COMPSCI 690T Coding Theory and Applications Apr 5, 2017

Lecture 18

Instructor: Arya Mazumdar *Scribe: Names Redacted*

1 Distributed Storage Systems

1.1 Introduction

In a distributed storage network, data is encoded and stored on a set of servers (nodes). Whenever a node fails, error correction can be used to recover the lost data. In the discussion that follows, we consider only data erasures (and not data corruption). Consider a distributed storage network with n nodes. Let k denote the amount of data we would like to encode; for example, a file could be divided into k equal parts, encoded into n fragments using an $[n, k]$-code, and stored on the n servers. Then, lost data can be recovered by k of the n servers.

Recall that for a code with minimum distance d, the number of erasures that can be corrected is $d - 1$. Recall the fundamental limit imposed by the Singleton Bound:

$$d - 1 \leq n - k$$

Or, if we use an MDS code:

$$d - 1 = n - k$$

In practice, there are various encoding techniques utilized on distributed storage systems. For example, Google uses triplication (*i.e.* a repetition code where the number of identical copies is three). Another example is Facebook, which uses an $[n, k] = [14, 10]$-RS code (*i.e.* if up to four servers fail, data can still be recovered).

1.2 Locality

Consider two examples of encoding schemes:

(a) A simple repetition scheme. Make one copy of every file. For example, if A and B are files, we have:

$$\boxed{A}\boxed{B} \longrightarrow \boxed{A}\boxed{B}\boxed{A}\boxed{B}$$

This scheme protects against one erasure per file.

(b) A parity check code. The files A and B are encoded as follows:

$$\boxed{A}\boxed{B} \longrightarrow \boxed{A}\boxed{B}\boxed{A+B}$$

Note that scheme ***(b)*** uses storage more efficiently. However, if either of the files A or B gets lost, more computation is required to recover the lost data than in scheme ***(a)***. This brings us to the concept of *locality*. Locality is the number of other symbols needed in order to recover an erased symbol. In scheme ***(a)***, the locality is one, since if, for example, A is erased, only one file (the copy of A) is necessary for the recovery. In scheme ***(b)***, the locality is two, since if, for example, A is erased, we would need $A + B$ and B in order to recover A.

We can incorporate this new notion into a coding scheme: let an $[n, k, d, r]$-code be an $[n, k, d]$-code with locality r. How does this new parameter, r, affect the limit on the rate of the code? Suppose that there is no requirement on the minimum distance of the code (ignore the d parameter). Assume that for

an erasure, there exist r servers which can be contacted for recovery. Here is a possible construction to achieve this property: for every $r+1$ servers, perform the parity operation. Then the rate is

$$R = \frac{k}{n} = \frac{r}{r+1},$$

since for every r bits of information, we add a bit containing their parity information. We will show that a scheme that achieves this rate is optimal.

Claim 0: The rate of any code with locality r is $\leq r/(r+1)$.

Proof: Consider a graph, G, with n vertices (each representing a server). Since, the locality is r, all servers can go to r other servers to recover data. To represent these dependencies, for each node, add outgoing edges to the r nodes that are used for the node's data recovery. The resulting graph is directed and r-regular in outgoing edges. We first make and prove two more claims:

Claim 1: If there exists a directed acyclic subgraph (DAG) in G of size L, then $k \leq n - L$ (Note: k is the amount of information that is being stored on n servers. Also note: L depends on r).

Claim 2: There exists a DAG of size $\frac{n}{r+1}$.

To prove **Claim 1**, we make the following observation: if there exists a DAG G' in G, then there must exist a node in G' such that all of its r outgoing edges are going outside of the subgraph G'; otherwise, there would exist a cycle in G'. Suppose that we have found such a node, v. This means that all of the information necessary to recover the data on v lies outside of the subgraph G'. Suppose we remove v from G'. But now, there must be some other node in the new DAG subgraph with the property that all of its r outgoing edges lie outside of the subgraph. We can remove this node as well, and iterate until there are no nodes left to remove. From this argument, we see that the amount of information that can be stored is upper-bounded by the maximum possible size of a DAG subgraph of G. In other words, $k \leq n - L$.

To prove **Claim 2**, we will use the probabilistic method. We begin by taking a random and uniform permutation of $\{1, 2, \ldots, n\}$ and assigning these values to the nodes (*i.e.* we randomly permute the vertices of G). Next, for a vertex v, if the value it was assigned is higher than the values of all of its neighbors (note: we are referring to the r neighbors that v uses for data recovery), then we include v in the subgraph. Repeat this step for all vertices. Note that the resulting graph is a DAG, since by its construction, it is not possible for a lower-indexed node to have an edge to a higher-indexed node. Next, let x_1, x_2, ..., x_n be indicators s.t.:

$$x_i = \begin{cases} 1, & \text{if the } i\text{th node is in the subgraph} \\ 0, & \text{otherwise} \end{cases}$$

Then the size of the subgraph is

$$\sum_{i=1}^{n} x_i,$$

and the average size of the subgraph is

$$E\left[\sum_{i=1}^{n} x_i\right] = \sum_{i=1}^{n} E[x_i] = nP(x_i = 1)$$

$P(x_i = 1)$ is the probability that the ith node is included in the subgraph, or, in other words, the probability that i's neighbors have lower values than i. Since we indexed the nodes uniformly and at random, this probability is simply $\frac{1}{r+1}$ (*i.e.* the probability that i is the largest of a total of $r+1$ nodes). Hence, the expected subgraph size is $\frac{n}{r+1}$. This means that there must exist some subgraph with a size $L \geq \frac{n}{r+1}$. This concludes the proof of **Claim 2**.

Combining the two claims, we have:

$$k \leq n - \frac{n}{r+1} = n\left(\frac{r}{r+1}\right)$$
$$R = \frac{k}{n} \leq \frac{r}{r+1}$$

This concludes the proof of **Claim 0**.

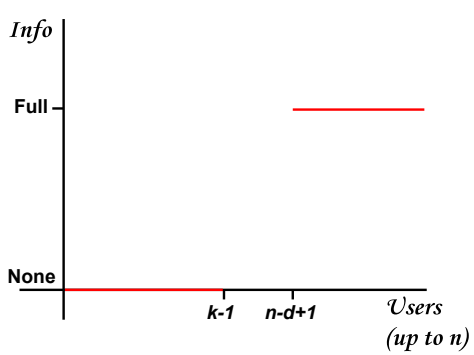
Here is a claim on the distance of locally reparable codes:

$$d \leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2$$

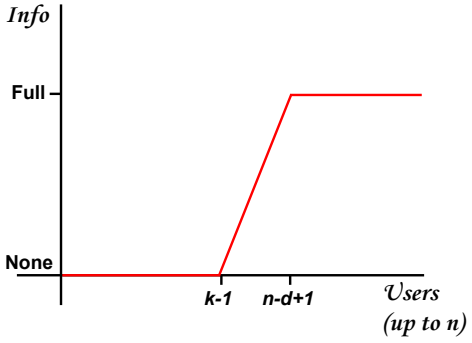
Proof is to come later in class (or you can refer to [1]). Note: the intuition is that $\left\lceil \frac{k}{r} \right\rceil$ is the penalty we pay for locality r.

2 Back to Secret Sharing

So far, we have discussed secret sharing schemes in which absolutely no information is revealed about the universe for less than the required number of cooperating users. The plot in Figure (1a) illustrates this idea. Another secret sharing scheme, known as the Ramp Scheme, allows some of the information to be revealed for less than the required number of users. The rate of the information revealed grows linearly, as shown in Figure (1b).



(a) A scheme where no information is revealed until there are at least k cooperating users.



(b) A scheme where information is revealed at a linear rate for less than the number of users required to reveal the entire secret.

Figure 1: Two different secret sharing schemes.

Recall that linear MDS codes can be used to implement the Threshold Secret Sharing scheme: if any $< k$ of n users collude, they get no information at all, and if any $\geq k$ users collude, they obtain the secret. Consider a code with length $n+1$ and dimension k. Assign the secret to the first coordinate, and random symbols to the remaining $k-1$ coordinates. Call this k-length row vector $\mathbf{v}$. Then the codeword is obtained in the usual manner: $\mathbf{s} = \mathbf{v}G$, where G is the $k \times n+1$ generator matrix of the code, and all but the first element of $\mathbf{s}$ are the n shares to be distributed to the users.

In general, suppose we have a codeword whose first symbol is the secret and $k-1$ symbols are chosen uniformly at random, such that the coordinates of these symbols, along with the first coordinate, form an information set. Then these k coordinates determine the remaining symbols of the codeword. Let

$\mathbf{s} = (s\ s_1\ s_2\ \dots\ s_n)$ be this codeword, where $s_1, \dots s_n$ are the shares. Then the minimal access structure is given by the minimal codewords of the dual code [2]. Suppose that $\mathbf{c} = (c_1\ c_2\ \dots\ c_{n+1}) \in \text{span}(H)$. By the definition of dual code, we have $\langle \mathbf{s}, \mathbf{c} \rangle = 0$.

$$sc_1 + s_1c_2 + \cdots + s_nc_{n+1} = 0$$

If $\mathbf{c}$ is a minimal codeword, then its leftmost value is 1 ($c_1 = 1$). Then we can decode s as follows:

$$s = -(s_1c_2 + \cdots + s_nc_{n+1})$$

Note that the dual code is known (it is public information), so that a group of users who form a set in the access structure can decode the secret this way.

3 Wiretap Channel

We introduce the wiretap channel in this section (more on this in the next lecture). The setup is as follows: k bits of information are encoded into n bits. Alice sends this codeword over a noiseless channel to Bob. However, Eve can intercept the codeword on this channel and look at any m bits of the transmission (she can choose which m bits to look at). The goal is to minimize the amount of information leakage. We will show that Eve gets no information if $k \to 2k$ and $m = k$.

References

[1] Gopalan, Parikshit, et al. "On the Locality of Codeword Symbols." IEEE Transactions on Information Theory 58.11 (2012): 6925-6934. https://arxiv.org/pdf/1106.3625.pdf

[2] Massey, James L. "Minimal Codewords and Secret Sharing." Proceedings of the 6th Joint Swedish-Russian International Workshop on Information Theory. 1993.