Time-Space Lower Bound for Parity Learning

Xuangui Huang   Willy Quach

Machine Learning Project Presentation
April 9th 2018

Learning Parity

A secret vector $\mathbf{s} = (s_1, \ldots, s_n) \in \{0,1\}^n$.

A secret vector $\mathbf{s} = (s_1, \ldots, s_n) \in \{0,1\}^n$.

Learning Parity

A secret vector $\mathbf{s} = (s_1, \ldots, s_n) \in \{0,1\}^n$.

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i \bmod 2$$

$$\xrightarrow[\mathbf{a} \leftarrow_R \{0,1\}^n]{\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle}$$

Learning Parity

A secret vector $\mathbf{s} = (s_1, \ldots, s_n) \in \{0,1\}^n$.

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i \bmod 2$$

$$\xrightarrow[\mathbf{a} \leftarrow_R \{0,1\}^n]{\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle}$$

$$\downarrow$$

$$\mathbf{s}$$

A secret vector $\mathbf{s} = (s_1, \ldots, s_n) \in \{0,1\}^n$.

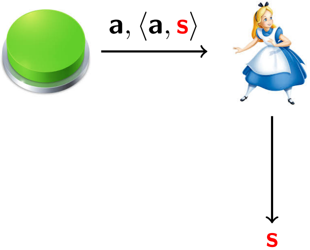
With **large memory**:

Gather n vectors $\mathbf{a}$ into an invertible $\mathbf{A}$,

$\longrightarrow$ obtain $(\mathbf{A}, \mathbf{A} \cdot \mathbf{s})$

Then use **Gaussian Elimination:**

$\longrightarrow \sim n$ samples and n^2 memory



Learning Parity: Two strategies

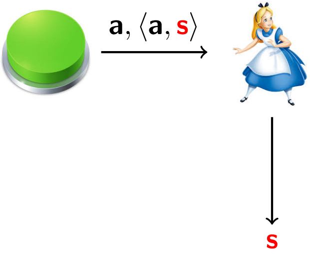
A secret vector $\mathbf{s} = (s_1, \ldots, s_n) \in \{0,1\}^n$.

With **large memory**:
Gather n vectors $\mathbf{a}$ into an invertible $\mathbf{A}$,
$\longrightarrow$ obtain $(\mathbf{A}, \mathbf{A} \cdot \mathbf{s})$
Then use **Gaussian Elimination:**

$\longrightarrow \sim n$ samples and n^2 memory

With **a lot of samples:**
Wait for a sample $\mathbf{a} = (1, 0, \ldots, 0)$;
read $s_1 = \langle \mathbf{a}, \mathbf{s} \rangle$.

$\longrightarrow \sim n \cdot 2^n$ samples and n memory



Time-Memory Tradeoff for Learning Parity

- Those are the **only two** strategies!

Main Theorem ([Raz, FOCS'16] , informal)

Any algorithm that learns parity either:

- Uses $\sim \mathbf{n^2}$ **memory**,

OR

- Uses an **exponential number of samples**.

Time-Memory Tradeoff for Learning Parity

- Those are the **only two** strategies!

Main Theorem ([Raz, FOCS'16])

For all $c < 1/20$, there exists an $\alpha > 0$ such that
any program using

- $\leq \mathbf{2^{\alpha n}}$ **samples**
- $\leq \mathbf{cn^2}$ **memory**

only succeeds with probability $\leq \mathcal{O}(2^{-\alpha n})$.

Time-Memory Tradeoff for Learning Parity

- Those are the **only two** strategies!

Main Theorem ([Raz, FOCS'16] , informal)

Any algorithm that learns parity either:

- Uses $\sim \mathbf{n^2}$ **memory**,

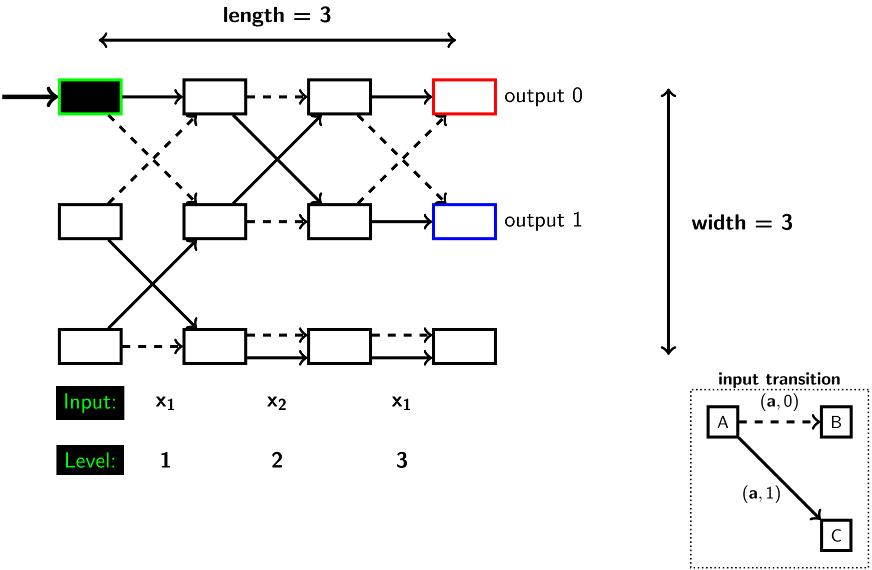
OR

- Uses an **exponential number of samples**.

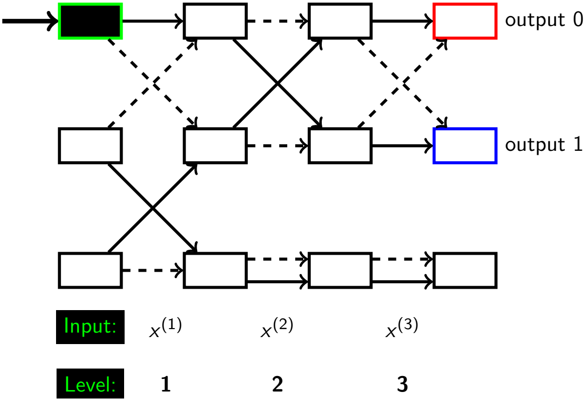
- Proof?

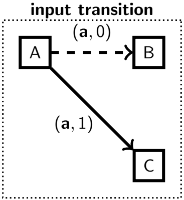
  Need a computational model for **bounded memory**.

Branching Programs

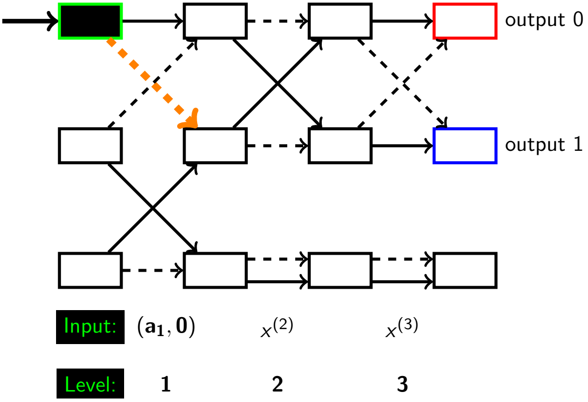


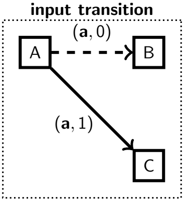
$$\text{Input: } x^{(1)} = (\mathbf{a}_1, 0)$$





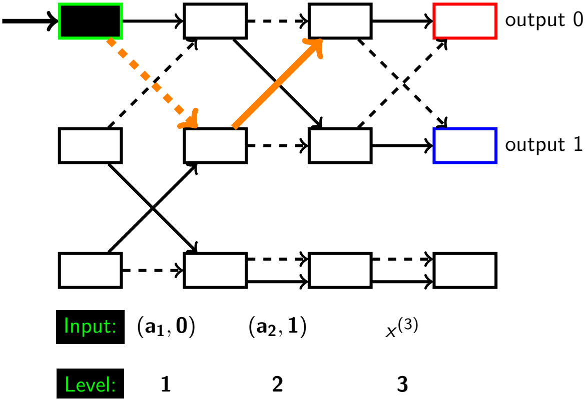
Input: $x^{(1)} = (\mathbf{a}_1, 0)$

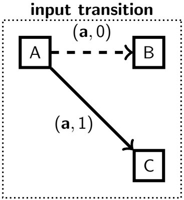




Branching Programs

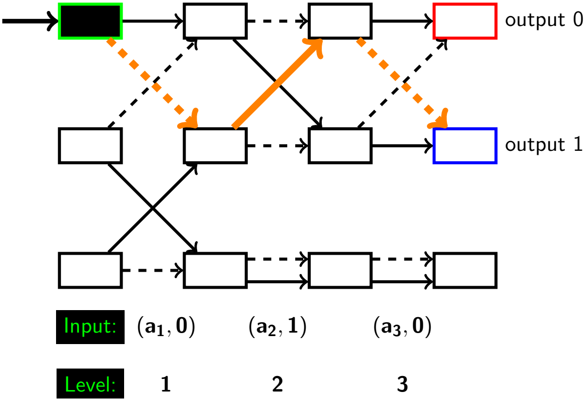
Input: $x^{(2)} = (\mathbf{a}_2, 1)$

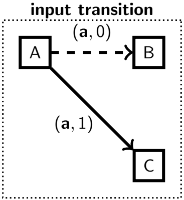




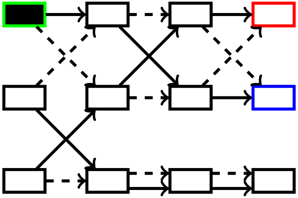
Branching Programs

$$\text{Input: } x^{(3)} = (\mathbf{a}_3, 0)$$



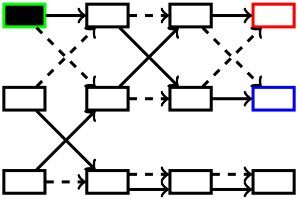


- Idea: embed **structure** into the computation.



Affine Branching Programs

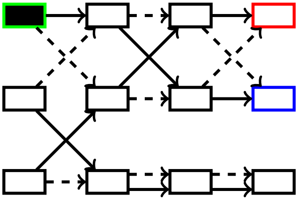
- Idea: embed **structure** into the computation.
- Memorize **subspace** where **s** lives in **nodes**.



Affine Branching Programs

- Idea: embed **structure** into the computation.
- Memorize **subspace** where **s** lives in **nodes**.
- Associate **nodes** with **affine subspaces.**
  - Starting node is **the whole space**.

$$E_{start} = \{0,1\}^n.$$



Affine Branching Programs

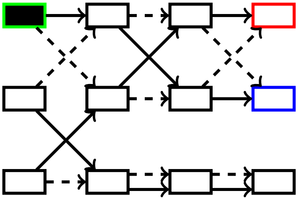
- Idea: embed **structure** into the computation.
- Memorize **subspace** where **s** lives in **nodes**.
- Associate **nodes** with **affine subspaces.**
  - Starting node is **the whole space**.

    $$E_{start} = \{0,1\}^n.$$

  - Edge (u, v) for input $(\mathbf{a}, b)$,

    Restricts **s** to a **smaller subspace**.

    $$E_v \supseteq E_u \cap \{\mathbf{s} \mid \langle \mathbf{a}, \mathbf{s} \rangle = b\}$$



Proof Outline

The proof goes in two steps:

Step 1, easy-ish

Any **Affine Branching Program** that Learns Parity using *small width* **needs** an exponential number of samples

Proof Outline

The proof goes in two steps:

Step 1, easy-ish

Any **Affine Branching Program** that Learns Parity using *small width* **needs** an exponential number of samples

Main Idea: Subspace in output node must have **large** dimension.

Proof Outline

The proof goes in two steps:

Step 1, easy-ish

Any **Affine Branching Program** that Learns Parity using *small width* **needs** an exponential number of samples

Main Idea: Subspace in output node must have **large** dimension.
$\Longrightarrow$ Outputs **s** with **exponentially small probability**.

Proof Outline

The proof goes in two steps:

Step 1, easy-ish

Any **Affine Branching Program** that Learns Parity using *small width* **needs** an exponential number of samples

Main Idea: Subspace in output node must have **large** dimension.
$\Longrightarrow$ Outputs **s** with **exponentially small probability**.

Step 2, much harder

Any **Branching Program** can be simulated by an **Affine Branching Program**.

Summary

Main Theorem ([Raz, FOCS'16], informal)

Any algorithm that learns parity either:

- Uses $\sim$ $\mathbf{n^2}$ **memory**,

OR

- Uses an **exponential number of samples**.

Follow-up works: ([Kol-Raz-Tal, STOC'17], [Raz, FOCS'17], [Garg-Raz-Tal, STOC'18]...)

- Generalize lower-bound for **larger class of problems**
  - **Sparse** parities
  - Low-degree equations...

Summary

Main Theorem ([Raz, FOCS'16], informal)

Any algorithm that learns parity either:

- Uses $\sim \mathbf{n^2}$ **memory**,

OR

- Uses an **exponential number of samples**.

Follow-up works: ([Kol-Raz-Tal, STOC'17], [Raz, FOCS'17], [Garg-Raz-Tal, STOC'18]...)

- Generalize lower-bound for **larger class of problems**
  - **Sparse** parities
  - Low-degree equations...

Thank you for your attention