Asking Pythia

Alapan Chaudhuri and Pratishtha Abrol

December 22, 2021

Contents

Introduction

Sophocles is wise, Euripides is wiser, but of all men Socrates is wisest.

- Pythia

Introduction

Query complexity is the study of complexity of one of the fundamental paradigms of computation, namely — queries.

So, what is a query? It is a mapping from "structures of one signature to structures of another vocabulary".

Introduction

Introduction

So, why is this of any importance?

- Simplicity and its importance in studying complexity of computable functions.
- Ability to analyse (all major) quantum algorithms.

Basics

In query complexity we deal with the notion of a blackbox. Imagine that we are provided with a blackbox x computing some unknown function, and all we are allowed to do is to make queries to the blackbox.

Our goal would be to calculate some property $f(x)$ of this blackbox with minimal queries.

Queries entail feeding in an input i to x, and observe the output $x(i)$.

Basics

Considering x as a function, we can agree that x is completely determined by the outputs to all the different types of inputs that it accepts.

Say that the domain of x is $[n] = \{1, 2, 3, \ldots, n\}$. Then, we can use a string s to represent x where:

- $\text{len}(s) = n$
- $s_i = x(i)$

Basics

Typically, the blackbox x will output Boolean outputs, which makes the string x a Boolean string in $\{0,1\}^n$.

However, it is also possible to consider blackboxes with non-Boolean outputs, in which case the string x will be in Σ^n for some alphabet Σ.

Basics

This beautifully transforms the problem into computing some function f of the input x by computing as few bits of x as possible.

Thus,

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

where f is a known function.

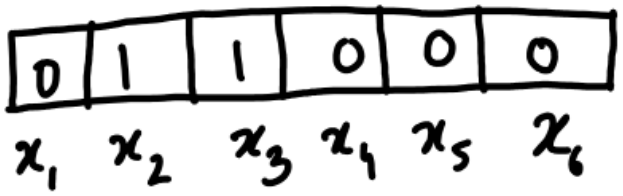
For example, f might be the **OR** function, in which case our goal would be to determine whether the input string x is all zeros (equivalently, whether the input blackbox x ever returns a 1).

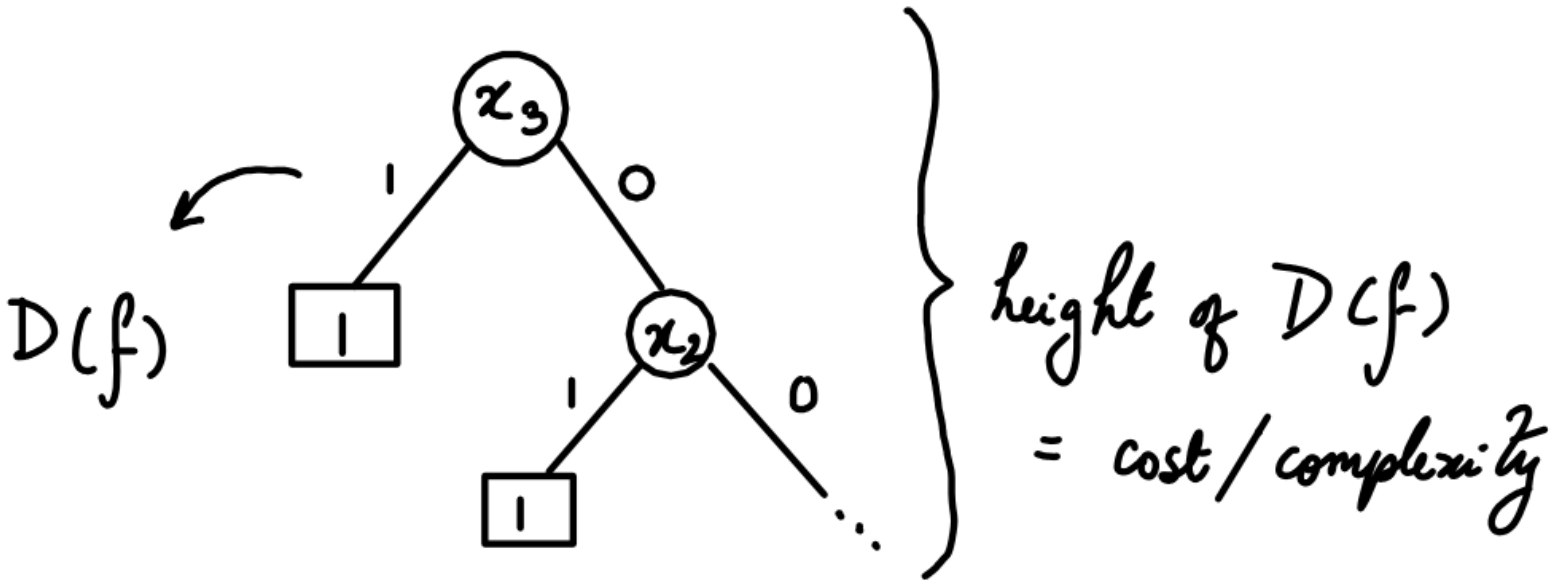
Can we model this problem better?

Umm yes, we shall use **Decision Trees**!

Basics

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

$x =$ 



Basics

If D is a decision tree, we use $D(x)$ to denote the output the decision tree when run on x. So, basically it is the leaf node we reach across the tree.

Furthermore, we say D computes f iff $D(x) = f(x)\ \forall x \in \{0,1\}^n$.

Determinism

Thus, as described above a deterministic query algorithm will be defined as a decision tree.

Definition

A (deterministic) decision tree on inputs of size n is either a leaf in $0, 1^n$ or a tuple (i, D_0, D_1) where $i \in [n]$ and where D_0 and D_1 are decision trees (on inputs of size n) which do not use i in any internal tuple.

Determinism

On that note, we define the deterministic query complexity as follows.

Definition

The decision tree complexity of $f : \{0,1\}^n \to \{0,1\}$ is namely,

$$min \text{ height}(D) \ \ \forall D \in D_f$$

where D_f is the set of all decision trees that compute f.

Computational Power of Decision Trees

We can associate several measures of complexity with a Boolean function. For example:

- Certificate Complexity: to measure how many of the n variables have to be given a value in order to fix the value of f.
- Sensitivity (Critical Complexity) and Block sensitivity: to measure how sensitive the value of f is to changes in the input.
- Degree of representing or approximating polynomials: We can represent a function f by means of a polynomial, or approximate it with one. The approximating polynomial may even be of a smaller degree.

Certificate Complexity

Let, $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function:

Definition

Let C be an assignment $C : S \to \{0,1\}$ of values to some subset S of the n variables. We say that C is consistent with $x \in \{0,1\}^n$ if $x_i = C(i)\ \forall i \in S$.
For $b \in \{0,1\}$, a $b-$certificate for f is an assignment C such that $f(x) = b$ whenever x is consistent with C. The size of C is $|S|$, the cardinality of S.

Certificate Complexity

Definition

The certificate complexity $C_x(f)$ of f on x is the size of a smallest $f(x)-$certificate that is consistent with x. The certificate complexity of f is $C(f) = max_x C_x(f)$.
The 1-certificate complexity of f is $C^{(1)}(f) = max_{\{x|f(x)=1\}} C_x(f)$, and similarly, for $C^{(0)}(f)$.

For example, $C^{(1)}(OR_n) = 1$ since it suffices to set one variable $x_i = 1$ to force the OR function to 1. On the other hand, $C(OR_n) = C^{(0)}(OR_n) = n$.

Sensitivity

Definition

The sensitivity $s_x(f)$ of f on x is the number of variables x_i for which $f(x) \neq f(x^i)$. The sensitivity of f is $s(f) = max_x s_x(f)$.

Block Sensitivity

Definition

The block sensitivity $bs_x(f)$ of f on x is the maximum number b such that there are disjoint sets $B_1, ...B_b$ for which $f(x) \neq f(x^{B_i})$. $bs(f) = max_x bs_x(f)$

Note that i f f is constant, $s(f) = bs(f) = 0$.

Degree of a Representing Polynomial

Definition

A polynomial $p : \boldsymbol{R}^n \to \boldsymbol{R}$ represents f if $p(x) = f(x) \forall x \in \{0,1\}^n$. The degree $deg(f)$ of f is the degree of the multilinear polynomial that represents f.

For example, $deg(AND_n) = n$ because the representing polynomial is the monomial $x_1 ... x_n$. The degree $deg(f)$ may be larger than $s(f), bs(f)$ and $C(f)$.

Degree of an Approximating Polynomial

Definition

A polynomial $p : \boldsymbol{R}^n \to \boldsymbol{R}$ approximates f if $|p(x) - f(x)| \leq \frac{1}{3} \forall x \in \{0,1\}^n$.
The approximate degree $deg(f)$ of f is the minimum degree among all the multilinear polynomial that approximate f.

For example, $\frac{2}{3}x_1 + \frac{2}{3}x_2$ approximates OR_2 so, $approxdeg(OR_2) = 1$, while $deg(OR_2) = 2$.

Randomness

A randomized algorithm can make random queries to the input x, and must still compute $f(x)$.

It will basically be a probability distribution over deterministic decision trees and which deterministic algorithm (tree) we are going to run will depends on the random seed of the randomized algorithm.

$$R = 0.1 \times (D_1) + 0.9 \times (D_2)$$

Randomness

In this case, we have two notions for applicable complexity measures.

Definition

$\text{height}(R) = \max \text{height}(D) \;\; \forall D \in \text{support}(R)$

Randomness

The second one is as follows.

Here, we are still considering the worst-case input, but we average out over the internal randomness of R when measuring the query cost on that input. In the former case, we take the worst case over both inputs and randomness.

Definition

$\text{cost}(R) = E[height(R, x)]$

Randomness

If R is a randomized decision tree, we let $R(x)$ denote the random variable we get by picking a decision tree according to R (the distribution) and running it on x.

We say R computes f to error ϵ if $Pr[R(x) \neq f(x)] \leq \epsilon$ for all x.

Randomness

If we relax our cost measure but not the correctness measure, we get a called zero-error randomized query complexity, denoted $R_0(f)$.

Zero-error randomized algorithms (also called Las Vegas algorithms) always output the right answer, and for all inputs they terminate quickly on average, but they may use a very large number of queries if you get unlucky.

Randomness

Alternatively, we can relax the correctness measure but not the cost measure.

Thus, we have:

- $R_0(f) = \min \text{cost}(R)$ where R computes f to error 0
- $R(f) = \min \text{height}(R)$ where R computes f to error $1/3$

Randomness

Thus, given $\epsilon, \epsilon' < 1/2$ we have:

Theorem

$R_\epsilon(f) = \Theta(R_{\epsilon'}(f))$

Further, we have $R_0(f) \leq D(f)$.

Intro to Quantum Systems

Quantum Theory can be described as an extension of Classical Probability.

Just as Classical Probability Theory, Quantum Mechanics can be fundamentally formulated by pure mathematical constructs alone without any particular appeal to experiment.

It is what we get upon conserving the L_2 norm rather than the L_1 norm (as in Classical Probability) along with addition of the continuity axiom and the idea of measurement.

Thus, the axiomatic formulation of Quantum Mechanics can be stated as follows.

- The state of a system encodes probability of outcomes in a vector, say $[\alpha_1, \alpha_2, \ldots \alpha_n]$, such that $\sum_{\forall i} |\alpha_i|^2 = 1$, or that the L_2 norm is preserved.
- There exists a continuous reversible transformation on a system between any two pure states of that system. This is called the axiom of continuity.
- Measurement in a standard basis results in a collapse of the state to whatever outcome is obtained. The outcome is governed by the probability distribution.

Intro to Quantum Systems

The unit of computation is the qubit, which is a superposition (linear combination) of the 2 classical bits. That is, an $m-$qubit state $|\phi\rangle$ is a superposition of all classical $m-$bit strings:

$$|\phi\rangle = \sum_{i \in \{0,1\}^m} a_i |i\rangle$$

where a_i is a complex number, the amplitude of the basis state $|i\rangle$.

Measuring a $m-$qubit register, we see the basis state with only a probability of amplitude squared, collapsing $|\phi\rangle$, and all information will be lost.

Applying a unitary transform, a new state can be obtained, these transformations are called quantum gates.

Quantum Query Complexity

Aim: We need to define quantum query algorithms, which are capable of making queries to the input x in superposition.

- Since Quantum Computing is reversible, we will need a reversible notion of queries.
- $(i, b) \rightarrow (i, b \oplus x_i)$ where $b \in \{0, 1\}$
- $U^x : |i\rangle |b\rangle \rightarrow |i\rangle |b \oplus x_i\rangle$
- A quantum algorithm can call this map on a superposition of single basis vectors $|i\rangle |b\rangle$, which is an arbitrary vector on this space whose 2-norm still equals 1.

Quantum Query Complexity

- After each query, we would like to allow the quantum algorithm to select the next query in an arbitrary way.
- To do so, we give the algorithm a work space of arbitrary size.
- That is, let W be a set representing the possible basis states of the work space. Then the quantum algorithm will act on the Hilbert space with basis states corresponding to $W \times [n] \times 0, 1$.
- The algorithm will alternate between applying an arbitrary transformation on this state which is independent of x (that is, an arbitrary unitary matrix which does not depend on the input), and applying the unitary U^x implementing the query specified by the query register.
- The unitary U^x will be extended to act as identity on the work space register.

Quantum Query Complexity

Theorem

The action of the quantum algorithm on x will be $U_T U^x U_{T-1} U^x \dots U_1 U^x U_0 |\psi\rangle$ where $|\psi\rangle$ is some fixed initial state.

- T is the number of queries the quantum algorithm makes, and that the quantum algorithm itself is specified by the matrices $U_0, U_1, \dots, U_T$.
- The unitary U_0 can transform the initial state to an arbitary state that does not depend on x.
- Thus, we can assume $|\psi\rangle = |0\rangle_W |1\rangle_I |0\rangle_B$ where we use the subscripts W , I, and B to denote the different registers, namely, work register, index register, and query-answer register.

Quantum Query Complexity

Actually, we will add yet another register, which should be viewed as a special part of the work register. This new register will be the output register, and has basis states in $\{0, 1\}$ if the quantum algorithm returns Boolean values. Hence, we will actually set $|\psi\rangle = |0\rangle_O |0\rangle_W |1\rangle_I |0\rangle_B$.

- Once the algorithm terminates, the output $Q(x)$ of the algorithm Q on the input x will be the random variable we get by measuring the output register in the final state.
- $\exists\ p, q$ such that $p + q = 1$ and we set the random variable $Q(x)$ to be 0 with probability p and 1 with probability q.

Quantum Decision Trees

As defined above, we now have a T-query algorithm. This serves as our Quantum Decision Tree model.

We say that a quantum decision tree computes f exactly if the output equals $f(x)$ with probability 1 $\forall x \in \{0,1\}^n$. The tree computes f with bounded error if the output equals $f(x)$ with probability atleast $2/3$.

$Q_E(f)$ denotes the number of queries (min number of T) of an optimal quantum decision tree (T-query quantum algorithm) that computes f exactly ($\epsilon = 0$), $Q_2(f)$ is the number of queries of an optimal quantum decision tree that computes f with bounded-error.

For now, we will not define expected-cost quantum algorithms, and so there will be no quantum analogue for $R_0(f)$ or $R(f)$.

Application to Deterministic Decision Tree Complexity

Lower Bounds:

Theorem

$bs(f) \leq D(f)$

Theorem

$deg(f) \leq D(f)$

Application to Deterministic Decision Tree Complexity

Upper Bounds:

Theorem

$C^{(1)}(f)bs(f) \geq D(f)$

Theorem

$C^{(1)}(f)deg(f) \geq D(f)$

Application to Randomized Decision Tree Complexity

Theorem

$approxdeg(f) \leq R_2(f)$

Proof: For a randomized decision tree of depth $R_2(f)$, viewed as a probability distribution μ over different deterministic decision trees T, we can write each T as a $0/1$ valued polynomial af degree at most $R_2(f)$. If we define $p = \sum_T \mu(T) p_T$, then it is easy to see that p gives the acceptance property of R, successfully approximating f.

Application to Randomized Decision Tree Complexity

Theorem

$bs(f) \leq 3R_2(f)$

Proof: Considering an algo with $R_2(f)$ queries and input x that achieves the block sensitivity, for every set S, such that $F(x) \neq f(x^S)$, the probability that a variable in S is set is $\geq 1/3$. The total expected number of queries of input should be at least $bs(f)/3$.

Application to Quantum Decision Tree Complexity

Theorem

$deg(f) \leq 2Q_E(f)$

Proof: For an exact quantum algorithm f with $Q_E(f)$ queries, if S is the set of basis states corresponding to 1-output, then acceptance probability is $P(x) = \sum_{k \in S} |a_k(x)|^2$. Since, a_k are polynomials of degree $\leq Q_E(f)$, so $P(x)$ is a poly of degree $\leq 2Q_E(f)$, while, since P represents f it has degree $deg(f)$.

Similarly,

Theorem

$approxdeg(f) \leq 2Q_2(f)$