Space Lower Bounds for Learning Problems

Andrew Gu
Franklyn Wang
Eric Zhang

Harvard University
May 6, 2021

Agenda

Work Covered

This work covers [Raz17], which subsumes [Raz18].

Work Covered

This work covers [Raz17], which subsumes [Raz18].

- (No, that's not a typo!)

Work Covered

This work covers [Raz17], which subsumes [Raz18].

- (No, that's not a typo!)
- Simpler proof, different method of attack, but still proves against branching programs.

Work Covered

This work covers [Raz17], which subsumes [Raz18].

- (No, that's not a typo!)
- Simpler proof, different method of attack, but still proves against branching programs.
- Main theorem can be applied to a broad class of learning problems, which includes but is not limited to parity learning.

Setup

- In our setup, we are trying to properly learn a binary function $f_\theta : \mathcal{X} \to \{-1, 1\}$ where $\theta \in \Theta$ where $\mathcal{X}$, Θ are finite sets.
- We are learning θ from samples $(x_i, f_\theta(x_i))$ in the streaming setting.

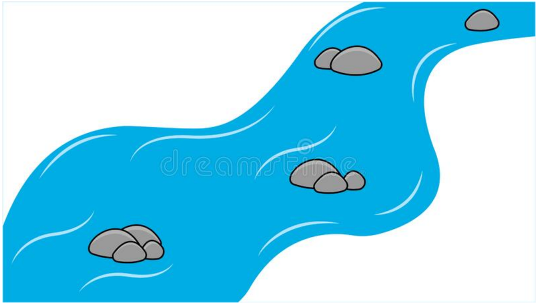


Figure: A stream.

Agenda

Space-Lower Bounds for Learning

- Frequently, lower bounds (whether on space / time) are useful in measuring the absolute limits of what we can accomplish under certain models of learning.

Space-Lower Bounds for Learning

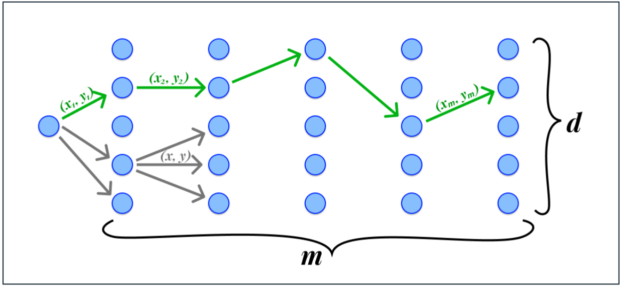
- Frequently, lower bounds (whether on space / time) are useful in measuring the absolute limits of what we can accomplish under certain models of learning.
- For example: this task takes $\geq X$ amount of space to complete, or $\geq Y$ amount of time. No program more efficient than this can exist.

Why streaming lower bounds?

- In streaming bounds, the model is given sequential access to examples

$$(x_1, y_1), (x_2, y_2), \ldots$$

- Streaming bounds penalize the number of times a sample is inspected.
- This type of bound fits in quite well with the branching program framework.



When should lower bounds exist?

We present here two function classes on $\mathbb{F}_2^n \to \{-1, 1\}$ of size 2^n.

- An easily learnable function class:

$$\{f_k(x) \equiv (-1)^k \mid 0 \leq k \leq 2^n - 1\}.$$

- A not so easily learnable function class [Raz18]:

$$\left\{f_c(x) = (-1)^{\sum_{i=1}^n c_i x_i} \mid c_i \in \{0, 1\}\right\}$$

- One sense in which a learning problem can be “hard” is that you have to know θ exactly to find $f_\theta(x)$.

Spectral Norm Condition

The result of [Raz17] says that learning is particularly hard when the matrix

$$M = \begin{bmatrix} f_{\theta_1}(x_1) & f_{\theta_2}(x_1) & \dots & f_{\theta_n}(x_1) \\ f_{\theta_1}(x_2) & f_{\theta_2}(x_2) & \dots & f_{\theta_n}(x_2) \\ \vdots & \vdots & & \vdots \\ f_{\theta_1}(x_k) & f_{\theta_2}(x_k) & \dots & f_{\theta_n}(x_k) \end{bmatrix}$$

has a low spectral norm ($\|M\|_2$ small). **Why is this important?**

Spectral Norm Intuition

- Take $\Theta = \begin{bmatrix} p(\theta_1) & p(\theta_2) & \dots & p(\theta_n) \end{bmatrix}^\top$, letting this be the prior that we have on θ.

Spectral Norm Intuition

- Take $\Theta = \begin{bmatrix} p(\theta_1) & p(\theta_2) & \dots & p(\theta_n) \end{bmatrix}^\top$, letting this be the prior that we have on θ.
- Low spectral norm corresponds to $\|M\Theta\|_2$ being small when $\|\Theta\|_2$ is small. $M\Theta$ is
$$\begin{bmatrix} \mathbb{E}_{\theta\sim p}[f_\theta(x_1)] & \mathbb{E}_{\theta\sim p}[f_\theta(x_2)] & \dots & \mathbb{E}_{\theta\sim p}[f_\theta(x_n)] \end{bmatrix}^\top.$$

Spectral Norm Intuition

- Take $\Theta = \begin{bmatrix} p(\theta_1) & p(\theta_2) & \dots & p(\theta_n) \end{bmatrix}^\top$, letting this be the prior that we have on θ.
- Low spectral norm corresponds to $\|M\Theta\|_2$ being small when $\|\Theta\|_2$ is small. $M\Theta$ is
$$\begin{bmatrix} \mathbb{E}_{\theta\sim p}[f_\theta(x_1)] & \mathbb{E}_{\theta\sim p}[f_\theta(x_2)] & \dots & \mathbb{E}_{\theta\sim p}[f_\theta(x_n)] \end{bmatrix}^\top.$$
- Note that when this vector has small norm, it effectively means that we are **uncertain about** $f_\theta(x_i)$ just from knowing what Θ is. In some sense, when the spectral norm is small, there are no "shortcuts" to knowing $f_\theta(x_i)$ without knowing θ exactly.

Main Theorem

Theorem ([Raz17])

Let $\Theta, \mathcal{X}$ be two finite sets. Let $n = \log_2 |\Theta|$. Let $M : \Theta \times \mathcal{X} \to \{-1, 1\}$ be a matrix, such that $\|M^\top\|_2 \leq 2^{\gamma n}$ where $\gamma < 1$. For any constant $c' < \frac{1}{3}$, there exists a constant $\epsilon' > 0$, such that the following holds: Let $c = c' \cdot (1-\gamma)^2$, and let $\epsilon = \epsilon' \cdot (1-\gamma)$. Let B be a branching program of length at most $2^{\epsilon n}$ and width at most 2^{cn^2} for the learning problem that corresponds to the matrix M. Then, the success probability of B is at most $O(2^{-\epsilon n})$.

Thus, to learn problems with we either need quadratic space or exponentially many samples.

Agenda

Walking Along the Branching Program

- A branching program is a *layered* graph whose leaves correspond to the output values of θ (the predicted parameters).

Walking Along the Branching Program

- A branching program is a *layered* graph whose leaves correspond to the output values of θ (the predicted parameters).
- **Key idea:** Instead of executing the entire program until reaching a leaf, we *truncate* the path after reaching a certain threshold of *significance* at a node.

Walking Along the Branching Program

- A branching program is a *layered* graph whose leaves correspond to the output values of θ (the predicted parameters).
- **Key idea:** Instead of executing the entire program until reaching a leaf, we *truncate* the path after reaching a certain threshold of *significance* at a node.

Theorem

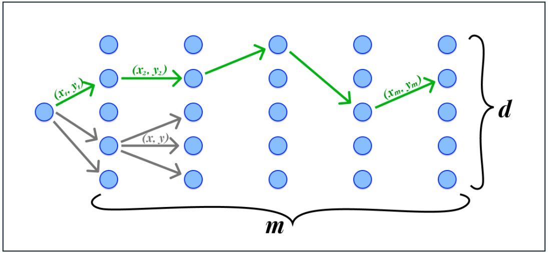
The probability that $\mathcal{T}$ reaches a significant vertex is $O(2^{-\epsilon n})$.

Measuring Progress of the Branching Program

- The proof defines

$$\mathcal{Z}_i = \sum_{v \in L_i} \Pr(v) \cdot \langle \mathbb{P}_{\theta|v}, \mathbb{P}_{\theta|s} \rangle^n, \quad i = 1, 2, \dots, m.$$

- Using an upper bound on $\mathcal{Z}_i$, it is shown that any particular significant vertex is reached with low probability.



Agenda

Implications for Learning

- There are surprisingly deep ramifications of space lower bounds on practical machine learning.
- Notably, streaming encapsulates most first-order methods, which use $\mathcal{O}(D)$ space, where D is the number of parameters.
- **Examples:** SGD, Adam, AdaGrad, etc.
- Thus, unless $D \gg n^2$, it may actually not be possible to solve the "hard problems" that we've described.
- Partially generalizes why learning parities is hard [SSSS17].

Practical Applications: Filecoin

- Another application of space lower bounds is to cryptocurrencies.
- An example of a “Proof-of-Space” system is the proof of replication used in Filecoin, part of the largest distributed filesystem in the world.

Figure: Filecoin

- Using space lower bounds is similar to asking a client to prove that it can store your object, but asking it to solve tasks which require space.

References

Ran Raz.
A time-space lower bound for a large class of learning problems.
In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 732–742. IEEE, 2017.

Ran Raz.
Fast learning requires good memory: A time-space lower bound for parity learning.
Journal of the ACM (JACM), 66(1):1–18, 2018.

References

Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah.
Failures of gradient-based deep learning.
In *International Conference on Machine Learning*, pages 3067–3075. PMLR, 2017.