6.897 Algorithmic Introduction to Coding Theory	November 20, 2002
Lecture 19	
Lecturer: Amir Shpilka	*Scribe: Jonathan Herzog*

The subject today was a construction of a code with a linear-time encoding algorithm and a linear time list-decoding algorithm. The construction proceeds in three parts:

1 Yet Another Code Based On Expander Graphs

The first step is to consider another code based on expander graphs, introduced by Alon, Bruck, Naor, Naor and Roth in 1993. The code is made from two components:

- A code $[n, k, \delta n]_2$ code C, and
- A c-regular, (γ, δ)-weak expander, (δ', ϵ')-mixer bipartite graph $G = (A, B, E)$ (where $|A| = |B| = n$).

This begs the question: what is are weak expander and mixer graphs? A weak expander graph is one that expands only sufficiently large sets of nodes:

Definition 1 *A bipartite graph $G = (A, B, E)$ is a (γ, δ)-weak expander if for every set $T \subseteq A$ where $|T| \geq \delta n$, $|\Gamma(T)| \geq \gamma\delta n$.*

(Recall that $\Gamma(T)$ is the set of neighbors of T.) A graph is a mixer if most left-hand nodes touch a large number of the right-hand nodes in an arbitrary set T, given that T is sufficiently large:

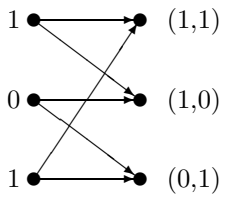
Definition 2 *Fix c. A bipartite graph $G = (A, B, E)$ is a (δ', ϵ')-weak mixer if for every set $T \subseteq B$ where $|T| \geq (\frac{1}{2} + \epsilon')n$, $\left|\left\{a \in A : |\Gamma(T) \cap T| < \frac{c}{2}\right\}\right| \leq \delta' n$.*

Once we have this graph G, we can compose it with C to produce $G(C)$, a $[n, \frac{k}{c}, \delta\gamma n]_{2^c}$ code. Note that we increase the alphabet, but we also increase the distance. How do we do this composition?

To encode a message m:

- First, m is a $\frac{k}{c}$-vector of elements in $\mathbf{GF}(2^c)$. Re-interpret the message as a k-vector of elements in $\mathbf{GF}(2)$.
- Encode the re-interpreted message using the encoding algorithm of C to get $C(m)$.
- Now, label each of the nodes in A, the left-hand size of G, with one of the coordinates of $C(m)$.
- For each of the nodes in B, the left-hand side of G, list the labels of its neighbors in some canonical order.

  For example: in the graph below, the nodes on the left are labeled $(1, 0, 1)$. The nodes on the right are labeled with a list of their neighbors' labels, higher before lower.



- Because the graph is c-regular, each nodes has exactly c neighbors. Hence each list on the right-hand side has c elements, and can be interpreted as an element of $\mathbf{GF}(2^c)$. The codeword is the n-vector of $\mathbf{GF}(2^c)$ elements created by reading down the right-hand side.

The length of this code is n. Since the message length of C is k, and we interpret it in $\mathbf{GF}(2^c)$, we divide the message length by c to produce a message length of $\frac{k}{c}$. As for the minimum distance, suppose that two messages m_1 and m_2 differ in at least one component. Then when they are interpreted in $\mathbf{GF}(2)$, they will continue to differ in at least one coordinate. Since the code C has a minimum distance of δn, $\Delta(C(m_1), C(m_2)) \geq \delta n$. Since the graph is a (γ, δ)-weak expander, at least $\gamma\delta n$ of the right-hand lists will see different labels on the left-hand side. Hence that many lists will differ in one place, and so that many of the $\mathbf{GF}(2^c)$ elements on the right-hand-side will differ. Hence, the minimum distance of $G(C)$ will be $\gamma\delta n$. Lastly, note that if C has a linear-time encoding algorithm, then so goes $G(C)$.

What about decoding? Suppose that we take in some corrupted codeword that has less than $\left(\frac{1}{2} - \epsilon'\right) n$ errors.

- Re-label each right-hand node of G with a $\mathbf{GF}(2^c)$-element.
- Re-interpret each element as a length-c list of bits.
- Each right-hand node will have an opinion about the labels of its c neighbors on left-hand side. Similarly, for each left-hand node, there will be c opinions about its label from its right-hand neighbors.
- For each left-hand node, label it with the *majority* of the relevant opinions.
- This gives you an n-bit vector. Apply the decoding algorithm for C to get a k-bit vector.
- Re-interpret as a $\frac{k}{c}$-list of elements from $\mathbf{GF}(2^c)$.

Will this work? We know that there are less than $\left(\frac{1}{2} - \epsilon'\right) n$ errors. Let T be the set of nodes with *correct* labels, of which there are more than $\left(\frac{1}{2} + \epsilon'\right) n$. Since the graph G is a (δ', ϵ')-weak mixer, we know that there are only $\delta' n$ nodes on the left-hand side which do not have over $\frac{c}{2}$ neighbors in T. All the nodes with more than $\frac{c}{2}$ neighbors in T will get the correct label by majority vote. Hence, at most $\delta' n$ nodes on the left-hand side will get in incorrect label. If $\delta' \leq \frac{\delta}{2}$, then we can apply the decoding algorithm of C to correct those remaining errors.

Note that if C has a linear-time decoding algorithm that corrects a δ'-fraction errors, then $G(C)$ has a linear-time decoding algorithm that corrects $(\frac{1}{2} - \epsilon')$-fraction errors.

What are the parameters of this?

- Suppose that $\frac{k}{n}$ is a constant.
- Suppose that $\delta = \Omega(1)$ is a constant as well.
- The normalized distance is $\gamma\delta$. If you set $c \gg \frac{1}{\delta}$ then you get that $\gamma\delta \leq 1 - O(\frac{1}{c})$. Why can it not reach 1?

  Pick a set of $\frac{n}{2c}$ nodes on the right-hand side. Because the graph is c-regular, they can have at most $\frac{n}{2}$ neighbors on the left-hand side. Pick a set of δ nodes from outside this set of neighbors. They can be neighbors to at most $(1 - \frac{1}{2c})n$ nodes on the right-hand side. If two codewords m_1 and m_2 produce a node-labeling on the left-hand nodes that differ in those δ places, then the labeling on the right will not differ in the $\frac{n}{2c}$ places we chose at the beginning of this paragraph. Hence, the distance can not be more than $(1 - \frac{1}{2c})n$.

So, letting $\gamma\delta = 1 - \epsilon$:

- The rate is $\Omega(\epsilon)$,
- The distance is $1 - \epsilon$, and
- The alphabet size is $2^{O(\frac{1}{\epsilon})}$

Hence, these codes are almost as good as the Gilbert-Vashamov bounds allow.

2 List Decoding

We would like a code that has a list-decoding algorithm that runs in linear time. To pull off this remarkable feat, we need a new definition:

Definition 3 *A code C is (α, l, L)-recoverable if given n sets S_1, S_2, $\ldots S_n$ (where each $S_i \subseteq \Sigma$) of size at most l, there are at most L codewords $c \in C$ such that $c_i \in S_i$ for at least αn indices.*

In other words, given sets S_1 through S_n, there are few codewords with many coordinates in the corresponding sets. Note that if the sets are of size 1, then this is exactly the definition of list decodable.

Theorem 1 *If C is $(1-\epsilon, l, L)$-list recoverable and G is a mixer, then $G(C)$ is $(1-(\frac{1}{l+1}+\epsilon'))$-list decodable.*

Proof Suppose that we get in a word w with coordinates $w_1 \ldots w_n$. Let α be a codeword that agrees with w in at least $(\frac{1}{l+1}+\epsilon')n$ coordinates. We want to find α.

How to do this? Since we are attempting to do this for $G(C)$, we start by looking at the graph. Label the right-hand sides as usual, which gives us c opinions about the value of each left-hand node. For each left-hand node, make a list of the l most popular opinions.

Will α have a coordinate in enough of the sets S_i? Let T be the set of right-hand nodes where $w_i = \alpha_i$. Now, consider a left-hand node. How many of its neighbors will be in T? If G is random, then we would expect each left-hand node to have $c\frac{|T|}{n} = c\left(\frac{1}{l+1}+\epsilon'\right) > \frac{c}{l+1}$ of its neighbors in T. Furthermore all the nodes in right-hand nodes in T agree on the label for the left-hand node in question. Hence, that label will be among the l most popular opinions. Hence, we would expect α_i to be in S_i all the time.

But G is not random; it is constructed. How do we get around this? We use the fact that G is a mixer. If G is a mixer then the above situation occurs often. The number of left-hand nodes with less than $\frac{c}{l+1}$ neighbors in T will be small: δ in fact.

Hence, every codeword α that is $(\frac{1}{l+1}+\epsilon')$ close to c will have its coordinates in $1-\delta$ of the sets S_i. Then, if C is list-recoverable, it can pick out the appropriate labels from the sets $S_1 \ldots S_n$, and then we can use the labels to generate α. ■

Note that if C has a linear-time list-recovering algorithm, then $G(C)$ has a linear-time list-decoding algorithm.

3 Constructing List-Recoverable Codes

Great. If we can make a linear-time list-recoverable code (and a mixing, regular, and expanding bipartite graph) then we've got a linear-time list-decodable code. Can we make a list-recoverable code that can be recovered in linear time?

Here, we will only make a $(1,2,2)$-list recoverable code.

Theorem 2 *Take any code C that can be encoded in linear time and can be decoded from 90% erasures. Let G be a good expander. Then $G(C)$ is a $(1,2,2)$-recoverable code.*

Proof What do we want? We want to take in n sets, each of two elements. Then we want to pick an element of each set so that we get a codeword at the end.

How do we do this? We pick elements and use them to label the right-hand side of G. We use that to determine come of the left-hand side of G.If we get enough of the left hand side, like 10%, then we've got a codeword of C with 90% erasures. We reconstruct the codeword, and feed it back through G to get the codeword of $G(C)$.

So, all we need to do is get 10% of the left-hand side of G right. Do so in the following way:

- Look at an arbitrary set $S_i = \{s_1, s_2\}$. We have to choose one of them. If s_1 and s_2 agree about the label of some left-hand node n, then it doesn't matter which one we pick. The label of n is determined. If we can get 10% of the left-hand side in this way, then we're done.

- Otherwise, remove all the left nodes with determined labels. (We will see why in one second.)
- Now choose one of the right-hand nodes arbitrarily. Associated with that node is a set of two elements. Choose on of the elements arbitrarily.
- That element has opinions about the labels for c of the left-hand nodes. Call this set T and label those nodes according to the element we just chose.
- Now look at the right-hand neighbors of T. Since we threw out all the determined nodes, no node in T was determined by any of its neighbors. Furthermore, this is a $(1, 2, 2)$-recoverable code, and so *all* the left-hand nodes must be consistent with right-hand nodes. Hence, the labels we just applied to T determine which elements we choose for the sets associated with $\Gamma(T)$.
- The elements we chose for $\Gamma(T)$ determine the labels for $\Gamma(\Gamma(T))$, which determine the elements chosen for nodes in $\Gamma(\Gamma(\Gamma(T)))$, and so on.
- In this way, we determine all the left-hand nodes for a connected sub-graph of G. If that sub-graph contains 10% of the left-hand nodes, we recreate the complete labeling on the left by reconstructing from 90% erasures.
- If the sub-graph doesn't contain 10% of the left-hand nodes, we repeat by picking an arbitrary right-hand node *not* in the sub-graph we just found.
- Will we eventually find a large enough sub-graph? Yes, because the graph is a regular mixing expander. (Trust me.) Hence, we will eventually be able to reconstruct from 90% erasures.

So, this allows us to extract one codeword from the right-hand sets. How do we show that there are only two such codewords? Recall that we started by choosing an arbitrary right-hand node, and then picking an arbitrary element from that node's set. From that point on, the process of recovering the codeword was deterministic. Hence, if we chose the *other* element from that set, we would have found at most one other codeword. So, there are at most two codewords that we could recover, making this a $(1, 2, 2)$-recoverable set. ■

Notes: The construction of codes based on weak expanders is due to Alon, Bruck, Naor, Naor and Roth (IEEE Transactions on Information Theory, ca. 1993). The decoding algorithms (both the unambiguous decoding and the list-decoding versions) for codes based on these constructions are due to Guruswami and Indyk (FOCS 2001, STOC 2002 and more to come). The notion of list-recoverable codes is also due to Guruswami and Indyk.