Average case Complexity

Samaris Michalis

February 2, 2015

introduction

So far we only studied the complexity of algorithms that solve computational task on every possible input; that is, worst-case complexity.

introduction

So far we only studied the complexity of algorithms that solve computational task on every possible input; that is, worst-case complexity.
One frequent objection to this whole framework is that practitioners are only interested in instances of the problem that arise "in practice".

So far we only studied the complexity of algorithms that solve computational task on every possible input; that is, worst-case complexity.
One frequent objection to this whole framework is that practitioners are only interested in instances of the problem that arise "in practice".
Algorithm designers have tried to formalize this in various ways and to design efficient algorithm that work for "many" or "most" of these instances, this body of work is known variously as average-case analysis.One way to formalize "average" instances is that is generated randomly.

introduction

So far we only studied the complexity of algorithms that solve computational task on every possible input; that is, worst-case complexity.
One frequent objection to this whole framework is that practitioners are only interested in instances of the problem that arise "in practice".
Algorithm designers have tried to formalize this in various ways and to design efficient algorithm that work for "many" or "most" of these instances, this body of work is known variously as average-case analysis.One way to formalize "average" instances is that is generated randomly.
The question arises whether we can come up with a theory analogous to **NP** completeness for average-case complexity, and to identify problems that are "hardest" or "complete" with respect to some appropriate notion of reducibility.

distributional problem

What is distribution?

distributional problem

What is distribution?
Examples:
1. $G_{n,p}$ be the distribution over n-vertex graph where each edge is chosen to appear in the graph independently with probability p.The time to produce an instance was polynomial in the instant size.
2.Random 3SAT: A random 3CNF formula on n literals and m clauses can be by choosing each clause as the OR of three random literals.

What is distribution?
Examples:
1. $G_{n,p}$ be the distribution over n-vertex graph where each edge is chosen to appear in the graph independently with probability p.The time to produce an instance was polynomial in the instant size.
2.Random 3SAT: A random 3CNF formula on n literals and m clauses can be by choosing each clause as the OR of three random literals.

Definition

A distributional problem is a pair $\langle L, D\rangle$ where $L \subseteq \{0,1\}^*$ is a language and $D = \{D_n\}$ is a sequence of distributions, with D_n being an distribution over $\{0,1\}^n$.

The class distP

Our next step is to define the class dist**P**,the average-case analog of **P**
For every algorithm A and input x, let $time_A(x)$ denote the number of steps A takes on input x.

The class distP

Our next step is to define the class dist**P**,the average-case analog of **P**

For every algorithm A and input x, let $time_A(x)$ denote the number of steps A takes on input x.

A natural candidate definition is to say that $\langle L, D\rangle$ is solvable in polynomial time on the average if there is an algorithm A such that $A(x) = L(x)$ for every x and a polynomial p that for every n, $E_{x \in_R D_n}[time_A(x)] \leq p(n)$.

The class distP

Our next step is to define the class dist**P**,the average-case analog of **P**

For every algorithm A and input x, let $time_A(x)$ denote the number of steps A takes on input x.

A natural candidate definition is to say that $\langle L, D \rangle$ is solvable in polynomial time on the average if there is an algorithm A such that $A(x) = L(x)$ for every x and a polynomial p that for every n, $E_{x \in_R D_n}[time_A(x)] \leq p(n)$.

Why NOT?

The class distP

Our next step is to define the class dist**P**,the average-case analog of **P**
For every algorithm A and input x, let $time_A(x)$ denote the number of steps A takes on input x.
A natural candidate definition is to say that $\langle L, D \rangle$ is solvable in polynomial time on the average if there is an algorithm A such that $A(x) = L(x)$ for every x and a polynomial p that for every n, $E_{x \in_R D_n}[time_A(x)] \leq p(n)$.
Why NOT?
If we change the model of a computation to a different model(for example change from multiple tape TM to one-tape TM), then a polynomial-time algorithm can suddenly turn into an exponential-time algorithm,as demonstrades by the following simple claim.

The class distP

Claim: There is an algorithm A such that for every n we have $E_{x \in_R \{0,1\}^n}[time_A(x)] \leq n+1$ but $E_{x \in_R \{0,1\}^n}[{time_A}^2(x)] \geq 2^n$.

The class distP

Claim: There is an algorithm A such that for every n we have $E_{x \in_R \{0,1\}^n}[time_A(x)] \leq n + 1$ but $E_{x \in_R \{0,1\}^n}[{time_A}^2(x)] \geq 2^n$.

Definition

A distributional problem $\langle L, D \rangle$ is in dist**P** if there is an algortihm A for L and constants C and $c > 0$ such that for every n

$$E_{x \in_R D_n}[\frac{time_A(x)^c}{n}] \leq C$$

Notice that $P \subseteq distP$.

"real-life distributions"

Polynomial time computable (or **P***-computable) distributions*.
Such distributions have an associated deterministic polynomial time machine that, given input $x \in \{0,1\}^n$, can compute the *cumulative probability* $\mu_{D_n}(x)$, where

$$\mu_{D_n}(x) = \sum_{y \in \{0,1\}^n : y \leq x} Pr_{D_n}[y]$$

Here $Pr_{D_n}(y)$ denotes the probability assigned to string y and $y \leq x$ means y either precedes x in lexicographic order or is equal to x.
Denoting the lexicographic prodecessor of x by $x - 1$, we have

$$Pr_{D_n}[y] = \mu_{D_n}(x) - \mu_{D_n}(x-1)$$

which shows that if μ_{D_n} is computable in polynomial time, then so is $Pr_{D_n}[x]$

"real-life distributions"

Polynomial time samplable(or **P***-samplable distributions)*
These distributions have an associated probabilistic polynomial time machine that can produce samples from the distribution. Specifically, we say that $D = \{D_n\}$ is **P**-samplable if there is a polynomial p and a probabilistic $p(n)$-time algorithm S such that for every n, the random variables $A(1^n)$ and D_n are identically distributed.

"real-life distributions"

Polynomial time samplable(or **P***-samplable distributions)*
These distributions have an associated probabilistic polynomial time machine that can produce samples from the distribution. Specifically, we say that $D = \{D_n\}$ is **P**-samplable if there is a polynomial p and a probabilistic $p(n)$-time algorithm S such that for every n, the random variables $A(1^n)$ and D_n are identically distributed.
If a distribution is **P**-computable then it is **P**-samplable.

distNP and average case reduction

Definition

A distributional problem $\langle L, D\rangle$ is in dist**NP** if $L \in NP$ and D is **P**-computable

distNP and average case reduction

Definition

A distributional problem $\langle L, D \rangle$ is in dist**NP** if $L \in NP$ and D is **P**-computable

Definition

We say that a distributional problem $\langle L, D \rangle$ *average-case reduces* to a distributional problem $\langle L', D' \rangle$, denoted by $\langle L, D \rangle \leq_p \langle L', D' \rangle$, if there is a polynomial time computable f an polynomials $p, q : \mathbb{N} \to \mathbb{N}$ satisfying

1.(*Correctness*) For every $x \in \{0,1\}^*, x \in L \Leftrightarrow f(x) \in L'$

2.(*Length regularity*) For every $x \in \{0,1\}^*, |f(x)| = p(|x|)$

3.(*Domination*) For every $n \in \mathbb{N}$ and $y \in \{0,1\}^{p(n)}, Pr[y = f(D_n)] \leq q(n) Pr[y = D'_{p(n)}]$

Theorem

If $\langle L, D\rangle \leq_p \langle L', D'\rangle$ *and* $\langle L', D'\rangle \in dist\mathbf{P}$ *then* $\langle L, D\rangle \in dist\mathbf{P}$

Proof.

A' is polynomial algorithm for $\langle L', D'\rangle$.
There are constants $C, c > 0$ that for every m

$$E[\frac{{time_{A'} D'_m}^c}{m}] \leq C$$

Algorithm A for L: Given input x, compute $f(x)$ and then output $A'(f(x))$.
Since A decides L, it is left to show that A runs on polynomial time on the average with respect to D.
Assume that for every x, $|f(x)| = |x|^d$ and that computing f in length n inputs is faster than running time of A' on length n^d inputs and hence $time_A(x) \leq 2time_{A'}(f(x))$ □

Proof.

Using definition of A,our assumption and domination of reduction:

$$E[\frac{(\frac{1}{2}time_A(D_n))^c}{q(n)n^d}] \leq \sum_{y\in\{0,1\}^{n^d}} Pr[y = f(D_n)]\frac{time_{A'}(y)^c}{q(n)n^d} \leq$$

$$\sum_{y\in\{0,1\}^{n^d}} Pr[y = f(D'_{n^d})]\frac{time_{A'}(y)^c}{n^d} = E[\frac{(time_{A'}(D'_{n^d}))^c}{n^d}] \leq C$$

□

A complete proble for distNP

We say that $\langle L', D' \rangle$ is *dist***NP***-complete* if $\langle L', D' \rangle$ is in *dist***NP** and $\langle L, D \rangle \leq_p \langle L', D' \rangle$ for every $\langle L, D \rangle \in$ *dist***NP**.

A complete proble for distNP

We say that $\langle L', D' \rangle$ is *dist***NP***-complete* if $\langle L', D' \rangle$ is in *dist***NP** and $\langle L, D \rangle \leq_p \langle L', D' \rangle$ for every $\langle L, D \rangle \in dist\mathbf{NP}$.
We give a useful lemma for the proof of the theorem of existence

Lemma

*Let $D = \{D_n\}$ be a **P**-computable distribution. Then there is a polynomial-time computable function $g : \{0,1\}^* \to \{0,1\}^*$ such that*

1. *g is one-to-one*
2. *For every $x \in \{0,1\}^*$, $|g(x)| = |x| + 2$*
3. *For every string $y \in \{0,1\}^m$, $Pr[y = g(D_m)] \leq 2^{-m+1}$*

Theorem

Let V contain all tuples $\langle M, x, 1^t \rangle$ where there exists a string $y \in \{0,1\}^l$ such that NDTM M outputs 1 on input x within t steps.
For every n we let U_n be the following distribution on length n tuples $\langle M, x, 1^t \rangle$: the string representing M is chosen at random from all strings of length at most $\log n$, t is chosen at random in the set $\{0, ..., n - |M|\}$ and x is chosen at random from $\{0,1\}^{n-t-|M|}$.
*This distribution is polynomial-time computable. Then $\langle V, U \rangle$ is dist***NP***-complete*

Proof.

Let $\langle L, D\rangle$ be in $dist\mathbf{NP}$ and let M be the polynomial-time NDTM M accepting L.
Define the following NDTM M': On input y, guess x such that $y = g(x)$ and execute $M(x)$
Let p be the polynomial running time of M'.

Proof.

Let $\langle L, D\rangle$ be in *dist***NP** and let M be the polynomial-time NDTM M accepting L.
Define the following NDTM M': On input y, guess x such that $y = g(x)$ and execute $M(x)$
Let p be the polynomial running time of M'.
To reduce $\langle L, D\rangle$ to $\langle V, U\rangle$, we simply map every string x into the tuple $\langle M', g(x), 1^k\rangle$ where $k = p(n) + \log n + n - |M'| - |g(x)|$.

Proof.

Let $\langle L, D \rangle$ be in *dist*$\mathbf{NP}$ and let M be the polynomial-time NDTM M accepting L.
Define the following NDTM M': On input y, guess x such that $y = g(x)$ and execute $M(x)$
Let p be the polynomial running time of M'.
To reduce $\langle L, D \rangle$ to $\langle V, U \rangle$, we simply map every string x into the tuple $\langle M', g(x), 1^k \rangle$ where $k = p(n) + \log n + n - |M'| - |g(x)|$.
The reduction obviously satisfies the **length regularity** requirement.

Proof.

Let $\langle L, D \rangle$ be in *dist*$\mathbf{NP}$ and let M be the polynomial-time NDTM M accepting L.
Define the following NDTM M': On input y, guess x such that $y = g(x)$ and execute $M(x)$
Let p be the polynomial running time of M'.
To reduce $\langle L, D \rangle$ to $\langle V, U \rangle$, we simply map every string x into the tuple $\langle M', g(x), 1^k \rangle$ where $k = p(n) + \log n + n - |M'| - |g(x)|$.
The reduction obviously satisfies the **length regularity** requirement.
Because function g is one-to-one by the previous lemma,reduction satisfies also the **correctness** condition.

Proof.

Let $\langle L, D\rangle$ be in *dist***NP** and let M be the polynomial-time NDTM M accepting L.
Define the following NDTM M': On input y, guess x such that $y = g(x)$ and execute $M(x)$
Let p be the polynomial running time of M'.
To reduce $\langle L, D\rangle$ to $\langle V, U\rangle$, we simply map every string x into the tuple $\langle M', g(x), 1^k\rangle$ where $k = p(n) + \log n + n - |M'| - |g(x)|$.
The reduction obviously satisfies the **length regularity** requirement.
Because function g is one-to-one by the previous lemma,reduction satisfies also the **correctness** condition.
Also by the previous lemma the probability that a length m tuple $\langle M', y, 1^t\rangle$ is obtained by the reduction, is at most $2^{-|y|+1}$.This tuple is obtained with probability at least $2^{-\log n}2^{-|y|}\frac{1}{m}$ by U_m.Hence also the **domination** condition is satisfied □

Thank you.