Notes for Lecture 23

In these notes we introduce Levin's theory of average-case complexity.

This theory is still in its infancy: in these notes we will introduce the notion of "distributional problems," discuss various formalizations of the notion of "algorithms that are efficient on average," introduce a reducibility that preserves efficient average-case solvability, and finally prove that there is a problem that is complete for the distributional version of **NP** under such reductions. It is still an open question how to apply this theory to the study of standard NP problems and natural distributions of inputs.

1 Distributional Problems

Definition 1 (Distributional Problem) *A* distributional problem is a pair $\langle L, \mu \rangle$, where L is a decision problem and $\mu = (\mu_1, \ldots, \mu_n, \ldots)$ is an ensemble of distributions such that μ_n is a distribution over $\{0,1\}^n$.

We will restrict to the study of distributional problems where μ is "polynomial-time computable."

Definition 2 (Polynomial Time Computable Distributions) *Let μ be an ensemble of distributions, and for $x \in \{0,1\}^n$, let*

$$\mu(x) := \Pr_{y \sim \mu_n} [y \leq x] \ . \tag{1}$$

where '$\leq$' denotes lexicographic ordering. Then μ is a polynomial time computable distribution if and only $\mu(x)$ is computable in $\text{poly}(|x|)$ *time.*

Clearly this notion is at least as strong as the requirement that $\Pr_{y \sim \mu_n}[y = x]$ be computable in polynomial time, because

$$\Pr[y = x] = \mu'(x) := \mu(x) - \mu(x-1), \tag{2}$$

$x - 1$ being the lexicographic predecessor of x. Indeed one can show that, under reasonable assumptions, there exist distributions that are efficiently computable in the second sense but not polynomial-time computable in our sense.

2 DistNP

We define the complexity class

$$\textbf{DistNP} := \{\langle L, \mu \rangle : L \in NP, \mu \text{ polynomial-time computable}\}. \tag{3}$$

There are at least two good reasons for looking only at polynomial-time computable distributions.

1. One can show that there exists a distribution μ such that every problem is as hard on average under μ as it is in the worst case. Therefore, unless we place some computational restriction on μ, the average-case theory is identical to the worst-case one.

   This is undesirable, because we would like to prove a completeness result stating that a specific distributional problem is hard-on-average unless *all* problems in **DistNP** are easy-on-average. So, if we defined **DistNP** in terms of all possible distributions, our completeness result would have to state that a specific distributional problem is hard-on-average unless $P = NP$, and there is some evidence [FF93, BT03] that the reductions needed to establish such a "worst-case-to-average-case" completeness result cannot exist.

2. Someone, somewhere, had to generate the instances we are trying to solve. If we place computational restrictions on ourselves, then it seems reasonable also to place restrictions on whoever/whatever generated the instances.

It should be clear that we need a whole *class* of distributions to do reductions; that is, we can't just parameterize a complexity class by a single distribution. This is because a problem can have more than one natural distribution; it's not always obvious what to take as the 'uniform distribution.'

3 Reductions

Definition 3 (Reduction) *We say that a distributional problem $\langle L_1, \mu_1 \rangle$* reduces *to a distributional problem $\langle L_2, \mu_2 \rangle$ (in symbols, $\langle L_1, \mu_1 \rangle \leq \langle L_2, \mu_2 \rangle$) if there exists a polynomial-time computable function f such that:*

1. *f* **is a reduction.** *$x \in L_1$ iff $f(x) \in L_2$.*
2. *$f(\mu_1)$* **dominates** *μ_2. For every n and m, and for every $y \in \{0,1\}^m$,*

$$\sum_{x:f(x)=y} \mu\prime_{1,n}(x) \leq \text{poly}(n)\, \mu\prime_{2,m}(y). \tag{4}$$

In Levin's original definition, there is a requirement that for every x we have $|f(x)| \geq |x|^{\Omega(1)}$, and the domination condition requires

$$\sum_{x:f(x)=y} \mu\prime_{1,n}(x) \leq \text{poly}(|y|)\, \mu\prime_{2,m}(y),$$

so our definition is more general.

The first condition is the standard condition of many-to-one reductions in complexity theory: it ensures that an algorithm that is always correct for L_2 can be converted into an algorithm that is always correct for L_1.

To motivate the domination condition, consider that we want reductions to preserve the existence of algorithms that are efficient on average. Suppose that we have an algorithm A_2 for problem L_2 such that, when we pick y according to distribution μ_2, $A(y)$ is efficient

on average; if we want to solve L_1 under distribution μ_1, then, starting from an input x distributed according to μ_1, we compute $f(x)$ and then apply algorithm A_2 to $f(x)$. This will certainly be correct, but what about the running time? Intuitively, it could be the case that A_2 is very slow on some inputs, but such inputs are unlikely to be sampled according to distribution μ_2; the domination condition ensures us that such inputs are also unlikely to be sampled when we sample x according to μ_1 and then consider $f(x)$.

4 Polynomial-Time on Average

Given a problem $\langle L, \mu \rangle$ and an algorithm A that runs in time $t(x)$ on input x, what does it mean to say that A solves $\langle L, \mu \rangle$ in polynomial time on average?

A first attempt could be to say that an algorithm is polynomial on average if its expected running time is polynomial.

$$\mathbb{E}_{x \sim \mu_n}[t(x)] \leq O(n^c)$$

This definition is quite appealing, but is subject to the fatal flaw of not being *robust*, in that: (1) reductions do not preserve this definition of polynomial solvability on average and (2) the definition is sensitive to trivial representation changes such as replacing a matrix representation of a graph by an adjacency list.

To see why these problems arise, let μ be the uniform distribution, and let

$$t(x) = 2^n \text{ if } x = \overrightarrow{0}, \, t(x) = n^2 \text{ otherwise.} \tag{5}$$

The average running time is about n^2. But suppose now that n is replaced by $2n$ (because of a change in representation, or because of the application of a reduction), then

$$t(x) = 2^{2n} \text{ if } x = \overrightarrow{0}, \, t(x) = 4 \cdot n^2 \text{ otherwise.} \tag{6}$$

Similarly, if $t(x)$ is replaced by $t^2(x)$, the average running time becomes exponential.

We now come to a satisfying definition.

Definition 4 (Polynomial on average) *Suppose A is an algorithm for a distributional problem $\langle L, \mu \rangle$ that runs in time $t(x)$ on input x. We say that A has polynomial running time on average if there are constants $\delta > 0$ and c such that*

$$\mathbb{E}_{x \sim \mu_n}\left[t(x)^\delta\right] \leq O(n^c)$$

Notice, first, that this definition is satisfied by any algorithm that runs in worst-case polynomial time or in expected polynomial time (take $\delta = 1$). More interestingly, suppose $t()$ is a time bound for which the above definition is satisfied; then an algorithm whose running time is $t'(x) = t(x)^2$ also satisfies the definition, unlike the case of the previous definition. In fact we have the following result, whose proof would be non-trivial at this time. We shall prove it later after finding an alternative characterization (due to Impagliazzo) of algorithm that in time polynomial on average.

Theorem 1 *If $\langle L_1, \mu_1 \rangle \leq \langle L_2, \mu_2 \rangle$ and $\langle L_2, \mu_2 \rangle$ admits an algorithm that is polynomial on average, then $\langle L_1, \mu_1 \rangle$ also admits an algorithm that is polynomial on average.*

There is an additional interesting property of the definition of polynomial on average: there is a high probability that the algorithm runs in polynomial time.

Suppose that

$$\mathbb{E}_{x \sim \mu_n} \left[t\left(x\right)^{\delta} \right] = O\left(n^{c}\right). \tag{7}$$

and that we wish to compute $\Pr[t(x) \geq k \cdot |x|^{c/\delta}]$. Such a probability is clearly the same as

$$\Pr\left[t(x)^{\delta} \geq k^{\delta} |x|^{c} \right]$$

and by Markov's inequality this is at most $O(1/k^{\delta})$, which can be made fairly small by picking k large enough. Since the algorithm runs in time at most $kn^{c/\delta}$ for a subset of inputs having probability $1 - O(k^{-\delta})$, we see that our definition gives a smooth quantitative tradeoff for how much time we need to solve an increasing fraction of inputs.

This observation motivates the following definition.

Definition 5 (Errorless Heuristic Scheme) *An errorless heuristic scheme for a distributional problem $\langle L, \mu \rangle$ is an algorithm A that, on input x and ϵ, runs in time polynomial in $|x|$ and in $1/\epsilon$, and either correctly decides whether $x \in L$, or outputs "FAIL." Furthermore, for every n and ϵ*

$$\mathbf{Pr}_{x \sim \mu_n}[A(x, \epsilon) = FAIL] \leq \epsilon$$

Errorless heuristic schemes capture polynomial time on average solvability.

Theorem 2 *A distributional problem $\langle L, \mu \rangle$ admits a polynomial time on average algorithms if and only if it allows an errorless heuristic scheme.*

Proof: We have already established that an algorithm that is polynomial on average implies an errorless heuristic scheme: if the algorithm runs in time polynomial on average with parameters δ and c, then given inputs x and ϵ we run the algorithm for at most $O(\epsilon^{-1/\delta} \cdot |x|^{c/\delta})$ steps, and output FAIL if the algorithm does not halt within such a number of steps. By the Markov inequality calculations we performed above, the algorithm fails with probability at most ϵ when x is sampled from μ_n.

For the other direction, if $A(\cdot, \cdot)$ is an errorless heuristic scheme, we compute $A(x, 1/2)$, and output the result, unless the computation fails. If it fails, we compute $A(x, 1/4)$ and output the result, if different from FAIL, otherwise we go on to compute $A(x, 1/8)$ and so on. Eventually, the computation will not fail, because $A(x, \epsilon)$ where ϵ is smaller than the probability of x under μ_n must output the correct answer. If x is an input that requires k rounds of the above process in order to find the answer, then the running time $t(x)$ for x is $|x|^{a} \cdot 2^{kb}$ for some exponents a and b. Let us choose $\delta = 2b$, so that $t^{\delta}(x) \leq |x|^{\delta a} \cdot 2^{k/2}$. Notice that we have $\mathbf{Pr}[t^{\delta}(x) \geq |x|^{\delta a} 2^{k/2}] \leq 2^{-k}$. So we have

$$\begin{aligned}
\mathbb{E}_{x\in\mu_n}[t^\delta(x)] &\leq |x|^{\delta a}\mathbb{E}_{x\sim\mu_n}\left[|x|^{\delta a}\cdot t^\delta(x)\right] \\
&= |x|^{\delta a}\sum_t \mathbf{Pr}_{x\sim\mu_n}\left[1|x|^{\delta a}t^\delta(x)\geq t\right] \\
&= |x|^{\delta a}\sum_t \mathbf{Pr}_{x\sim\mu_n}[t^\delta(x)\geq |x|^{\delta a}\cdot t] \\
&\leq |x|^{\delta a}\sum_t \frac{1}{t^2} \\
&= O(|x|^{\delta a})
\end{aligned}$$

□

Given this characterization, we can now prove Theorem 1.

PROOF:[Of Theorem 1] Suppose that $\langle A,\mu_A\rangle \leq \langle angle B,\mu_B\rangle$ via the reduction f, and that $\langle angle B,\mu_B\rangle$ admits a polynomial time errorless heuristic scheme S; we shall show that $\langle angle A,\mu_A\rangle$ admits an errorless heuristic scheme too.

On input x sampled from μ_A and a parameter ϵ, we simply run $S(f(x),\gamma)$ for an appropriately small γ that we will determine later. (But such that $\gamma^{-1} = poly(n\cdot\epsilon^{-1})$, so that our heuristic scheme for $\langle A,\mu_A\rangle$ runs in polynomial time.) If the computation of $S(f(x),\gamma)$ does not fail, then we correctly determine whether $x\in A$. We need to prove that the probability of failing is at most ϵ, over the random choices of x from μ_A. We know that for every m,

$$\mathbf{Pr}_{y\sim\mu_{B,m}}[S(y,\gamma)\text{ fails }]\leq\gamma$$

and that, for every m, and every $b\in\{0,1\}^m$,

$$\mathbf{Pr}_{x\sim\{0,1\}^n}[f(x)=b]\leq poly(n)\cdot\mathbf{Pr}_{y\sim\mu_{B,m}}[y=b]$$

Finally, we note that $|f(x)|\leq poly(|x|)$, because f is computable in polynomial time. So we can see that

$$\begin{aligned}
\mathbf{Pr}_{x\sim\{0,1\}^n}[S(f(x),\gamma)\text{ fails }] &= \sum_{m=1}^{poly(n)}\mathbf{Pr}_{x\sim\{0,1\}^n}[S(f(x),\gamma)\text{ fails }\wedge|f(x)|=m] \\
&\leq \sum_{m=1}^{poly(n)} poly(n)\cdot\mathbf{Pr}_{y\sim\{0,1\}^m}[S(y,\gamma)\text{ fails }] \\
&\leq poly(n)\cdot\gamma \\
&\leq \epsilon
\end{aligned}$$

provided we set γ small enough. □

5 Existence of Complete Problems

We now show that there exists a problem (albeit an artificial one) complete for **DistNP**. Let the inputs have the form $\langle M, x, 1^t \rangle$, where M is an encoding of a non-deterministic Turing machine and 1^t is a sequence of t ones. Then we define the following "bounded halting" problem BH.

- Decide whether there is an accpeting path of $M(x)$ that takes at most t steps.

Define a uniform distribution U over the $\langle M, x, 1^t \rangle$ as follows:

$$U'_n\left(\langle M, x, 1^t \rangle\right) := \frac{1}{2^{|M|}} \cdot \frac{1}{2^{|x|}} \cdot \frac{1}{n}. \tag{8}$$

The intuition is that first we pick t at random (uniformly from 1 to n), and then we fill up the first $n - t$ entries of the string at random. In a correctly formatted string, M and x are encoded in a prefix-free way, so that if M has length m and x has length ℓ, the length of the string is $O(\log m + \log \ell) + m + \ell + t$. We allow incorrectly formatted strings in our distributions. (Those are all NO instances of the language.) Alternative conventions could be used and could be accommodated with small changes to the argument.

That BH is **NP**-complete follows directly from the definition. Recall the definition of **NP**: we say that $L \in \mathbf{NP}$ if there exists a machine M running in $t = \text{poly}(|x|)$ steps such that $x \in L$ iff at least one of the computational paths of $M(x)$ accepts within time t. Thus, to reduce L to BH we need only map x onto $f(x) := \langle M, x, 1^t \rangle$ where t is a sufficiently large bound.

Such a reduction, however, fails to reduce a generic **DistNP** problem $\langle L, \mu \rangle$ to $\langle BH, U \rangle$.

The trouble is that, because of the domination condition, we can't map x onto $f(x)$ unless $\mu\prime_n(x) \leq \text{poly}(|x|)\, 2^{-|x|}$. We work around this problem by *compressing* x to a shorter string if $\mu\prime_n(x)$ is large. Intuitively, by mapping high-probability strings onto shorter lengths, we make their high probability less conspicuous. The following lemma shows how to do this.

Lemma 3 *Suppose μ is a polynomial-time computable distribution over x. Then there exists a polynomial-time algorithm C such that*

1. *C is injective over $\{0,1\}^n$: for every $x, y \in \{0,1\}^n$, if $x \neq y$ then $C(x) \neq C(y)$.*
2. *$|C(x)| \leq 1 + \min\left\{|x|, \log \frac{1}{\mu\prime_n(x)}\right\}$.*

Proof: If $\mu\prime_n(x) \leq 2^{-|x|}$ then simply let $C(x) = 0x$, that is, 0 concatenated with x. If, on the other hand, $\mu\prime_n(x) > 2^{-|x|}$, then let $C(x) = 1z$. Here z is the longest common prefix of $\mu_n(x)$ and $\mu_n(x-1)$ when both are written out in binary. Since μ_n is computable in polynomial time, so is z. C is injective because only two binary strings s_1 and s_2 can have the longest common prefix z; a third string s_3 sharing z as a prefix must have a longer prefix with either s_1 or s_2. Finally, since $\mu\prime_n(x) \leq 2^{-|z|}$, $|C(x)| \leq 1 + \log \frac{1}{\mu\prime_n(x)}$. □

Now the reduction is to map x onto

$$f(x) := \left\langle \overline{M}, (n, C(x)), 1^{\bar{t}} \right\rangle .$$

Here $\overline{M}$ is a machine that on input n, z accepts if and only if there exists an $x \in \{0,1\}^n$ such that $C(x) = z$ and $M(x)$ accepts. The running time of $\overline{M}$ is $\bar{t}$. Clearly $x \in L$ iff $\overline{M}$ accepts. To show that domination holds, observe that, since the map is injective, we need only show that

$$\mu\prime_n(x) \leq \text{poly}(n) \cdot U'_{|f(x)|}(f(x))$$

and we have $|f(x)| = O(1) + O(\log n) + |C(x)| + \bar{t}(n)$, so that

$$U'_{|f(x)|}(f(x)) = \frac{1}{O(1) + O(\log n) + |C(x)| + \bar{t}} \cdot \frac{1}{O(1)} \cdot \frac{1}{2^{O(\log n + |C(x)|)}} \geq \frac{1}{poly(n)} \mu_n(x)$$

and we are done.

6 Polynomial-Time Samplability

Definition 6 (Samplable distributions) *We say that a distribution μ is* polynomial-time samplable if there exists a probabilistic algorithm A that, on input n, runs in expected $poly(n)$ time and outputs x with probability $\mu\prime(x)$.

Any polynomial-time computable distribution is also polynomial-time samplable, provided that for all x,

$$\mu\prime_{|x|}(x) \geq 2^{-\text{poly}(|x|)} \text{ or } \mu\prime(x) = 0. \tag{9}$$

For a polynomial-time computable μ satisfying the above property, we can indeed construct a sampler A that first chooses a real number r uniformly at random from $[0,1]$, to poly $(|x|)$ bits of precision, and then uses binary search to find the first x such that $\mu(x) \geq r$.

On the other hand, under reasonable assumptions, there are efficiently samplable distributios μ that are not efficiently computable.

In addition to **DistNP**, we can look at the class

$$\langle \mathbf{NP}, \mathbf{P}\text{-samplable} \rangle = \{ \langle L, \mu \rangle : L \in \mathbf{NP}, \mu \text{ polynomial-time samplable} \} . \tag{10}$$

A result due to Impagliazzo and Levin states that if $\langle L, \mu \rangle$ is **DistNP**-complete, then $\langle L, \mu \rangle$ is also complete for the class $\langle \mathbf{NP}, \mathbf{P}\text{-samplable} \rangle$.

This means that the completeness result established in the previous section extends to the class of NP problems with samplable distributions. The completeness, however, is proved under a different notion of reducibility, that preserves heuristic but not average polynomial time algorithms.

7 References

Levin's theory of average-case complexity was introduced in [Lev86]. Ben-David et al. [BDCGL92] prove several basic results about the theory. Impagliazzo and Levin [IL90] show that the theory can be generalized to samplable distributions. Impagliazzo [Imp95] wrote a very clear survey on the subject. Other good reference are a survey paper by Goldreich [Gol97] and a more recent one by Bogdanov and Trevisan [BT06]

References

[BDCGL92] Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. On the theory of average case complexity. *Journal of Computer and System Sciences*, 44(2):193–219, 1992. 8

[BT03] Andrej Bogdanov and Luca Trevisan. On wost-case to average-case reductions for NP problems. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science*, pages 308–317, 2003. 2

[BT06] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Foundations and Trends in Theoretical Computer Science*, 2, 2006. 8

[FF93] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22:994–1005, 1993. 2

[Gol97] Oded Goldreich. Notes on Levin's theory of average-case complexity. Technical Report TR97-058, Electronic Colloquium on Computational Complexity, 1997. 8

[IL90] Russell Impagliazzo and Leonid Levin. No better ways to generate hard NP instances than picking uniformly at random. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, pages 812–821, 1990. 8

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the 10th IEEE Conference on Structure in Complexity Theory*, pages 134–147, 1995. 8

[Lev86] Leonid Levin. Average case complete problems. *SIAM Journal on Computing*, 15(1):285–286, 1986. 8

Exercises

1. For a parameter c, consider the ensemble of distributions $D_c = (D_{n,cn})_{n=1,2,\ldots}$ over instances of 3SAT with n variables generated by picking cn times independently a random a clause out of the $8\binom{n}{3}$ possible clauses that can be constructed from n variables. (Note that the same clause could be picked more than once.)

   (a) Show that an instance from $D_{n,cn}$ is satisfiable with probability at least $(7/8)^{cn}$ and at most $2^n \cdot (7/8)^{cn}$.

   (b) Argue that, using the definition given in this lecture, $\langle 3SAT, D_{15} \rangle$ cannot be reduced to $\langle 3SAT, D_{30} \rangle$.

   [Hint: take a sufficiently large n, and then look at the probability of satisfiable instances of length n under D_{15} and the probability that their image is generated by D_{30}.]