Adaptivity vs Postselection, and Hardness Amplification in Polynomial Approximation

Lijie Chen

Tsinghua University

December 14, 2016

Story Time

- "Adaptivity vs Postselection" ? What is that? Why would anyone on earth study this question?

- Well, need some background story...

Story Time

- Back in this April, I was visiting MIT, worked with Prof. Scott Aaronson.

- I read a paper [Aar10] by him, and find a bug in a corollary: it claims the main result implies an oracle separation $\mathsf{BQP}^{\mathcal{O}} \not\subset \mathsf{PostBPP}^{\mathcal{O}}$.
  - I: Oops, the proof seems not right...!
  - Prof. Aaronson: Oops, can you fix that?
  - I: Let me have a try...
- Then this paper somehow came out...

My Challenge

In about **20 minutes**, explain what is the following (some not so standard) complexity classes and our results, then (probably) give you a taste of our techniques.

- PP.
- PostBPP.
- PostBQP.
- SBQP.
- SBP.
- A0PP.

Too many definitions...
I have to skip many formal discussions and some results.

Background: Relativization, Oracle Separation and Query Complexity

What is oracle separation? Quick overview some backgrounds.

- **Relativized Techniques**: Techniques that works equally well when given an *arbitrary* oracle.
- **Oracle Separation**: For two complexity classes $\mathcal{C}$ and $\mathcal{D}$, find an oracle function $\mathcal{O} : \{0,1\}^* \to \{0,1\}$ such

$$\mathcal{C}^{\mathcal{O}} \not\subset \mathcal{D}^{\mathcal{O}}.$$

  - Implies that relativized techniques along are not enough to show $\mathcal{C} = \mathcal{D}$. (A warning sign that new techniques are needed.)
  - An **evidence** that $\mathcal{C}$ actually does not equal to $\mathcal{D}$.

Background: Relativization, Oracle Separation and Query Complexity

- **Oracle Separation $\Leftrightarrow$ Query Complexity**
  - The usual way to find oracle separation is to show query complexity lower bound.
  - Quick example of P vs NP.
    - Imagine the oracle encodes a string of 2^n length.
    - A question on oracle: "Does that 2^n-bit string contains a 1"? (OR of 2^n-bits).
    - NP algorithm: simply guess the position of the 1-bit. $\Rightarrow O(n)$.
    - P algorithm: needs to query all the 2^n bits. $\Rightarrow \Omega(2^n)$. (**A query complexity lower bound**)
    - Some standard diagonalization $\Rightarrow$ an oracle separation that $\mathsf{NP}^{\mathcal{O}} \not\subset \mathsf{P}^{\mathcal{O}}$.

Background: Relativization, Oracle Separation and Query Complexity

- **Oracle Separation $\Leftrightarrow$ Query Complexity**
- So in general, to find an oracle separation between $\mathcal{C}$ and $\mathcal{D}$, we:
  - Find a Boolean function $f\colon \{0,1\}^{2^n} \to \{0,1\}$ on oracles. (Probably a partial function.)
  - Such that given an oracle $\mathcal{O} \in \{0,1\}^{2^n}$, to compute $f(\mathcal{O})$:
    - A $\mathcal{D}$ algorithm needs super-polynomial queries to $\mathcal{O}$.
    - There is a poly-time $\mathcal{C}$ algorithm to solve $f(\mathcal{O})$.
- Some examples:
  - OR function $\Rightarrow$ $\mathbf{NP}^{\mathcal{O}} \not\subset \mathbf{P}^{\mathcal{O}}$.
  - GapMaj function $\Rightarrow$ $\mathbf{BPP}^{\mathcal{O}} \not\subset \mathbf{P}^{\mathcal{O}}$.
  - Simon function $\Rightarrow$ $\mathbf{BQP}^{\mathcal{O}} \not\subset \mathbf{BPP}^{\mathcal{O}}$.
  - Collision function $\Rightarrow$ $\mathbf{SZK}^{\mathcal{O}} \not\subset \mathbf{BQP}^{\mathcal{O}}$.
  - ODD-MAX-BIT function $\Rightarrow$ $\mathbf{P}^{\mathbf{NP}^{\mathcal{O}}} \not\subset \mathbf{PP}^{\mathcal{O}}$.

Background: Postselection

An interesting idea in computation, basically, it means that you can condition on some rare event.
Best illustrated by an example:
A foolproof way to solve 3-SAT is:

- Given a 3-SAT formula φ, need to output whether it is satisfiable.
- Output NO and terminate with probability 2^{-2n}.
- Guess a random assignment $x \in \{0,1\}^n$.
- **Kill yourself** if x does not satisfy φ, output YES otherwise.

Analysis:

- Condition on **you are alive**.
  - Answer is NO: you always output NO.
  - Answer is YES: you output YES w.p. $\geq 2^n/(2^n+1)$ (Simple Bayesian).
  - You are correct w.h.p.

Background: PostBPP and PostBQP

- **PostBPP** [HHT97]: problems can be solved in poly-time by **classical** postselection algorithm.
  - So 3-SAT $\in$ PostBPP, and NP $\subseteq$ PostBPP from the previous slide.
- **PostBQP** [Aar05]: problems can be solved in poly-time by **quantum** postselection algorithm.
  - Certainly PostBPP $\subseteq$ PostBQP.

Background: PostBQP and PP

- **PostBQP**: problems can be solved in poly-time by **quantum** postselection algorithm.
  - Certainly PostBPP $\subseteq$ PostBQP.

- **PP**: problems can be solved by a polynomial-time randomized Turing Machine with correct probability $1/2 + 2^{-\mathsf{poly}(n)}$.
  - A relaxation of BPP, in which you need to be correct w.p. $\geq 2/3$.
  - A fundamental classes in computational complexity theory.
  - Surprisingly, **PostBQP $=$ PP [Aar05]**.

Story Time: Cont

...Finally we have went through the definitions...

- Recall that I want to rescue Prof. Aaronson's oracle separation $\mathsf{BQP}^{\mathcal{O}} \not\subset \mathsf{PostBPP}^{\mathcal{O}}$. (Hopefully now you know what is PostBPP!).
- From the previous discussion, I need to find a Boolean function $f\colon \{0,1\}^{2^n} \to \{0,1\}$ such that:
  - It is easy for quantum algorithm (only need $\mathrm{poly}(n)$ queries).
  - Hard for any postselection algorithms.
- But, what is hard for postselection algorithms?
  - **Adaptive queries** (this work)!

Our Results: Informal Statement

- **Small Bounded Error** Computation [BGM06]:
  - There exist a real α (can be exponentially small) such that:
    - Answer is YES: your algorithm accept with probability $> \alpha$.
    - Answer is NO: your algorithm accept with probability $\leq \alpha/2$.
  - Yet another generalization of BPP (in which α must be $2/3$).
  - SBP: poly-time classical small bounded error computation.
  - SBQP: poly-time quantum small bounded error computation.
- Informally, we showed that, (classically or quantumly) for a partial Boolean function f:
  - If there is no efficient **small bounded-error** algorithm for f,
  - then no efficient **postselection bounded-error** algorithm can answer $\log n$ adaptive queries to f.

Some Applications

- The Simon function is hard for SBP, so the adaptive version of it is hard for PostBPP.
  - Its adaptive version is also obviously easy for BQP.
  - $\Rightarrow$ an oracle separation $\mathsf{BQP}^{\mathcal{O}} \not\subseteq \mathsf{PostBPP}^{\mathcal{O}}$!
  - Good, rescued the separation.
- Since PostBQP is equivalent to PP and PP is closely related to **polynomial approximation**.
  - Our work implies a polynomial hardness amplification scheme with the same effect in a recent work by Thaler [Tha14] but a much simpler *amplifier* (not cover in this talk.)
- Using AND, reproved an old oracle separation $\mathsf{P}^{\mathsf{NP}^{\mathcal{O}}} \not\subset \mathsf{PP}^{\mathcal{O}}$ by Beigel [Bei94].
- Also implies a new oracle separation $\mathsf{P}^{\mathsf{SZK}^{\mathcal{O}}} \not\subset \mathsf{PP}^{\mathcal{O}}$.

$\mathsf{P}^{\mathsf{NP}^{\mathcal{O}}} \not\subset \mathsf{PP}^{\mathcal{O}}$: A Toy Example

- To avoid too many technical details, we illustrate our techniques by constructing an oracle separation between P^{NP} and PP.

- The approach can be generalized to our full formal statement easily.

The Adaptive Construction

We need to formally define what is the adaptive version of a Boolean function:

Definition (Adaptive Construction)

- *Given a function $f: D \to \{0,1\}$ with $D \subseteq \{0,1\}^M$ and an integer d, we define $\mathsf{Ada}_{f,d}$, its depth d adaptive version, as follows:*

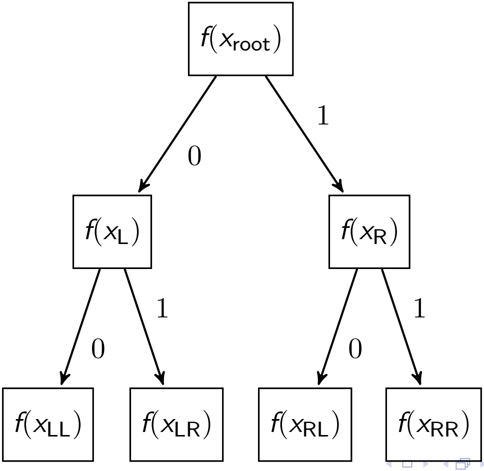
$$\mathsf{Ada}_{f,0} := f \quad \textit{and} \quad \begin{aligned} &\mathsf{Ada}_{f,d} : D \times D_{d-1} \times D_{d-1} \to \{0,1\} \\ &\mathsf{Ada}_{f,d}(w,x,y) := \begin{cases} \mathsf{Ada}_{f,d-1}(x) & f(w) = 0 \\ \mathsf{Ada}_{f,d-1}(y) & f(w) = 1 \end{cases} \end{aligned}$$

- *where D_{d-1} denotes the domain of $\mathsf{Ada}_{f,d-1}$.*

The Adaptive Construction: Example when $d = 2$

An example for $\mathrm{Ada}_{f,2}$, given input

$$x = \Big(x_{\mathrm{root}}, (x_{\mathrm{L}}, x_{\mathrm{LL}}, x_{\mathrm{LR}}), (x_{\mathrm{R}}, x_{\mathrm{RL}}, x_{\mathrm{RR}})\Big) \in D^7.$$



Warm Up: PP and Polynomials

Lemma (PP-to-Polynomial Lemma)

Given a Boolean function $f\colon \{0,1\}^M \to \{0,1\}$, suppose there is a d-time PP algorithm, then there is polynomial $p : \mathbb{R}^M \to \{0,1\}$:

1. *p is of degree at most d.*
2. *$p(x) \geq 1$ when $f(x) = 1$.*
3. *$p(x) \leq -1$ when $f(x) = 0$.*
4. *$|p(x)|_\infty = \max_{x\in\{0,1\}^M} |p(x)| \leq 2^d$.*

Why?

- Simply let $p(x) = \#\text{accept paths} - \#\text{rejected paths}$.

Also, if a polynomial p satisfies (2) and (3) above, then we say it is a **valid** polynomial for f.

A Lemma from Minimax Theorem

We have the following interesting lemma proved using the Minimax Theorem.

Lemma (Base-Case Lemma)

- *Let* $f = \mathrm{AND}_n$ *(*AND *on* n*-bits).*
- *Then there exist two distributions:*
- $\mathcal{D}_0$ *supported on* $f^{-1}(0)$ *and* $\mathcal{D}_1$ *supported* $f^{-1}(1)$*, such that*

$$-p(\mathcal{D}_0) > 2 \cdot p(\mathcal{D}_1)$$

- *where* $p(\mathcal{D}) = \mathbb{E}_{x \sim \mathcal{D}}[p(x)]$*,*
- *for all degree-*$\sqrt{n}$ *valid polynomial* p *for* f*.*

Very easy to prove using the *one-sided approximate degree* lower bound [NS94] on OR_n ($\neg\mathrm{AND}_n$), omit here.

The Proof Details: An Induction

We want to prove the following theorem by an induction.

Theorem (Induction Theorem)

- *Let* $f = \mathsf{AND}_n$ *(*AND *on n-bits). Then for each integer d,*
- *there exist two distributions* $\mathcal{D}_1^d$ *supported on* $\mathsf{Ada}_{f,d}^{-1}(1)$ *and* $\mathcal{D}_0^d$ *supported on* $\mathsf{Ada}_{f,d}^{-1}(0)$*, such that*
- $$-p(\mathcal{D}_0^d) > 2^{2^d} \cdot p(\mathcal{D}_1^d)$$
- *for any degree-*$\sqrt{n}$ *valid polynomial p for* $\mathsf{Ada}_{f,d}$.

The Oracle Separation

- Let $d = \log n$, then for any degree-$\sqrt{n}$ valid polynomial p for $\mathsf{Ada}_{f,d}$:
- $\|p\|_\infty \geq -p(\mathcal{D}_0^d) > 2^{2^d} \cdot p(\mathcal{D}_1^d) \geq 2^{2^{\log d}} = 2^n$.
- Comparing with the PP-to-Polynomial Lemma, $\Rightarrow$ a PP algorithm need $\Omega(\sqrt{n})$ time to solve $\mathsf{Ada}_{\mathsf{AND},\log n}$.
- On the other side: there is a trivial polylog(n)-time P^{NP} algorithm.
- Big separation!
- So $\mathsf{Ada}_{\mathsf{AND},\log n}$ implies an oracle separation $\mathsf{P}^{\mathsf{NP}^{\mathcal{O}}} \not\subseteq \mathsf{PP}^{\mathcal{O}}$!

Proof for the Induction Theorem: Base Case when $d = 0$

Now we prove our induction theorem.

- Consider the base case when $d = 0$.
- Simply set $\mathcal{D}_0^0 = \mathcal{D}_0$ and $\mathcal{D}_1^0 = \mathcal{D}_1$ as in the Base-Case Lemma.
- From the definition, $\mathsf{Ada}_{f,0} := f$, the base case just follows from the Base-Case Lemma.

$$-p(\mathcal{D}_0^0) > 2 \cdot p(\mathcal{D}_1^0) = 2^{2^0} \cdot p(\mathcal{D}_1^0).$$

Proof for the Induction Theorem: when $d \geq 1$ Construction of $\mathcal{D}_0^d$ and $\mathcal{D}_1^d$

- Suppose that we have already constructed the required distributions $\mathcal{D}_0^{d-1}$ and $\mathcal{D}_1^{d-1}$ for $\mathsf{Ada}_{f,d-1}$.

- Decompose the input to $\mathsf{Ada}_{f,d}$ as $(w, x, y) \in D \times D_{d-1} \times D_{d-1}$ as in the definition.

- We claim that

$$\mathcal{D}_0^d = (\mathcal{D}_0, \mathcal{D}_0^{d-1}, \mathcal{D}_0^{d-1}) = \mathcal{D}_0 \times \mathcal{D}_0^{d-1} \times \mathcal{D}_0^{d-1}$$

  and

$$\mathcal{D}_1^d = (\mathcal{D}_1, \mathcal{D}_1^{d-1}, \mathcal{D}_1^{d-1}) = \mathcal{D}_1 \times \mathcal{D}_1^{d-1} \times \mathcal{D}_1^{d-1}$$

  satisfy our conditions.

Proof for the Induction Theorem: Outline

- From definition, easy to see that $\mathcal{D}_d^0$ and $\mathcal{D}_d^1$ are supported on $\mathsf{Ada}_{f,0}$ and $\mathsf{Ada}_{f,1}$.
- We are going to show for any degree-$\sqrt{n}$ valid polynomial p for $\mathsf{Ada}_{f,d}$:

$$
\begin{aligned}
-p(\mathcal{D}_0, \mathcal{D}_0^{d-1}, \mathcal{D}_0^{d-1}) &> 2^{2^{d-1}} \cdot p(\mathcal{D}_0, \mathcal{D}_1^{d-1}, \mathcal{D}_0^{d-1}) && \text{(Step I)} \\
p(\mathcal{D}_0, \mathcal{D}_1^{d-1}, \mathcal{D}_0^{d-1}) &> -p(\mathcal{D}_1, \mathcal{D}_1^{d-1}, \mathcal{D}_0^{d-1}) && \text{(Step II)} \\
-p(\mathcal{D}_1, \mathcal{D}_1^{d-1}, \mathcal{D}_0^{d-1}) &> 2^{2^{d-1}} \cdot p(\mathcal{D}_1, \mathcal{D}_1^{d-1}, \mathcal{D}_1^{d-1}) && \text{(Step III)}
\end{aligned}
$$

- Putting them together:

$$
-p(\mathcal{D}_0^d) = -p(\mathcal{D}_0, \mathcal{D}_0^{d-1}, \mathcal{D}_0^{d-1}) > 2^{2^d} \cdot p(\mathcal{D}_1, \mathcal{D}_1^{d-1}, \mathcal{D}_1^{d-1}) = 2^{2^d} \cdot p(\mathcal{D}_1^d).
$$

- DONE!

Step I: $(\mathcal{D}_0, \mathcal{D}_0^{d-1}, \mathcal{D}_0^{d-1}) \Rightarrow (\mathcal{D}_0, \mathcal{D}_1^{d-1}, \mathcal{D}_0^{d-1})$.

- For any degree-$\sqrt{n}$ valid polynomial p for $\mathsf{Ada}_{f,d}$,
- for any fixed $W \in \mathrm{support}(\mathcal{D}_0)$ and $Y \in \mathrm{support}(\mathcal{D}_0^{d-1})$,
- let

$$p_L(x) := p(W, x, Y).$$

- From definition, p_L is a valid polynomial for $\mathsf{Ada}_{f,d-1}$.
- Hence,

$$-p_L(\mathcal{D}_0^{d-1}) > 2^{2^{d-1}} \cdot p_L(\mathcal{D}_1^{d-1}).$$

- By linearity, we have

$$-p(\mathcal{D}_0, \mathcal{D}_0^{d-1}, \mathcal{D}_0^{d-1}) > 2^{2^{d-1}} \cdot p(\mathcal{D}_0, \mathcal{D}_1^{d-1}, \mathcal{D}_0^{d-1}).$$

Step II: $(\mathcal{D}_0, \mathcal{D}_1^{d-1}, \mathcal{D}_0^{d-1}) \Rightarrow (\mathcal{D}_1, \mathcal{D}_1^{d-1}, \mathcal{D}_0^{d-1})$.

Similarly,

- for any degree-$\sqrt{n}$ valid polynomial p for $\mathsf{Ada}_{f,d}$,
- for any fixed $X \in \mathrm{support}(\mathcal{D}_1^{d-1})$ and $Y \in \mathrm{support}(\mathcal{D}_0^{d-1})$.
- Let
$$p_M(w) := -p(w, X, Y),$$
- from definition, p_M is a valid polynomial for f.
- Hence,
$$-p_M(\mathcal{D}_0) > 2 \cdot p_M(\mathcal{D}_1).$$
- Again by linearity, we have
$$p(\mathcal{D}_0, \mathcal{D}_1^{d-1}, \mathcal{D}_0^{d-1}) > -2 \cdot p(\mathcal{D}_1, \mathcal{D}_1^{d-1}, \mathcal{D}_0^{d-1}) > -p(\mathcal{D}_1, \mathcal{D}_1^{d-1}, \mathcal{D}_0^{d-1}).$$

Step III: $(\mathcal{D}_1, \mathcal{D}_1^{d-1}, \mathcal{D}_0^{d-1}) \Rightarrow (\mathcal{D}_1, \mathcal{D}_1^{d-1}, \mathcal{D}_1^{d-1})$.

Finally,

- for any degree-$\sqrt{n}$ valid polynomial p for $\mathrm{Ada}_{f,d}$,
- for any fixed $W \in \mathrm{support}(\mathcal{D}_1)$ and $X \in \mathrm{support}(\mathcal{D}_1^{d-1})$,
- let

$$p_R(y) := p(W, X, y).$$

- From definition, p_R is a valid polynomial for $\mathrm{Ada}_{f,d-1}$.
- Hence,

$$-p_R(\mathcal{D}_0^{d-1}) > 2^{2^{d-1}} \cdot p_R(\mathcal{D}_1^{d-1}).$$

- Still by linearity, we have

$$-p(\mathcal{D}_1, \mathcal{D}_1^{d-1}, \mathcal{D}_0^{d-1}) > 2^{2^{d-1}} \cdot p(\mathcal{D}_1, \mathcal{D}_1^{d-1}, \mathcal{D}_1^{d-1}).$$

- Q.E.D.

Open Question

- In this work, we found a **sufficient** condition for a function's adaptive version to be hard for PostBPP(PostBQP).
  - Can we find a **necessary and sufficient** condition?
  - Our condition here is not necessary.

- The $\mathrm{Ada}_{f,d}$ construction seems very interesting, are there any other applications?

The End

Thanks for listening!

Scott Aaronson.
Quantum computing, postselection, and probabilistic polynomial-time.
In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 461, pages 3473–3482. The Royal Society, 2005.

Scott Aaronson.
Bqp and the polynomial hierarchy.
In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 141–150. ACM, 2010.

Richard Beigel.
Perceptrons, pp, and the polynomial hierarchy.
Computational Complexity, 4(4):339–349, 1994.

Elmar Böhler, Christian Glaßer, and Daniel Meister.
Error-bounded probabilistic computations between ma and am.
Journal of Computer and System Sciences, 72(6):1043–1076, 2006.

Yenjo Han, Lane A Hemaspaandra, and Thomas Thierauf.
Threshold computation and cryptographic security.

SIAM Journal on Computing, 26(1):59–78, 1997.

Noam Nisan and Mario Szegedy.
On the degree of boolean functions as real polynomials.
Computational complexity, 4(4):301–313, 1994.

Justin Thaler.
Lower bounds for the approximate degree of block-composed functions.
In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 21, page 150, 2014.