# Learning Fast Requires Good Memory: Time-Space Tradeoff Lower Bounds for Learning

**Ran Raz**

**Princeton University**

Based on joint works with:

**Sumegha Garg, Gillat Kol, Avishay Tal**

[R16, KRT17, R17, GRT18]

## This Talk:

**A line of recent works studies time-space (memory-samples) lower bounds for learning**

[S14, SVW16, R16, VV16, KRT17, MM17, R17, MM18, BOGY18, GRT18, DS18, AS18, DKS19, SSV19, GRT19, GKR19]

## Main Message:

**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**

## Original Motivation: Online Learning Theory:

**Initiated by:**

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

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

(when each sample is viewed only once - also known as online learning)

## Example: Parity Learning:

$x = (x_1, \dots, x_n) \in_R \{0,1\}^n$ is unknown

A learner gets a stream of random linear equations (mod 2) in $x_1, \dots, x_n$, one by one, and tries to learn $x$

## Formally:

The learner gets a stream of samples: $(a_1, b_1), (a_2, b_2) \dots$ , where $\forall t$ :

$a_t \in_R \{0,1\}^n$ and $b_t = a_t \cdot x$ (inner product mod 2)

The learner needs to solve the equations and find $x$

(no noise)

# Ready to Play?

$x = (x_1, x_2, x_3, x_4, x_5)$ is unknown

# Ready to Play?

$x = (x_1, x_2, x_3, x_4, x_5)$ is unknown

$$a_1 = (1, 1, 0, 1, 1)\,, \qquad b_1 = 0$$

$$x_1 + x_2 + x_4 + x_5 = 0 \pmod 2$$

# Ready to Play?

$x = (x_1, x_2, x_3, x_4, x_5)$ **is unknown**

$$a_2 = (0, 1, 1, 0, 0), \qquad b_2 = 0$$

$$x_2 + x_3 = 0 \pmod 2$$

# Ready to Play?

$x = (x_1, x_2, x_3, x_4, x_5)$ **is unknown**

$$a_3 = (0, 0, 1, 1, 1)\,, \qquad b_3 = 0$$

$$x_3 + x_4 + x_5 = 0 \pmod 2$$

# Ready to Play?

$x = (x_1, x_2, x_3, x_4, x_5)$ **is unknown**

$$a_4 = (0, 1, 1, 1, 0)\,, \qquad b_4 = 0$$

$$x_2 + x_3 + x_4 = 0 \pmod 2$$

# Ready to Play?

$x = (x_1, x_2, x_3, x_4, x_5)$ **is unknown**

$$a_5 = (1, 1, 0, 0, 1)\,, \qquad b_5 = 0$$

$$x_1 + x_2 + x_5 = 0 \pmod 2$$

# Ready to Play?

$x = (x_1, x_2, x_3, x_4, x_5)$ **is unknown**

$$a_6 = (0, 0, 1, 1, 0)\,, \qquad b_6 = 1$$

$$x_3 + x_4 = 1 \pmod 2$$

# Ready to Play?

$x = (x_1, x_2, x_3, x_4, x_5)$ **is unknown**

$$a_7 = (0, 1, 0, 1, 1)\,, \qquad b_7 = 0$$

$$x_2 + x_4 + x_5 = 0 \pmod 2$$

# Ready to Play?

$x = (x_1, x_2, x_3, x_4, x_5)$ **is unknown**

$$a_8 = (1, 0, 0, 0, 1)\,, \qquad b_8 = 1$$

$$x_1 + x_5 = 1 \pmod 2$$

# Ready to Play?

$x = (x_1, x_2, x_3, x_4, x_5)$ **is unknown**

$$a_9 = (1, 1, 1, 1, 0)\,, \qquad b_9 = 0$$

$$x_1 + x_2 + x_3 + x_4 = 0 \pmod 2$$

# Ready to Play?

$x = (x_1, x_2, x_3, x_4, x_5)$ **is unknown**

$$a_{10} = (0, 1, 1, 1, 1)\,, \qquad b_{10} = 1$$

$$x_2 + x_3 + x_4 + x_5 = 1 \pmod 2$$

# Ready to Play?

$x = (x_1, x_2, x_3, x_4, x_5)$ **is unknown**

$$a_{11} = (0, 0, 0, 0, 0)\,, \qquad b_{11} = 0$$

$$0 = 0 \pmod 2$$

**Parity Learning:** $x \in_R \{0,1\}^n$ **is unknown**

**A learner gets a stream of samples:** $(a_1, b_1), (a_2, b_2) \ldots$ **,**

**where** $\forall t$ **:** $a_t \in_R \{0,1\}^n$ **and** $b_t = a_t \cdot x$ **(mod 2)**

**and needs to solve the equations and find** $x$

**Parity Learning:** $x \in_R \{0,1\}^n$ **is unknown**

**A learner gets a stream of samples:** $(a_1, b_1), (a_2, b_2) \ldots$ **,**
**where** $\forall t$ **:** $a_t \in_R \{0,1\}^n$ **and** $b_t = a_t \cdot x$ **(mod 2)**
**and needs to solve the equations and find** $x$

**By solving linear equations:**

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

**Parity Learning:** $x \in_R \{0,1\}^n$ **is unknown**

**A learner gets a stream of samples:** $(a_1, b_1), (a_2, b_2) \ldots$ **,**
**where** $\forall t$ **:** $a_t \in_R \{0,1\}^n$ **and** $b_t = a_t \cdot x$ **(mod 2)**
**and needs to solve the equations and find** $x$

**By solving linear equations:**

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

**By trying all possibilities:**

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

**Parity Learning:** $x \in_R \{0,1\}^n$ **is unknown**

**A learner gets a stream of samples:** $(a_1, b_1), (a_2, b_2) \ldots$ **,**

**where** $\forall t$ **:** $a_t \in_R \{0,1\}^n$ **and** $b_t = a_t \cdot x$ **(mod 2)**

**and needs to solve the equations and find** $x$

**Parity Learning:** $x \in_R \{0,1\}^n$ **is unknown**

**A learner gets a stream of samples:** $(a_1, b_1), (a_2, b_2) \ldots$ **,**
**where** $\forall t$ **:** $a_t \in_R \{0,1\}^n$ **and** $b_t = a_t \cdot x$ **(mod 2)**
**and needs to solve the equations and find** $x$

**[R 2016]: Any algorithm for parity learning requires either** $\frac{n^2}{10}$
**memory bits or an exponential number of samples**
**(Conjectured by Steinhardt, Valiant and Wager [2015])**

**Parity Learning:** $x \in_R \{0,1\}^n$ **is unknown**

**A learner gets a stream of samples:** $(a_1, b_1), (a_2, b_2) \ldots$ **,**
**where** $\forall t$ **:** $a_t \in_R \{0,1\}^n$ **and** $b_t = a_t \cdot x$ **(mod 2)**
**and needs to solve the equations and find** $x$

**[R 2016]: Any algorithm for parity learning requires either** $\frac{n^2}{10}$
**memory bits or an exponential number of samples**
**(Conjectured by Steinhardt, Valiant and Wager [2015])**
**Previously: no lower bound on the number of samples, even if the memory size is** $n$ **(for any learning problem)**

(for memory of size $< n$, relatively easy to prove lower bounds, since inner product is a good two-source extractor)

**Parity Learning:** $x \in_R \{0,1\}^n$ **is unknown**

**A learner gets a stream of samples:** $(a_1, b_1), (a_2, b_2) \ldots$ **,**
**where** $\forall t$ **:** $a_t \in_R \{0,1\}^n$ **and** $b_t = a_t \cdot x$ **(mod 2)**
**and needs to solve the equations and find** $x$

**[R 2016]: Any algorithm for parity learning requires either** $\frac{n^2}{10}$
**memory bits or an exponential number of samples**
**(Conjectured by Steinhardt, Valiant and Wager [2015])**
**Previously: no lower bound on the number of samples, even if the memory size is** $n$ **(for any learning problem)**

I will focus on super-linear lower bounds on the memory size

**Parity Learning:** $x \in_R \{0,1\}^n$ **is unknown**

**A learner gets a stream of samples:** $(a_1, b_1), (a_2, b_2) \ldots$ **,**
**where** $\forall t$ **:** $a_t \in_R \{0,1\}^n$ **and** $b_t = a_t \cdot x$ **(mod 2)**
**and needs to solve the equations and find** $x$

**[R 2016]: Any algorithm for parity learning requires either** $\frac{n^2}{10}$
**memory bits or an exponential number of samples**
**(Conjectured by Steinhardt, Valiant and Wager [2015])**

**Parity Learning:** $x \in_R \{0,1\}^n$ **is unknown**

**A learner gets a stream of samples:** $(a_1, b_1), (a_2, b_2) \ldots$ **,**
**where** $\forall t$ **:** $a_t \in_R \{0,1\}^n$ **and** $b_t = a_t \cdot x$ **(mod 2)**
**and needs to solve the equations and find** $x$

**[R 2016]: Any algorithm for parity learning requires either** $\frac{n^2}{10}$
**memory bits or an exponential number of samples**
**(Conjectured by Steinhardt, Valiant and Wager [2015])**

**Best upper bound on the memory size :** $\approx \frac{n^2}{4}$

(when the number of samples is sub-exponential)

# Motivation: Machine Learning Theory:

**For some online learning problems, access to a relatively large memory is crucial**

**In some cases, learning is infeasible, due to memory constraints (if each sample is viewed only once)**

## Motivation: Machine Learning Theory:

For some online learning problems, access to a relatively large memory is crucial

In some cases, learning is infeasible, due to memory constraints (if each sample is viewed only once)

Very interesting to understand how much memory is needed for learning

Our result gives a concept class that can be efficiently learnt if and only if the learner has a quadratic-size memory

## Motivation: Machine Learning Theory:

For some online learning problems, access to a relatively large memory is crucial

In some cases, learning is infeasible, due to memory constraints (if each sample is viewed only once)

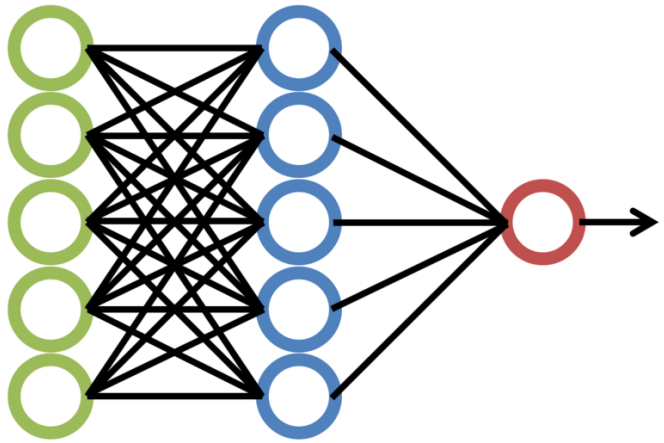
Very interesting to understand how much memory is needed for learning

Our result gives a concept class that can be efficiently learnt if and only if the learner has a quadratic-size memory

“Good” memory may be crucial in learning processes

# Example: Neural Networks

**Many learning algorithms try to learn a concept by modeling it as a neural network. The algorithm keeps in the memory some neural network and updates the weights when new samples arrive. The memory used is the size of the network**

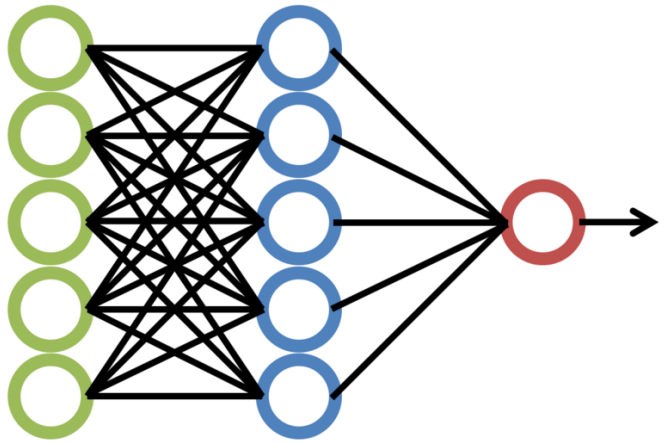


# Example: Neural Networks

**Many learning algorithms try to learn a concept by modeling it as a neural network. The algorithm keeps in the memory some neural network and updates the weights when new samples arrive. The memory used is the size of the network**

**Conclusion: such algorithms cannot learn certain concept classes, if each sample is viewed only once and the size of the neural network is not sufficiently large**

**For example, for learning parities, the memory size must be quadratic**



# Motivation: Bounded Storage Cryptography [Maurer 92]:

## [Maurer, CM, AR, ADR, Vadhan, DM,…]

**In the bounded storage model. Alice and Bob want to interact securely, in the presence of an adversary with bounded memory size**

Alice → Bob

## Applications to Bounded Storage Cryptography:

**[R16]:**

Secret Key Encryption/Decryption protocol

**[Guan-Zhandry 2019]:**

Key Agreement, Bit Commitment, Oblivious Transfer

**(all these protocols have advantages over previous works)**

## Motivation: Complexity Theory:

**Time-Space Lower Bounds for computing a function $f(x_1, .., x_n)$ have been studied for a long time, in various models [BJS 98, Ajt 99, BSSV 00, For 97, FLvMV 05, Wil 06,…]**

## Motivation: Complexity Theory:

Time-Space Lower Bounds for computing a function $f(x_1, .., x_n)$ have been studied for a long time, in various models [BJS 98, Ajt 99, BSSV 00, For 97, FLvMV 05, Wil 06,…]

These results proved lower bounds of at most $n^{1+\delta}$ on the time needed, under space constraints

How come we prove exponential bounds on the time, under space constraints?

## Motivation: Complexity Theory:

Time-Space Lower Bounds for computing a function $f(x_1, .., x_n)$ have been studied for a long time, in various models [BJS 98, Ajt 99, BSSV 00, For 97, FLvMV 05, Wil 06,…]

These results proved lower bounds of at most $n^{1+\delta}$ on the time needed, under space constraints

How come we prove exponential bounds on the time, under space constraints? The models are different

## Motivation: Complexity Theory:

Time-Space Lower Bounds for computing a function $f(x_1, .., x_n)$ have been studied for a long time, in various models [BJS 98, Ajt 99, BSSV 00, For 97, FLvMV 05, Wil 06,…]

These results proved lower bounds of at most $n^{1+\delta}$ on the time needed, under space constraints

How come we prove exponential bounds on the time, under space constraints? The models are different

Lower bounds for computing $f(x_1, .., x_n)$ assume that $x_1, .., x_n$ can always be accessed (The inputs are stored for free)

In our case, after the learner saw $(a_t, b_t)$, the sample $(a_t, b_t)$ cannot be accessed again (unless stored in the memory)

## Motivation: Pseudo Randomness:

**Alternative way to construct pseudorandom generators for log-space, using $\mathrm{polylog}(n)$ truly-random bits**

**(doesn't match the best known generators that use only $\mathrm{O}(\log^2(n))$ bits [Nisan 90, INW 94, GR 14])**

**[Kol-R-Tal 2017]:**

**Learning sparse parities requires either super-linear memory size or a super-polynomial number of samples**

**Sparsity-$k$ parity learning: Same as parity learning but it is known in advance that at most $k$ of the variables $x_1, \dots, x_n$ are 1**

## [Kol-R-Tal 2017]:

**Learning sparse parities requires either super-linear memory size or a super-polynomial number of samples**

## [Kol-R-Tal 2017]:

**Learning sparse parities requires either super-linear memory size or a super-polynomial number of samples**

**For sparsity $k < n^{0.99}$: Any algorithm requires**

**1) $\Omega(n \cdot k)$ memory bits or $2^{\Omega(k)}$ samples**

**2) $\Omega(n \cdot k^{0.99})$ memory bits or $k^{\Omega(k)}$ samples**

**[Kol-R-Tal 2017]:**

**Learning sparse parities requires either super-linear memory size or a super-polynomial number of samples**

**For sparsity $k < n^{0.99}$: Any algorithm requires**

**1) $\Omega(n \cdot k)$ memory bits or $2^{\Omega(k)}$ samples**

**2) $\Omega(n \cdot k^{0.99})$ memory bits or $k^{\Omega(k)}$ samples**

**Conclusion: Learning $\log(n)$-sparse parities, linear-size DNFs, linear-size CNFs, linear-size decision trees, $\log(n)$-size Juntas requires super-linear memory size or a super-polynomial number of samples**

**[R 17, MM 18, BOGY 18, GRT 18]:**

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

**[BOGY 18, GRT 18] build on [R 17]**

**[MM 18] builds on an earlier paper [MM 17] that obtained a similar result but with a linear lower-bound of $1.25 \cdot n$ on the memory size**

## [R 17, MM 18, BOGY 18, GRT 18]:

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

[R 17, MM 18, BOGY 18, GRT 18]:

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

I will focus on [R 17, GRT 18]

Additional follow-up works (building on [R 17]):

Memory-Sample lower bounds for:

linear regression with small error [SSV 19]

two-pass learning [GRT 19]

## 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 ( $= \{0, 1\}^n$ in parity learning)

$A$ : possible samples ( $= \{0, 1\}^n$ in parity learning)

## Theorem [R 17], [Garg-R-Tal 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 requires either $\Omega(k \cdot \ell)$ memory bits or $2^{\Omega(r)}$ samples**

**In particular, 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**

(A new general proof technique. Implies all previous results)

(A related result by Beame, Oveis-Gharan, Yang (building on [R 17]))

## Applications: (examples)

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

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

## Applications: (examples)

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

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

**Low degree polynomials:** A learner tries to learn an $n$-variate multilinear polynomial of degree $d$ over $F_2$, from random evaluations:

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

## Applications: (examples)

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

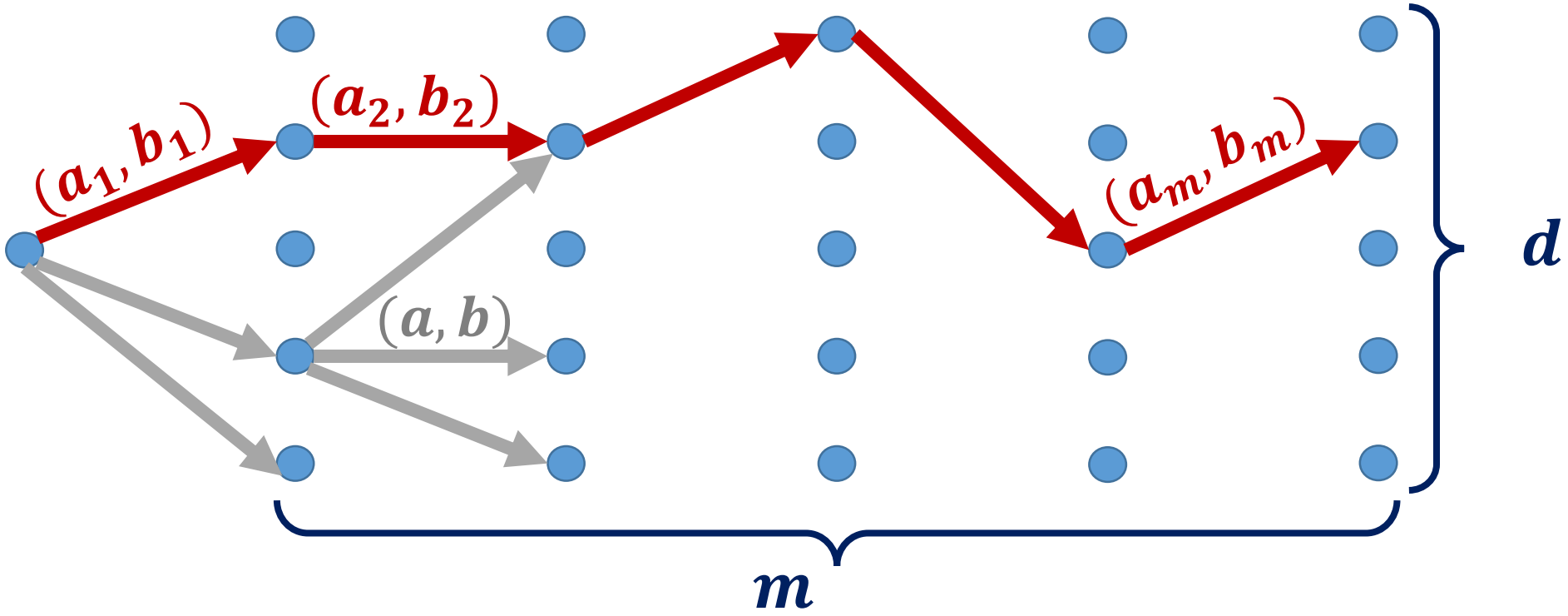
Requires $\Omega(n^{d+1})$ memory or $2^{\Omega(n)}$ samples

**Low degree polynomials:** A learner tries to learn an $n$-variate multilinear polynomial of degree $d$ over $F_2$, from random evaluations:

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

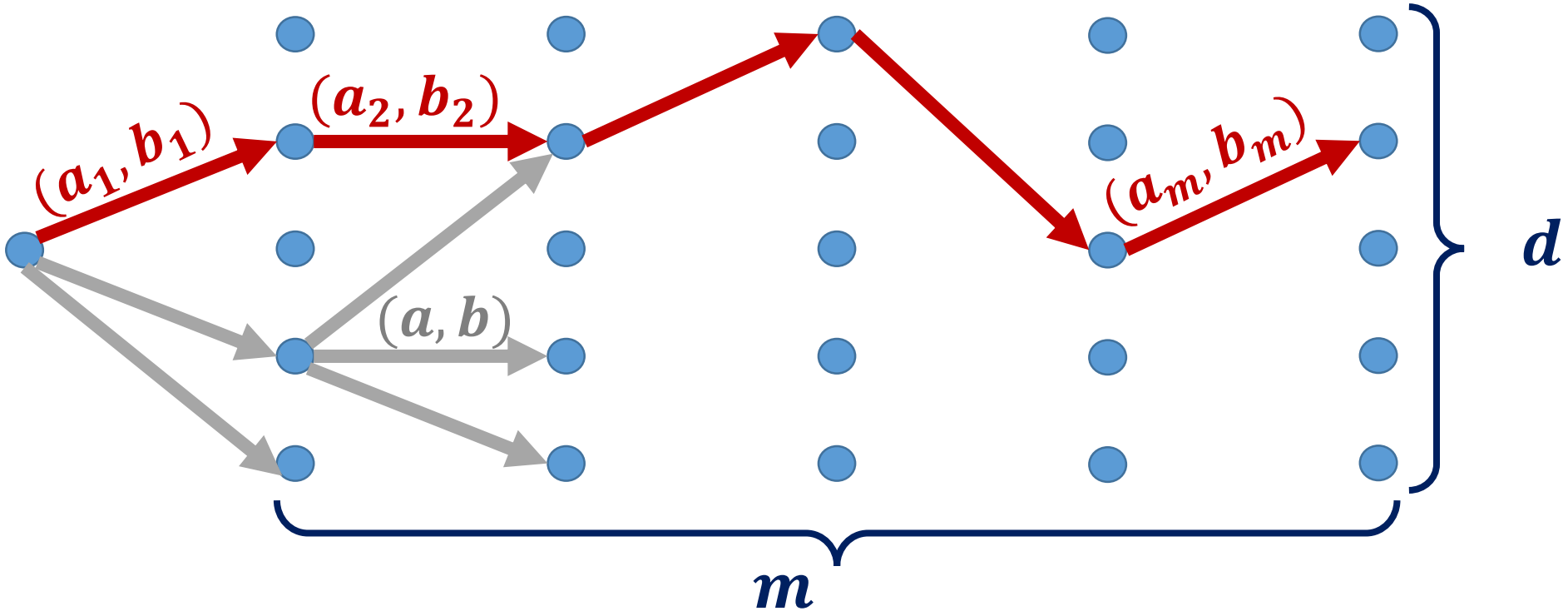
**Error correcting codes…** **Random matrices…**

# Branching Program (length $m$, width $d$): (for parity learning)



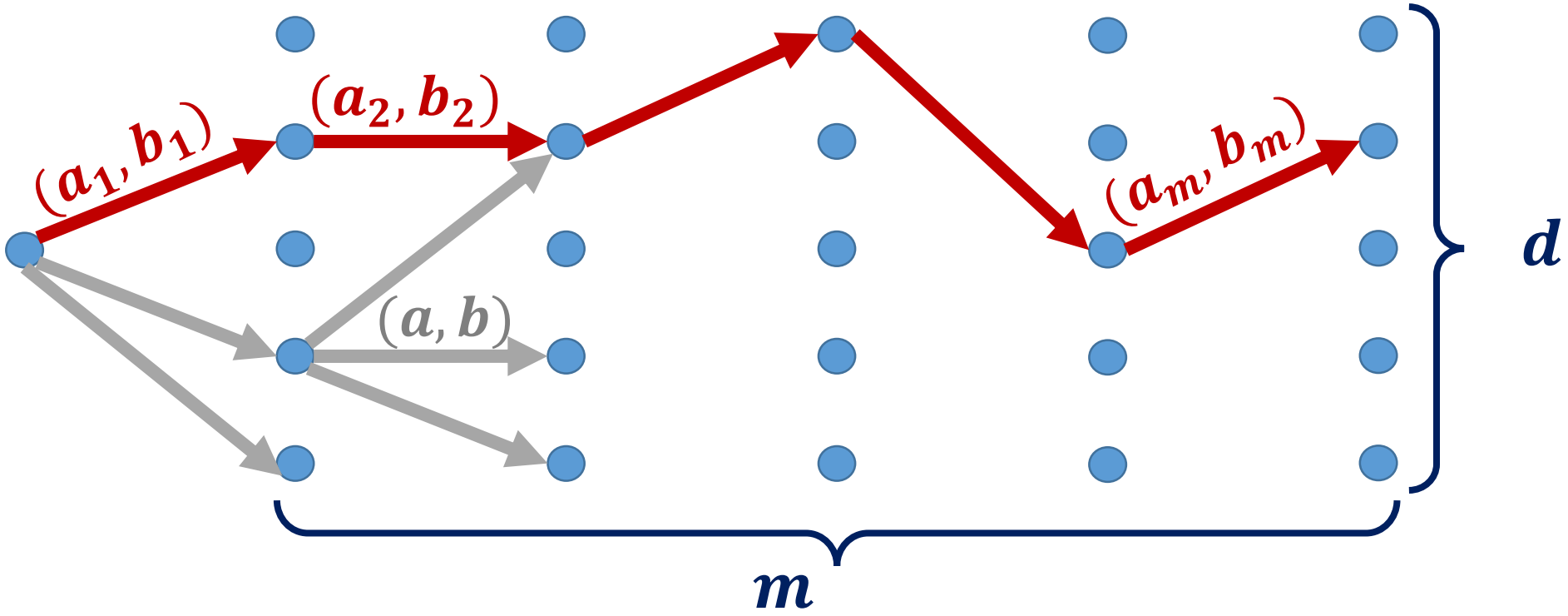

Each layer represents a time step. Each vertex represents a memory state of the learner. Each non-leaf vertex has $2^{n+1}$ outgoing edges, one for each $(a, b) \in \{0,1\}^n \times \{-1, 1\}$

# Branching Program (length $m$, width $d$): (for parity learning)




The samples $(a_1, b_1), .., (a_m, b_m)$ 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

# Branching Program (length $m$, width $d$): (for parity learning)




**Example: BP for Parity learning: Any BP with width $d \leq 2^{n^2/20}$ and length $m \leq 2^{n/100}$, outputs the correct $x$ with exponentially small probability**

# Proof Outline (for parity): [R17]

**same proof techniques was used in several follow-up works:**
[BOGY18, GRT18, SSV19, GRT19]

# Interesting Idea in the Proof: (very high level)

Significant vertices: $v$ s.t. conditioned on the event that the computation-path reaches $v$, $x$ can be guessed with non-negligible probability

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

We want to prove: If $v$ is significant, $Pr(v) \leq 2^{-\Omega(n^2)}$

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

$B$ = the event that some "atypical" things happen

We show that $Pr(B) \leq 2^{-\Omega(n)}$ (but much larger than $2^{-\Omega(n^2)}$)

We show that if $v$ is significant, $Pr(v|\overline{B}) \leq 2^{-\Omega(n^2)}$

## Proof Outline:

$T$ = same as the computation path, but stops when "atypical" things happen (stopping rules). All definitions are with respect to $T$

$Pr(T \text{ stops})$ is exp small (but much larger than $2^{-\Omega(n^2)}$)

$\mathbf{P}_{x|v}$ = distribution of $x$ conditioned on the event that the path $T$ reaches $v$

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

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

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

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

## Proof Outline:

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

Progress Function: For layer $L_i$,

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

1) $Z_0 = 2^{-2\delta n^2}$
2) $Z_i$ is very slowly growing: $Z_0 \approx Z_m$
3) If $s \in L_m$, then $Z_m \geq Pr(s) \cdot 2^{(\delta n)^2} \cdot 2^{-2\delta n^2}$

Hence: If $s$ is significant, $Pr(s) \leq 2^{-(\delta n)^2} = 2^{-\Omega(n^2)}$

(the hard step is step 2)

## How we prove that $Z_i$ is very slowly growing:

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

$$Z_{i+1} = \sum_{v \in L_{i+1}} Pr(v) \cdot \langle \mathbf{P}_{x|v} , \mathbf{P}_{x|s} \rangle^{\delta n}$$

$$Z' = \sum_{e:\, L_i \to L_{i+1}} Pr(e) \cdot \langle \mathbf{P}_{x|e} , \mathbf{P}_{x|s} \rangle^{\delta n}$$

**By a simple convexity argument,** $Z_{i+1} \leq Z'$

**The hard part is to show that** $Z'$ **is only negligibly larger than** $Z_i$

## How we prove that $Z'$ is only negligibly larger than $Z_i$:

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

$$Z' = \sum_{e:\, L_i \to L_{i+1}} Pr(e) \cdot \langle \mathbf{P}_{x|e} , \mathbf{P}_{x|s} \rangle^{\delta n}$$

**We show that, on average,**

$$\sum_{e:\, v \to L_{i+1}} Pr(e) \cdot \langle \mathbf{P}_{x|e} , \mathbf{P}_{x|s} \rangle^{\delta n}$$

**is only negligibly larger than** $Pr(v) \cdot \langle \mathbf{P}_{x|v} , \mathbf{P}_{x|s} \rangle^{\delta n}$

How we prove that, on average,

$$\sum_{e:\, v \to L_{i+1}} Pr(e) \cdot \langle \mathbf{P}_{x|e}\, , \mathbf{P}_{x|s} \rangle^{\delta n}$$

is only negligibly larger than $Pr(v) \cdot \langle \mathbf{P}_{x|v}\, , \mathbf{P}_{x|s} \rangle^{\delta n}$

Roughly speaking: For parity:

$\langle \mathbf{P}_{x|e}\, , \mathbf{P}_{x|s} \rangle$ are close, up to a normalization, to the Fourier coefficients of $\mathbf{P}_{x|v} \cdot \mathbf{P}_{x|s}$. By introducing stopping rules for the path $T$, we are able to bound the $L_2$-norm of $\mathbf{P}_{x|s}$ and the $L_\infty$-norm of $\mathbf{P}_{x|v}$ and hence the $L_2$-norm of $\mathbf{P}_{x|v} \cdot \mathbf{P}_{x|s}$, so that the Fourier coefficients of $\mathbf{P}_{x|v} \cdot \mathbf{P}_{x|s}$ are small on average. Another stopping rule ensures that the normalization doesn't distort by much

## Stopping Rules:

Stop on a vertex $v$ if:

1) $||\mathbf{P}_{x|v}||_2$ is large
2) $\mathbf{P}_{x|v}(x)$ is large
3) The next edge corresponds to a large Fourier coefficient of $\mathbf{P}_{x|v}$

The stopping rules do not depend on $s$

They do depend on $x$

## Summary:

**For a large class of learning problems, any learning algorithm requires either super-linear memory size or a super-polynomial number of samples**

## Main Message:

**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**

[S14, SVW16, R16, VV16, KRT17, MM17, R17, MM18, BOGY18, GRT18, DS18, AS18, DKS19, SSV19, GRT19, GKR19]

# Thank You!