Z_0 \approx Z_m$

3) If $s \in L_m$, then $Z_m \geq Pr(s) \cdot 2^{2l \cdot k} \cdot 2^{-2n \cdot k}$

Hence: If s is significant, $Pr(s) \leq 2^{-\Omega(k \cdot l)}$

Generalization to Non-Product Distributions

A, X : finite sets

$P: A \times X \to [0,1]$: a joint distribution

$x \in_R X$ is unknown

A learner tries to learn x from a stream

$a_1, a_2 \ldots$, where $\forall t$:

a_t is drawn randomly according to $P_{A|X=x}$

Generalization to Non-Product Distributions

A, X : finite sets

$P: A \times X \to [0,1]$: a joint distribution

$x \in_R X$ is unknown

A learner tries to learn x from a stream

$a_1, a_2 \ldots$, where $\forall t$:

a_t is drawn randomly according to $P_{A|X=x}$

For example: what if we get only positive samples, large output…

Generalization to Non-Product Distributions

A, X : finite sets

$P: A \times X \to [0,1]$: a joint distribution

$x \in_R X$ is unknown

A learner tries to learn x from a stream

$a_1, a_2 \ldots$, where $\forall t$:

a_t is drawn randomly according to $P_{A|X=x}$

$\widetilde{M}: A \times X \to \mathrm{R}: \widetilde{M}(a, x) = \frac{P_{A|X=x}(a)}{P_A(a)} - 1$

Generalization to Non-Product Distributions

$\widetilde{M}: A \times X \to \mathrm{R}: \widetilde{M}(a,x) = \frac{P_{A|X=x}(a)}{P_A(a)} - 1$

If $\max_{a,x} \widetilde{M}(a,x) \leq 2^p$ and assume that any submatrix of M of fractional weight $2^{-k} \times 2^{-\ell}$ has bias of at most 2^{-r}. Then:

Generalization to Non-Product Distributions

$\widetilde{M}: A \times X \to \mathrm{R}: \widetilde{M}(a,x) = \frac{P_{A|X=x}(a)}{P_A(a)} - 1$

If $\max_{a,x} \widetilde{M}(a,x) \leq 2^p$ and assume that any submatrix of M of fractional weight $2^{-k} \times 2^{-\ell}$ has bias of at most 2^{-r}. Then:

Any algorithm requires either $\Omega(\frac{k \cdot \ell}{p})$ memory bits or $2^{\Omega(r)}$ samples

Generalization to Non-Product Distributions

$\widetilde{M}: A \times X \to \mathrm{R}$: $\widetilde{M}(a,x) = \frac{P_{A|X=x}(a)}{P_A(a)} - 1$

If $\max_{a,x} \widetilde{M}(a,x) \le 2^p$ and assume that any submatrix of M of fractional weight $2^{-k} \times 2^{-\ell}$ has bias of at most 2^{-r}. Then:

Any algorithm requires either $\Omega(\frac{k \cdot \ell}{p})$ memory bits or $2^{\Omega(r)}$ samples

Application: For any finite field F, learning a string $x \in F^n$ from random linear equations, requires either a memory of size $\Omega(\mathrm{n}^2 \log(\mathrm{F}))$, or an exponential number of equations

Open Problems

Secret/Samples over Reals

Optimal tradeoffs for DNFs..

Read-k learning

Thank You ☺