CSE522/Math581EB Coding Theory **Fall 2001**

Lecture 1: October 1

Lecturer: Chris Umans *Scribe: Chris Umans*

If you did not receive a syllabus, copies are available from the course web site:

`http://www.research.microsoft.com/~umans/522/course.htm`

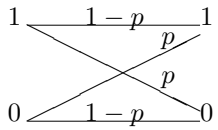
In this first lecture we'll give some background, notation, and then see two example applications.

1.1 Some background and basic notation

The basic scenario is the following. We want to send information over a noisy channel. The idea is to use encoding and decoding functions (E and D, respectively) to overcome the errors introduced by the channel:

$$\text{message} \longrightarrow E \longrightarrow \text{codeword} \longrightarrow \text{noisy channel} \longrightarrow \text{received word} \longrightarrow D \longrightarrow \text{message}$$

How do we model the noisy channel? In his 1948 paper, Shannon [S48] introduced a class of models for this purpose. One of the simplest (but still interesting enough to be the focus of a large portion of modern coding theory) is the *binary symmetric channel* (BSC). The picture associated with this channel is:



where p is a probability. In general, the labels on the left represent the transmitted symbols, and the labels on the right represent the received symbols. In the binary symmetric channel, the transmitted and received symbols are bits (0 or 1), and they are "flipped" with probability p, and remain intact with probability $1-p$.

Shannon proved the "Noisy Coding Theorem," which we will see in its full glory in the next lecture. For now, we state (slightly informally) a special case relevant to the binary symmetric channel above: for all $p < 1/2$, there exists a constant c an encoding function $E : \{0,1\}^k \to \{0,1\}^c k$ and a decoding function $D : \{0,1\}^{ck} \to \{0,1\}^k$ such that a random message $m \in \{0,1\}^k$ is recovered with probability $1 - o(1)$ as $k \to \infty$. The important thing here is that the encoding needs only increase the message length by a constant in order to overcome the noise in the channel. In fact, we will see that any c such that $1/c$ is less than the channel *capacity*, which is $1 - H(p)$, works. (We will define capacity and the *binary entropy* function $H(p)$ later).

However, the encoding and decoding functions given by Shannon's theorem are not at all nice – E is a random function, and D requires exponential time to compute. To use codes in practice we need an explicit construction of a code with efficient encoding and decoding functions. The first example of such a code was given by Hamming in 1950 [H50]. A couple of observations are important here:

- Rather than analyzing the probability of recovering a message using the model of a noisy channel above, it is easier to aim for *guaranteed recovery* if fewer than t errors occur. Notice that for $t \approx p \cdot$ (length of codeword), such a guarantee also ensures that the message is recovered with high probability in the BSC model.

- Two desirable properties of a code depend on the *distance* of a code. If the code is *t-error-detecting* – it can determine if at most t errors have occurred – then we cannot have two codewords x and y that differ in $\leq t$ positions (otherwise t errors can turn x into another valid codeword y and we won't notice that errors have occurred). If the code is *t-error-correcting* – it can correct at most t errors in the received word – then we cannot have two codewords x and y that differ in $\leq 2t$ positions (otherwise there is some z such that t errors can turn x into z and y into z, and then there is no way to know if z came from x or y).

This second item justifies minimum distance as one of the four main parameters of a code. We proceed with some formal definitions.

Definition 1.1 *Let Σ be a finite alphabet. A* code *is a subset $C \subseteq \Sigma^n$. The elements of this set are* codewords*.*

It will be convenient to think of this subset as the range of an encoding function $E : \Sigma^k \to \Sigma^n$. The *Hamming distance* (or just *distance*) between two strings in Σ^n is the number of positions in which they differ. We write $\Delta(x, y)$ for the distance between x and y.

Definition 1.2 *The* distance *of a code C, denoted $\Delta(C)$ is the minimum distance between any pair of codewords; i.e., $\Delta(C) = \min_{x \neq y \in C} \Delta(x, y)$.*

The opposing goals of the code designer are to make the distance of the code large while keeping the blocklength small. In summary there are four main parameters of a code:

- *blocklength n*
- *message length k*
- *distance d*
- *alphabet size q.*

We often use the shorthand "C is an $[n, k, d]_q$ code" to mean C is a code with blocklength n, dimension k, and distance d over an alphabet Σ of q elements. A useful rule-of-thumb is that q-ary codes can achieve minimum distance close to $(1 - 1/q)n$, while other parameters remain "reasonable." The fact that larger distances can be achieved with larger alphabets is one of the reasons it is useful to consider codes with $q > 2$.

Two additional quantities of interest are

- *rate k/n*
- *relative distance d/n*

A major goal is to build families of codes with *constant* rate and *constant* relative distance; such codes are called *asymptotically good*.

1.2 Application: perfect hash functions

Without knowing anything more than the existence of efficiently encodable and decodable codes with constant relative distance, we can present a simple application.

A function $h : \{0,1\}^k \to \{0,1\}^\ell$ is called a perfect hash function with respect to $T \subseteq \{0,1\}^k$ if $h(x) \neq h(y)$ for all $x \neq y \in T$. Clearly this requires that $\ell \geq \log_2 |T|$, and the goal is to have an efficiently computable function in which ℓ is as close to this lower bound as possible.

In some applications, it is easier to have a family of efficiently computable functions:

$$H = \left\{ h_i : \{0,1\}^k \to \{0,1\}^\ell \right\}$$

with the property that for all $T \subseteq \{0,1\}^k$ with $\log_2 |T| \leq t$

$$\Pr[h_i \text{ is a perfect hash function with respect to } T] \geq 1/2. \tag{1.1}$$

That is, no matter what T is given to us, picking a random $h_i \in H$ is likely to give us the desired hash function. Again, we want ℓ as close to t as possible.

A well known solution to this problem is to take H to be a "universal" family of hash functions. We'll see an alternative approach based on codes that has some advantages, especially when the aim is to have the h_i extremely easy to compute.

Let $E : \{0,1\}^k \to \{0,1\}^n$ be the encoding function for an $[n, k, \delta n]_2$ code for some constant δ. For each multiset S of $[n]$ of size ℓ, we define $h_S : \{0,1\}^k \to \{0,1\}^\ell$ by:

$$h_S(x) = \{E(x)_i\}_{i \in S}.$$

That is, to compute h_S, we encode x, and then select the ℓ positions indexed by S as our ℓ-bit output. We take H to be the collection of all such h_S.

We claim that H satisfies (1.1) for $\ell = O(t)$. Fix $T \subseteq \{0,1\}^k$ with $\log_2(|T|) \leq t$. For $x \neq y \in T$, we have:

$$\Pr_S[h_S(x) = h_S(y)] \leq (1-\delta)^\ell.$$

This is true because $\Delta(E(x), E(y)) \geq \delta n$, or equivalently, $\Pr_{i \in [n]}[E(x)_i = E(y)_i] \leq (1-\delta)$, and we are choosing S by picking ℓ elements of $[n]$ independently and uniformly at random.

By the union bound, we have:

$$\Pr_S[\exists x \neq y \in T \text{ such that } h_S(x) = h_S(y)] \leq (1-\delta)^\ell \binom{|T|}{2}.$$

Picking $\ell \geq (1/\delta) 2 \ln |T|$ (so $\ell = O(t)$) ensures that this probability is at most $1/2$.

1.3 Application: hard-core predicates

This application is very important in cryptography. In order to present it we state a couple of theorems that we will prove later in the course.

Theorem 1.3 *Let $E : \{0,1\}^k \to \{0,1\}^n$ be the encoding function for an $[n, k, d = (1/2 - \delta)n]_2$ code. Then for all $y \in \{0,1\}^n$, at most $O(1/\delta)$ messages x satisfy $\Delta(E(x), y) < (1/2 - \sqrt{\delta})n$.*

Here, think of δ as being very small, e.g. $\delta = 1/n^{1/100}$. We already know that in the Hamming ball of radius $\lfloor d/2 \rfloor$ around any string $y \in \{0,1\}^n$, there is at most one codeword. This theorem states that in a Hamming ball of radius *close to* d around any string $y \in \{0,1\}^n$, there still aren't too many codewords. This is an example of one of the bounds we'll prove later in the course.

Perhaps more important is the algorithmic version, given in the following theorem:

Theorem 1.4 *There exist* $[n = poly(k, 1/\delta), k, d = (1/2 - \delta)n]_2$ *codes with efficient* list-decoding *algorithms that given* $y \in \{0,1\}^n$ *compute the list of* $O(1/\delta)$ *messages* x *for which* $\Delta(E(x), y) \leq (1/2 - \sqrt{\delta})n$. *Here efficient means that the algorithm runs in time* $poly(n, 1/\delta)$.

Now we turn to the application. Cryptography is based on hard functions, and there are many functions of the form

$$f : \{0,1\}^k \to \{0,1\}^k$$

that we believe are hard (e.g. factoring). However, many applications require a hard *predicate*:

$$f' : \{0,1\}^{k'} \to \{0,1\}.$$

We'd like a way to transform a generic hard f into a similarly hard predicate f'. If we are concerned with worst-case hardness, this is easy: given $f : \{0,1\}^k \to \{0,1\}^k$ we can define $f' : \{0,1\}^k \times [k] \to \{0,1\}$ by

$$f'(x, i) = f(x)_i. \tag{1.2}$$

Then, if f' can be computed by a circuit of size s, then f can be computed by a circuit of size $s' = sk$; this implies (in a precise way) that if f is "hard" then f' is also.

The problem is that for cryptographic applications, we need f' to be hard to compute for a randomly chosen input. The above transformation can fail significantly in this regard. For example, suppose we started with $f : \{0,1\}^k \to \{0,1\}^k$ defined as follows:

$$f(x) = \begin{cases} 000\ldots0 & \text{if } x \notin \text{SAT} \\ 100\ldots0 & \text{if } x \in \text{SAT} \end{cases}$$

Even if determining membership in SAT was hard on average, the function $f'(x, i) = f(x)_i$ would be extremely easy to compute correctly on a $1 - 1/k$ fraction of its inputs – just output 0 all the time.

To transform an average-case hard function into an average-case hard predicate, we'd like to solve the following problem:

Given: $f : \{0,1\}^k \to \{0,1\}^k$ for which no circuit of size s can compute f with probability $\geq \delta$

Produce: $f' : \{0,1\}^{k'} \to \{0,1\}$ for which no circuit of size s' can compute f' with probability $\geq 1/2 + \delta'$, for s' not much smaller than s and δ' not much larger than δ

We further assume that f is a *one-way permutation* – that is, f is a permutation, and there is a small circuit, of size poly(k) that computes f^{-1}. Notice that it is always easy to compute f' with probability $1/2$ by always outputting 0 or always outputting 1; this is why our goal is to produce a function that is hard to compute with probability *more* than $1/2$.

The original solution to the above problem is due to Goldreich and Levin [GL89], and the generic version using error-correcting codes is attributed to R. Impagliazzo.

Theorem 1.5 *Let* $E : \{0,1\}^k \to \{0,1\}^n$ *be the encoding function for an* $[n, k, (1/2 - \delta^2)n]_2$ *code. Given* $f : \{0,1\}^k \to \{0,1\}^k$ *as specified above, the function* $f' : \{0,1\} \times [n] \to \{0,1\}$ *defined by:*

$$f'(x, i) = E(f(x))_i \tag{1.3}$$

cannot be computed with probability $\geq 1/2 + 2\delta$ *by circuits of size* $s' = s/poly(n, 1/\delta)$.

Proof: Suppose for the purpose of contradiction there is a circuit C of size s' that computes f' with probability $\geq 1/2 + 2\delta$. We want to describe a small circuit C' that computes f with probability $\geq \delta$.

Given $x' \in \{0,1\}^k$, our circuit C' will use C to compute the n-bit string $y = (C(x,i))_{i \in [n]}$. Using Theorem 1.4, it will find $O(1/\delta^2)$ messages $m \in \{0,1\}^k$ for which $\Delta(E(m), y) \leq (1/2 - \delta)n$. Finally, it will compute $f^{-1}(m)$ for each such message, and output m if $f^{-1}(m) = x$. The size of circuit C' is most $s' \cdot \text{poly}(n, 1/\delta) < s$.

We now argue that $\Pr_x[C'(x) = f(x)] \geq \delta$. We know that $\Pr_{x,i}[C(x,i) = f'(x,i)] \geq 1/2 + 2\delta$. By an averaging argument,

$$\Pr_{x \in \{0,1\}^k}[\Pr_i[C(x,i) = f'(x,i)] \geq 1/2 + \delta] \geq \delta.$$

So, with probability $\geq \delta$, we are applying list-decoding to an n-bit string y for which $\Delta(y, E(f(x))) \leq (1/2 - \delta)n$. Therefore, $f(x)$ will appear in the returned list. In this case, when we compute $f^{-1}(f(x))$ we get x, so we output $f(x)$, the correct value.

We have described a circuit C' of size $< s$ which computes f with probability $\geq \delta$, a contradiction. ■

It is instructive to compare with (1.3) with (1.2). Intuitively, we've overcome the problem of having some bits in the output of f that are very easy to compute by "spreading" the k bits of information evenly onto n bits using an error-correcting code.

1.4 Entropy and the noiseless coding theorem

Entropy is a measure of the "information" in elements drawn from a given distribution.

Definition 1.6 *Given a distribution $D : U \to [0,1]$ over a finite set U, the entropy of D is*

$$H(D) = \sum_{x \in U} D(x) \log_2 \frac{1}{D(x)}.$$

The simplest non-trivial distribution is one in which $U = 0, 1$, $D(0) = p$ and $D(1) = 1 - p$. For this distribution, instead of writing $H(D)$, we often write $H(p)$ (or sometimes $H_2(p)$ – the "binary entropy" function) where $H(p) = p \log_2(1/p) + (1-p)\log_2(1/(1-p))$.

A distribution D with all elements equally likely achieves the maximum entropy of $\log_2 |U|$. This is particularly natural in computer science where if one is sending messages uniformly drawn from $U = \{0,1\}^n$, then each one conveys n bits of information.

The justification for calling H a measure of "information" comes from Shannon's "noisless coding theorem:"

Theorem 1.7 *Let $D : U \to [0,1]$ be a distribution over a finite set U. There exist function $Enc : U \to \{0,1\}^*$ and $Dec : \{0,1\}^* \to U$ for which $Dec(Enc(x)) = x$ for all $x \in U$, and*

$$H(D) \leq E[|Enc(x)|] \leq H(D) + 1.$$

Moreover, no prefix-free, error-free encoding can do better.

References

[GL89] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC 89)*. pages 25–32. 1989.

[H50] R. W. Hamming. Error detecting and error correcting codes. *Bell System Tech. J.* **29**, 147–160, 623–656. April 1950.

[S48] C. E. Shannon. A mathematical theory of communication. *Bell System Tech. J.* **27**, 379–423, 623–656. 1948.