Probabilistic Computation

Lecture 15
Computing with Less Randomness, or with Imperfect Randomness
Soundness Amplification for BPP
Soundness Amplification for BPP

- Repeat $M(x)$ $t$ times and take majority
  - i.e. estimate $\Pr[M(x)=\text{yes}]$ and check if it is $> 1/2$
  - Error only if $|\text{estimate}-\text{real}| \geq \text{gap}/2$

Estimation error goes down exponentially with $t$: Chernoff bound

- $\Pr[|\text{estimate}-\text{real}| \geq \delta/2] \leq 2^{-\Omega(t \cdot \delta^2)}$

- $t = O(n^d/\delta^2)$ enough for $\Pr[\text{error}] \leq 2^{-n^d}$
Randomness Efficient
Soundness Amplification
Randomness Efficient Soundness Amplification

In repeating $t$ times (to reduce error to $2^{-\Omega(t)}$) number of coins used = $t.m$
Randomness Efficient Soundness Amplification

- In repeating $t$ times (to reduce error to $2^{-\Omega(t)}$) number of coins used = $t.m$

- Used independent random tapes to get error $2^{-\Omega(t)}$
Randomness Efficient Soundness Amplification

- In repeating $t$ times (to reduce error to $2^{-\Omega(t)}$) number of coins used $= t.m$

- Used independent random tapes to get error $2^{-\Omega(t)}$

- Can use very dependent tapes and still get error $2^{-\Omega(t)}$! (but with a smaller constant inside $\Omega$)
Randomness Efficient
Soundness Amplification

In repeating $t$ times (to reduce error to $2^{-\Omega(t)}$) number of coins used = $t \cdot m$

Used independent random tapes to get error $2^{-\Omega(t)}$

Can use very dependent tapes and still get error $2^{-\Omega(t)}$ (but with a smaller constant inside $\Omega$)

Random tapes produced using a random walk on an “expander graph”
Randomness Efficient Soundness Amplification

- In repeating $t$ times (to reduce error to $2^{-\Omega(t)}$) number of coins used = $t \cdot m$
- Used independent random tapes to get error $2^{-\Omega(t)}$
- Can use very dependent tapes and still get error $2^{-\Omega(t)}!$ (but with a smaller constant inside $\Omega$)
- Random tapes produced using a random walk on an “expander graph”
- No. of coins used = $m + O(t)$
Randomness Efficient Soundness Amplification
Randomness Efficient Soundness Amplification

Space of all random tapes = \(\{0,1\}^m\). Consider a subset ("yes" set). To estimate its weight \(p\).
Randomness Efficient Soundness Amplification

Space of all random tapes = \{0,1\}^m. Consider a subset ("yes" set). To estimate its weight \( p \).
Randomness Efficient Soundness Amplification

Space of all random tapes = \{0,1\}^m. Consider a subset ("yes" set). To estimate its weight p.
Randomness Efficient
Soundness Amplification

Space of all random tapes = \{0,1\}^m. Consider a subset ("yes" set). To estimate its weight p.

By Chernoff, if p’ is the estimate from t independent samples, then \Pr[|p’-p| > \varepsilon p] < 2^{-\Omega(t \varepsilon^2)}
Space of all random tapes = \{0,1\}^m. Consider a subset ("yes" set). To estimate its weight \(p\).

By Chernoff, if \(p'\) is the estimate from \(t\) independent samples, then \(\Pr[|p' - p| > \varepsilon p] < 2^{-\Omega(t\varepsilon^2)}\)

Random walk: superimpose an "expander graph" on this space. Pick first point at random, and then do random walk of length \(t\) using the graph edges. Estimate \(p' = \text{fraction of yes nodes along the path}\).
Randomness Efficient Soundness Amplification

Space of all random tapes = \{0,1\}^m. Consider a subset ("yes" set). To estimate its weight p.

By Chernoff, if p' is the estimate from t independent samples, then \Pr[|p'-p| > \varepsilon p] < 2^{-\Omega(t\varepsilon^2)}

Random walk: superimpose an "expander graph" on this space. Pick first point at random, and then do random walk of length t using the graph edges. Estimate p' = fraction of yes nodes along the path
Space of all random tapes = \{0,1\}^m. Consider a subset (“yes” set). To estimate its weight \( p \).

By Chernoff, if \( p' \) is the estimate from \( t \) independent samples, then \( \Pr[|p'-p| \geq \epsilon p] < 2^{-\Omega(t \epsilon^2)} \)

Random walk: superimpose an “expander graph” on this space. Pick first point at random, and then do random walk of length \( t \) using the graph edges. Estimate \( p' = \text{fraction of yes nodes along the path} \)
Randomness Efficient Soundness Amplification

Space of all random tapes = \{0,1\}^m. Consider a subset (“yes” set). To estimate its weight \(p\).

By Chernoff, if \(p'\) is the estimate from \(t\) independent samples, then
\[
\Pr[|p' - p| > \varepsilon p] < 2^{-\Omega(t\varepsilon^2)}
\]

Random walk: superimpose an “expander graph” on this space. Pick first point at random, and then do random walk of length \(t\) using the graph edges. Estimate \(p' = \) fraction of yes nodes along the path

Expander’s degree is constant: coins needed = \(m + O(t)\)
Randomness Efficient
Soundness Amplification

Space of all random tapes = \{0,1\}^m. Consider a subset (“yes” set). To estimate its weight \( p \).

By Chernoff, if \( p' \) is the estimate from \( t \) independent samples, then \( \Pr[|p' - p| > \varepsilon p] < 2^{-\Omega(t\varepsilon^2)} \)

Random walk: superimpose an “expander graph” on this space. Pick first point at random, and then do random walk of length \( t \) using the graph edges. Estimate \( p' \) = fraction of yes nodes along the path

Expander’s degree is constant: coins needed = \( m + O(t) \)

Expander “mixing”: \( \Pr[|p' - p| > \varepsilon p] < 2^{-\Omega(t\varepsilon^2)} \) (but with a smaller constant inside \( \Omega \))
Soundness Amplification
Soundness Amplification

- Probabilistic Approximately Correct estimation of Pr[yes]
Soundness Amplification

- Probabilistic Approximately Correct estimation of $\Pr[\text{yes}]$
- Bounded gap: so enough to approximate
Soundness Amplification

- Probabilistic Approximately Correct estimation of $\Pr[\text{yes}]$
  - Bounded gap: so enough to approximate
  - A small probability of error still allowed
Soundness Amplification

- Probabilistic Approximately Correct estimation of $Pr[\text{yes}]$
  - Bounded gap: so enough to approximate
  - A small probability of error still allowed
  - Not “derandomization”
Soundness Amplification

- Probabilistic Approximately Correct estimation of $\Pr[\text{yes}]$
  - Bounded gap: so enough to approximate
  - A small probability of error still allowed
    - Not “derandomization”
  - Trying to minimize amount of randomness used
Soundness Amplification

- Probabilistic Approximately Correct estimation of $\Pr[\text{yes}]$
  - Bounded gap: so enough to approximate
  - A small probability of error still allowed
  - Not “derandomization”
- Trying to minimize amount of randomness used
  - Still need perfectly random bits (fair, independent coin tosses)
Soundness Amplification

- Probabilistic Approximately Correct estimation of Pr[yes]
  - Bounded gap: so enough to approximate
  - A small probability of error still allowed
    - Not “derandomization”
  - Trying to minimize amount of randomness used
    - Still need perfectly random bits (fair, independent coin tosses)
    - Not a realistic assumption on random sources
Soundness Amplification

- Probabilistic Approximately Correct estimation of $\Pr[\text{yes}]$
  - Bounded gap: so enough to approximate
  - A small probability of error still allowed
    - Not "derandomization"
- Trying to minimize amount of randomness used
  - Still need perfectly random bits (fair, independent coin tosses)
    - Not a realistic assumption on random sources
  - Can we work with imperfect random sources?
Philosophical Issues with Randomness/Probability
Philosophical Issues with Randomness/Probability
Imperfect Randomness
Imperfect Randomness

Perfect
Imperfect Randomness

- Perfect
- Fair coin flips
Imperfect Randomness

- Perfect
  - Fair coin flips
- Slightly imperfect
Imperfect Randomness

- Perfect
  - Fair coin flips
- Slightly imperfect
  - Don’t know the exact distribution, but belongs to a known class of distributions with:
Imperfect Randomness

- Perfect
  - Fair coin flips
- Slightly imperfect
  - Don’t know the exact distribution, but belongs to a known class of distributions with:
    - Sufficient unpredictability (entropy)
Imperfect Randomness

- Perfect
  - Fair coin flips
- Slightly imperfect
  - Don’t know the exact distribution, but belongs to a known class of distributions with:
    - Sufficient unpredictability (entropy)
    - Sufficient independence
Imperfect Randomness
Imperfect Randomness

- Bit-wise guarantee
Imperfect Randomness

- Bit-wise guarantee
- von Neumann source
Imperfect Randomness

- Bit-wise guarantee
- von Neumann source

Independent but not fair: Each bit is independent of previous bits, but with a bias. Bias is same for all bits.
Imperfect Randomness

- Bit-wise guarantee
- von Neumann source
  - Independent but not fair: Each bit is independent of previous bits, but with a bias. Bias is same for all bits.
- Santha-Vazirani source
Imperfect Randomness

- Bit-wise guarantee
  - von Neumann source
    - Independent but not fair: Each bit is independent of previous bits, but with a bias. Bias is same for all bits.
  - Santha-Vazirani source
    - Dependent bits of varying bias: Each bit can depend on all previous bits, but \( \Pr[b_i=0], \Pr[b_i=1] \in [1/2-\delta/2, 1/2+\delta/2] \), even conditioned on all previous bits (i.e., sufficiently unpredictable)
Imperfect Randomness

- Bit-wise guarantee
  - von Neumann source
    - Independent but not fair: Each bit is independent of previous bits, but with a bias. Bias is same for all bits.
  - Santha-Vazirani source
    - Dependent bits of varying bias: Each bit can depend on all previous bits, but \( \Pr[b_i=0], \Pr[b_i=1] \in [1/2-\delta/2, 1/2+\delta/2] \), even conditioned on all previous bits (i.e., sufficiently unpredictable)

- Weaker guarantees: e.g. Block source
BPP using imperfect randomness
BPP using imperfect randomness

Small bias (1/m, where m coins in all) SV source is harmless:
BPP using imperfect randomness

- Small bias \((1/m, \text{ where } m \text{ coins in all})\) SV source is harmless:
  - Any string has weight at most \((1/2 + \delta/2)^m\)
BPP using imperfect randomness

- Small bias ($1/m$, where $m$ coins in all) SV source is harmless:
- Any string has weight at most $(1/2 + \delta/2)^m$
BPP using imperfect randomness

- Small bias (1/m, where m coins in all) SV source is harmless:
  - Any string has weight at most (1/2 + \(\delta/2\))^m
  - t strings can have weight at most t.(1/2 + \(\delta/2\))^m

Using bound on conditional probability
BPP using imperfect randomness

Small bias (1/m, where m coins in all) SV source is harmless:

Any string has weight at most \((1/2+\delta/2)^m\)

t strings can have weight at most \(t.(1/2+\delta/2)^m\)

\[ t.(1/2+\delta/2)^m = (t/2^m).(1+\delta)^m \leq (t/2^m).e^{m\delta} \]
BPP using imperfect randomness

- Small bias \(\frac{1}{m}\), where \(m\) coins in all) SV source is harmless:
  - Any string has weight at most \((1/2+\delta/2)^m\)
  - \(t\) strings can have weight at most \(t.(1/2+\delta/2)^m\)
  - \(t.(1/2+\delta/2)^m = (t/2^m).(1+\delta)^m \leq (t/2^m).e^{m\delta}\)

Using bound on conditional probability:
\[(1+x) \leq e^x\]
BPP using imperfect randomness

- Small bias \((1/m, \text{where } m \text{ coins in all})\) SV source is harmless:
  - Any string has weight at most \((1/2+\delta/2)^m\)
  - \(t\) strings can have weight at most \(t.(1/2+\delta/2)^m\)
  - \(t.(1/2+\delta/2)^m = (t/2^m).(1+\delta)^m \leq (t/2^m).e^{m\delta}\)

- Wt. of \(t\) strings from a \(\delta < 1/m\) bias SV source < \(e\). (Wt of \(t\) strings under uniform distribution)
BPP using imperfect randomness

- Small bias (1/m, where m coins in all) SV source is harmless:
  - Any string has weight at most \((1/2+\delta/2)^m\)
  - \(t\) strings can have weight at most \(t.(1/2+\delta/2)^m\)
  - \(t.(1/2+\delta/2)^m = (t/2^m).(1+\delta)^m \leq (t/2^m).e^{m\delta}\)
  - Wt. of \(t\) strings from a \(\delta < 1/m\) bias SV source < \(e\). (Wt of \(t\) strings under uniform distribution)

- If on perfect randomness, \(\Pr[\text{error}] < 1/(e2^n)\), then on imperfect randomnness with bias < 1/m, \(\Pr[\text{error}] < 1/2^n\)
BPP using imperfect randomness
BPP using imperfect randomness

Handling more imperfectness
BPP using imperfect randomness

- Handling more imperfectness
- by pre-processing the randomness
BPP using imperfect randomness

- Handling more imperfectness
  - by pre-processing the randomness
    - Randomness extraction
BPP using imperfect randomness

- Handling more imperfectness
  - by pre-processing the randomness
  - Randomness extraction

- Deterministic Extractor:
BPP using imperfect randomness

- Handling more imperfectness
  - by pre-processing the randomness
  - Randomness extraction

- Deterministic Extractor:
BPP using imperfect randomness

- Handling more imperfectness
- by pre-processing the randomness
- Randomness extraction

Deterministic Extractor:
Handling more imperfectness
by pre-processing the randomness

Randomness extraction

Deterministic Extractor:
Deterministic extractor for von Neumann Sources
Deterministic extractor for von Neumann Sources

Extraction for von Neumann sources
Deterministic extractor for von Neumann Sources

 Extraction for von Neumann sources
Deterministic extractor for von Neumann Sources

- Extraction for von Neumann sources

Case $r_{2i} \; r_{2i+1}$:
- 01: output 0
- 10: output 1
- *: discard
Deterministic extractor for von Neumann Sources

- Extraction for von Neumann sources
- Perfectly random output

Case $r_{2i} r_{2i+1}$:
01: output 0
10: output 1
*: discard
Deterministic extractor for von Neumann Sources

- Extraction for von Neumann sources
- Perfectly random output
- Fewer output bits

Case $r_{2i} \ r_{2i+1}$:
- 01: output 0
- 10: output 1
- *: discard
Deterministic extractor for von Neumann Sources

- Extraction for von Neumann sources
  - Perfectly random output
  - Fewer output bits
  - Running time (per bit): constant number of tries, expected
Deterministic extractor for von Neumann Sources

- Extraction for von Neumann sources
- Perfectly random output
- Fewer output bits
- Running time (per bit): constant number of tries, expected
- Can be generalized to sources which are (hidden) Markov chains

Case $r_{2i} r_{2i+1}$:
- 01: output 0
- 10: output 1
- *: discard
Extractor for SV sources?
Extractor for SV sources?

- No deterministic extractor, for even one bit output
Extractor for SV sources?

- No deterministic extractor, for even one bit output
- For any extractor, can find an SV-source on which the extractor “fails”
Extractor for SV sources?

- No deterministic extractor, for even one bit output
- For any extractor, can find an SV-source on which the extractor “fails”
- Output bias no better than input bias
Extractor for SV sources?

- No deterministic extractor, for even one bit output
- For any extractor, can find an SV-source on which the extractor “fails”
  - Output bias no better than input bias

Exercise
Randomized Extractors
Randomized Extractors

Randomized extractor
Randomized Extractors

- Randomized extractor
- Some perfect randomness as a catalyst
Randomized Extractors

Randomized extractor

Some perfect randomness as a catalyst

Ext

Biased input

Almost unbiased output
Randomized Extractors

Randomized extractor

Some perfect randomness as a catalyst

Seed randomness

Almost unbiased output

Biased input

Ext
Randomized Extractors

- Randomized extractor
- Some perfect randomness as a catalyst
- Running a BPP algorithm with only the imperfect source

Seed randomness

Almost unbiased output

Biased input
Randomized Extractors

- Randomized extractor
  - Some perfect randomness as a catalyst
  - Running a BPP algorithm with only the imperfect source
    - Draw one string from the biased source and generate random tapes, one for each seed. If the algorithm accepts on more than half of these random tapes, accept.
Randomized Extractors

Randomized extractor

Some perfect randomness as a catalyst

Running a BPP algorithm with only the imperfect source

Draw one string from the biased source and generate random tapes, one for each seed. If the algorithm accepts on more than half of these random tapes, accept.

Polynomial time, if seed logarithmically short
Randomized Extractors

- Randomized extractor
- Some perfect randomness as a catalyst
- Running a BPP algorithm with only the imperfect source

  Draw one string from the biased source and generate random tapes, one for each seed. If the algorithm accepts on more than half of these random tapes, accept.

- Polynomial time, if seed logarithmically short
- Error probability remains bounded [Exercise]
Extractor for SV sources
Extractor for SV sources

Randomized extractor
Extractor for SV sources

Randomized extractor

Input: SV(δ) for a constant δ<1
Extractor for SV sources

- Randomized extractor
- Input: $SV(\delta)$ for a constant $\delta < 1$
Extractor for SV sources

Randomized extractor

Input: SV(δ) for a constant δ < 1

Plan: to get to a small (conditional) bias (O(1/m)) for each output bit.
Extractor for SV sources

- Randomized extractor
- Input: SV(δ) for a constant δ<1
- Plan: to get to a small (conditional) bias (O(1/m)) for each output bit.
- Weak extraction
Extractor for SV sources

- Randomized extractor
- Input: SV(δ) for a constant δ<1
- Plan: to get to a small (conditional) bias (O(1/m)) for each output bit.
- Weak extraction
Extractor for SV sources

- Randomized extractor
  - Input: SV(δ) for a constant δ<1
  - Plan: to get to a small (conditional) bias (O(1/m)) for each output bit.
  - Weak extraction
Extractor for SV sources

- Randomized extractor
- Input: SV(\(\delta\)) for a constant \(\delta<1\)
- Plan: to get to a small (conditional) bias (\(O(1/m)\)) for each output bit.
- Weak extraction

\[ a_i = \langle R_i, S \rangle \]
Extractor for SV sources

Randomized extractor

Input: SV(δ) for a constant δ<1

Plan: to get to a small (conditional) bias (O(1/m)) for each output bit.

Weak extraction

Using seed-length d = O(log m)
Extractor for SV sources

- Randomized extractor
  - Input: SV(δ) for a constant δ<1
  - Plan: to get to a small (conditional) bias (O(1/m)) for each output bit.
- Weak extraction
  - Using seed-length d = O(log m)
- Analysis: Need to bound only the collision probability for an input block of length d [Exercise]
Extractor for SV sources

- Randomized extractor
  - Input: $SV(\delta)$ for a constant $\delta < 1$
  - Plan: to get to a small (conditional) bias ($O(1/m)$) for each output bit.
  - Weak extraction

- Using seed-length $d = O(\log m)$

- Analysis: Need to bound only the collision probability for an input block of length $d$ [Exercise]

- Collision prob $\leq$ max prob $\leq (1/2 + \delta/2)^d = 1/poly(m)$
Extractors

Extractors with logarithmic seed-length known for more general classes of sources (block sources)
Extractors

- Extractors with logarithmic seed-length known for more general classes of sources (block sources)
- Which extract “almost all” the entropy in the input
Extractors

- Extractors with logarithmic seed-length known for more general classes of sources (block sources)
- Which extract “almost all” the entropy in the input
- Output can be made “arbitrarily close” to uniform
Extractors

- Extractors with logarithmic seed-length known for more general classes of sources (block sources)
  - Which extract “almost all” the entropy in the input
  - Output can be made “arbitrarily close” to uniform
- Bottom line: Can efficiently run BPP algorithms using very general classes of sources of randomness
Extracting from independent sources
Extracting from independent sources

- Deterministic extraction possible!
Extracting from independent sources

- Deterministic extraction possible!
Extracting from independent sources

Deterministic extraction possible!
Extracting from independent sources

Deterministic extraction possible!
Extracting from independent sources

Deterministic extraction possible!
Extracting from independent sources

- Deterministic extraction possible!
Extracting from independent sources

Deterministic extraction possible!
Extracting from independent sources

Deterministic extraction possible!
Extracting from independent sources

- Deterministic extraction possible!

- $a = \langle R, S \rangle$
Extracting from independent sources

- Deterministic extraction possible!
- Challenge: extract almost all the entropy from two independent sources

\[ a = \langle R, S \rangle \]
Extracting from independent sources

- Deterministic extraction possible!
- Challenge: extract almost all the entropy from two independent sources
- Known, with a few more sources

\[ a = \langle R, S \rangle \]
Today
Today

Efficient soundness amplification using expanders
Today

- Efficient soundness amplification using expanders
- Imperfect random sources
Today

- Efficient soundness amplification using expanders
- Imperfect random sources
- von Neumann, SV, and more
Today

- Efficient soundness amplification using expanders
- Imperfect random sources
  - von Neumann, SV, and more
- Extractors
Today

- Efficient soundness amplification using expanders
- Imperfect random sources
  - von Neumann, SV, and more
- Extractors
  - For von Neumann, SV sources and more
Today

- Efficient soundness amplification using expanders
- Imperfect random sources
  - von Neumann, SV, and more
- Extractors
  - For von Neumann, SV sources and more
- Can extract almost all entropy into almost uniform output using log seed-length
Today

- Efficient soundness amplification using expanders
- Imperfect random sources
  - von Neumann, SV, and more
- Extractors
  - For von Neumann, SV sources and more
- Can extract almost all entropy into almost uniform output using log seed-length
- Closely related to other tools: pseudorandomness generators, list decodable codes
Today

- Efficient soundness amplification using expanders
- Imperfect random sources
  - von Neumann, SV, and more
- Extractors
  - For von Neumann, SV sources and more
- Can extract almost all entropy into almost uniform output using log seed-length
- Closely related to other tools: pseudorandomness generators, list decodable codes
- Useful in “derandomization”