Complexity Theory II Course Instructor: V. Arvind

Lecture 7 : ??, 2006

Lecturer: V. Arvind *Scribe: Srikanth Srinivasan*

1 Concatenated codes

In this lecture, we will look at *concatenated codes*, a method by which two different codes (with certain suitable parameters) can be composed to give a code over a more useful, smaller alphabet. More formally:

Assume the existence of an $[n_o, k_o, d_o]_{q_o}$ *outer code*, with encoding function C_o over a large alphabet Σ_o, and an $[n_i, k_i, d_i]_{q_i}$ *inner code* with encoding function C_i over a smaller alphabet Σ_i such that $q_o = q_i^{k_i}$. Then the *concatenation* of the two codes is an $[n_o.n_i, k_o.k_i, d_o.d_i]_{q_i}$ code over Σ_i, the encoding function (denoted by $C_o.C_i$) of which is computed as follows:

- The input is first divided into blocks of k_i symbols each. Each block can be considered to be a member of Σ_o, and the message itself to be a member of $\Sigma_o^{k_o}$. The encoding function C_o is applied to this message.
- The output of C_o is a member of $\Sigma_o^{n_o}$, and each symbol can be considered a member of $\Sigma_i^{k_i}$. The inner code is now applied to each symbol to obtain a member of $\Sigma_i^{n_i}$.
- The concatenation of the n_o messages obtained above is the output.

Firstly, let us look at the distance of $C_o.C_i$. Given distinct inputs x and y, the distance (over alphabet Σ_o) after the application of C_o is atleast d_o. That is, at least d_o blocks of $C_o(x)$ (interpreted over Σ_i) differ from the corresponding blocks of $C_o(y)$. Each of these blocks, after the second decoding phase, will differ in atleast d_i places. Hence, the distance between $C_o.C_i(x)$ and $C_o.C_i(y)$ is atleast d_od_i.

Also note that, assuming the existence of an efficiently computable bijection $\pi : \Sigma_o \mapsto \Sigma_i^{k_i}$, the concatenated code is efficiently encodable if both the inner and outer codes are. Note also that if C_o and C_i are both linear, and π is also linear, then so is $C_o.C_i$.

1.1 A simple decoding function

The decoding function does simply the opposite of the encoding function. It splits its input into n_o blocks of n_i symbols each and decodes each block using the decoding function for the inner code. It then interprets this output as a member of $\Sigma_o^{n_o}$, and decodes that using the decoding function of the inner code. This decoding function can correct up to $\frac{(d_o-1)(d_i-1)}{2}$ errors, which is not the maximum possible. More on this in the next lecture.

2 Justesen codes

Our goal in this section is to construct an $[n, Rn, \delta n]_2$ codes for some constants R and δ. To do this, we will use the idea of concatenated codes above. First, given any n_1 and n_2 we can construct two Reed-Solomon codes such that one is an $[n_1, n_1/2, n_1/2]_{n_1}$ code, and the other an $[n_2, n_2/2, n_2/2]_{n_2}$ code, such that both are codes over fields of characteristic 2. If we further set $n_1 = 2^{n_2}$, we clearly have a vector space isomorphism $\pi : \mathbb{F}_{n_1} \mapsto \mathbb{F}_{n_2}^{\log n_1/\log\log n_1}$. Hence, we can concatenate the two codes obtained above to obtain a linear code, which is a $[n_1n_2, n_1n_2/4, n_1n_2/4]_{n_2}$ code, i.e, an $[n, n/4, n/4]_q$ code, where $q < \log n$, which is very small in terms of the input length.

If we can further obtain a $[O(\log q), \log q, \frac{\log q}{c}]_2$ code, we will clearly be done (we can reach our goal by concatenating this code with the one we already have). But how do we obtain such a code? The answer lies in the Gilbert-Varshamov bounds, which assures us of the existence of such (linear) codes. Since the number of such (linear) codes is only $2^{O((\log q)^2)}$, i.e $2^{O((\log\log n)^2)}$, which is polylogarithmic in n, we can go through all possible codes and find the optimal code, which is guaranteed to be as efficient as outlined above. However, the *Justesen code* tells us that we can do the above even more easily. Instead of searching for the above code, one can simply use all possible codes (a different one for each symbol), and obtain a good code. To prove this, however, we need the following technical lemma:

Lemma 1. *There are* $2^{4l^2}(1 - \frac{1}{2^{\Omega(l)}})$ *linear* $[4l, l, l/8]_2$ *codes.*

Proof. Assume a random $4l \times l$ matrix C over $\mathbb{F}_2$ is picked. Given any $x \neq 0$ from $\mathbb{F}_2^l$, and any row of C, the probability that the inner product of the row with x is 0 is $\frac{1}{2}$. Now, using Chernoff bounds, we get

$$\Pr_C[wt(C(x)) < l/8] \leq 2^{-c.l}$$

where $c > 1$. Hence,

$$\Pr_C[\exists x \neq 0.wt(C(x)) < l/8] \leq 2^{-\Omega(l)}$$

Thus, we are done. $\square$

Returning to the Justesen code, let $q = 2^l$, where l is as above. Let $E_1, E_2, \ldots, E_{2^{4l^2}}$ be all the $4l \times l$ matrices over $\mathbb{F}_2$. The encoding function of the Justesen code $C : \mathbb{F}_2^{nl/4} \mapsto \mathbb{F}_2^{4nl}$ is computed as follows:

- Obtain outer code $[n, n/4, n/4]_q$ code as outlined above.
- Divide the input into blocks of l bits each, and obtain message in the outer alphabet. Encode using outer code. The output is a member of $\mathbb{F}_q^n$, and can be converted into a member of $\mathbb{F}_2^{nl}$.
- Divide the above into blocks of l bits and to the ith block, apply the encoding function $E_{i \bmod 2^{4l^2}}$. The resulting message, a member of $\mathbb{F}_2^{4nl}$, is the output.

Let us analyze the distance of the above encoding function. Clearly, at the last stage, at least $n/4$ of the n l-bit blocks are non-zero. The number of blocks that are encoded by codes that are *not* $[4l, l, l/8]_2$ codes is at most $n.\frac{1}{2^{\Omega(l)}}$. Hence, at least $n/4 - n/2^{\Omega(l)}$ of the non-zero blocks are encoded by $[4l, l, l/8]_2$ codes. Therefore, the number of non-zero bits in the output is atleast $(n/4 - n/2^{\Omega(l)}).\frac{l}{8}$, which is $\frac{nl}{32}(1 - \frac{1}{2^{\Omega(l)}})$. Hence, we have obtained a $[4nl, \frac{nl}{4}, \frac{nl}{32}(1 - o(1))]_2$ code.