6.897 Algorithmic Introduction to Coding Theory	December 10, 2001
Lecture 22	
Lecturer: Madhu Sudan	*Scribe: Eleni Drinea*

In today's lecture, we shall continue our discussion about *pseudorandomness*; in particular, we will be interested in the following topics:

- ϵ–biased spaces,
- Pseudorandomness from hard functions,
- Extractors of randomness.

We will overview how coding theory explains these notions, without actually giving detailed proofs of they theorems.

1 ϵ–biased spaces

In the last lecture we presented a randomized algorithm for MAX–3–SAT and answered the questions what kinds of pseudorandom sets would be useful for this algorithm and how we can construct such sets using error correcting codes. Today we answer the same questions for MAX–10–SAT and present techniques that build smaller pseudrandom sets, with a rather insignificant loss in the expected number of satisfied clauses.

MAX–10–**SAT**

Instance: m clauses $c_1, \dots, c_m$ on n variables $x_1, \dots, x_n$ such that each clause has 10 distinct literals.

Goal: Find an assignment that satisfies the maximum number of clauses.

Note that the purely random assignment (i.e., pick a random n–bit string $\alpha_1 \dots \alpha_n \in \{0,1\}^n$ and set $x_i = \alpha_i$, $i = 1, \dots, n$) satisfies an $1 - 2^{-10}$ fraction of the clauses. This requires a 2^n sample space of randomness. If we use a 10–wise independent source of randomness, we can still satisfy the same fraction of clauses using only an n^5–sized space. We can construct a more efficient sample space by allowing a slightly smaller fraction of clauses to be satisfied. Before this, we recall the definition of l–wise independent sets from last lecture.

Definition 1 *Let $S \subseteq \{0,1\}^n$. If for all distinct $i_1, \dots, i_l$ in $[n]$ and $b_1, \dots, b_l \in \{0,1\}$,*

$$\Pr[(X_{i_1} = b_1) \wedge \cdots \wedge (X_{i_l} = b_l)] = 2^{-l}$$

then S is l–wise independent.

In [3] the notion of ϵ–biased spaces was introduced.

Definition 2 *A set S is ϵ–biased if $\forall \alpha \in \{0,1\}^n - \{\overline{0}\}$,*

$$\left| \Pr_{x \in S}[< \alpha, x >= 1] - \frac{1}{2} \right| \leq \epsilon.$$

Before we can state a theorem about ϵ–biased sets, we need the notion of the *distance* between two distributions.

Definition 3 *Let D_1, D_2 be two distributions on t–bit strings, i.e., $D_i : \{0,1\}^t \to [0,1]$ and $\sum_{x \in \{0,1\}^t} D_i(x) = 1$. Then the* distance *between D_1, D_2 is*

$$\frac{1}{2} \sum_{x \in \{0,1\}^t} | D_1(x) - D_2(x) |$$

Consider a set $S \subseteq \{0,1\}^n$; this is a $|S| \times n$ matrix. The *projection* of S onto coordinates $i_1, i_2, \ldots, i_l$ is a $|S| \times l$ matrix formed by the columns $i_1, i_2, \ldots, i_l$ of S.

Theorem 4 *If S is ϵ–biased, then the projection of S onto any l coordinates is $2^l \cdot \epsilon$–close to the uniform distribution.*

Note that the set S is l–wise independent if its projection onto any l coordinates is 0–close to the uniform distribution.

1.1 Construction of ϵ–biased spaces

In this section we show how to construct an ϵ–biased probability space S using error correcting codes. It turns out that ϵ–biased spaces slightly augment the definition of a code. Recall that S is a $|S| \times n$ matrix. One fact we know about this matrix is that if we right multiply it by a fixed vector $\alpha \in \{0,1\}^n$, then we expect the resulting $|S| \times 1$ vector to consist of a fraction of between $\frac{1}{2} - \epsilon$ and $\frac{1}{2} + \epsilon$ 1's.

Now consider the collection of vectors $S \cdot \alpha$. This collection corresponds to a linear code of relative minimum distance $\frac{1}{2} - \epsilon$. Still, this is not sufficient to construct an ϵ–biased space, since for the latter, we also need our code to achieve a maximum relative distance of $\frac{1}{2} + \epsilon$. However, most families of codes we have already seen exhibit this property! Once we know that we can construct ϵ–biased probability spaces, we wish to improve their properties. Specifically, we want S to be small. We can view S as the transpose of the generator matrix of a linear code, of dimension n and block length $|S|$. We wish to minimize $\frac{|S|}{n}$, or equivalently, maximize $\frac{n}{|S|}$. The latter quantity is simply the rate of the code, conditioned on the relative distance of this code being $\frac{1}{2} - \epsilon$. Thus the goal is to find binary codes of maximum rate, with minimum relative distance $\frac{1}{2} - \epsilon$ and maximum relative distance $\frac{1}{2} + \epsilon$. We already know of such codes:

- Reed Solomon codes concatenated with Hadamard codes (dimension is n, block length is $\left(\frac{n}{\epsilon}\right)^2$)
- Algebraic Geometry codes concatenated with Hadamard codes (dimension is n, block length is $O\left(\frac{n}{\epsilon^3}\right)$)

The latter were suggested in [1] and form the third row in Table 1; it is not clear though whether these are always better than the former.

1.2 Construction of l–wise ϵ–biased spaces

We can do even better in terms of the sample space needed by using l–wise ϵ–biased spaces, which we define after [3].

Definition 5 *A set $S \subseteq \{0,1\}^n$ is l–wise ϵ–biased if for all projections of S onto l coordinates, the induced distribution is ϵ–biased.*

Theorem 4 also holds for l–wise ϵ–biased sets S.

In order to construct such spaces, let's recall l-wise independence from last lecture. Small l–wise independent spaces correspond to the parity check matrix $H_{n \times t}$ of some code of large distance (e.g., BCH codes), with $t = \log n^{l/2}$. If we multiply a purely random vector $x \in \{0,1\}^t$ by H^T, the resulting vector xH^T is an n–bit vector consisting of l–wise independent bits. The following lemma summarizes what we proved in the last lecture:

Lemma 6 *If H is the parity check matrix of an $[n, n-t, l+1]_2$ code, then $S = \{x \cdot H^T \mid x \in \{0,1\}^t\}$ is l–wise independent.*

In the sequel, we will use ϵ–biased strings x instead of purely random strings. The following lemma holds:

Lemma 7 *If H is the parity check matrix of an $[n, n-t, l+1]_2$ code, the number of rows in H^T is t and S' is a collection of ϵ–biased vectors in $\{0,1\}^t$, then $S = \{x \cdot H^T \mid x \in S'\}$ is l–wise ϵ–biased.*

The fourth row in Table 1 arises from 10–wise 2^{-22}–biased spaces.

Table 1

MAX–10–SAT	E[fraction of clauses satisfied]	Sample Space
Purely random assignmemt	$1 - 2^{-10}$	2^n
10–wise independent source of randomness	$1 - 2^{-10}$	n^5
ϵ–biased with $\epsilon = 2^{-22}$	$1 - 2^{-10} - 2^{-11}$	$O\left(\frac{n}{\epsilon^3}\right)$
10–wise ϵ–biased with $\epsilon = 2^{-22}$	$1 - 2^{-10} - 2^{-11}$	$\left(\frac{\log n}{\epsilon}\right)^{O(1)}$

2 Pseudorandomness from hard functions

Where does randomness come from? Physisists answer "it exists!", mathematicians answer "it can be generated". There exist several approaches to pseudorandomness. We already discussed pseudorandomness for specific algorithms, where we consider an algorithm that uses n truly random bits and try to determine how we can replace those bits by n bits that are "less random" (e.g., pairwise independent); we can generate these bits by creating smaller truly random spaces.

Another approach is to start with n bits that are not truly random (but we do not necessarily know how much they differ from random bits) and figure out ways to use these. In other words, we assume that an Adversary picks a large space $S \subseteq \{0,1\}^n$ of very impure randomness, and gives a random element of S to the algorithm. The algorithm does not know S but there is randomness in S that it can exploit.

Today, we will discuss how to use hard functions to simulate randomness and what their connection to error–correcting codes is. As a brief history to the use of hard functions in generating randomness, we mention that in [2], Blum and Micali construct a pseudorandom bit generator that passes all polynomial-size statistical tests (as defined in [5]), under the intractability assuption of the Discrete Logarithm Problem; this simply means that if DLP is not solvable in probabilistic polynomial time, then no probabilistic polynomial time algorithm can guess the generator's next output bit wih probability more than $1/2$.

In the sequel, we show how the existence of certain kinds of hard functions implies BPP=P, following [4].

Assumption 8 (NW Assumption) *There exists a family of functions $\{f_t\}_t$ such that $f_t : \{0,1\}^t \to \{0,1\}$ and f_t is computable in time 2^t, but* no *boolean circuit C with $2^{2t/3}$ AND, OR, NOT gates, approximates f_t with probability more than $\frac{1}{2} + 2^{-t/3}$. This means*

$$\Pr_{x \in \{0,1\}^t}[f(x) = C(x)] \leq \frac{1}{2} + 2^{-t/3}$$

Theorem 9 *Under the NW assumption, BPP=P.*

Proof We construct a pseudorandom generator G. Then we run the BPP algorithm on every possible input to G to obtain a deterministic algorithm.

Construction of G: We will construct a generator $G : \{0,1\}^l \to \{0,1\}^n$, such that for any circuit C' of size $2^{l/3}$

$$\Pr_{x \in \{0,1\}^l}[C'(x) = 1] \approx \Pr_{y \in \{0,1\}^l}[C'(G(y)) = 1].$$

Intuitively, G stretches poly–log(n) truly random bits into n bits that look random to every sufficiently small circuit. More precisely, let y be an l–bit string. We project y onto a set S of t bits and apply f onto these bits; this way we extract one random bit. Now if we take a completely disjoint set T (i.e., $S \cap T = \emptyset$) of t bits and do as above, then we extract another bit. Clearly, we can repeat this process $\frac{l}{t}$ times and get $\frac{l}{t}$ random bits. But this does not gain us any bits. The idea is to apply f on sets of bits with small but nonempty intersections. For example, assume $|S \cap T| \leq \frac{t}{3}$, then these bits are still essentially independent. This way, we get a large collection of sets $S_1, S_2, \ldots, S_n$, where $S_i \subseteq [l]$, $|S_i| = t$, $|S_i \cap S_j| \leq \frac{t}{3}$. Now we project y onto each of these sets. The generator's output is $G(y) = f(y|_{s_1}), f(y|_{s_2}), \ldots, f(y|_{s_n})$. These bits look independent and have the properties we want. ■

How large can n be? This turns out to be a classic question in the theory of design (or block designs), asking for the construction of a fixed size family of sets over the smallest universe with (specified) pairwise small intersections.

Exercise: Codes over large alphabets give nice designs...

References

[1] N. Alon, O. Goldreich, J. Hastad, R. Peralta. Simple constructions for almost k–wise independent random variables. Random Structures and Algorithms, Vol.3 (1992).

[2] M. Blum, S. Micali. How to generate cryptographically strong sequences of pseudo–random bits. FOCS, 1982.

[3] J. Naor, M. Naor. Small–bias probability spaces: efficient constructions and applications. STOC, 1990.

[4] N. Nisan and A. Widgerson. Hardness vs. Randomness. FOCS, 1988.

[5] A. Yao. Theory and Applications of Trapdoor functions. FOCS, 1982.