CSCI 5060: Techniques in the theory of computing **Lecture 2**
The Chinese University of Hong Kong 18 January 2012

In the last lecture we looked at error correcting codes in the regime where the distance is small in terms of the block length. We used these codes towards a randomness-efficient construction of variables of bounded independence. Bounded independence can substitute for full independence in various scenarios, for example when we argued the existence of two-source hitters. Recall that an (N, K) two-source hitter is a function $f\colon [N] \times [N] \to \{0,1\}$ so that f is not constant on $S \times T$ for every pair of sets S, T of size at least K. Here $[N] \equiv \{0,1\}^n$ represents the possible pairs of data sequences, and K is the smallest possible number of uncertain outcomes in each data sequence.

We showed the existence of $(N, K = 2\log N + 1)$ two-source hitters via the probabilistic method: A random function f whose values are chosen uniformly and independently of one another is a hitter with nonzero probability. Yet finding such a hitter by brute-force search requires examining $2^{2^{2n}}$ possible functions f. To improve upon this, we observed that it is not necessary for our analysis that the values of f be fully independent, but K^2-wise independence suffices. So if we could get hold of a K^2-wise independent distribution of much smaller support, this would improve the running time. This led us to the study of t-wise independent random variables. We showed a close to optimal construction of such variables when t is small and used them to reduce the size of the search space down to $2^{n^3+o(n^3)}$.

Can we do even better? Our construction of t-wise independent distributions is close to optimal in support size (when t is not too large in terms of n), so it appears unlikely that the same approach can yield much bigger gains. But maybe t-wise independence is too strong a property for our purposes. In this lecture we will see an even weaker property, called almost t-wise independence, which can sometimes be used to approximate t-wise independence but admits more efficient implementations.

From what you know so far it seems that to get an improvement, we should take a closer look at what happens to linear codes at small distances. However, it turns out that instead of looking deeper, we get a more useful perspective by going broader: We will need to understand what happens to codes when the distance is large. How many messages can we encode in such a regime?

1 The Plotkin bound and the Hadamard code

Recall the pigeonhole bound which tells us that in an $[n, k, d]$ code we must have $k \leq n - d + 1$. Last time we looked at the regime where d was small and asked about the best value of k we can actually achieve. Today let's start at $d = n$ and see what happens when we start moving d to the left.

When d is close to n we cannot expect to have too many codewords that are all at distance d from one another. In fact, even if $d = 2n/3 + 1$, C can have at most two codewords. For suppose C has three distinct codewords c_1, c_2, c_3. Then c_1 and c_2 differ on more than $2n/3$ coordinates and so do c_1 and c_3. It follows that c_2 and c_3 must match on more than $n/3$ coordinates – the coordinates on which c_1 differs from both c_2 and c_3, a contradiction.

The following bound shows that k is quite small all the way down to distance $n/2$. The proof is very short, but to understand what really goes on it helps to have a geometric view of codewords. To get the right geometric picture in mind we apply a very useful trick. Instead of thinking of codewords as strings of 0s and 1s, we will think of them as vectors in $\{-1,1\}^n$. If two such vectors differ in $2n/3+1$ positions, it means the angle between them has to be obtuse. Our low-dimensional intuition suggests that we cannot pack too many such vectors inside $\mathbb{R}^n$.

Theorem 1 (Plotkin bound)**.** *Let $\epsilon > 0$. If C is an $[n, k, (1+\epsilon)n/2]$ code, then $k \leq \log(1/\epsilon + 1)$.*

Proof. We will view the code C as a subset of not $\{0,1\}^n$, but $\{-1,1\}^n$. Then for every pair of codewords $c \neq c'$, their inner product $\langle c, c' \rangle_{\mathbb{R}}$ can be upper bounded by

$$\langle c, c' \rangle_{\mathbb{R}} \leq \frac{(1+\epsilon)n}{2} \cdot (-1) + \frac{(1-\epsilon)n}{2} \cdot 1 = -\epsilon$$

and so

$$0 \leq \Big\|\sum_{c \in C} c\Big\|^2 = \sum_{c \in C} \|c\|^2 + \sum_{c \neq c'} \langle c, c' \rangle_{\mathbb{R}} \leq K - K(K-1)\epsilon$$

where $K = 2^k$. It follows that $K \leq 1/\epsilon + 1$. $\square$

For a fixed $\epsilon > 0$ the Plotkin bound tells us that we can have only a constant number of codewords at distance $(1+\epsilon)/2$ no matter how large n is. This is not terribly useful so we will instead look at what happens when the distance is just under $n/2$.

What happens at distance exactly $n/2$? Theorem 1 doesn't give us anything for $\epsilon = 0$ but we can do a heuristic argument. Since the distance is an integer, ϵ takes values in multiples of $1/n$ so for $\epsilon > 0$ we get that $k \leq \log(n+1)$. This suggests k should depend logarithmically in n when $\epsilon = 0$. Indeed it can be shown that if C is an $[n, k, n/2]$ code then $k \leq \log 2n$, and $[n, \log 2n, n/2]$ codes exist for infinitely many n.

An important example of a $[n, \log n, n/2]$ code when n is a power of two is the Hadamard code. Recall that the linear functions from $\mathbb{F}_2^k$ to $\mathbb{F}_2$ are those functions of the form

$$\ell_a(x) = \langle a, x \rangle = a_1 x_1 + \cdots + a_k x_k$$

where a is a string in $\mathbb{F}_2^k$ and the addition is modulo two. There are 2^k such functions. The Hadamard encoding $Had(x)$ of $x \in \{0,1\}^k$ consists of the evaluations of all $n = 2^k$ linear functions $\ell_a(x)$ as a ranges over $\mathbb{F}_2^k$. The Hadamard code is linear ($Had(x+y) = Had(x) + Had(y)$), so its minimum distance equals the smallest weight of a nonzero codeword. Again it will be helpful to work with codewords over $\{-1,1\}$ instead of $\{0,1\}$; we achieve this via the substitution $c \to (-1)^c$. When $x \neq 0$ and $a \sim \{0,1\}^n$ is chosen at random, we have

$$\mathrm{E}_{a \sim \{0,1\}^n}[(-1)^{\ell_a(x)}] = \mathrm{E}[(-1)^{x_1 a_1 + \cdots + x_n a_n}] = \prod_{i:\, x_i = 1} \mathrm{E}[(-1)^{a_i}] = 0$$

and so every nonzero codeword in the Hadamard code must have an equal number of zeros and ones.

2 The Gilbert-Varshamov bound

We now look at codes whose distance is a little bit less than $n/2$. We will parametrize it by setting $d = (1-\epsilon)n/2$ and ask for the asymptotically best value of k when ε is small and n is large. Some interesting phenomena occur in this range of parameters.

From the Plotkin bound we would expect that the rate k/n should approach zero as the *relative distance* d/n moves closer towards $1/2$. The singleton bound is too weak to support this intuition; it only tells us that $k/n < 1/2$. The volume bound gives the better but unsatisfying $k/n \prec 0.188$. Obtaining a good upper bound on the rate in this regime is surprisingly tricky, so let us start with some examples of codes instead.

Theorem 2. *There exists a linear* $[n, k, d]$ *code whenever* $k/n \leq 1 - H(d/n)$.

Here $H(p) = \log 1/p + (1-p)\log 1/(1-p)$ is the binary entropy function.

We will show the existence of such codes via the probabilistic method: We argue that a random linear code has the proscribed distance with nonzero probability.

Proof. Let H be a random $(n-k) \times n$ matrix whose entries are uniform independent values in $\{0,1\}$. We think of H as the parity check matrix of a code C. We will show that for a suitable value of k, the probability that C has distance less than d is strictly less than one. Recall that if $Hx \neq 0$ for any nonzero vector x of hamming weight less than d, then H has distance at least d. Therefore

$$\begin{aligned}\Pr_H[C \text{ has distance less than } d] &= \Pr_H[Hx = 0 \text{ for some } x \text{ of weight less than } d,\ x \neq 0] \\ &\leq \sum_{x\colon 0<\mathrm{wt}(x)<d} \Pr_H[Hx = 0] = 2^{-n+k}\sum_{k=1}^{d-1}\binom{n}{k}\end{aligned}$$

When $d \leq 1/2$ we have the elegant estimate

$$\sum_{k=0}^{d}\binom{n}{k} \leq \frac{1}{d^d(n-d)^{n-d}}\sum_{k=1}^{d}\binom{n}{k}d^k(n-d)^{n-k} = \frac{n^n}{d^d(n-d)^{n-d}} = 2^{nH(d/n)}.$$

It follows that C has distance less than d with probability less than $2^{k-n(1-H(d/n))} \leq 1$ as long as $k/n \leq 1 - H(d/n)$. □

Let's see what happens for $d = (1-\varepsilon)n/2$ and small ε. Recall the Taylor expansion $\log(1+x) = x + O(x^2)$, we get that

$$H((1-\varepsilon)/2) \geq 1 - K\varepsilon^2 \qquad \text{for } K \text{ sufficiently large}$$

which shows the existence of linear codes of distance $(1-\varepsilon)/2$ and message length $k = O(\varepsilon^2 n)$.

Where can we find such codes? One way is by doing brute force search over all possible linear codes. A linear map $C\colon \{0,1\}^k \to \{0,1\}^n$ can be described by a $k \times n$ matrix, so the desired code can be found by going over all $2^{kn} = 2^{O((\varepsilon n)^2)}$ such matrices. This brute-force search can be improved a bit using the "method of conditional probabilities", and in fact the code can be found in time $\mathrm{poly}(n)2^n$. You will do this in the homework.

We will now see an explicit construction that achieves somewhat worse tradeoffs between k, n and ε.

3 Concatenation

As for the case of small distances, a good starting point are the Reed-Solomon codes, which are optimal but unfortunately require a large alphabet. Recall that the Reed-Solomon code $[n, k, d]_{\mathbb{F}}$ over a field $\mathbb{F}$ achieves the pigeonhole bound $d = n - k + 1$ as long as $|\mathbb{F}| \geq n$.

When d was small, we managed to turn the Reed-Solomon code over $\mathbb{F}_n$, $n = 2^m$ into a pretty good binary code B simply by replacing each field element by its binary representation. How does this binary code behave at large distances? Let us attempt a heuristic argument. In a Reed-Solomon codeword, at most a k/n fraction of positions are zero, and these map to a sequence of zeros in the binary representation. For the nonzero positions, we may expect that their binary representation contains about the same number of zeros and ones. So we may expect the relative distance – that is the fraction of zeros in B – to be about $k/n + 1/2 \cdot (1 - k/n) = 1/2 + k/2n$. This suggests starting with a Reed-Solomon code with $k \approx \varepsilon n$ and $d \approx (1 - \varepsilon)n$.

However this intuition is incorrect, as we made the optimistic assumption that on average, a nonzero element of a Reed-Solomon codeword over $\mathbb{F}_{2^m}$ will have a balanced representation over $\mathbb{F}_2$ (with roughly the same number of zeros and ones). It is possible to construct Reed-Solomon codewords whose $\mathbb{F}_2$-representation is very sparse, and the relative distance of the resulting code will be far from half. How do we prevent such sparse representations from occurring? The idea is to encode the symbols in the Reed-Solomon encoding themselves by a new code.

Let us model this scenario more generally. We are given two codes: An outer code C_{out} over some alphabet Σ_{out}, and an inner code C_{in} that encodes messages of length k_{in} over some alphabet Σ, where each symbol of Σ_{out} is represented by k_{in} symbols in Σ. The codewords of the concatenated code $C = C_{\text{out}} \circ C_{\text{in}}$ are obtained by replacing the symbol of each codeword c_{out} of C_{out} by its encoding in C_{in}. We view C as a code over alphabet Σ.

Suppose C_{out} and C_{in} are $[n_{\text{out}}, k_{\text{out}}, d_{\text{out}}]_{\Sigma_{\text{out}}}$ and $[n_{\text{in}}, k_{\text{in}}, d_{\text{in}}]_{\Sigma}$ codes, respectively. Then the codeword length of C is $n_{\text{out}} n_{\text{in}}$ and its message length (over Σ) is $k_{\text{out}} k_{\text{in}}$. What about the distance? Two distinct codewords in C_{out} must differ in d_{out} positions, and each one of these gives rise to at least d_{in} differences in C, so the distance is also $d_{\text{out}} d_{\text{in}}$.

Theorem 3. *Suppose C_{out} is an $[n_{\text{out}}, k_{\text{out}}, d_{\text{out}}]_{\Sigma_{\text{out}}}$ code and C_{in} is a $[n_{\text{in}}, k_{\text{in}}, d_{\text{in}}]_{\Sigma}$ code where $\Sigma_{\text{out}} \equiv \Sigma^{k_{\text{in}}}$. Then $C_{\text{out}} \circ C_{\text{in}}$ is a $[n_{\text{out}} n_{\text{in}}, k_{\text{out}} k_{\text{in}}, d_{\text{out}} d_{\text{in}}]_{\Sigma}$ code.*

At large relative distances, notice that if the relative distance of the outer code is $1 - \varepsilon_{\text{out}}$ and the relative distance of the inner code is $1/2(1 - \varepsilon_{\text{in}})$, then the relative distance of the concatenated code is at least $1/2(1 - \varepsilon_{\text{out}} - \varepsilon_{\text{in}})$.

When C_{out} is linear over $\mathbb{F}_n$, $n = 2^{k_{\text{in}}}$ and C_{in} is linear over $\mathbb{F}_2$, you can check that $C_{\text{out}} \circ C_{\text{in}}$ will also be linear over $\mathbb{F}_2$ under the standard representation of elements in F_n as vectors in $\mathbb{F}_2^{k_{\text{in}}}$.

The advantage of concatenation is that it splits task of constructing a large code over a small alphabet, like $\mathbb{F}_2$ into two simpler parts: An outer code which should be large, but can use a large alphabet and an inner code which must be over a small alphabet, but can be small. For the outer code we have good candidates, like the Reed-Solomon code. Since the inner code is small, we may be able to find it by brute force search, or sometimes even highly suboptimal constructions may be adequate. Here are some examples:

- By concatenating the Reed-Solomon code $RS\colon \mathbb{F}_n^k \to \mathbb{F}_n^n$, $n = 2^m$, $k = \varepsilon n$ with the Hadamard code $Had\colon \mathbb{F}_2^m \to \mathbb{F}_2^n$ we obtain an $[n^2, \varepsilon n \log n, (1-\varepsilon)n^2/2]$ code.

- Take the Reed-Solomon code with $RS\colon \mathbb{F}_n^k \to \mathbb{F}_n^n$, $n = 2^m$, $k = \varepsilon n/2$. By Theorem 2, we know there exists an inner code GV that maps m bits into $O(m/\varepsilon^2)$ bits of relative distance $1/2 - \varepsilon/4$. The concatenated code maps $k = \varepsilon mn$ bits into $O(k/\varepsilon^3)$ bits and has relative distance $(1-\varepsilon)/2$.

  How do we get hold of such a code? The bulk of the work is in finding the required inner code GV. A brute force search over all such codes takes time $2^{O(m^2/\varepsilon^2)}$, and in the homework you will show an improvement to $2^{O(m/\varepsilon^2)} = n^{O(1/\varepsilon^2)}$.

  Later in the class we will see a faster (more explicit) construction which achieves comparable parameters.

4 Small-bias distributions

Let $C\colon \{0,1\}^k \to \{0,1\}^n$ be any binary linear code of relative distance $(1-\varepsilon)/2$. We can represent this code by a $k \times n$ generator matrix $C(x) = Mx$. Then the codewords of C are the different linear combinations of the rows of M. Take any nonzero such linear combination

$$\ell_a(x) = a_1x_1 + \cdots + a_kx_k$$

what happens when we evaluate this linear combination on the different columns of M? Since C has relative distance $(1-\varepsilon)/2$, it means that $\ell_a(x) = 0$ on at most a $(1-\varepsilon)/2$ fraction of columns x of M.

Now suppose that in addition of C having relative minimum distance $(1-\varepsilon)/2$, we could also guarantee that C has relative *maximum* distance $(1+\varepsilon)/2$. Then by the same reasoning we can say that

$$\frac{1}{2} - \frac{\varepsilon}{2} \leq \Pr_x[\ell_a(x) = 1] \leq \frac{1}{2} + \frac{\varepsilon}{2} \tag{1}$$

for any nonzero linear combination ℓ_a, where the probability is taken over the choice of a random column x of M. If x was a truly random $\{0,1\}$ vector, then the probability would be exactly $1/2$; so from the point of view of linear functions ℓ_a, a random column of M looks like a truly random vector in $\{0,1\}^n$.

Definition 4. A probability distribution $\mathcal{D}$ over $\{0,1\}^k$ is ε*-biased* if condition (1) holds for all nonzero linear functions ℓ_a.

Essentially all our discussion about codes so far applies to maximal distance as well as minimal distance and the constructions we discussed so far (the Hadamard code, the Gilbert-Varshamov argument, and the concatenations) yield small-biased distributions.

Here is one reason why condition (1) is interesting and significant. As we said, a truly uniform distribution is 0-biased. The converse is also true: If a distribution over $\{0,1\}^k$ is 0-biased, then it must be uniform. So we can view ε-biased distributions as approximations of the uniform distribution. However, to specify a uniformly random string in $\{0,1\}^k$ requires k bits of randomness; in contrast, to specify a sample from an ε-biased distribution we merely need to index a column of the generator matrix M, which requires $\log n$ bits, which is often much smaller. For example, if C is the concatenation of the Reed-Solomon and Hadamard codes, then $n \le (k/\varepsilon)^2$, so $\log n = O(\log(k/\varepsilon))$.

One specific property of the uniform distribution which we found useful for two-source extractors is that it is t-wise independent for every t. Small-bias distributions enjoy the following approximate t-wise independence property:

Lemma 5. *If the random variable X taking values in $\{0,1\}^k$ is ε-biased, then*

$$2^{-t} - \varepsilon \le \Pr[X_{i_1} = a_1 \text{ and } \dots \text{ and } X_{i_t} = a_t] \le 2^{-t} + \varepsilon$$

for every subset $\{i_1, \dots, i_t\} \subseteq [k]$ and values $a_1, \dots, a_t \in \{0,1\}$.

Proof. The trick is to use the identity $c = (1 - (-1)^c)/2$ for $c \in \{0,1\}$. Then condition (1) can be rewritten as

$$|\mathrm{E}[(-1)^{\langle a,X\rangle}]| = |\mathrm{E}[1 - 2\ell_a(x)]| = |1 - 2\Pr[\ell_a(x) = 1]| \le \varepsilon \quad \text{for every nonzero } a \in \{0,1\}^k.$$

Now to simplify notation let us assume $i_1 = 1, \dots, i_t = t$. We arithmetize as in the previous lecture:

$$\begin{aligned}
\Pr[X_1 = a_1 \cdots \text{ and } \cdots X_t = a_t] &= \mathrm{E}\Big[\prod_{j=1}^{t} \frac{1 - (-1)^{X_j + a_j}}{2}\Big] \\
&= 2^{-t} \sum_{S \subseteq [t]} \mathrm{E}\Big[\prod_{j \in S} (-1)^{j + X_j + a_j}\Big] \\
&= 2^{-t} \sum_{S \subseteq [t]} (-1)^{\sum_{j \in S} a_j}\, \mathrm{E}\big[(-1)^{\sum_{j \in S} X_j}\big].
\end{aligned}$$

The term $S = \varnothing$ gives a contribution of 2^{-t}. Each of the other $2^t - 1$ terms contributes at most $\varepsilon 2^{-t}$ in absolute value, so the value of the expression must be between $2^{-t} - \varepsilon$ and $2^{-t} + \varepsilon$. □

Now let us go back to the existence proof for two-source extractors and see what happens if the values of the candidate extractor f were chosen from an ε-biased distribution instead of the uniform one:

$$\begin{aligned}
\Pr[f \text{ is not a two-source hitter}] &= \Pr[\exists S, T, b\colon f(x,y) = b \text{ for all } x \in S, y \in T] \\
&\le \sum_{S,T,b} \Pr[f(x,y) = b \text{ for all } x \in S, y \in T] \\
&\le \sum_{S,T,b} (2^{-K^2} + \varepsilon) = 2\binom{N}{K}\binom{N}{K}(2^{-K^2} + \varepsilon).
\end{aligned}$$

if we chose $\varepsilon = 2^{-K^2}$ we would have to pay a factor of two in this probability, which you can check is negligible in the final estimate. Using the small-bias set construction based on the concatenation of Reed-Solomon and Hadamard codes, we can very efficiently obtain an ε-biased sample space of size $O(N^2 2^{2K^2})$. Recalling that $K = 2\log N + 1 = n + 1$, we conclude that the resulting search space has size $2^{O(n^2)}$, which is an improvement over the K^2-wise independent space of size 2^{n^3}.