Approximate Degree of Boolean Functions and Applications in Quantum Query Complexity

Undergraduate Project - 2017-18/II

Sagnik Bhattacharya
Advisor: Prof Rajat Mittal
Co-advisor: Prof Ketan Rajawat

April 18, 2017

Outline

Outline

Boolean Functions

- In the $\{0, 1\}$ basis, can be represented as:

$$f : \{0, 1\}^n \to \{0, 1\}$$

- In the Fourier $\{-1, 1\}$ basis, can be represented as:

$$f : \{-1, 1\}^n \to \{-1, 1\}$$

- Examples: OR, AND

Representing Boolean Functions as Polynomials

- We can represent each such function exactly by polynomial in n variables.
- Consider only multinlinear polynomials.
- Natural to talk about polynomials while discussing Boolean functions.

Approximate Degree of Boolean Functions

- A real polynomial p is said to be an ϵ-approximation to a Boolean function f if the following holds:

$$|p(x) - f(x)| < \epsilon \ \forall x \in \{-1, 1\}^n$$

- The minimum degree required to approximate a given function f is called the approximate degree of the function.
- Note that the upper bound on the approximate degree is n.
- Question: Can we do better?

Quantum Algorithms

In this talk...

- **Why should we care about approximate degree?** We will look at its relation with quantum query complexity.
- **How to find upper bounds on the approximate degree?** Use quantum algorithms (follows from answer to previous point) or provide an explicit construction.

Outline

Hardness of Functions

We can have several measure of how hard it is to compute a given boolean function. These measures include:

- Decision tree complexity/query complexity
- Block sensitivity
- Quantum query complexity

Nisan and Szegedy (1994) showed that the **approximate degree of a polynomial is polynomially related** to the first two hardness measures.

Quantum Query Complexity

Beals et al (1998, 2001) proved the following result.

Theorem

Let $\mathcal{A}$ be a quantum algorithm that makes $\mathcal{T}$ queries to to a black-box $\mathcal{X}$. Let $\mathcal{B}$ be a subset of basis states. Then there exists a real valued multilinear polynomial p of degree at most $2\mathcal{T}$ which equals the probability that observing the final state of the algorithm yields a state from $\mathcal{B}$.

We will now prove this more general theorem that implies the result we are looking for.

Proof

Relating Approximate Degree and Quantum Query algorithms

Using the previous theorem, we have the following result [Beals et al 1998]

Theorem

Let a quantum algorithm $\mathcal{A}$ compute a Boolean function f using T queries with bounded error. Let $\widetilde{deg}(f)$ be the approximate degree of the associated polynomial. Then,

$$T \geq \frac{\widetilde{deg}(f)}{2}$$

Outline

Approximating OR

Kothari, Bun and Thaler (2017) : Key ideas

- Isolate the 'hard' cases, solve them and reduce the general case to the hard case by some other computation.
- Give an explicit polynomial that approximates NOR on the hard cases.
- Think of polynomials as algorithms

Symmetric functions

- Functions whose value remains the same for every permutation of a given input string.
- Can be treated as a function of the Hamming weight of the input only.
- Example OR, AND

Approximating NOR

What is the hard case?

- First question - **what is a hard case?**
- We are looking for two inputs which are 'close' but for which the function value differs.
- In the case of NOR, consider the all-zero input and an input with just one 1.
- for the NOR function, we generalize this - we regard inputs with Hamming weight $\leq T$ to be hard, where T is a parameter which will be chosen later.

$\mathcal{P} = \mathcal{H}^n_{\leq T} =$ set of all strings with Hamming weight less than T

Chebyshev Polynomial of degree *d*

- $T_d(x) \in [-1, 1]$ for all $x \in [-1, 1]$
- $T_d(1 + \mu) \geq \frac{1}{2} exp(d\sqrt{\mu})$ for all $\mu \in (0, 1)$
- For any polynomial $p : \mathbb{R} \to \mathbb{R}$ of degree d with $|p(x)| \leq 1$ for all x in $[-1, 1]$ we have

$$|p(x)| \leq T_d(x) \leq (2|x|)^d \text{ for all } |x| > 1$$

The approximating polynomial on the hard cases

Suppose we have a promise that the only inputs are from the hard set. We claim that the following polynomial approximates $\textsc{Or}$ in this case.

$$V_{T,\epsilon}(x) = \left(1 - \frac{1}{M}\right) - \frac{1}{M} \cdot T_d\left(1 + \frac{1 - |x|}{T}\right)$$

We have $d = O\left(\sqrt{T} log\left(\frac{1}{\epsilon}\right)\right)$ and $M = T_d(1 + \frac{1}{T}) + 1$

The approximating polynomial on the hard cases - continued

We can show that this polynomial satisifies the following properties.

- $V_{T,\epsilon} \in [0, \epsilon]$ for the zero vector
- $V_{T,\epsilon} \in [1-\epsilon, 1]$ all inputs with Hamming weight $\leq T$
- $V_{T,\epsilon} \in [-a, a]$ for all other inputs

Therefore, it approximates $\textsc{or}$ on the hard inputs and has degree $\tilde{O}\left(\sqrt{T}\right)$. Using the properties of the Chebyshev Polynomials, we can also show that $a = exp\left(O\left(\sqrt{T} logn\right)\right)$

Reducing to the Hard case

- We claim that a polynomial $\tilde{q}$ exists that can distinguish the cases $|x| = 0$ and $x \notin \mathcal{P}$, and that such a polynomial has degree $O\left(\sqrt{\frac{N}{T}}\right)$.
- This polynomial can be explicitly constructed or its existence can be proven using the **Quantum Counting** algorithm [Brassard et al, 1998]
- More explicitly, $\tilde{q}$ satisfies the following:
  - $\tilde{q} \in \left[\frac{9}{10}, 1\right]$ for $|x| = 0$
  - $\tilde{q} \in [0, 1]$ for $|x| \leq T$
  - $\tilde{q} \in \left[0, \frac{1}{10}\right]$ for $|x| \geq T$

Reducing to the Hard case - continued

- Using $\tilde{q}$ we can construct another polynomial q with the following properties
  - $\tilde{q} \in \left[1 - \frac{1}{3a}, 1\right]$ for $|x| = 0$
  - $\tilde{q} \in [0, 1]$ for $|x| \leq T$
  - $\tilde{q} \in \left[0, \frac{1}{3a}\right]$ for $|x| \geq T$
- The degree of q is equal to $deg(\tilde{q}) \cdot O\left(log(3a)\right)$
- Therefore, $deg(q) = \tilde{O}(\sqrt{n})$.

Polynomial that approximates NOR

Claim: The polynomial $p \cdot q$ approximates NOR

Proof by picture.

Outline

Approximating NOR

Key ideas

- Isolate the 'hard' cases, solve them and reduce the general case to the hard case by some other computation.
- Give an explicit polynomial that approximates NOR on the hard cases.
- Think of polynomials as algorithms

The SURJECTIVITY Function

- What is the SURJECTIVITY function?
- What are the hard cases?
- Representation in terms of AND and OR on a restricted set of inputs
- Reduction to the hard case

Summary

- Approximate degree and relation with hardness measures
- Quantum query complexity and approximate degree
- Algorithms as polynomials
- Explicit approximation polynomial for OR
- Sketch of how to extend to SURJECTIVITY