CS395T Pseudorandomness and Combinatorial Constructions
Lecture 25, April 17, 2001
Lecturer: Prof. David Zuckerman, Scribe: Vladimir Trifonov

Applications of Extractors

1 Generator for Space-Bounded Computation

A randomized algorithm running in space S, $S(n) \geq \log n$, could use up to $2^{O(S)}$ random bits. Such algorithms can be fooled by the Nisan's PRG for space-bounded computation which stretches $O(S^2)$ to $2^{O(S)}$ bits. The question one can ask is what if the algorithm we want to derandomize uses less then $2^{O(S)}$ random bits, e.g. poly(S); how many random bits we can use to fool a randomized space-bounded computation using a polynomial number of random bits? Nisan's PRG can stretch $O(S \log S)$ random bits to poly(S) random bits, but can we do better? Nisan and Zuckerman showed a generator (NZ PRG), using extractors, which converts $O(S)$ random bits to poly(S) bits that look random to any randomized algorithm running in space S. Thus a randomized space-bounded computation running in space S and using poly(S) random bits can be completely derandomized in space $O(S)$.

We will see the building block of this generator – a generator which stretches $O(S)$ bits to $O(S^2/\log S)$ bits. The generator of any polynomial number of bits is obtained by a straightforward composition of the generator which we will describe.

Besides its significance as a step towards the ultimate goal of producing $2^{O(S)}$ bits from S bits, this generator has some practical value. For example the task of obtaining a random n-bit prime number requires $O(n^3)$ random bits, which can be reduced to $O(n)$ by the NZ PRG. We can also use this generator to produce bits for a random walk on a weak expander until it mixes.

1.1 Construction

Let $E : \{0,1\}^{4S} \times \{0,1\}^t \to \{0,1\}^S$, for $t = O(\log S)$, be an extractor with error ε for min-entropy at least $2S$. Define $G : \{0,1\}^{4S} \times (\{0,1\}^t)^l \to (\{0,1\}^S)^l$, where $l = S/t$, in the following way:

$$G(x, y_1, \ldots, y_l) = E(x, y_1) \circ \cdots \circ E(x, y_l).$$

We can depict the computation of the space-bounded algorithm as a layered multi-graph with l layers, each with 2^S nodes and each node with out-degree 2^S. Each node corresponds to a configuration of the computation and each edge going out of a node describes the transition of the computation after reading S random bits. The number of layers is exactly S^{-1} times the number of random bits required by the algorithm.

In these terms the above construction can be described as converting the S bits required to move out of a node, to t bits (the y's). In the new transition we use $4S$ additional bits (the x), but these are the same for all layers.

1.2 Correctness

The idea of the proof is to consider the nodes of a layer which have high probability of being visited. The total probability of visiting of the rest will be small and thus they will not pose an obstacle in our argument. It will turn out that conditioning on being in a node of high probability, induces a source with high min-entropy on x. This is not a surprise since x is uniform and $|x|$ is larger than S, so even after conditioning on an event of large probability, there will be randomness left in x. The extractor will extract this randomness from x and supply almost fresh S bits for the next transition.

Define U_i to be the distribution on the nodes of layer i using uniform bits for the transitions, and D_i to be the distribution using the output of the generator. We will prove the following lemma from which the correctness follows taking $i = l$.

Lemma 1 $|U_i - D_i| \leq i(\varepsilon + 2^{-S})$.

Proof: For $i = 0$ the statement is trivial since $U_0 = D_0$. Assume that the statement is true for $i - 1 \geq 0$.

Let W_{i-1} be the state at time $i-1$ using the output of the generator and X is the distribution from which x is chosen (i.e. the uniform distribution on $\{0,1\}^{4S}$). Then for every state j and for every $x \in \{0,1\}^{4S}$,

$$\begin{aligned}
\Pr[X = x | W_{i-1} = j] &= \frac{\Pr[X = x \wedge W_{i-1} = j]}{\Pr[W_{i-1} = j]} \\
&\leq \frac{\Pr[X = x]}{\Pr[W_{i-1} = j]} \\
&= \frac{2^{-4S}}{\Pr[W_{i-1} = j]}.
\end{aligned}$$

Hence if $\Pr[W_{i-1} = j] \geq 2^{-2S}$, then $\Pr[X = x | W_{i-1} = j] \leq 2^{-2S}$, i.e. $H_\infty(X | W_{i-1} = j) \geq 2S$. Denote with U_i^j and D_i^j the distributions on layer i conditioned on $W_{i-1} = j$. Then, by the properties of the extractor E, $|U_i^j - D_i^j| \leq \varepsilon$.

Let $B = \{j : D_{i-1}(j) \geq 2^{-2S}\}$. The following chain of inequalities completes the proof.

$$\begin{aligned}
2|U_i - D_i| &= \|U_i - D_i\|_1 \\
&= \|\sum_j U_{i-1}(j) U_i^j - \sum_j D_{i-1}(j) D_i^j\| \\
&\leq \|\sum_j U_{i-1}(j) U_i^j - \sum_j D_{i-1}(j) U_i^j\| + \|\sum_j D_{i-1}(j) U_i^j - \sum_j D_{i-1}(j) D_i^j\| \\
&\leq \|U_{i-1} - D_{i-1}\| + \sum_{j \in B} D_{i-1}(j) \|U_i^j - D_i^j\| + \sum_{j \notin B} D_{i-1}(j) \|U_i^j - D_i^j\| \\
&\leq 2(i-1)(\varepsilon + 2^{-S}) + 1 \cdot 2\varepsilon + 2^S \cdot 2^{-2S} \cdot 2 \\
&= 2i(\varepsilon + 2^{-S})
\end{aligned}$$

□

2 Error-Correcting Codes

Let $C : \Sigma^R \to \Sigma^T$ be an error-correcting code. From Lecture 8 we know that $\text{dist}(C) = \min_{x \neq y} \text{distance}(C(x), C(y))$ and that we can uniquely decode from $\text{dist}(C) - 1$ erasures or $(\text{dist}(C) - 1)/2$ errors. In the same lecture we also defined Reed-Solomon code C_{RS} as the evaluation of polynomials of degree less than R over $\mathbb{F}_q$ and saw that $\text{dist}(C_{RS}) = T - R + 1$.

In Lecture 10 we saw that if we are willing to give up unique decoding we can decode with more than $E \geq \text{dist}(C)/2$ errors using the list-decoding technique. We showed that for every binary code C there are at most $\text{poly}(T)$ codewords within a certain distance E of every word. We also showed the polynomial time list-decoding algorithm of Sudan for Reed-Solomon codes, which worked as long as the agreement $A = T - E$ is greater than $\sqrt{RT}$.

List-decoding has many applications. Among them are applications in cryptography for determining a hardcore predicate of 1-way permutations and in complexity theory for obtaining the average case hardness of the permanent. To exemplify the usefulness of extractors we will show a generalization of list-decoding and codes which allow for this generalized list-decoding. This is a recent development due to Ta-Shma and Zuckerman, which has application in obtaining multiple hardcore bits for 1-way permutations.

2.1 Generalized List-Decoding

Consider the following extremely noisy channel: for every symbol received we do not know the exact symbol, but only a set which contains it. We will model this as a set $S \subseteq [T] \times \Sigma$. Let w_i be the i-th symbol received. An example of the model is shown in the following diagram.

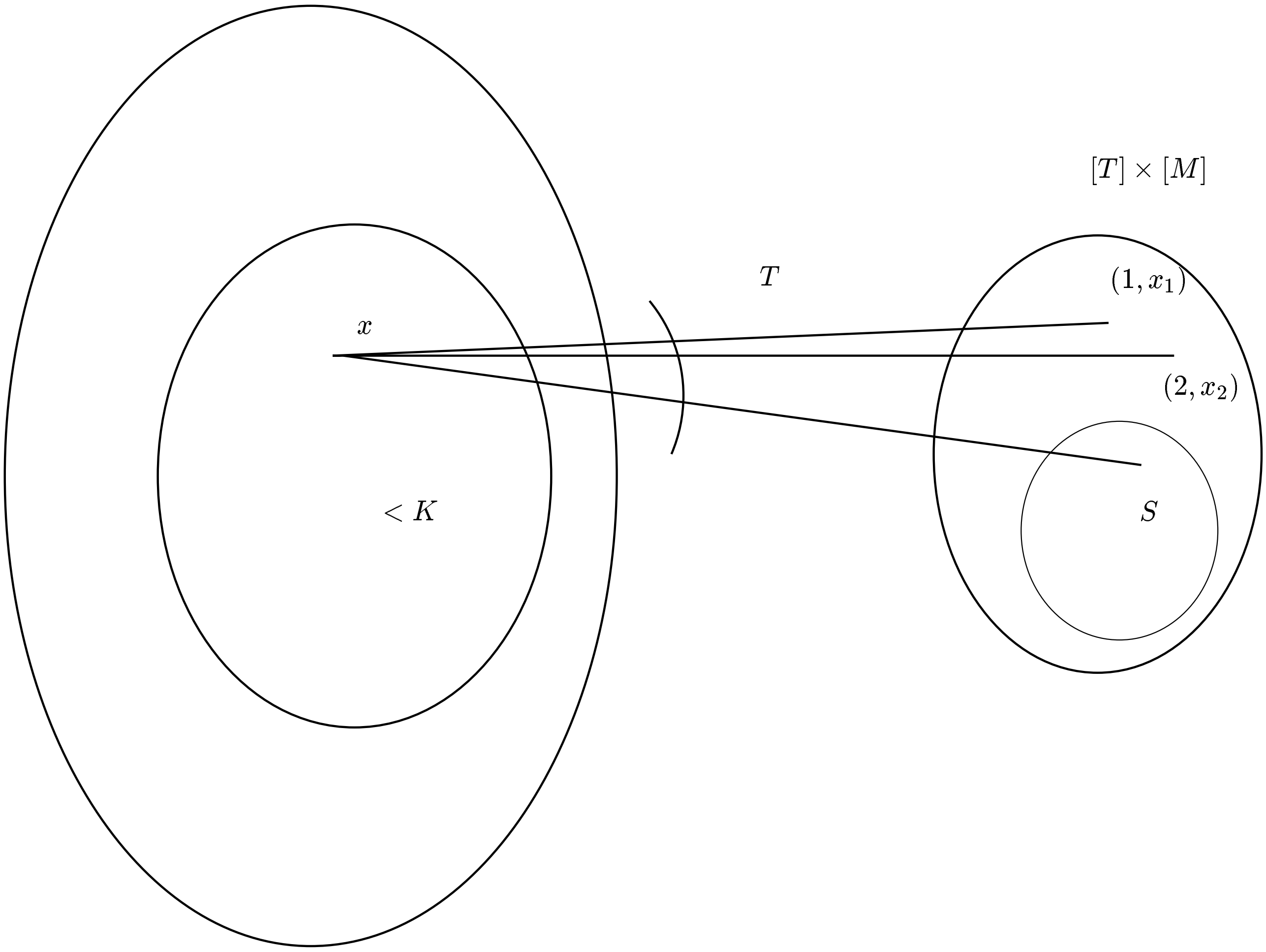
$$\begin{array}{lcl} w_1 \in \{2,3,7\} & \longleftrightarrow & (1,2),(1,3),(1,7) \\ w_2 \in \{3,4,6,8\} & \longleftrightarrow & (2,3),(2,4),(2,6),(2,8) \\ \vdots & & \vdots \\ w_T \in \{1,5\} & \longleftrightarrow & (T,1),(T,5) \end{array}$$

We want to exhibit a code C which for every S and every ε does not have too many codewords $C(x) = (w_1, \ldots, w_T)$ agreeing with S in at least $(\rho(S) + \varepsilon)T$ places, where $\rho(S) = |S|/(T|\Sigma|)$. Given such a code our goal will be to produce efficiently a list of these codewords.

First let us see that Reed-Solomon codes do not allow efficient list-decoding in the above model. Assume that everything we know is that the received symbols are quadratic residues, i.e. $S = [T] \times \{\text{quadratic residues}\}$. From number theory we know that $\rho(S) \approx 1/2$. The values of every polynomial p over $\mathbb{F}_q$, such that $p = p_0^2$, for some polynomial p_0, are quadratic residues, i.e. every such polynomial agrees with S in T places. Therefore there are at least $q^{\lceil (R+1)/2 \rceil} \geq \sqrt{|C_{RS}|}$ codewords of C_{RS} which have large agreement with S. This means that an efficient general procedure for list-decoding of Reed-Solomon codes in the generalized model is not possible (we must have at most $\text{poly}(T)$ agreeing codewords for every possible S).

2.2 Extractor Codes

Let $E : [N] \times [T] \to [M]$ be a strong extractor with error ε for min-entropy at least $\log K$. Define the *extractor code* $C_E(x) = (E(x,1), \ldots, E(x,T))$. The following diagram depicts the definition of extractor code.



Let $S \subseteq [T] \times [M]$ be the set we want to agree with. By the extractor property, less than K words have agreement at least $(\rho(S) + \varepsilon)T$ with S (here we use the alternative view on strong extractors as defined in Lecture 22), which is an improvement over Reed-Solomon codes for large sets S. Hence, explicit constructions of extractors yield explicit codes which are list-decodable (at least non-constructively) in the generalized sense. An efficient polynomial time list-decoding algorithm for the extractor code C_{TR} induced by the Trevisan's Extractor is shown in the Extractor Codes paper.

Let us see how we can apply extractor codes to the construction of hardcore function for any 1-way permutation $f : \{0,1\}^n \to \{0,1\}^n$. Out goal is to find $h(x, y)$, which is hard to compute given $f(x)$ and y (h is hardcore for f). In Lecture 21 we saw that due to Goldreich and Levin we can take $h(x, y)$ to be $x \cdot y$. Thinking of the Goldreich-Levin predicate as the y-th bit in the encoding of x with the Hadamard code, Impagliazzo showed that it is enough to take $h(x, y) = C(x)_y$, where C is any list-decodable binary code. From Lecture 8 we know that we can construct efficiently such codes with constant rate, which essentially reduces the additional randomness required to generate one hardcore bit from n to $O(\log n)$.

The definition of hardcore function requires the function to be hard to predict in variation distance sense. Therefore to apply the above idea to multiple hardcore bits, i.e. we do not require the list-decodable code to be binary, we will need list-decodability in the generalized sense, i.e. for every possible statistical test S.

Using that C_{TR} has an efficient list-decoding procedure, we obtain a hardcore function $h_{TR}(x,y) = (C_{TR}(x))_y = E_{TR}(x,y)$, where the additional randomness is $O(\log^2(n/\varepsilon))$ and the number of output bits is $m \leq n/\varepsilon$. h_{TR} has the following hardcore property: $h_{TR}(x,y)$ has security $(2^{2m}, \varepsilon)$ for every 1-way permutation f with security $(2^{4m}, \varepsilon/2)$. The security of f is in terms of inverting, i.e. every probabilistic algorithm running in time 2^{4m} has advantage at most $\varepsilon/2$ of computing x given $f(x)$. The security of h_{TR} is in the usual hardcore terms, i.e. every probabilistic algorithm running in time 2^{2m} has advantage at most ε of computing $h(x,y)$ given $f(x)$ and y.

2.3 Slice Extractors

Looking closely at the definition of extractor code, we notice that we actually use a weaker form of extractor. Namely the statistical tests used to distinguish the output of the extractor from uniform are restricted from being of any size to being of size at most pM (pTM in the strong version, actually used in the construction of extractor codes), for fixed p. We will call such extractors *slice extractors*. It is known that for the optimal extractor we have $T = \Theta(n/\varepsilon^2)$, where $n = \log N$. It can be shown that for an optimal slice extractors with parameter $p \geq \varepsilon$, $T = \Theta(pn/\varepsilon^2)$.

Since extractor codes imply generalized list-decodability, we can apply slice extractors to construct codes, list-decodable in the usual sense, i.e. $|S| = T$. For Reed-Solomon codes we also have that $T = M$. On the other hand, by the definition of the generalized model of noisy channel, $\varepsilon = A/T - \rho(S) = (A-1)/T$. Plugging this in the bound for T for an optimal extractor, we obtain that optimal extractors give list-decodable codes with $A \approx \sqrt{nT}$. This is similar to the bound $A > \sqrt{RT}$ obtained by Sudan for Reed-Solomon codes. For optimal slice extractors taking $p = \varepsilon$, in a similar way we obtain that $A \approx n$, which is the same as having agreement $\log |C_{RS}| = R \log M$ in the Reed-Solomon codes notation. But this is exactly the agreement required by a random code, i.e. if we can construct optimal slice extractors we will be able to construct list-decodable codes as good as random codes.

Optimal slice extractors also imply optimal bipartite Ramsey graphs, i.e. with $t = \Theta(\log n)$. This again is an improvement over the usual optimal extractors (and actually over all current constructions), since they give bipartite Ramsey graphs with $t = O(\sqrt{n})$.

3 Summary of the Course

Here is a somewhat entangled road map of the course.