Today

- Applications of Codes in Computer Science: Randomness Extractors

Randomness and Computation

- Randomness useful in design of algorithms.
- In reasonable number of cases - only efficient algorithms known are randomized algorithms.
- What happens in practice?

Randomness in nature

- One hope: Computational pseudo-randomness. Universal algorithm that given $t, m$ produces $\text{poly}(t)$ strings of length $m$ that look “random” for any algorithm $A$ running in time $t$.

- Other hope: Randomness inherent in physics. But, even then:
  - Algorithms assume $m$ unbiased independent bits.
  - Sources of randomness produce dependent bits.
  - How to “extract” pure randomness?

Notions of imperfect randomness

- Good imperfectness: statistically close to uniform.
  - Prob. distribution is a vector of $\ell_1$ norm 1.
  - Statistical distance between $\pi$ and $\sigma$ is $\frac{1}{2}||\pi - \sigma||_1$.
  - Statistical distance between $\pi$ and $\sigma$ at most $\epsilon$ implies $\Pr_{x \in \sigma}[A(x) = 1] - \Pr_{x \in \pi}[A(x) = 1] \leq \epsilon$.
  - While would be ideal to convert imperfect randomness into $m$ independent uniform bits, it is good enough to generate distribution that is $\epsilon$-close to $U_m$ the uniform distribution on $m$ bits.
Notions of imperfect randomness (contd.)

- Bad imperfectness: \( k \) bits of min-entropy.
- Distribution \( \pi \) on \( \{0,1\}^n \) has \( k \) bits of min-entropy if no string \( x \in \pi \) has probability more than \( 2^{-kn} \).
- Example: Some \( k \) bits random, others fixed in advance.
- Worse example: Uniform on some \( 2^k \) strings.
- How to use such “randomness”?
- Non-trivial!

\[ \text{Ex} \text{tractors} \]

- \( \text{Ext} : \{0,1\}^n \times \{0,1\}^\ell \rightarrow \{0,1\}^m \) is a \((k, \epsilon)\)-extractor if for every distribution \( D \) of min-entropy \( k \), the distribution \( \{\text{Ext}(x, y)\}_{x \in D, y \in U_1} \) is \( \epsilon \)-close to uniform.
- Usage: Given \( n \) bit string \( x \in D \) and algorithm \( A \) using \( m \) bit random strings, run \( A \) on \( \{D(x, y)\}_y \).
- W.p. \( 1 - \sqrt{\epsilon} \), \( x \) is such that \( E_y[A(\text{Ext}(x, y))] \) is \( \sqrt{\epsilon} \) close to its expectation on uniform.

\[ \text{Trevisan Extractors} \]

- Ingredients:
  - \([N, n, *]_2\) code \( E \) list-decodable upto \( 1/2 + \delta \) fraction error with \( \text{poly}(1/\delta) \) codewords. Will let \( N = 2^\ell \).
  - \((t, \ell, a)\)-block design \( B \) with \( |B| = m \):
    i.e., \( B = \{s_i\}_{i \in [m]} \), where \( s_i \subset [\ell] \) and \( |s_i| = \ell \) and \( |s_i \cap s_j| \leq a \).
- \( y \in \{0,1\}^\ell \) defines projection \( \pi_y : \{0,1\}^N \rightarrow \{0,1\}^m \) as follows: \( \pi_y(z) = z_{y|s_1} \cdots z_{y|s_m} \).
- \( \text{Ext}(x, y) = \pi_y(E(x)) \) !

\[ \text{Analysis} \]

- Consider \( x \)’s such that \( A \) not fooled by \( \text{Ext}(x, y) \).
- Then \( A \) can predict many next bits of \( \text{Ext}(x, y) \).
- Step 1: Show by careful argument that this gives a succinct description of some \( r \) close to \( E(x) \) (for fixed \( A \)).
- Step 2: this implies that \( x \) has small description.
- By PHP, can’t have too many \( x \)’s with small description (even with fixed \( A \)).
• For us Step 2 is trivial: If $E$ is $((\frac{1}{2} - \epsilon)N, L)$-error-correcting, then $\log L$ additional bits specify $x$ provided $\Delta(E(x), r) \leq (\frac{1}{2} - \epsilon)N$.

• So we can focus on Step 1.

Details of Step 1

• Fix $A, x$. Let $w(y) = \text{Ext}(x, y)$ and $z = E(x)$.

• Step 1.1: Suppose $A$ has different acceptance probability on $\text{Ext}(x, y)$ than on uniform, then there exists $i \in [m]$ and function $f$ such that $f(w(y_1), \ldots, w(y_{i-1}))$ equals $w(y_i)$ with high probability for random $y$.

• Step 1.2: There exist $y_1, \ldots, y_n$ such that $w(y_j) = z_j$; the string $\{w(y_j)\}_{i < j, j \in [n]}$ can be specified with much less than $n$ bits (specifically $m2^n$ bits); and $f$ retains its advantage on $y_1, \ldots, y_n$.

• Step 1.3: Put two & two together.

Details of Step 1.1

• Disclaimer 1: Standard argument. Goes back to [[Yao,unpublished]].

• Let $D_0, \ldots, D_m$ be distributions moving from extractor to uniform: Pick random $w$ from extractor, and $u$ uniformly. $D_i =$ last $i$ bits from $u$, and first $m - i$ bits from $w$.

• Triangle inequality implies $A$ has different biases on $D_{i-1}$ and $D_i$ for some $i$.

• $f$ follows somehow ...
Details of Step 1.2

- Natural choice for \( y_1, \ldots, y_n \) when we think about it.
  - Fix \( y_\ast \) on all but \( S_i \) to fixed random values and on \( S_i \) let is vary over all \( n \) possibilities.
  - \( f \) should retain its bias on this set to, by averaging.
  - How many possibilities for \( y_j|S_i \)? All \( n! \)
  - How many possibilities for \( y_j|S_{i'} \)? At most \( 2^a \), since \( |S_i \cap S_{i'}| \leq a \).
  - Can specify \( x_{y_j|S_{i'}} \) for all \( i' \) by specifying \( m \cdot 2^a \) values.
  - Obtain properties needed.