9/22/2003

Notes for Lecture 8

In this lecture we see more applications of bipartite expander graphs to the construction of optimal up to constant factor error correcting codes with efficient encoding and decoding algorithms. We begin with an expander based construction of Spielman that allows both linear time encoding and linear time error correction from a small but linear fraction of errors over binary alphabets. We then increase the error correcting capabilities of these using a different expander based approach initiated by Alon et al. and perfected by Guruswami and Indyk. This construction yields linear time encoding and decoding algorithms for $[n, \Omega(\epsilon^2 n), d]_{2^{O(1/\epsilon^2)}}$ codes that fix up to $1/2 - \epsilon$ errors, and for $[n, \Omega(\epsilon^4 n), d]_2$ codes with $1/4 - \epsilon$ errors.

8.1 Spielman Codes

Recall that a (γ, δ) bipartite expander is a bipartite graph (L, R, E) such that for all $S \subseteq L$ of size at most $\delta|L|$, S has at least $\gamma|S|$ neighbors in R. For Spielman Codes, we will use expanders with $|L| = k, |R| = k/2$, constant degree D in L, degree $O(1)$ in R, constant δ, and $\gamma > \frac{3}{4}D + 1$.

We will use the expander to produce $k/2$ parity constraints between the bits of the message: Consider the function $R_k : \{0,1\}^k \to \{0,1\}^{k/2}$ such that $(R_k(x))_j = \bigoplus_{i \in \Gamma(j)} x_i$. One naive idea would be to use the encoding $x \rightsquigarrow (x, R_k(x))$, but this doesn't have good distance since flipping one bit of x cannot change the value of more than $D + 1$ bits in $(x, R_k(x))$. However, this mapping exhibits a weaker error correcting property, which will turn out useful nonetheless.

Lemma 1 (Main Lemma) *Let $a, x \in \{0,1\}^k, b \in \{0,1\}^{k/2}$. Call (a, b) "good" for x if $d(a, x) < \delta k/2$ and $e \triangleq d(b, R_k(x)) \leq \delta k/2$. There exists a linear time algorithm D_R which, on input (a, b) that is good for some x, outputs a string $x' \in \{0,1\}^k$ such that $d(x, x') < e/2$.*

We will prove this shortly, but first let us see how this helps us construct a family of optimal within a constant factor, binary, linear time encodable and decodable codes. The encoding will be the mapping $x \rightsquigarrow (x, C_k(x))$, where $C_k : \{0,1\}^k \to \{0,1\}^{3k}$ is given by the inductive formula

$$C_k(x) = (R_k(x), C_{k/2}(R_k(x)), R_{2k}(R_k(x), C_{k/2}(R_k(x)))).$$

This specifies the encoding only over input lengths that are a power of two, but can be extended to arbitrary lengths by padding without affecting the asymptotic properties of the code.

Let $T_E(k)$ denote the encoding time complexity on inputs of length k. By looking at the components of C_k, it is easy to see that $T_E(k) = T_E(k/2) + O(k)$, so (by Master's Theorem) encoding can be performed in linear time. For decoding the analysis is only slightly trickier. Consider the following decoding algorithm D_C:

On input $(x' \in \{0,1\}^k, r_1 \in \{0,1\}^{k/2}, c \in \{0,1\}^{3k/2}, r_2 \in \{0,1\}^k)$,

1. Apply the decoding algorithm D_R to $((r_1, c), r_2)$. Say the algorithm outputs $(r' \in \{0,1\}^{k/2}, c' \in \{0,1\}^{3k/2})$.

2. Apply the decoding algorithm D_C to (r', c'). (This is well defined by recursion, since (r', c') is half the size of the input we started with.) Say the algorithm outputs $r'' \in \{0,1\}^{k/2}$.

3. Apply D_R to (x', r'') and output the result.

Claim 2 *Algorithm D_C runs in linear time and recovers up to $\delta k/2$ errors.*

Proof: The running time T_D of the decoding algorithm satisfies a recursion of the form $T_D(k) = T_D(k/2) + O(k)$, so decoding can be done in linear time.

We show correctness by induction: Suppose the claim is true when the message length is $k/2$. Also suppose $x' \in \{0,1\}^k, c' = (r_1, c, r_2) \in \{0,1\}^{3k}$ satisfy $d((x', c'), (x, C_k(x))) < \delta k/2$. In particular, $d((r_1, c), (R_k(x), C_{k/2}(R_k(x)))) < \delta k/2$ and $d(r_2, R_{2k}(R_k(x), C_{k/2}(R_k(x)))) < \delta k/2$, so by the Main Lemma, after Step 1 of the algorithm, (r', c') is within $\delta k/4$ of $(R_k(x), C_{k/2}(R_k(x)))$. This falls within the realm of the inductive hypothesis, so after applying Step 2, it must be that $r'' = R_k(x)$. This, in turn, implies that (x', r'') is within $\delta k/2$ of $(x, R_k(x))$, so by the Main Lemma again, $D_R(x', r'') = x$. $\square$

We now prove the Main Lemma. The argument is similar to the correctness proof for Sipser-Spielman codes from last lecture. We associate a pair of strings $a \in \{0,1\}^k, b \in \{0,1\}^{k/2}$ with the vertices of the expander (L, R, E) by thinking of bits a_i and b_j as the labels of vertices $i \in L$ and $j \in R$, respectively. We think of the vertices of R as parity checks of their neighbors in L. Along these lines, we call a vertex $j \in R$ *satisfied* with respect to (a, b) if parity "checks out", i.e., $\bigoplus_{i \in \Gamma(j)} a_i = b_j$. Otherwise, we call j *unsatisfied*. The decoding algorithm is:

Algorithm. While there is a vertex $i \in L$ such that more than half of its neighbors are unsatisfied, flip the value a_i.

Since each step of the algorithm reduces the number of unsatisfied vertices, the number of steps the algorithm takes is at most linear. In addition, it is not difficult to find data structures with linear overhead that implement each step in constant time, yielding a linear time implementation.

We now show correctness. Fix a time in the execution of the algorithm and let V denote the set of vertices in L on which x and a differ, i.e., $V = \{i \in L : x_i \neq a_i\}$. Suppose that $|V| < \delta k$; the assumption of the Lemma guarantees that this is true at the beginning of the algorithm, but we will have to show that this invariant remains in power throughout. We partition $\Gamma(V)$ into (S, U), where S and U are the satisfied and unsatisfied vertices, respectively, with respect to (a, b) (at the current time.) Also, let $S' = \{j \in S : b_j = (R_k(x))_j\}$.

We compute the size of the cut $C = E(V, \Gamma(V))$. Counting from the left, we obtain $|C| \leq D|V|$. Counting from the right, we have the following: Any $j \in \Gamma(V)$ must have at least one neighbor in V. Moreover, if $j \in S'$, then j has at least two neighbors, so that

$$|C| \geq |\Gamma(V)| + |S'| \geq (|U| + |S|) + (|S| - e) = 2|S| + |U| - e.$$

Combining the bounds on C, we obtain, on the one hand, $D|V| + e \geq |U| + 2|S|$. On the other hand, by expansion $|U| + |S| = |\Gamma(S)| \geq (\frac{3}{4}D + 1)|V|$. This pair of inequalities implies $|U| \geq D|V|/2 + 2|V| - e$. If $|U| > D|V|/2$, then some vertex in $|V|$ has more than half of its neighbors unsatisfied, and the algorithm keeps going. So when the algorithm stops, we must have $|V| \leq e/2$.

It remains to show that $|V| < \delta k$ throughout the algorithm. Initially, there are at most $\frac{1}{2}D\delta k + \frac{1}{2}\delta k$ unsatisfied vertices in R, and this quantity decreases monotonically. By an edge counting argument, as long as $|V| \leq \delta k$, $\Gamma(V)$ has at least $(\frac{1}{2}D + 2)|V|$ unique neighbors in V. In particular, if $|V|$ ever becomes δk, there would be as many as $(\frac{1}{2}D + 2)\delta k$ unique neighbors in V, which implies at least $(\frac{1}{2}D + 2)\delta k - e \geq \frac{1}{2}D\delta k + \frac{3}{2}\delta k$ unsatisfied vertices in R. Contradiction.

8.2 Improving the error correcting capabilities

Spielman codes are close to optimal in many ways, but can correct only a small fraction of errors. As we must have $\gamma > \frac{3}{4}Dk$ for the argument to work, it cannot be that $\delta \geq \frac{3}{4D}$, which even for optimal D is much smaller than $\frac{1}{4}$. Therefore, to improve the error correcting capability, we will need to use a new construction.

In general, suppose we have an $[n, k, d]_2$ code C with a linear time decoding algorithm that can correct up to a δ fraction of errors. Using this code, we will construct a new code C' over a bigger but constant size alphabet whose rate is worse by a constant factor but can correct up to a $1/2 - \epsilon$ fraction of errors. The basic block in the construction is a bipartite graph (L, R, E) with $|L| = |R| = n$ and constant odd degree D on both sides. We will need this graph to have certain random-like properties which we will specify shortly.

First let us describe the construction. Let $v[t]$ denote the tth neighbor of vertex v (in an arbitrary ordering). Think of the ith vertex in L as labeled by the symbol $(C(x))_i$ and the jth vertex in R as labeled by $(C'(x))_j$. The alphabet of C' is $\{0,1\}^D$, which we interpret as a string of D bits. The jth symbol of $C'(x)$ is obtained by concatenating the symbols at js neighbors:

$$(C'(x))_j = ((C(x))_{j[1]}, (C(x))_{j[2]}, \ldots, (C(x))_{j[D]}).$$

Intuitively, by replicating and mixing the symbols of $C(x)$ on the right side of the partition, we reduce the risk of corrupting a symbol. The recovery is straightforward: Given a possibly corrupted word $w \in (\{0,1\}^D)^n$, we label R with the symbols of w, and we label each $i \in L$ with the *majority* of the bits corresponding to its neighbors. More precisely, if i is the $n(i,t)$th neighbor of $i[t]$, we label $i \in L$ by u_i, where

$$u_i = \text{majority}(w_{i[1]}[n(i,1)], \ldots, w_{i[D]}[n(i,D)]).$$

Once we complete this labeling of L (which takes time linear in n), we pass the string u to the decoding algorithm for C and output the result.

Now suppose that we start with a word $w \in (\{0,1\}^D)^n$ at distance $(\frac{1}{2} - \epsilon)n$ from a codeword $C'(x)$. To show correctness of the decoding procedure, it will be sufficient that the word u is within $\delta n/2$ of the codeword $C(x)$. If S and T are the sets of corrupted bit indices for u and w respectively, it must be that each $i \in S$ neighbors more than $D/2$ elements of T, so that the size of the cut $E(S,T)$ is at least $|S|D/2$. Therefore a sufficient condition for successful decoding is the following:

Goodness condition. The graph (L, R, E) is *good* if for all $S \subseteq L, T \subseteq R$ such that $|S| > \delta n/2$ and $|T| \leq (\frac{1}{2} - \epsilon)n$, we have $|E(S,T)| < |S|D/2$.

The existence of good graphs with $D = 4/\epsilon^2\delta$ will follow from two lemmas which we won't prove. Let $\lambda_1 \geq \ldots \geq \lambda_n$ denote the eigenvalues of an n vertex graph and $\lambda(G) \triangleq \max\{\lambda_2, |\lambda_n|\}$.

Lemma 3 *For arbitrarily large constant D, we can construct in polynomial time a D-regular graph G on n vertices such that $\lambda(G) \leq 2\sqrt{D}$.*

Lemma 4 *Given a D-regular graph G, let (L, R, E) be the bipartite graph with $L = V(G), R = V(G)$ and $E = E(G)$. Then for all $S \subseteq L$ and $T \subseteq R$,*

$$||E(S,T)| - |S||T|D/n| \leq \lambda(G)\sqrt{|S||T|}.$$

The last condition says that the graph should look somewhat "random-like": If (L, R, E) were random bipartite D-regular, we would expect exactly $|S||T|D/n$ edges between S and T, and the standard deviation would be roughly $\sqrt{|S||T|D/n}$.

In our application, we have $|T| \leq (\frac{1}{2} - \epsilon)n$, so that

$$||E(S,T)| - |S|D/2 + |S|\epsilon D| \leq 2\sqrt{D|S|(\frac{1}{2} - \epsilon)n}.$$

So $|E(S,T)| < |S|D/2$ will hold as long as $2\sqrt{D|S|n/2} < |S|\epsilon D$, or $D > 2n/S\epsilon^2 \geq 4/\delta\epsilon^2$.

In the Spielman code δ is constant, so we obtain codes over alphabet size $2^{O(1/\epsilon^2)}$. Over this alphabet, each input symbol consists of $1/\epsilon^2$ bits, so that C' is a $[n, \Omega(\epsilon^2 n), d]_{2^{O(1/\epsilon^2)}}$ code that can correct up to $(\frac{1}{2} - \epsilon)n$ errors in linear time.