Coding and Cryptography

May 14, 2008

We shall consider a sender who has a message m which they transform into a coded message $c^\star(m)$; it is then sent over some channel to a receiver who decodes $c^\star(m)$ to recover m; each of these steps is interesting and we shall study them in this course.

Our central idea, originally found by Shannon, is to consider the information content of the message; no compression method can reduce this, since if we did that we would then not be able to recover the original message.

0.1 Defns

An alphabet $\mathcal{A}$ is a finite set of symbols, called leters. A message or word is a finite sequence of symbols from the alphabet; we write these by simple concatenation $m = a_1 a_2 \dots a_n$. The set of all messages from the alphabet $\mathcal{A}$ is called $\mathcal{A}^\star$.

To convey m we will need to transform it into some new alphabet, the transmission alphabet [?] $\mathcal{B}$, so we need a coding function $c : \mathcal{A} \to \mathcal{B}^\star$ which encodes each element of A as a word from $\mathcal{B}$; then $c^\star(m) = c(a_1)c(a_2)\dots c(a_n)$.

1 Things we will consider

The requirement that we can decode a message places some restrictions on this coding function; what are they?

Most channels have some cost to sending long messages - how ca we encode our messages most efficiently? This relates to the notion of information content.

Can we encode in such a way that we can correct small errors in the transmitted message, for use in noisy channels?

Encryption

Examples are ASCII, morse code, reed-solomon error correction, "txtish", gzip, and the encryption applied to messages between ATMs and their bank; we won't consider "lossy compression" at all.

For background the reader should be familiar with the concepts of finite probability spaces, expectation and the WLLN from the Probability course, modulo arithmetic and Euclid's algorithm from Numbers and Sets, vector spaces from Linear Algebra and finite fields from GRM.

The recommended books for this course are Goldie and Pinch's Communication Theory (which follows this course most closely since it was written for an earlier version of this course), Welsh's Codes and Cryptography (a slightly

more understandable work written for an equivalent course in Oxford), Cover and Thomas' Elements of Information Theory (a very large, mathematically interesting tome, which is more focussed on ideas of information than the coding aspects), plus two other books mentioned in the schedules the notes about which I lacked the time to take down.

2 Entropy

Let $(\mathbb{P}, \Omega)$ be a discrete probability on a sample space; for each $A \subset \Omega$ the information of A is $I(A) = -\log_2 P(A)$ (In this course, logs are always base 2 unless otherwise stated, and I shall write P instead of $\mathbb{P}$ due to laziness); it is clear that this is ≥ 0 with equality if and only if $P(A) = 1$. Since e.g. the information of a sequence of results from N fair coin tosses is N, but we could also regard such a sequence of results as a string of N bits, we consider the unit of information to be bits. We shall see this definition is reasonable and gives reasonable results; a problem which is not an issue at this stage but will become one later is that this is not defined for A for which $P(A) = 0$.

Let X be a discrete random variable on Ω; for each x, $\{X = x\}$ has an information value as above; the expected value of this, $H(X) = -\sum_x P(X = x) \log_2 P(X = x)$ is called the entropy of X. Note that this does not depend on the particular values X takes, only their probabilities. This definition fails when one of the $\{X = x\}$ has probability 0, but if we define $-p \log_2 p = 0$ when $p = 0$ (which is the correct limit, from the graph of this as a function of p), then everything works. Entropy is also ≥ 0, with equality iff X is almost surely constant (i.e. takes a particular value with probability 1).

If $Y = f(X)$, intuitively Y can contain no more information than X, and indeed this is so, since $P(f(X) = y) = \sum_{x:f(x)=y} P(X - x)$ so $H(f(X)) = -\sum_y P(f(X) = y) \log P(f(X) = y) = -\sum_x P(X = x) \log P(f(X) = f(x)) \leq -\sum_x P(X = x) \log P(X = x) = H(X)$, since $P(X = x) \leq P(f(X) = f(x))$.

Example

Let $X : \Omega \to \{0, 1\}$ be a Bernoulli random variable taking the values 1 with probability p and 0 otherwise; then $H(X) = -p \log p - (1-p) \log(1-p)$, a concave function of p, with maximal value 1, obtained when $p = \frac{1}{2}$.

The entropy is the average amount of information that we gain by knowing the value of X; we will see later it is (to within 1) the expected number of yes/no questions we would need to ask to establish the value of X. We will use Entropy to measure the amount of information in a message, and thus determine how much it can be compressed.

2.1 Lemma: Gibbs' Inequality

Let X be a discrete random variable that takes different values with probabilities p_k ($\sum p_k = 1$). If q_k is another set of positive numbers summing to 1 then $-\sum p_k \log p_k \leq -\sum p_k \log q_k$ with equality iff $p_k = q_k \forall k$: the inequality is implied by $-\sum p_k \ln p_k \leq -\sum p_k \ln q_k \Leftarrow \sum p_k(\ln q_k - \ln p_k) \leq 0 \Leftarrow \sum p_k ln \frac{q_k}{p_k} \leq 0$. Now ln is strictly concave so its graph lies below the tangent to it at $(1, 0)$; that is, $\ln t \leq t - 1$

with equality only when $t = 1$, so the above is $\leq \sum p_k(\frac{q_k}{p_k} - 1) = \sum q_k - p_k = 0$ with equality iff $\frac{q_k}{p_k} = 1 \forall k$.

Example

A random variable that takes K different values with equal probability $\frac{1}{k}$ has entropy $\log K$; if X is any random variable that takes K different values then its entropy is at most $\log K$; the entropy is maximised when the values taken by X are all equally likely (compare this with the concept of etropy in thermodynamics)

Let X, Y be two random variables, then the entropy of the vector-valued random variable (X, Y) is called the joint entropy of X and Y; from 2.1 we have:

2.2 Corollary

$H(X, Y) \leq H(X) + H(y)$ with equality iff X, Y independent: $H(X, Y) = -\sum P(X = x, Y = y) \log P(X = x, Y = y)$; $\sum P(X = x)P(Y = y) = 1$ so by Gibbs this is $\leq -\sum P(X = x, Y = y) \log P(X = x)P(Y = y) = -\sum P(X = x, Y = y)(\log P(X = x) + \log P(Y = y) = -\sum P(X = x) \log P(X = x) - \sum P(Y = y) \log P(Y = y)$ with equality if and only if $P(X = x, Y = y) = P(X = x)P(Y = y) \forall x, y$ i.e. X, Y independent.

Example

If $X_1, \ldots, X_N$ are independent random variables with the distribution in the previous examplethen $(X_1, \ldots, X_N)$ has entropy $N \log K$.

We can refine corollary 2.2 by considering conditional distributions: if x is a value taken by X with nonzero probability then the conditional distribution of Y given $X = x$ is $P(Y = y|X = x) = \frac{P(X=x,Y=y)}{P(X=x)}$; we can see this as giving a new random variable $Y \mid X = x$. Now observe that $H(X, Y) - H(X) = \sum P(X = x, P = p) \log P(X = x, Y = y) - \sum P(X = x, Y = y) \log P(X = x) = -\sum P(X = x, Y = y) \log \frac{P(X=x,Y=y}{P(X=x)} = -\sum P(X = x)P(Y = y|X = x) \log P(Y = y|X = x) = \sum P(X = x)H(Y|X = x)$; this is called the conditional entropy $H(Y|X) = H(X, Y) - H(X)$; observe that $\forall X\ H(Y|X = x) \geq 0$ with equality if and only if $Y|X = x$ is almost surely constant; $H(Y|X) \geq 0$ with equality iff Y is almost surely a function of X. SO $H(X) + H(Y) \geq H(X, Y) \geq H(X)$.

3 Coding

3.1 Prefix-free Codes

We encode a message $m = a_1 \ldots a_n$ using the function $c : \mathcal{A} \to \mathcal{B}^\star$ by replacing each letter a with the code word $c(a)$; the entire message is then coded as $c^\star(m) = c(a_1) \ldots c(a_N)$; $c^\star : \mathcal{A}^\star \to \mathcal{B}^\star$ is the induced map. We want $c^\star$ to be injective, so we can recover m from $c^\star(m)$; in this case we say c is decodable or decipherable. It is clearly necessary that c is injective, but this is not sufficient, as e.g. c given by $a \mapsto 1, b \mapsto 11$ is injective. Some obvious ways to make a decipherable code are fixed-length codes like ASCII, or "comma codes" like

morse code where a special symbol is used to indicate the end of a codeword; however, we will consider the following more general class of codes:

Let $w \in \mathcal{B}^\star$; then $w' \in \mathcal{B}^\star$ is a prefix of w if $w = w'w''$ for some $w'' \in \mathcal{B}^\star$. A code $c : \mathcal{A} \to \mathcal{B}^\star$ is prefix-free if $c(a)$ is never a prefix of $c(a')\forall a, a' \in \mathcal{A}$. Prefix-free codes are decipherable by "greedy matching"; we look at the first n letters of the message, and if they form a codeword we remove it and continue, otherwise we look at the first $n+1$ letters and so on. However, ot all decipherable codes are prefix-free, e.g. $c : \{0, 1, 2, 3\} \to \{0, 1\}^\star$ given by $0 \mapsto 0, 1 \mapsto 10, 2 \mapsto 110, 3 \mapsto 111$ is prefix-free, but the same code with each codeword reversed is not prefix-free, but still decipherable as we could apply the same method to the message in reverse. But we will concentrate on prefix-free codes because they have many nice properties; for example, we can decode them "on the fly" as it is received rather than having to wait for the entire message before decoding; and we will see later they are in some sense "as good as" any other decipherable code. Prefix-free codes are also called instantaneous or self-punctuating.

We can consider prefix-free codes in terms of trees; we use the empty string ϕ as the root of the tree ("level 0"), then the verticies at level N are all the possible code words of length N, where each node's parent is the same word without its final letter. Then a code is prefix-free if the path from each codeword up to the root contains no other codeword.

3.2 Kraft's Inequality

Proposition (Kraft's Inequality I): Let $c : \mathcal{A} \to \mathcal{B}^\star$ be a prefix-free code, with the length of each codeword $c(a)$ being $l(a)$, then $\sum_{a\in\mathcal{A}} D^{-l(a)} \leq 1$ where $D = |\mathcal{B}|$. Let $L = \max_a l(a)$, then in the tree constructed as above there are clearly D^L verticies at level L; a codeword $c(a)$ is a prefix of $D^{L-l(a)}$ verticies at level L [if we consider a word is a prefix of itself, which it technically is], so since D is prefix-free the total number of level L verticies with code words as prefixes is $\sum_a D^{L-l(a)} \leq D^L$ the total number of verticies there, so $\sum_a D^{-l(a)} \leq 1$. Conversely:

Proposition (Kraft's Inequality II): Let $\mathcal{A}, B$ (finite) alphabets, $|\mathcal{B}| = D$, then if $l(a) > 0\forall a$ with $\sum_a D^{-l(a)} \leq 1$ then $\exists$ a prefix-free code $c : \mathcal{A} \to \mathcal{B}^\star$, where each $c(a)$ has length $l(a)$: we re-order the a_i so we ahve $l_1 \leq l_2 \leq \cdots \leq l_K$, then define c inductively: choose $c(a_1)$ of length l_1, then having chosen $c(a_j)\forall j < k$ with $c(a_i)$ never a prefix of $c(a_j)$ for $i < j < k$ (we do not have to check the $j < i$ case by our ordering), consider the words of length l_k; there are D^{l_k} of them; $D^{l_k-l_j}$ of them have $c(a_j)$ as a prefix for each j, but by the inequality $1 + \sum_{j=1}^{k-1} D^{l_k-l_j} = \sum_{j=1}^{k} D^{l_k-l_j} \leq D^{l_k}$, so we can choose a word of length l_k to be $c(a_k)$; note that this construction is not unique.

McMillan showed that Kraft's inequality holds for any decodable code; thus, if we only wish to minimise the lengths of codewords, we only need to consider prefix-free codes:

Proposition (McMillan): Let $c : \mathcal{A} \to \mathcal{B}^\star$ be a decodable code with $c(a)$ having length $l(a)$, then $\sum_a D^{-l(a)} \leq 1$ where $D = |\mathcal{B}|$: let $L = \max_a l(a)$, then consider $\left(\sum_a D^{l(a)}\right)^R$ for $R \in \mathbb{N}$; expanding this bracket we obtain a sum of the form $\sum_{a_1,\ldots,a_R\in\mathcal{A}} D^{-(l(a_1)+l(a_2)+\cdots+l(a_R))}$; we can also view the sum as running over words $a_1 \ldots a_R$ of length R. Now the word $w = c(a_1)c(a_2)\ldots c(a_R) \in \mathcal{B}^\star$ has length $|w| = l(a_1) + \cdots + l(a_R) \leq RL$; every such word can come from at most

one sequence $a_1 \dots a_R$ so $\left(\sum_a D^{-l(a)}\right)^R \leq \sum_{m=1}^{RL} n(m)D^{-1}$ where $n(m)$ is the number of words of length m of the form $c(a_1)\dots c(a_R)$; this is $\leq D^m$ so the above is $\leq \sum_{m=1}^{RL} D^m D^{-m} = RL$; taking the Rth root we have $\sum_a D^{-l(a)} \leq \sqrt[R]{RL}$ and taking the limit as $R \to \infty$ we have the result.

4 Efficient Codes

We want to minimise the expected length of codewords; a code where this length is as small as possible is called optimal.

4.1 Shannon-Fano Codes

Let $c : \mathcal{A} \to \mathcal{B}^\star$ be a code; assume decodable, more, prefix-free; we want to minimise $E(|c(a)|)$; to do this we need to fix a probability distribution on $\mathcal{A}$ so $p(a)$ is the probability of choosing the letter a. Then the expected length of a code word is $\sum_a p(a)|c(a)|$. If we let A be a random variable taking values in $\mathcal{A}$ with probability $p(a)$ then this is just $E|c(A)|$. By Kraft's inequality we have that $l(a) = |c(a)|$ satisfies $\sum D^{-l(a)} \leq 1$, and for any set of such $l(a)$ we know how to construct a prefix-free code with code word lengths $l(a)$, so we reduce to the optimization problem of minimizing $\sum p(a)l(a)$ over integers $l(a)$ subject to $\sum D^{-l(a)} \leq 1$.

Proposition: for any decodable $c : \mathcal{A} \to \mathcal{B}^\star$ we have $E|c(A)| \geq \frac{H(A)}{\log_2 D}$: by Kraft we have $\sum D^{-l(a)} \leq 1$; set $S = \sum D^{-l(a)}, q(a) = \frac{D^{-l(a)}}{S}$. Then $\sum q(a) = 1$ and by Gibbs, $-\sum p(a)\log p(a) \leq -\sum p(a)\log q(a) = \sum p(a)(l(a)\log D + \log S) = (\sum p(a)l(a))\log D + \log S \leq (\sum p(a)l(a))\log D$ as required.

We cannot always achieve equality in the above, but we can easily get close:

Proposition: Shannon-Fano encoding: for A as above there is a prefix-free code $c : \mathcal{A} \to \mathcal{B}^\star$ for which $E|c(A)| < 1 + \frac{H(A)}{\log D}$: by Kraft, sufficient to find $l(a)$ such that $\sum p(a)l(a) < 1 + \frac{H(A)}{\log D}, \sum D^{-l(a)} \leq 1$. We want to take $l(a) = -\log_D p(a)$ but the $l(a)$ must be integers so we instead take $\lceil -\log_D p(a) \rceil$. Then we have $-\log_D p(a) \leq l(a)$ so $p(a) \geq D^{-l(a)}$, so $\sum D^{-l(a)} \leq 1$ and we can find a prefix-free code with such $l(a)$, and we have $\sum p(a)l(a) < \sum p(a)(1 - \log_D p(a)) = 1 - \sum p(a)\log_D p(a) = 1 + \frac{H(A)}{\log D}$ as required. The code constructed by this is called a Shannon-Fano code; while it is not optimal it is easy to construct and its exected word lengths are close to their optimal value.

By the two above propositions we have:

Theorem: Shannon's noiseless coding theorem. For A as above an optimal code c satisfies $\frac{H(A)}{\log D} \leq E|c(A)| < \frac{H(A)}{\log_2 D} + 1$.

Example: if we ask yes/no questions how many do we need to ask to determine the value of A? Each question can be considered a map $\mathcal{A} \to \{0,1\}$; thus by the sequence of questions we have a code $c : \mathcal{A} \to \{0,1\}^\star$; the answers determine A iff this is decodable, so by this theorem we can choose the questions so $H(A) \leq E|c(A)| < H(A) + 1$; thus the average number of questions required is within 1 of $H(A)$, as asserted earlier.

4.2 Huffman Codes

This is an optimal code; for simplicity we shall only consider the case $\mathcal{B} = \{0, 1\}$; the codes are defined inductively and we shall proove they are optimal. Label the a_i so that the $p_k = p(a_k)$ satisfy $p_1 \geq p_2 \geq \cdots \geq p_k$; for $k = 2$ we have $h : \{a_1, a_2\} \to \{0, 1\}$ given by $h(a_1) = 0, h(a_2) = 1$, which is clearly optimal. Now supposing we have defined the Huffman code for alphabets of size $k - 1$; form the alphabet $\tilde{\mathcal{A}} = \{a_1, \ldots, a_{k-2}, a_{k-1} \cup a_k\}$ where the new letter $a_{k-1} \cup a_k$ has probability $p_{k-1} + p_k$. Then take $\tilde{h} : \tilde{A} \to \{0, 1\}$ to be a Huffman code for $\tilde{A}$, then the Huffman code for $\mathcal{A}$ is given by $h(a_j) = \tilde{h}(a_j)$ for $1 \leq a_j \leq k - 2$, $\tilde{h}(a_{k-1} \cup a_k)0$ for $j = k - 1$ and $\tilde{h}(a_{k-1} \cup a_k)1$ for $j = k$; this is clearly prefix-free.

Theorem: Huffman codes are optimal: we label the a_j, p_j as above; then the average code length is $E|c(A)|$.

Lemma: Optimal codes. There is an optimal code $c : \mathcal{A} \to \{0, 1\}^\star$ with lengths ordered inversely to probabilities so $l_1 \leq l_2 \cdots \leq l_k$, such that the code words $c(a_{k-1}), c(a_k)$ have the same length and differ only in the last bit: we know there is a prefix-free code for $\mathcal{A}$; let its expected code word length be C, then there are only a finite number of codes c with $\sum p_k l_k \leq C$, so there must be an optimal one. For the first property, if $p_i \geq p_j$ but $l_i > l_j$ then we could reduce $\sum p_k l_k$ by exchanging their codewords $c(a_i), c(a_j)$ so we have $l_1 \leq \cdots \leq l_k$; fir the second, let L be the maximum codeword length l_k. Then, deleting the final bit from $c(a_k)$ to get a new codeword w would reduce $\sum p_k l_k$, so there must be another codeword with w as a prefix, so we have some $c(a_j)$ of length L differing from $c(a_k)$ only in the last bit; by permuting the codewords of length L we can choose it to be $c(a_{k-1})$.

Now, to proove the above theorem: induct on the size k of $\mathcal{A}$ (base case 2); assume it is true for alphabets of size $k - 1$, then let $h : \mathcal{A} \to \{0, 1\}$ be a Huffman code and $c : \mathcal{A} \to \{0, 1\}$ a Huffman code. By construction of the Huffman code we have a Huffman code $\tilde{h}$ on $\tilde{\mathcal{A}} = \{a_1, \ldots, a_{k-2}, a_{k-1} \cup a_k\}$; then $E|h(A)| = E|\tilde{h}(\tilde{A})| + (p_{k-1} + p_k)$, where $\tilde{A}$ is a random variable takiing values in $\tilde{\mathcal{A}}$ with the obvious probabilities. By the lemma we can take $c(a_{k-1}), c(a_K)$ differing only in their last bit; wlog $c(a_{k-1}) = w0, c(a_k) = w1$ for some word w. Define $\tilde{c}(a_{k-1} \cup a_k) = w, \tilde{c}(a_k) = c(a_j) \forall j \leq k - 2$; this is a code on $\tilde{A}$, clearly prefix free, with $E|c(a)| = E|\tilde{c}(\tilde{A})| + (p_{k-1} + p_k)$. By inductive hipothesis $E|\tilde{h}(\tilde{A})| \leq E|tildec(\tilde{A})|$, so $E|h(a)| \leq E|c(A)|$ so by minimality of c we have equality and h is optimal.

5 Compression

We take $\mathcal{A}$ an alphabet of K letters; we want to send messages $m = a_1 a_2 \ldots$ from $\mathcal{A}^\star$. Let A_j be a random variable giving the jth letter, with each A_j identically distributed with $p(A_j = a) = p(a)$; we are encoding with strings from $\mathcal{B}$ with $D = |\mathcal{B}|$.

5.1 Block codes

For a code $c : \mathcal{A} \to \mathcal{B}^\star$, the expected code length is $E|c(A_1)| = \sum p(a)|c(a)|$; by the noiseless coding theorem above there is an optimal code c with $H(A_1) \leq E|c(A)| < 1 + H(A_1)$. However, rather than encoding each message, we can divide m into blocks of length r; consider each block as an element of a new alphabet

$\mathcal{A}^r$, then we can find an optimal code $c_r : \mathcal{A}^r \to \mathcal{B}^\star$ with $H(A_1, \dots, A_r) \le E|c_r(A_1, \dots, A_r)| < 1 + H(A_1, \dots, A_r)$; this is called a block code. Then the average code length per letter is $\frac{E|c_r(A_1,\dots,A_r)|}{r}$, with $\frac{H(A_1,\dots,A_r)}{r} \le \frac{E|c_r(A_1,\dots,A_r)|}{r} < \frac{1}{r} + \frac{H(A_1,\dots,A_r)}{r}$; thus [for large r] the expected code length per letter is approximately $\frac{H(A_1,\dots,A_r)}{r}$. Far above we have $H(A_1, \dots, A_r) \le H(A_1) + \cdots + H(A_r)$ with equality when the letters are independent; since all our A_j have the same distribution we have $H(A_1, \dots, A_r) \le rH(A_1)$.

If the A_j are independent this is an equality so we have $H(A_j) \le \frac{E|c_r(A_1,\dots,A_r)|}{r} < \frac{1}{r} + H(A_1)$, so the expected code length per letter $\to H(A_1)$ as $r \to \infty$.

Now we consider the case where $P(A_j = a_j)$ depends only on the previous letter A_{j-1}, and this is independent of j [i.e. a Markov chain model]; then $P(A_j = a_j \mid A_1 = a_1, \dots, A_{j-1} = a_{j-1}) = P(A_j = a_j \mid A_{j-1} = a_{j-1})$ so $H(A_j \mid A_1, \dots, A_{j-1}) = H(A_j \mid A_{j-1}) = H(A_2 \mid A_1)$. Now we have $H(A_1, \dots, A_r) = H(A_r \mid A_1, \dots, A_{r-1}) + H(A_1, \dots, A_{r-1}) = H(A_r \mid A_{r-1}) + H(A_{r-1}|A_{r-2}) + \cdots + H(A_2 \mid A_1) + H(A_1) = (r-1)H(A_2 \mid A_1) + H(A_1)$ so the expected code length per letter $\to H(A_2 \mid A_1)$ as $r \to \infty$.

5.2 Compression

The maximum value for $H(A_1)$ occurs when all the K letters are equally likely; in this case it is $\log K$ and the optimal code satisfies $\log_D K = \frac{H(A_1)}{\log D} \le E|c(A_1)| < 1 + \frac{H(A_1)}{\log D} = \log_D(DK)$; there are K possible letters in A and D possibilities for each letter of $c(a)$, so the number of code words of length L is D^L; this $= K$ when $L = \log_D K$ and this is approximately the expected code length, so the optimial code does not gain us any compression. However, for a non-uniform distribution of the letters, the entropy $H(A_1) < \log K$ so the optimal code compresses the message; if we use block codes and the letters are not independent then there will be much better compression, since we have $\frac{H(A_1,\dots,A_r)}{r} < H(A_1)$, so the expected code length per letter is $< \frac{1}{r} + H(A_1)$; the entropy $\frac{H(A_1,\dots,A_r)}{r}$ is the amount of information per letter in a block of length r, and gives a limit on the best we can compress the message (on average).

6 Noisy Channels

Here we consider the case where there is a small but nonzero probability that a letter will be altered in the channel; we want to devise codes which detect or correct these errors.

6.1 Memoryless Channels

If the probability of any letter b_n of a coded message $b_1 \dots b_N \in \mathcal{B}^\star$ being altered in the channel is independent of the other letters then we say the channel is memoryless; write B_n for the random variable giving the nth letter of the transmitted message and B'_n for the same thing for the received message, then $P(b'_1 \dots b'_N \mid b_1 \dots b_N = \prod_{n=1}^N P(B'_n = b'_n \mid B_n = b_n)$; we have a transition matrix $P_{ij} = P(B'_n = i \mid B_n = j)$; we are assuming the probability of an error is small, i.e.

P_{ii} is close to 1. We will assume that the channel is time independent; that is, the transition matrix is the same $\forall n$.

For example, suppose $\mathcal{B} = \{0, 1, \ldots, 9\}$ and $P_{ji} = \frac{3}{4}$ when $j = i$, $\frac{1}{4}$ when $j = i + 1$, 0 otherwise. Then if we naively encode an N letter message the probability it is correctly received is only $(\frac{3}{4})^n$; if we send each letter 3 times and assume that if two out of the three received letters are the same this is the correct letter, the probability a letter is received correctly is $\frac{27}{32} > \frac{3}{4}$ so we increase the probability of decoding the message without error to $(\frac{27}{32})^N$, though at the cost of sending a message three times as long. However, we can do much better by only using the letters $\{0, 2, 4, 6, 8\}$ in our encoding; then we can decode the message perfectly.

6.2 The Binary Symmetric Channel

We take $\mathcal{B} = \{0, 1\}$ for simplicity. Then the binary symmetric channel is a time independent, memoryless channel with transition matrix $\begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}$; p is the probability of an error in a single bit, usually small. If $p = 0$ there are no errors, if $p = \frac{1}{2}$ then no information is transmitted, only noise.

As an example, consider computer punch cards; we can use a check digit every 8 bits; for $x_1, \ldots, x_7$, x_8 is chosen so that $x_1 + \cdots + x_8 \equiv 0 \mathrm{mod} 2 (\star)$. If $p = 0.1$ the probability of a correct transmission of all 8 bits is 0.48, and the probability that 1 error occurs in the 8 bits and is therefore detected is 0.33. Note that if 2 errors occur, $(\star)$ holds and the errors are not detected.

6.3 Check digits

Example: ISBN-10s are codes for identifying books; we have $x_1 \ldots x_9$ identifying the books, and $x_{10} \in \{0, \ldots, 9, X\}$ is a check digit such that $10x_1 + 9x_2 + \cdots + x_{10} \equiv 0 \mathrm{mod} 11$. This means that if we alter any single digit or transpose two adjacent digits we will obtain an invalid code, so we can detect such errors.

6.4 The Hamming Code

This is in a way a converse to the fact that we can compress a message that contains redundant information; here we expand a code to add redundant information, allowing us to detect or even correct errors. Hamming1s code is a binary code that encodes binary words of length 4 as words of length 7; it is capable of correcting any single bit error in the 7 bits; when we receive a word we look for the closest possible code word, and this works.

Consider a binary string of length N as a vector in the N-dimensional vector space F_2^N; then Hamming's code is given by the linear map $C : F_2^4 \to F_2^7$ given by the matrix $C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}$. Thus $C\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_2 + x_3 + x_4 \\ x_1 + x_3 + x_4 \\ x_1 + x_2 + x_4 \end{pmatrix}$; the first 4 bits

carry the data, the remaining 3 are check digits. For $\vec{y} = C\vec{x}$ we can verify we will always have $y_1 + y_3 + y_5 + y_7 = 0, y_2 + y_3 + y_6 + y_7 = 0, y_4 + y_5 + y_6 + y_7 = 0$(†) since this is true for each column of C. The converse is also true; if $\vec{y}$ satisfies these equations then they determine y_5, y_6, y_7 in terms of y_1, y_2, y_3, y_4 (and any set of values for y_1, y_2, y_3, y_4 gives a codeword), so $\vec{y}$ is a codeword precisely when (†) are satisfied. We can rewrite these equations as $S\vec{y} = \vec{0}$ where the syndrome matrix $S = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$; we have seen $SC\vec{x} = 0 \forall \vec{x} \in F_2^4$, i.e. $S \circ C = 0$ and the image $C(F_2^4) = \ker S$.

If one error occurs in transmission, say in the jth bit, then we receive $\vec{y} = C\vec{x} + \vec{e_j}$; we then have the syndrome $S\vec{y} = SC\vec{x} + S\vec{e_j} = S\vec{e_j}$, the jth column of S; notice this is the binary representation of j in reverse, e.g. 6 = 110 in binary, and the sixth column of S is 011. So we can tell in which bit the error occurred, and therefore correct it. If there are two errors, e.g. $\vec{y} = C\vec{x} + \vec{e_i} + \vec{e_j}$, then $S\vec{y} = S\vec{e_i} + S\vec{e_j}$; this is the sum of two different syndromes so nonzero, which tells us that errors have occured; however, this is insufficient to correct the errors, because we cannot tell whether a given syndrome e.g. 100 is the result of one error or two errors. So the Hamming code detects 2 errors and can correct 1 error.

If we measure the distance between $\vec{y}, \vec{z} \in F_2^N$ by the Hamming distance $d(\vec{x}, \vec{y}) = |\{j : y_j \neq z_j\}|$, this is a metric; it is the number of bit errors required to change $\vec{y}$ to $\vec{z}$. If x, x' are two different strings in F_2^4 and $C\vec{x}, c\vec{x'}$ their corresponding codewords, then their difference $C\vec{x'} - C\vec{x} = C(\vec{x'} - \vec{x})$ is [lecturer says a sum of, but I think lol] a column of the matrix C, so has at least 3 nonzero bits; $d(C\vec{x'}, C\vec{x}) \geq 3$ (the distance can be exactly 3). So if there is one error in $\vec{y}$ we use the syndrome to find the unique $\vec{x} \in F_2^4$ with $d(\vec{y}, C\vec{x}) = 1$; if our received y has had two errors, it may not even have a unique closest $C\vec{x}$; it could be distance 2 from each of two possible $C\vec{x}$.

7 Error correcting codes

In this section we consider codes $c : \mathcal{A} \to \mathcal{B}^N$ of constant length N; we use the discrete metric on $\mathcal{B}$ and the Hamming distance on $\mathcal{B}^N$ as above.

Say a codeword $\vec{b}$ is sent through a noisy channel and received as $\vec{v}$. We will consider three possible decoding rules:

Ideal observer: decode $\vec{v}$ by the codeword $\vec{b}$ such that $P(\vec{b}|\vec{v})$ maximal; this is sensible if we can calculate it, but usually we cannot.

Maximum likelihood: decode $\vec{v}$ by the codeword $\vec{b}$ with $P(\vec{v} \mid \vec{b})$ maximal; this is easier to calculate in general.

Minimum distance: decode $\vec{v}$ by the codeword $\vec{b}$ with $d(\vec{b}, \vec{v})$ minimal.

Proposition: if all the code words are equally likely then the first two rules give the same result, since $P(\vec{v} \mid \vec{b}) = \frac{P(\vec{v} \cap \vec{b})}{P(\vec{b})} = P(\vec{b} \mid \vec{v}) \frac{P(\vec{v})}{P(\vec{b})}$.

Proposition: if letters are transmitted through a binary symmetric channel with error probability $p < \frac{1}{2}$ then the second and third rules give the same result, since $P(\vec{v} \mid \vec{b}) = \prod_{j=1}^{N} P(v_j \mid b_j) = p^d (1-p)^{N-d} = (1-p)^N (\frac{p}{1-p})^d$ where d is the Hamming distance; this is clearly maximal when d is minimal.

We will generally use the minimum distance rule.

In this section we will use binary codes for simplicity; write K for the number of code words. The volume of $B(\vec{b}, r)$ is the number of points it contains, $|B(\vec{b}, r)| = \sum_{0 \le k < r} \binom{N}{k}$. This is independent of $\vec{b}$; denote it by $V(N, r)$.

A code $c : \mathcal{A} \to \{0,1\}^N$ is e-error-detecting if when at most e letters of a codeword are altered the result is never a different codeword, i.e. we can tell there have been errors if there have been at most e of them. The code is e-error correcting if when at most e letters of a codeword are altered, the codeword is still decoded correctly using the minimum distance rule.

The minimum distance for a code δ is the minimum distance between two different codewords. We immediately have:

Proposition: For c with minimum distance $\delta > 0$, c is $(\delta - 1)$-error detecting and not δ-error detecting, and c is $\lfloor \frac{1}{2}(\delta - 1) \rfloor$-error correcting but not $\lfloor \frac{1}{2}(\delta + 1) \rfloor$-error correcting (by the triangle inequality).

Observe that c is e-error correcting precisely when the $B(c(a), e + 1) \forall a$ are disjoint.

Proposition (Hamming's bound): If $c : \mathcal{A} \to \{0,1\}^N$ is an e-error correcting code with K codewords, then $K \le \frac{2^N}{V(N,e+1)}$. We say a code is perfect when we have equality in this bound; the balls cover all of $\{0,1\}^N$, e.g. Hamming's code is perfect.

Proposition (Gilbert-Shannon-Varshamov bound): There is an e-error detecting code with $K \ge \frac{2^N}{V(N,e+1)}$: choose a maximal set C of codewords such that no $c(a')$ lies within $B(c(a), e + 1)$ (which is clearly a necessary condition). Now if there were any $\vec{v} \notin \bigcup_{\vec{c} \in C} B(\vec{c}, e + 1)$ then we could add it to C, contradicting maximality, so this union is precisely $\{0,1\}^N$. Since the volume of each ball is $V(N, e + 1)$ the volume of this union is at most $KV(N, e + 1)$ and we have the result.

Corollary: for c a code with minimum distance δ, $K \le \frac{2^N}{V(N, \lfloor \frac{1}{2}(\delta+1) \rfloor)}$; there is a code with minimum distance δ and $K \ge \frac{2^N}{V(N,\delta)}$.

We want asymptotic estimates on the size of error correcting codes using the previous bounds; for this we need bounds on the volumes $V(N, r)$. It will be useful to talk obut the entropy of a Bernoulli random variable taking two values with probability q and $1 - q$, $h(q) = -q \log q - (1 - q) \log(1 - q)$.

Lemma: for $0 \le r \le N$, $\frac{1}{N+1} 2^{Nh(q)} \le \binom{N}{r} \le 2^{Nh(q)}$, where $q = \frac{r}{N}$: let X be a binomial $B(N, q)$ random variable, $P(X = k) = \binom{N}{k} q^k (1 - q)^{N-k}$; the entropy $H(X) = Nh(q)$. We can find that this probability is maximal when $k = r = qN$, so $P(X = r) \le \sum_{k=0}^{N} P(X = k) = 1 \le (N + 1)P(X = r)$; thus $\frac{1}{N+1} \le P(X = r) = \binom{N}{r} q^r (1 - q)^{N-r} = \binom{N}{r} 2^{-Nh(q)} \le 1$. Since $2^{-Nh(q)} = q^{Nq}(1 - q)^{N(1-q)} = q^r (1 - q)^{N-r}$ we have the result.

Proposition: For N and $0 < r < \frac{1}{2}N$, $V(N, r) \le 2^{Nh(q)}$ for $q = \frac{r}{N}$, and $\frac{1}{N+1} 2^{Nh(q')} \le V(N, r)$ where $q' = (\lceil r \rceil - 1)/N$: $V(N, r) = \sum_{0 \le k < r} \binom{N}{k}$; since $q = \frac{r}{N} \le \frac{1}{2}$, $q^k (1 - q)^{N-k} \le q^r (1 - q)^{N-r} \forall 0 \le k < r$, so $1 = \sum_{0 \le k \le N} \binom{N}{k} q^k (1 - q)^{N-k} \ge \sum_{0 \le k < r} \binom{N}{k} q^k (1 - q)^{N-k} \ge \left(\sum_{0 \le k \le r} \binom{N}{k} \right) q^r (1 - q)^{N-r} = V(N, r) q^r (1 - q)^{N-r}$ so $V(N, r) \le 2^{Nh(q)}$. For the second part let $k = \lceil r \rceil - 1$, then $V(N, r) \ge \binom{N}{k} \ge \frac{1}{N+1} 2^{Nh(q')}$ by the above lemma.

For $c : \mathcal{A} \to \{0,1\}^N$, if A is a random variable taking values in $\mathcal{A}$ then

information is transmitted through the channel at the rate $\frac{H(A)}{N}$; this is lagest when A is equally distributed over its K possible values, then it is $\frac{\log_2 K}{N}$. If we take q with $0 < q < \frac{1}{2}$, then by Hamming's bound $\frac{\log_2 K}{N} \leq 1 - \frac{\log_2 V(N,qN)}{N}$ for $(qN-1)$-error correcting codes; by the previous proposition the right hand side $\to 1-h(q)$ as $N \to \infty$; similarly by the Gilbert-Shannon-Varshamov bound there are $(qN-1)$-error detecting codes with information rate $\frac{\log_2 K}{N} \geq 1 - \frac{\log V(N,qN)}{N}$, and the right hand side $\to 1 - h(q)$ as $N \to \infty$.

8 Information Capacity

We want to determine how much information can be passed through a noisy channel per bit transmitted; consider a time-independent, memoryless channel. Say $B \in \mathcal{B}$ transmitted and $B' \in \mathcal{B}$, possibly different, received. The average information given by B is $H(B)$; the average information received is $H(B')$ but this is partly due to noise. The conditional entropy $H(B'|B)$ is the information due to the noise in the channel, so the remaining informationhe mutual information of B and B', $I(B', B) := H(B') - H(B'|B)$ is the information due to the B that was sent.

Proposition: this satisfies $I(B', B) = H(B') + H(B) - H(B', B) = I(B, B')$ since from section 2, $H(B'|B) = H(B', B) - H(B)$ so $I(B', B) = H(B') - H(B' \mid B) = H(B') - H(B', B) + H(B)$, a) $I(B', B) \geq 0$ with equality iff B, B' independent b) $I(B', B) \leq H(B')$ with equality iff B' is a function of B c)$I(B', B) \leq H(B)$ with equality iff B is a function of B' by the symmetry above.

From the distribution $P(B = b)$ and transition probabilities $P(B' = b' \mid B = b)$, which we are taking to both be known, we can calculate $P(B' = b', B = b) = P(B' = b'|B = b)P(B = b)$ and $P(B' = b' = \sum_b P(B' = b'|B = b)P(B = b)$; therefore we can find the entropies $H(B), H(B')H(B', B)$ and the mutual information.

The information capacity of the channel is the supremum of $I(B', B)$ over all possible distributions of B. A probability distribution of B is given by a point $\vec{p} = (p(b_1), \ldots, p(b_k))$ in the compact set $[0, 1]^k$; the mutual information is a continuous function of $\vec{p}$ so the supremum is attained by some distribution. Note that the information capacity depends only on the transition probabilities $P(B'|B)$.

Proposition (Information capacity of a BSC): a binary symmetric channel with probability p of errors has information capacity $1 - h(p)$: let $t = P(B = 1)$, then $H(B) = H(1 - t, t) = h(t)$; $H(B'|B) = P(B = 1)H(B'|B = 1) + P(B = 1)H(B'|B = 0) = tH(1-p, p) + (1-t)H(p, 1-p) = h(p)$; $P(B' = 1) = t(1-p) + (1-t)p = t + p - 2tp$ so $H(B') = h(t + p - 2tp)$, so $I(B', B) = H(B') - H(B'|B) = h(t + p - 2tp) - h(p)$; we have free choice of $0 \leq t \leq 1$, and the maximum value for $h(t + p - 2tp)$ is 1, attained by $t = \frac{1}{2}$ so $t + p - 2tp = \frac{1}{2}$, so the information capacity is $1 - h(p)$.

We often want to consider repeated use of the same channel, e.g. sending a word from $\{0, 1\}^N$ through a BSC; we can consider this as having N copies of the channel in parallel, and sending one bit through each. It is then simple to compute the capacity:

Proposition (capacity of parallel channels): for Q a channel transmitting letters of $\mathcal{B}$ with capacity $\mathrm{Cap}(Q)$, the capacity of a new channel Q^N that transmits an N-tuple $(b_1, \ldots, b_N) \in \mathcal{B}^N$ one letter at a time through Q with each use independent of the others is $N\mathrm{Cap}(Q)$: let $\vec{B} = (B_1, \ldots, B_N)$ be a random variable

taking values in $\mathcal{B}^N$, $\vec{B}'$ the random variable we receive; for each $\vec{b}' = (b'_1, \dots, b'_N)$ we have $H(\vec{B}|\vec{B}' = \vec{b}') = \sum H(B_j|B'_j = b_j)$ since each B_j is independent of all the others, so $H(\vec{B}|\vec{B}') = \sum H(B_j|B'_j)$; also $H(\vec{B}) \leq \sum H(B_j)$ with equality iff the B_j are independent, so $I(\vec{B}', \vec{B}) = H(\vec{B}) - H(\vec{B}|\vec{B}') \leq \sum_{j=1}^N H(B_j) - H(B_j|B'_j) = \sum_{j=1}^N I(B'_j, B_j)$ with equality iff the B_j are independent; thus $\text{Cap}(Q^N) \leq N\text{Cap}(Q)$. Now if we choose the B_j to be independently distributed with $I(B'_j, B_j)$ =Cap(Q) then we have $I(\vec{B}', \vec{B}) = N\text{Cap}(Q)$ so we have the result. We can similarly show that the capacity of independent channels Q_j in parallel is $\sum \text{Cap}(Q_j)$.

We will later proove Shannon's Coding Theorem, which shows the existence of very good codes for noisy channels; however Shannon's argument is nonconstructive and it is difficult to construct such codes in practise.

Suppose we want to transmit a message of size K from $\mathcal{A}$; let A be a random variable taking values in $\mathcal{A}$. Then $H(A) \leq \log K$ with equality when all letters in $\mathcal{A}$ are equally likely; we shall assume this is so.

We take a constant length code $c : \mathcal{A} \to \mathcal{B}^N$ to encode the message one letter at a time. This gives a random variable $\vec{B} = c(A)$ taking values in $\vec{B}^N$; since c is decodable A determines $\vec{B}$ and v.v., so $H(A) = H(\vec{B})$. Each bit of $c(a)$ is passed through a channel, giving a new string $c(a)' \in \mathcal{B}^N$; let $\vec{B}' = c(\vec{A})'$. Let $c(a')$ be the closest codeword to $c(a)'$, then we decode this as $a' \in \mathcal{A}$; A' is the obvious thing, and $H(A') \leq H(\vec{B}')$. The probability of an error is $\sum AP(\text{error}|a)P(A = a)$; we want this to be small; however, what we really want is to make the probability of an error small $\forall a \in \mathcal{A}$, so we want the <u>maximum error</u> $\hat{e}(c) = \max\{P(\text{error}|a) : a \in \mathcal{A}\}$ to be small.

The <u>rate of transmission for c</u> is $\rho(c) = \frac{H(A)}{N} = \frac{\log K}{N}$; we want this to be as large as possible. We aim to relate this to the information capacity of the channel.

Theorem (Fano's inequality): Let X, Y random variables taking values in $\mathcal{A}, |\mathcal{A}| = K$. Then $H(X|Y) \leq p \log(K-1) + h(p)$ where $p = P(X \neq Y)$; we will apply this where $X = A, Y = A'$; then p is the probability of error. For the proof of the theorem, recall $H(X, I) = H(X|I) + H(I)$ for a random variable I; conditioning on $Y = y$ we have $H(X, I|Y = y) = H(X|I, Y = y) + (I|Y = y)$ and so $H(X, I|Y) = H(X|I, Y) + H(I|Y)$. Let I be the indicator random variable that $X \neq Y$. then $H(X, I|Y) = H(X|Y)$ as I is determined by X, Y. $H(X|I, Y) = P(I = 0)H(X|I = 0, Y) + P(I = 1)H(X|I = 1, Y) = (1-p)0 + pH(X|I = 1, Y) \leq p \log(K-1)$ as if $I = 1$ there are $K-1$ possible values for $X|Y$. $H(I|Y) = H(I, Y) - H(Y) \leq H(I)$ so $H(X|Y) \leq p \log(K-1) + H(I)$; since I takes only the two values 0,1 we have $H(I) = h(p)$ and the result.

We can think of the two terms of Fano's inequality as $h(p) = H(I)$ giving the information given by knowing whether there is an error, and $p \log(K-1)$ measuring the information on what the error is, when it occurs. The statement and proof of Fano's theorem do not depend on Y.

Using Fano's inequality we can compare the rate of transmission of information to the capacity of the channel; we have $H(A|A') \leq h(p) + p \log(K-1) < h(p) + P \log K$ so we have $I(A', A) = H(A) - H(A|A') \geq \log K - (h(p) + p \log K)$, so $N\text{Cap}(Q) \geq (1-p) \log K - h(p)$; dividing by N we have $\text{Cap}(Q) + \frac{h(p)}{N} \leq (1-p)\rho(c)$.

Theorem (Shannon's Noisy Coding Theorem I): Let $c : \mathcal{A} \to \mathcal{B}^N$ where each letter of a codeword is transmitted through a time-independent memoryless

channel with capacity Cap. Let p be the probability of decoding incorrectly. Then the rate of transmission satisfies $\rho(c) \leq \frac{\log K}{N} \leq \frac{\text{Cap}}{1-p} + \frac{h(p)}{N(1-p)}$; as we let $p \to 0$ downwards, $1-p \to 1$ from below and $h(p) \to 0$ from above so $\frac{\text{Cap}}{1-p} + \frac{h(p)}{N(1-p)} \to$ Cap from above; hense if we can find codes c_j with probabilities of error p_j that degrease to 0 and transmission rates that $\to$ a limit ρ, then $\rho \leq$ the information capacity of the channel.

9 Shannon's Noisy Coding Theorem

Theorem (Shannon's Noisy Coding Theorem: For any $\epsilon > 0$, for a binary symmetric channel of capacity Cap, for sufficiently large N there are codes $c_N : \mathcal{A}_N \to \{0,1\}^N$ with rate of transmission $\rho(c_N) = \frac{\log_2 |\mathcal{A}_N|}{N} \geq \text{Cap} - \epsilon$ and maximum error $\hat{e}(c_N) < \epsilon$. Unfortunately, constructing these codes is difficult in practice.

Theorem (Chebyshev's Inequality): For X a real-valued discrete random variable with mean EX, variance $\text{var}(X)$, $P(|X - EX| \geq t) \leq \frac{\text{var}(X)}{t^2}$ (the inequality also holds for X non-discrete provided $\text{var}(X)$ exists): $\text{var}(X) = E|X - EX|^2 = \sum P(X = x)|x - EX|^2 \geq \sum\{P(X = x)t^2 | x - EX| \geq t\} = t^2 P(|X - EX| \geq t)$.

Proof of Shannon's Noisy Coding Theorem: choose codewords in $\{0,1\}^N$ at random, independently of each other and the channel; we shall proove that on average this gives the inequalities we want, therefore there must be at least one choice of codewords for which the inequalities hold. Let the error probability of the BSC be $0 \leq p < \frac{1}{2}$. Then $\text{Cap} = 1 - h(p)$; set $K_N = \lfloor 2^{N(\text{Cap}-\epsilon)} \rfloor$ (note this is much less than 2^N), and let $\mathcal{A}_N$ be such that $|\mathcal{A}_N| = K_N$. For a letter $a_0 \in \mathcal{A}_N$ choose a random code word $\vec{c}_0 = c_N(a_0) \in \{0,1\}^N$, uniformly and independently of anything else. Say it is sent through the channel and $\vec{c}_0'$ is received; we want this to be close to $\vec{c}_0$. The Hamming distance $d(\vec{c}, \vec{c}')$ is a $B(N,p)$ random variable, so has mean Np, variance $Np(1-p)$; by Chebyshev we have $P(d(\vec{c}_0, \vec{c}_0') \geq r) \leq \frac{Np(1-p)}{(r-Np)^2}$ for $r > Np$; take $q > p$ and set $r = Nq$, then $P(d(\vec{c}_0, \vec{c}_0') \geq r) \leq \frac{p(1-p)}{N(q-p)^2}$ (1). Now for anothe rcodeword $\vec{c} = c_N(a)$ for some $a \neq 0$, we have $P(d(\vec{c}, \vec{c}_0') < r) = P(\vec{c} \in B(\vec{c}_0', r)) = \frac{V(N,r)}{2^N} \leq \frac{2^{Nh(q)}}{2^N} = 2^{-N(1-h(q))}$. So the probability that $d(\vec{c}, \vec{c}_0') < r$ for at least one of the $K_N - 1$ codewords $\vec{c} \neq \vec{c}_0$ is $\leq K_N 2^{-N(1-h(q))} = 2^{N(\text{Cap}-\epsilon-(1-h(q)))}$; since $\text{Cap} = 1 - h(p)$ this is $\leq 2^{N(h(q)-h(p)-\epsilon)}$ (2).

We shall use the minimal distance decoding rule, so we make an error and decode a_0 as $a \neq a_0$ only when $d(\vec{c}_0, \vec{c}_0') \geq r$ or $d(\vec{c}_N(a), \vec{c}_0') = 0$. So the probability of an error is $\leq \frac{p(1-p)}{N(q-p)^2} + 2^{N(h(q)-h(p)-\epsilon)}$ by (1) and (2). Take q with $p < q < \frac{1}{2}, h(q) < h(p) + \epsilon$ which we can do as $0 \leq p < \frac{1}{2}$; then $\frac{p(1-p)}{N(q-p)^2} \leq \frac{\epsilon}{2}, 2^{N(h(q)-h(p)-\epsilon)} \leq \frac{\epsilon}{2}$ for sufficiently large N; for such N the probability of an error is $< \epsilon$. This was the average for a random choice of codewords, so we must be able to choose the codewords for c_N such that the probability of error is $< \epsilon$. So we now have that the mean probability of an error is $< \epsilon$; we need that the maximum error $\hat{e}(c_N) < \epsilon$.

We have that the average error probability $\frac{1}{|\mathcal{A}_n|} \sum_a P(\text{error}|a \text{ sent}) < \epsilon$, so for at least half of the letters in $\mathcal{A}_N$ we have $P(\text{error}|a \text{ sent}) < 2\epsilon$; take these $\frac{1}{2}K_n$ letters as our new alphabet $\mathcal{A}_N'$, then the maximum error for this alphabet is

$< 2\epsilon$, and we have $|\mathcal{A}'_N| = \frac{1}{2}|\mathcal{A}_N| = 2^{N(\text{Cap}-\epsilon)-1}$ so the rate of transmission is $\frac{\log|\mathcal{A}'_N|}{N} = \text{Cap} - \epsilon - \frac{1}{N}$.

9.1 Typical Sequences

We are going to reinterpret the entropy $H(A)$ in terms of a "typical sequence" given by a sequence (A_n) of random variables IID with the same distribution as A.

Recall the WLLN from IA probability: we say a sequence of random variables S_n converge to a random variable L in probability if $P(|S_n - L| \geq \epsilon) \to 0$ as $n \to \infty$, $\forall \epsilon > 0$. Then the WLLN is that for X a discrete real-valued random variable, (X_N) IID with the same distribution as X, the averages $S_n = \frac{X_1+\cdots+X_n}{n}$ converge in probability to EX as $n \to \infty$: we have $ES_n = EX, \text{var}(S_n) = \frac{\text{Var}(X)}{n}$, so by Chebyshev we have $P(|S_n - ES_n| \geq t) \leq \frac{\text{var}(S_n)}{t^2}$ i.e. $P(|S_n - EX| \geq t) \leq \frac{\text{var}X}{nt^2}$ and the RHS $\to 0$ as $n \to \infty$.

Now, the reinterpretation: recall that the entropy of A is $H(A) = -\sum_a P(A = a) \log P(A = a)$; this is the expectation of the real-valued random variable $X(\omega) = -\log P(A = A(\omega)$ (i.e. X takes the values $-\log p_k$ with probability p_k, where A takes values a_k with probabilities p_k). Similarly we define X_n from A_n, so the X_n are IID with the same distribution as X. By the WLLN, $\frac{X_1+\cdots+X_n}{n} \to EX$ in probability as $n \to 0$, so $\forall \epsilon > 0 \exists N(\epsilon) : P(|\frac{X_1+\cdots+X_n}{n} - EX| \geq \epsilon) \leq \epsilon \forall n \geq N(\epsilon)$.

For a particular sequence of values (a_j) from $\mathcal{A}$, the probability of obtaining this as (A_j) is $P(A_j = a_j \forall j) = \prod_{j=1}^n P(A = a_j)$; if we do have $A_j = a_j$ then $\frac{X_1+\cdots+X_n}{n} = -\sum \log P(A = A_j) = -\log \prod P(A = a_j)$, so $|\frac{X_1+\cdots+X_n}{n} - EX| < \epsilon$ is equivalent to $2^{-n(H(A)+\epsilon)} < P(A_j = a_j \forall j) < 2^{-n(H(A)-\epsilon)} (\star)$, so the probability of obtaining (a_j) for which $(\star)$ holds is at least $1 - \epsilon$ for $n \geq N(\epsilon)$; such sequences are "typical sequences" and we have proven the following:

Theorem (Asymptotic equipartition property): Let A be a random variable taking values in $\mathcal{A}$ with entropy $H(A)$; let (A_n) as before; then $\forall \epsilon > 0 \exists N(\epsilon) \in \mathbb{N}$ such that $\forall n \geq N(\epsilon) \exists T \subset \mathcal{A}^n : P((A_j) \in T) > 1 - \epsilon$ and $2^{-n(H(A)+\epsilon)} < P(A_j = a_j \forall j) < 2^{-n(H(A)-\epsilon)} \forall (a_j) \in T$; we call the sequences in T "typical" and those not in T are "atypical". The sequences of T have approximately the same probability $2^{-nH(A)}$ of arising, so we say the sequence of random variables (A_n) has the asymptotic equipartition property.

10 Linear Codes

There is a finite field F_q, unique up to isomorphism, of q elements iff $q = p^n$ for some prime p, e.g. $F_p = \frac{\mathbb{Z}}{p\mathbb{Z}}$; its multiplicative group is denoted $F_q^\times$. This is a cyclic group of order $q - 1$; any generator therof is called a primitive element for F_q: for such an α the elements of F_q are $\alpha, \alpha^2, \ldots, \alpha^{q-1}$, all distinct. The order of α^k is $\frac{q-1}{(q-1,k)}$, so the number of primitive elements is given by Euler's totient function as $\phi(q-1) := |\{1 \leq k \leq q-1 : (q-1, k) = 1\}$. For example, the primitive elements for F_7 are $3, 5$.

We shall consider alphabets which are finite dimensional vector spaces over F_q, e.g. F_q^N; this has the usual scalar product. The scalar product allows us to

identify the dual space $(F_q^N)^\star$ with F_q^N: every linear map $A : F_q^N \to F_q$ in this dual space is given by $x \mapsto x \cdot y$ for some vector y. More generally, for any finite S, the set V of functions $f : S \to F_q$ is a vector space over F_q of dimension $|S|$.

We have the ring $F_q[x]$ of polynomials over F_q by the usual definition; we have degrees of polynomials as usual with $\deg 0 := \infty$; then deg is a Euclidean function and the ring becomes a Euclidean Domain. Then we have $X - \alpha \mid A \Leftrightarrow A(\alpha) = 0$ for $A \in F_q[x]$ and also that the polynomials form a PID; for any ideal $I \triangleleft F_q[x]$ the quotient $\frac{F_q[x]}{I}$ is a ring and we have the quotient homomorphism. The quotient is a vector space over F_q with dimension $\deg D$ where $I = (D)$.

An important example is $\frac{F_q[x]}{(x^n-1)} = \{a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} : a_i \in F_q\}$, a vector space of dimension n.

Suppose we are considering codes $c : \mathcal{A} \to \mathcal{B}^N$; if the orders of both alphabets $\mathcal{A}, \mathcal{B}^N$ are powers of the same prime power q then we can consider them as vector spaces over F_q; then c is a map $F_q^K \to F_q^N$ [note that this is not in general the same N as previously in this paragraph; lol the lecturer] for some K, N. Such a map can be much more easily specified when it is linear, as then we only need to specify the images of a basis for its domain; in fact such a map makes both coding and decoding considerably easier.

Definition: write F for a general finite field, then a code is <u>linear</u> if it is given by an injective linear map $c : F^K \to F^N$. The image of c, a subspace of F^N, is called the <u>code book</u> of c; it has dimension $\mathrm{rk} c$. Example: Hamming's code is a linear code.

A linear code $c : F^K \to F^N$ can be given by an $N \times K$ matrix C; since c is injective the matrix has nullity 0 and rank K. We are interested only in the code book, which can be specified by giving a basis for the image of C; the columns of C are one such basis. By column operations we can rearrange C to be in the form $\begin{pmatrix} I \\ A \end{pmatrix}$ where I is the $K \times K$ identity and A is some general $(N-K) \times K$ matrix; then the code is $\vec{x} \mapsto \begin{pmatrix} \vec{x} \\ A\vec{x} \end{pmatrix}$, so the first K terms of the codeword carry information about x and the remainder are check digits.

For a linear code c, the image of c is a subset of F^N of dimension K. So we can construct a linear map $s : F^N \to F^{N-K}$ with $\ker s = \Im c$; such an s is called a <u>syndrome</u> of c. For example, if we have written our matrix C for c in the standard form above, $S = \begin{pmatrix} -A \\ I \end{pmatrix}$ is a syndrome for c: $\ker S = \{\begin{pmatrix} \vec{u} \\ \vec{v} \end{pmatrix} : -A\vec{u} + \vec{v} = 0 = \{\begin{pmatrix} \vec{u} \\ A\vec{u} \end{pmatrix} : u \in F^k\} = \Im C$.

We can use the syndrome to easily check whether $\vec{x} \in F^N$ is a code word - we simply check whether $S\vec{x} = 0$. We can define a code by giving a syndrome; its code book is then $\ker s$.

Example: for $d \in \mathbb{N}$ there are $N = 2^d - 1$ nonzero elements of F_2^d; take these to be the columns of a $d \times N$ matrix S; then the kernel of S is the code book for a Hamming code $c : F_2^{N-d} \to F_2^N$ (e.g. $d = 3$ gives the earlier Hamming code), which will be a perfect 1-error correcting code.

The code $c : F^K \to F^N$ and syndrome $s : F^N \to F^{N-K}$ are dual to each other: their dual maps are $s^\star : F^{N-K} \to F_N, c^\star : F^N \to F^K$ with $\ker c^\star = \Im c^\perp = \ker s^\perp = \Im s^\star$ so $s^\star$ is a linear code with $c^\star$ as its syndrome, i.e. S^T is the matrix for a

linear code with C^T as its syndrome.

The minimum distance for a code is the minimum Hamming distance between two (distinct) codewords; for a linear code we have $d(\vec{c'}, \vec{c}) = d(\vec{0}, \vec{c} - \vec{c'})$ so the minimum distance is $\min d(\vec{0}, \vec{c})$ over nonzero codewords $\vec{c}$. The weight of a codeword $\vec{x}$ is its Hamming distance from 0, so the minimum distance is the minimum weight of a nonzero codeword.

We use minimum distance decoding; suppose we receive $\vec{x} \in F^N$. If $s(\vec{x}) = 0$ then $\vec{x}$ is a codeword, otherwise errors have occured and we must find the codeword closest to $\vec{x}$: for every $\vec{y} \in F^{N-K}$, let $u(\vec{y})$ be the vector $\vec{z} \in s^{-1}(\vec{y})$ with minimum weight; if F^{N-K} is not too large we can do this in advance for every $\vec{y} \in F^{N-K}$ before receiving any messages; we have $s(u(\vec{y})) = \vec{y}$. Now decode $\vec{x}$ as $\vec{c_0} = \vec{x} - u(s(\vec{x}))$; since $s(\vec{x} - u(s(\vec{x}))) = s(\vec{x}) - s(u(s(\vec{x}))) = 0$ $\vec{c_0}$ really is a codeword. For any other codeword $\vec{c}$ we have $d(\vec{c}, \vec{x}) = d(0, \vec{x} - \vec{c})$, but $s(\vec{x} - c) = s(\vec{x})$ so by the definition of u, $d(0, \vec{x} - \vec{x}) \geq d(0, u(s(\vec{x}))) = d(0, \vec{x} - \vec{c_0})$ i.e. $d(\vec{c}, \vec{x}) \geq d(\vec{c_0}, \vec{x})$ so $\vec{c_0}$ is the codeword closest to $\vec{x}$.

Reed-Muller Codes

This is a construction of linear coodes which are useful in dealing with very noisy channnels. Let $S = F_2^M$, a finite set with 2^M elements; let V be the set of $f : S \to F_2$. Then the functions $1_{\vec{y}}(\vec{x}) = 1$ for $\vec{x} = \vec{y}$, 0 otherwise are a basis for V, so $\dim V = 2^M$; using this basis we have the Hamming distance in V is $d(f, y) = |\{\vec{x} \in S : f(\vec{x} \neq g(\vec{x}\}|$. Define $\pi_i : \vec{x} \mapsto x_i$; for $I = \{i(1), \ldots, i(r)\} \subset \{1, \ldots, M\}$ define $\pi_I : \vec{x} \mapsto \pi_{(1)}(\vec{x}) \ldots \pi_{i(r)}(\vec{x}) = \prod_{i \in I} x_i$; this is a function in V. In particular, we have $\pi_\emptyset : x \mapsto 1$.

Lemma: the set of π_I for $I \subset \{1, \ldots, M\}$ forms a basis for V, as we can wrute $1_{\vec{y}}(\vec{x}) = \prod_{i=1}^{M}(x_i - y_i + 1) = \sum_I \alpha_I(\prod_{i \in I} x_i$ for some scalars $\alpha_I \in F_2$ depending on $\vec{y}$, so we can write $1_{\vec{y}} = \sum_I \alpha_I \pi_I$, so the π_I span V; since there are $2^M = \dim V$ such π_I they must form a basis for V.

By this lemma we can write any $f \in V$ as $f = \sum_I \alpha_I \pi_I$. The Reed-Muller code $RM(M, r)$ has code book $\{\sum(\alpha_i \pi_I : |I| \leq r) : \alpha_I \in F_2\}$; this is a subspace of dimension $\binom{M}{0} + \binom{M}{1} + \cdots + \binom{M}{r}$, so gives a linear code of this rank and length $\dim V = 2^M$. For example, $RM(M, 0)$ has two codewords 0,1 so is the repitition code repeating each bit 2^M times; we have $d(0, 1) = 2^M$ so this is also the minimum distance for $RM(M, 0)$. $RM(M, 1)$ has dimension $1 + M$ and the functions π_j as a basis; the Hamming distance has $d(0, \pi_j) = 2^{M-1}$ and we can see this is the minimum distance for $RM(M, 1)$.

Proposition: The Reed-Muller code $RM(M, r)$ has minimum distance 2^{M-r} for $0 \leq r \leq M$: we shall induct on M, and have the result for the base case $M = 1$. Suppose we have the result for $M - 1 \forall r$. Let $J = \{1, \ldots, r\}$; then $d(0, \pi_J) = 2^{M-r}$ so the minimum distance is at least this large. For a nonzero $f = \sum_I \alpha_I \pi_I$ in $RM(M, r)$, split this according to whether $I \ni M$ or not, then $f = \sum_{I \ni M} \alpha_I \pi_I + \sum_{I \not\ni M} \alpha_I \pi_I \pi_M = f_0 + f_1 \pi_M$; f_0, f_1 are not dependent on x_M, we can think of them as functions on F_2^{M-1}. Then $f_0 \in RM(M-1, r)$, $f_1 \in RM(M-1, r-1)$; let d_M, d_{M-1} be the Hamming distances in F_2^M, F_2^{M-1} respectively, and observe $f(x_1, \ldots, x_M - 1, 0) = f_0(x_1, \ldots, x_{M-1}$ on the set where $x_M = 0$, $f(x_1, \ldots, x_{M-1}, 1) = (f_0 + f_1)(x_1, \ldots, x_{M-1}$ on the set where $x_M = 1$. Since f is nonzero, then f_0 or $f_0 + f_1$ is nonzero; if $f_0 = 0$ we have f_1 nonzero and $d_M(0, f) = d_{M-1}(0, f_1) \geq 2^{(M-1)-(r-1)}$; if $f_0 \neq 0, f_1 = -f_0$ then $f_0 = -f_1 \in RM(M-1, r-1)$ and we similarly

have $d_M(0,f) = d_{M-1}(0,f_0) \geq 2^{(M-1)-(r-1)}$. Finally for $f_0 \neq 0, f_1 \neq -f_0$ we have $d_M(0,f) = d_{M-1}(0,f_0) + d_{M-1}(0,f_0+f_1) \geq 2 \times 2^{M-1-r}$ since both f_0, f_0+f_1 are nonzero elements of $RM(M-1,r)$ so, in all cases $d_M(0,f) \geq 2^{M-r}$ as required. Example: the $RM(5,1)$ code as used for the Mariner mission to mars has rank $1+5=6$ and length $2^5=23$, so its rate of transmission is $\frac{6}{32} = \frac{3}{16}$; by this proposition it has minimum distance $2^4 = 16$ so it is 15-error detecting and 7-error correcting.

11 Cyclic Codes

A linear code is cyclic if for any codeword $c_1c_2\dots c_N$, $c_Nc_1\dots c_{N-1}$ is also a codeword, i.e. the shift map $T : F^N \to F^N : \vec{x} \mapsto \begin{pmatrix} x_N \\ x_1 \\ \dots \\ x_{N-1} \end{pmatrix}$ maps the code book into itself. These are more easily described by identifying F^N with $\frac{F[x]}{x^N-1}$; then T corresponds to the map $P(X) \mapsto XP(X)$. Let q be the quotient homomorphism $F[x] \to \frac{F[x]}{x^N-1}$.

Proposition: For W a vector subspace of $\frac{F[x]}{x^N-1}$, $J = q^{-1}(W)$, W is the code book for a cyclic code iff J is an ideal of $F[x]$: the quotient map q is linear, so J is a vector subspace of $F[x]$. Suppose W is the codebook for a cylic code, then $P(x) \in W \Rightarrow xP(x) \in W \forall P(x)$, so $P(x) \in J \Rightarrow xP(x) \in J$, so $A(x)P(x) = \sum a_k X^k P(x) \in J$ for any polynomial $A(x)$, so J is an ideal of $F[x]$; conversely, if $J \triangleleft F[x]$ and $P(x) \in W$ then $P(x) \in J$ so $xP(x) \in J$, so $xP(x) \in W$ and W is a codebook as required.

Corollary (Generators of cyclic codes): Any W which is the codebook for a cyclic code in $\frac{F[x]}{X^N-1}$ has $W = \{A(x)G(x) : A(x) \in F[x]\}$ for some generator polynomial $G(x) \in F[x]$; also $G(x)$ is a divisor of $x^N - 1$: by the proposition $J = q^{-1}(W) \triangleleft F[x]$; since $F[x]$ is a PID this implies $J = G(x)F[x]$ for some $G(x)$, so we have the first part; then $0 = q(x^N-1) \in W$ so $x^N - 1 \in J$, so $G(x) \mid x^N - 1$. Take $G(x)$ monic, then it is uniquely determined by the cyclic code book W; since $G(x) \mid x^N - 1$ write $x^N - 1 = G(x)H(x)$; $H(x)$ is called the check polynomial of the code, and is unique up to scalar multiplication.

Proposition: let $G(x)$ as above, $D = \deg G$. then $G(x), xG(x), \dots, x^{N-D-1}G(x)$ form a basis for the code book in $\frac{F[x]}{x^N-1}$, so the code book has rank $N-D$: we have $x^kG(x) \in J \forall k$, so $x^kG(x) \in W \forall j$ [note I am not distinguishing, as the lecturer did, between classes $x^kG(x) + (x^N-1)F[x] \in \frac{F[x]}{x^N-1}$ and polynomials $x^kG(x) \in F[x]$]. Every vector of W is $G(x)P(x)$ for some $P(x)$; by reducing $G(x)P(x)$ modulo x^N-1 we may take $\deg P < N-D$, so $G(x)P(x) = p_0G(x) + \dots + p_{N-D-1}X^{N-D-1}G(x)$ for some p_i; this was true for a general element of W, so $G(x), xG(x), \dots, x^{N-D-1}G(x)$ span W; if we have $p_0G(x) + \dots + p_{N-D-1}X^{N-D-1}G(x) = 0$ for some p_i then let $P(x)$ be defined by these p_i, then $P(x)G(x) = 0$ so $(x^N-1) \mid P(x)G(x)$, so $H(x) \mid P(x)$; since $\deg P < N-D = \deg H$ this means $P(X) = 0$, so our spanning set is linearly independent so forms a basis as required.

Using the above basis for the code book, a matrix for the code is given by the

$N\times(N-D)$ matrix $\begin{pmatrix} g_0 & 0 & 0 & \dots & 0 \\ g_1 & g_0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ g_D & g_{D-1} & \dots & & \\ 0 & g_D & \dots & & \\ \dots & \dots & & & \\ 0 & \dots & \dots & 0 & g_D \end{pmatrix}$ where $G(x) = g_0+g_1x+\dots+g_Dx^D$.

Let S be the $D \times N$ matrix $\begin{pmatrix} 0 & 0 & \dots & h_1 & h_0 \\ 0 & 0 & \dots & h_0 & 0 \\ \dots & \dots & & & \\ 0 & h_{N-D} & \dots & 0 & 0 \\ h_{N-D} & h_{N-D-1} & \dots & 0 & 0 \end{pmatrix}$ where $H(x) = h_0+h_1x+\dots+h_{N-D}x^{N-D}$. Then $h_{N-D} \neq 0$ so the rows of S are linearly independent, so S has rank D so its nullity must be $N-D$; however, we have that each column is in the kernel of S, as $G(x)D(x) = x^N - 1$ so $\sum_j h_{M-j}g_j = 0 \forall 0 < m < N$. So the kernel of S is spanned by the columns of C, so $\ker S = W = \mathfrak{I}C$ and S is a syndrome matrix for the code.

12 BCH Codes

Recall the definition of the characteristic of a finite field; we shall take fields of characteristic 2 (so of order 2^r, but the arguments generalise to other characteristics.

We know that any cyclic code has a generator polynomial $G(X)$ with $G(X) \mid (X^N - 1)$, where N is the length of the code; assume N odd. From Galois theory we know there is a splitting field K for $X^N - 1/F$.

Lemma: $X^N - 1$ has N distinct roots in K forming a cyclic group: suppose we had a repeated root α so $X^N - 1 = (X - \alpha)^2P(X)$ for some polynomial P; taking the formal derivative, $NX^{N-1} = 2(X - \alpha)P(X) + (X - \alpha)^2P'(X) = (X - \alpha)(2P(X) + (X-\alpha)P'(X))$, so $X-\alpha$ divides X^N-1 and NX^{N-1}; since N is odd, $N1 \neq 0$ so $X-\alpha$ divides the highest common factor of X^N-1, X^{N-1}, but this is 1, a contradiction. So the set S of roots of $X^N - 1$ in K contains N elements; it is a subgroup of the cyclic group $K^\times$, so cyclic.

By the above lemma we can find a primitive root α of $X^N - 1$ in the field K, with the other roots being $\alpha^2, \dots, \alpha^{N-1}$; this is not true for N even, e.g. $X^2 - 1 = (X - 1)^2$.

12.1 BCH Codes

Let F a finite field of characteristic 2, N an odd integer, K a splitting field for $X^N - 1$ over F, α a primitive root of $X^N - 1$ in K. The BCH code with design distance $\delta \in 1, 2, \dots, N$ is a cyclic code of length N in F defined by the ideal $J = \{P(X) \in F[x] : P(\alpha) = P(\alpha^2) = \dots = P(\alpha^{\delta-1}) = 0\}$. The generating polynomial for this code is the minimal polynomial in $F[x]$ with zeroes at $\alpha, \alpha^2, \dots, \alpha^{\delta-1}$ in K; it is therefore the lcm of the minimal polynomials of these, so clearly a factor of $X^N - 1$, so the BCH code is a cyclic linear code of length N.

Lemma (van der Monde determinants): the determinant $\Delta(X_1, X_2, \dots, X_K) =$

$\begin{vmatrix} X_1 & X_2 & \dots & X_K \\ X_1^2 & X_2^2 & \dots & X_K^2 \\ \dots & \dots & \dots & \dots \\ X_1^K & X_2^K & \dots & X_K^K \end{vmatrix}$ has $\Delta(X_1,\dots,X_K) = (\prod_{k=0}^K X_k)(\prod_{i<j}(X_i - X_j))$: induct on K, clearly true for $K = 1$. Consider $\Delta(X_1,\dots,X_K$ as a polynomial in X_K with coefficients polynomials in $X_1,\dots,X_{K-1}$; it is of degree K. When $X_K = 0$ or one of the X_i, the determinant is clearly 0, so $X_K(\prod_{i<K}(X_i - X_K)$ divides the determinant, i.e. $\Delta(X_1,\dots,X_K) = c(X_1,\dots,X_{K-1}X_K(\prod_{i<K}(X_i - X_K))$ for some polynomial c; the leading coefficients of both sides here are $\Delta(X_1,\dots,X_{K-1}), c(X_1,\dots,X_{K-1})$, so $\Delta(X_1,\dots,X_K) = \Delta(X_1,\dots,X_{K-1})X_K(\prod_{i<K}(X_i - X_K))$ as required.

Theorem: the minimum distance for a BCH code with design distance δ is $\geq \delta$: let $P(X) = p_0 + \dots + p_{N-1}X^{N-1}$ a nonzero polynomial in the codebook, then $A\begin{pmatrix} p_0 \\ \dots \\ p_{N-1} \end{pmatrix} = \begin{pmatrix} 0 \\ \dots \\ 0 \end{pmatrix}$, where $A = \begin{pmatrix} \alpha & \alpha^2 & \dots & \alpha^{N-1} \\ \alpha^2 & \alpha^4 & \dots & \alpha^{2(N-1)} \\ \dots & \dots & \dots & \dots \\ \alpha^{\delta-1}\alpha^{2(\delta-1)} & \dots & \alpha^{(N-1)(\delta-1)} & \end{pmatrix}$.
By a lemma, above, any $\delta - 1$ columns of A are linearly independent, because no two of the N powers of α are equal or 0. So we must have at least $\delta - 1$ of the p_k being 0, so the minimum distance is at least $\delta - 1$.

Now, how to decode: by a proposition above our code is t-error correcting where $t = \lfloor \frac{1}{2}(\delta - 1) \rfloor$; suppose $\vec{c}$ is sent and we receive $\vec{r} = \vec{c} + \vec{e}$, for an error vector $\vec{e}$ with at most t non-zero entries. Let $C(X), R(X), E(X)$ be the polynomials corresponding to $\vec{c}, \vec{r}, \vec{e}$, of degrees $< N$; we have that $C(\alpha) = C(\alpha^2) = \dots = C(\alpha^{\delta-1}) = 0$, so $R(\alpha) = E^\alpha, \dots, R(\alpha^{\delta-1}) = E(\alpha^{\delta-1})$. Therefore, we calculate $R(\alpha^j)$ for $j = 1,\dots,\delta - 1$; if these are all 0 $R(X)$ is a codeword and there have been no (or at least $\delta + 1$) errors; otherwise let $\epsilon = \{i : e_i \neq 0\}$ be the set of indicies at which errors occur; the error locator polynomial is $\sigma(X) = \prod_{i\in\epsilon}(1 - \alpha^i X)$; this is a polynomial (over K) of degree $|\epsilon|$; once we know it we can find which α^{-1} are the roots of it, and hence find the indicies at which errors have occured, and then by changing these entries correct the error.

Consider the power series $\eta(X) = \sum_{j=1}^\infty E(\alpha^j)X^j$ (the coefficients repeat since $\alpha^N = 1$); we have $E(\alpha^j) = R(\alpha^j)$ for $j = 1, 2, \dots, \delta - 1$, so we can calculate these coefficients in terms of $\vec{r}$: $\eta(X) = \sum_{j=1}^\infty E(\alpha^j)X^j = \sum_{j=1}^\infty \sum_{i\in\epsilon} \alpha^{ij}X^j = \sum_{i\in\epsilon}\sum_{j=1}^\infty \alpha^{ij}X^j = \sum_{i\in\epsilon} \frac{\alpha^i X}{1-\alpha^i X}$; write this as $\frac{\omega(X)}{\sigma(X)}$ where $\omega(X) = \sum_{i\in\epsilon} \alpha^i X \prod_{j\neq i}(1 - \alpha^j X)$; note that both σ and η have degree $|\epsilon| \leq t$. So $\sigma(X)\eta(X) = \omega(X)$; using $E(\alpha^k) = R(\alpha^k)$ for $k = 1, 2, \dots, 2t$ we have $(\sigma + \dots + \sigma_t X^t \times (R(\alpha)X + \dots + R(\alpha^{2t})X^{2t} + E(\alpha^{2t+1})X^{2t+1} + \dots) = \omega_0 + \dots + \omega_t X^t$; the coefficients of X^n for $t < n \leq 2t$ give $\sum_{j=0}^t \sigma_j R(\alpha^{n-j}) = 0$, which do not involve the terms $E(\alpha^k)X^k$, so we get the equations $\begin{pmatrix} R(\alpha^{t+1}) & R(\alpha^t) & \dots & R(\alpha) \\ R(\alpha^{t+2}) & R(\alpha^{t+1}) & \dots^R(\alpha^2) & \\ \dots & \dots & \dots & \dots \\ R(\alpha^{2t} & \dots & R(\alpha^t) & \end{pmatrix}\vec{\sigma} = \vec{0}$; the matrix is a $t \times (t+1)$ matrix so has a vector $\sigma \neq 0$ in its kernel; this σ gives us the error locator polynomial $\sigma(X)$ as required.

The special case $F = F_q, N = q - 1$ is called a Reed-Solomon code.

13 Linear Feedback Shift Registers

Suppose we have K registers each taking values in the finite field $F = F_q$; initially they are set to $x_0, \dots, x_{K-1}$, called the initial fill; at each time step we shift the values in the registers down, and fill the final register with a new value determined by the old values in the registers, i.e. if the register i takes the value $X_i(t)$ at time $t = 0, 1, \dots$, we have $X_i(t+1) = X_{i+1}(t) \forall 0 \le i \le K-2$, and $X_{K-1}(t+1) = f(X_0(t), \dots, X_{K-1}(t))$, where f is some feedback function. We call this system a linear feedback shift register when f is linear, say $f(X_0, \dots, X_{K-1}) = -c_0X_0 - \dots - c_{K-1}X_{K-1}$; then $c_0X_0(t) + \dots + c_{K-1}X_{K-1}(t) + X_K(t) = 0 \forall t$.

This system gives (and is determined by) a stream of values $X_0(0), X_0(1), \dots$; this begins with the initial fill $X_0(0) = x_0, X_0(1) = x_1, \dots, X_0(K-1) = x_{K-1}$, and satisfies the recurrence relation $c_0X_0(t) + c_1X_0(t+1) + \dots + c_{K-1}X_0(t+K-1) + X_0(t+K) = 0$; the feedback polynomial $C(X) = c_0 + c_1X + \dots + c_{K-1}X^{K-1} + X_K$ is the characteristic polynomial for this recurrence relation, so determines the solutions. The stream of values from a system such as this appears superficially random, but is not.

We shall assume $c_0 \neq 0$; otherwise, the value in the 0th register does not affect anything, so we can consider the LFSR formed by the remaining $K-1$ registers.

Proposition: a LFSR is periodic: let the vector $\vec{V} \in F^K$ be given by $V_i(t) = X_0(t+i)$. There are only finitely many vectors in F^K, so the sequence $(\vec{V}(t))$ for $t \in \mathbb{N}$ must eventually repeat an earlier value; suppose this first occurs when $\vec{V}(N) = \vec{V}(j)$ for some $0 \le j < N$. Then we have by the definition of the LFSR that $\vec{V}(t+1) = M\vec{V}(t)$ where $M = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \\ -c_0 & -c_1 & -c_2 & \dots & -c_{K-1} \end{pmatrix}$; this has determinant $\pm c_0 \neq 0$ so is invertible. We have $\vec{V}(t) = M^t\vec{V}(0)$, so $M^N\vec{V}(0) = M^j\vec{V}(0)$; if $j \neq 0$ then multiplying by M^{-1} we have an earlier repeat, so $j = 0$, $M^N\vec{V}(0) = \vec{V}(0)$ and the squence of $\vec{V}(t)$ is periodic with period N, so the sequence $X_0(t)$ is also.

We clearly have $M\vec{0} = \vec{0}$, so the largest possible period is $N = q^K - 1$; in this case the sequence $\vec{V}(0), \dots, \vec{V}(N-1)$ takes each value in $F^K \setminus \{0\}$ exactly once; therefore in any period of length $q^K - 1$ the number 0 occurs $q^{K-1} - 1$ times and every other number of F occurs q^{K-1} times; therefore in a very weak sence the sequence appears random, but it is entirely predictable given one complete period.

Say we have a LFSR of period N; define $\vec{W}(t) \in F^N$ by $W_i(t) = X_0(t+i)$, then $\vec{W}(t+1)$ is a cyclic shift of $\vec{W}(t)$, as $X_0(t+N) = X_0(t)$; we also have $c_0\vec{W}(0) + c_1\vec{W}(1) + \dots + c_{K-1}\vec{W}(K-1) + \vec{W}(K) = \vec{0}$. So the vectors $\vec{W}(0), \dots, \vec{W}(K-1)$ span the code book for a cyclic code of length N.

If we receive a sequence a_t elements of F from a linear feedback shift register, and want to determine the register which produced it, we simply need to solve linear equations to find the coefficients of the feedback polynomial: suppose the LFSR had K registers and feedback polynomial $C(X) =$

$c_0 + \cdots + c_{K-1}X^{K-1} + X^K$. Then we have $c_0 a_t + \cdots + c_{K-1}a_{t+K-1} + a_{t+K} = 0$, i.e. $\begin{pmatrix} a_0 & a_1 & \dots & a_{K-1} & a_K \\ a_0 & a_2 & \dots & a_K & a_{K+1} \\ \dots & \dots & \dots & \dots & \dots \\ a_K & a_{K+1} & \dots & a_{2K-1} & a_{2K} \end{pmatrix} \begin{pmatrix} c_0 \\ \dots \\ c_{K-1} \\ 1 \end{pmatrix} = \vec{0}(\star)$, so the matrix here has determinant 0. Therefore, to find the LFSR, following Berlekamp and Massey, we take the smallest possible value of K and compute the determinant of that matrix; if it is nonzero we cannot solve the problem with K registers, so increment K and continue. If the determinant is 0 we find the c_j by solving $(\star)$, then check whether this works (i.e. set up a LFSR with the feedback polynomial $C(X) = c_0 + c_1 X + \cdots + c_{K-1}X^{K-1} + X^K$ and initial fill $a_0, \dots, a_{K-1}$, then check whether this produces the same stream as that we received); if this does not work then we consider other possible solutions c_j, or increase K if there are no such.

13.1 Power Series

Let $A(x) = a_0 + a_1 x + \cdots + a_D x^D, B(x) = 1 + b_1 X + \cdots + b_K x^K \in F[x]$. Writing $B(x) = 1 - \beta(x)$ we have $\frac{1}{B(x)} = \frac{1}{1-\beta(x)} = \sum_{j=0}^{\infty} \beta(x)^j$; expanding the powers of $\beta(x)$ we obtain a formal power series for $\frac{1}{B(x)}$. Multiplying this by $A(x)$ we have a formal power series $\frac{A(x)}{B(x)} = \sum_{j=0}^{\infty} u_j x^j$. We are not concerend with convergence here, but only require that the coefficients of each power of x on both sides of the equation match; thus we need $A(x) = B(x)\sum_{j=0}^{\infty} u_j x^j$, or equivalently $a_n = \sum_{j=0}^{n} b_j u_{n-j} \forall n \in 0, 1, \dots$. So the sequence u_j satisfies $u_n = a_n - \sum_{j=1}^{n} b_j u_{n-j}$ for $n \in 0, 1, \dots, D$ and $u_n = -\sum_{j=1}^{n} b_j u_{n-j}$ for $n > D$. Thus the sequence is the stream output by a LFSR with feedback polynomial $C(x) = b_K + b_{K-1}x + \cdots + b_1 x^{K-1} + x^K$.

Therefore, determining whether a stream of elements of F is the output of a LFSR is equivalent to determining whether a formal power series is a quotient of two polynomials; compare with the method of decoding BCH codes, above.

Exercise: a decimal repeats iff it represents a rational number; relate this to LFSRs and the periodicity proven in the above proposition.

14 Cryptography

Suppose we are transmitting a message, the plaintext, from a finite alphabet $\mathcal{A}$; the encrypting function $e_k : \mathcal{A} \to \mathcal{B}$ is taken as varying depending on a key k from some finite set $\mathcal{K}$; the encrypted text is then the ciphertext; since we must be able to decipher the message there is a decrypting function $d_k : \mathcal{B} \to \mathcal{A}$ such that $d_k(e_k(a)) = a \forall a \in \mathcal{A}$. For example, in a [monoalphabetic] substitution cipher the key is a permutation κ of $\mathcal{A}$ and $e_\kappa(a) = \kappa(a)$; for an English message this is easy to decode by frequency analysis.

We shall assume an attacker knows the encryption method (i.e. the functions e_k, d_k) but not the specific k we are using; levels of attack we might consider are 1) a ciphertext only attack, where the attacker has only a piece of ciphertext 2) a known plaintext attack, where the attacker knows some plaintext and corresponding ciphertext, and finally 3) a chosen plaintext attack, where the attacker can choose arbitrary plaintexts and obtain the corresponding ciphertext.

A simple substitution cipher or even the Vigenère cipher are vulnerable at level 1 given a sufficiently long ciphertext; for a modern cipher we want to be resistant to a level 3 attack. Of course every cipher is vulnerable to such an attack in some sense: we can perform a brute force attack by encrypting some fixed plaintext with every possible key $k \in \mathcal{K}$, but we can hope to make the amount of work involved prohibitively large.

14.1 Equivocation

Suppose we choose a random plaintext message M according to some probability distribution on the set of possible messages $\mathcal{M}$, then independently choose a random key $K \in \mathcal{K}$. The ciphertext is then a random variable $C = c_K(M)$. Consider this as transmitting M through a noisy channel to produce C; the noise comes from the choice of key. We call the entropy $H(M|C)$ the message equivocation; this measures the amount of uncertainty we have about the plaintext given the ciphertext; similarly we have the key equivocation $H(K|C)$; we unsurprisingly have:

Proposition: the message equivocation is $\leq$ the key equivocation: M is a function of (K, C) so we have $H(K|C) = H(K, C) - H(C) = H(K, M, C) - H(C) = H(K, M, C) - H(M, C) + H(M, C) - H(C) = H(K|M, C) + H(M|C) \geq H(M|C)$.

We say a cipher has perfect secrecy if the ciphertext gives us no information about the plaintext: $H(M|C) = H(M)$, i.e. $H(M, C) = H(M) + H(C)$ so M, C are independent (Equivallenty $I(M, C) = 0$).

Proposition: if a cipher has perfect secrecy, there must be at least as many possible keys as possible plaintext messages (by which we mean, keys/messages with strictly positive probabilities): fix a message $m_0 \in \mathcal{M}$ and key $k_0 \in \mathcal{K}$, both with strictly positive probability; then $c_0 = e_{k_0}(m_0)$ has strictly positive probability. For any message m, we have $P(C = c_0) = P(C = c_0|M = m)$, so $\exists k \in \mathcal{K}$ with $c_0 = e_k(m)$; if two message m_1, m_2 give the same key then we have $e_k(m_1) = c_0 = e_k(m_2) \therefore m_1 = m_2$, so we have an injective map $m \mapsto k$.

This result means perfect secrecy is usually impractical; a one-time pad cipher is an example of perfect secrecy (if our key is $k_1 k_2 \dots$ and $|\mathcal{A} = q$, our message $m = a_1 a_2 \dots a_N$ is enciphered as $b_1 \dots b_N$ with $b_i = a_i + k_i \mod q$; then $P(M = \vec{m}|C = \vec{c}) = P(M = \vec{M}, K = \vec{c} - \vec{m}) = P(M = \vec{m})P(K = \vec{c} - \vec{m}) = \frac{1}{q^N} P(M = \vec{m})$ (working modulo q), so M, C are independent.

As well as the key length issue, a one-time pad requires a genuinely random key sequence, which is difficlet to produce. If we try and use some pseudo-random sequence, e.g. the output of a LFSR, then we are vulnerable; in this case a level 2 attack will work, since given a plaintext and ciphertext we can compute the key sequence, then use Berlekamp-Massey to compute the feedback polynomial and hence the key.

14.2 Unicity

When trying to break a simple substitution cipher by frequency analysis, we are aided by the fact that we know the original message made sense and, for a long enough ciphertext, there is only one key giving a sensible message for the plaintext. We ask: how long must our ciphertext be for this to apply?

Suppose $m = a_1 \dots a_N$; the letters are random variables A_j giving a random message $\vec{M} = A_1 \dots A_N$. Assume the entropy of this sequence is NH for some constant H, the entropy per letter (cf section 5). When this is enciphered we get a ciphertext $\vec{C} = C_1 \dots C_N$; the unicity is the least N for which $H(K|\vec{C}) = 0$.

$K, \vec{M}$ determine $\vec{C}$ and conversely $K, \vec{C}$ determine $\vec{M}$, so we have $H(K|\vec{C}) = H(K, \vec{C}) - H(\vec{C}) = H(\vec{M}, K, \vec{C}) - H(\vec{C}) = H(\vec{M}, K) - H(\vec{C}) = H(\vec{M}) + H(K) - H(\vec{C})$; we assume the ciphertext will be uniformly distributed (as is the case with most useful ciphers) so $H(\vec{C}) = N \log |\mathcal{C}|$, and K was chosen uniformly from $\mathcal{K}$, so $H(K) = \log |\mathcal{K}|$ and we have $0 = NH + \log |\mathcal{K}| - N \log |\mathcal{C}|$, so $N = \frac{\log |\mathcal{K}|}{\log |\mathcal{C}| - H}$. So we should not use a single key for messages longer than this N.

Example: for English, $H \approx 1.2$; suppose we use a substitution cipher with alphabets the 26 letters plus space, then $|\mathcal{K}| = 27!, \log |\mathcal{K}| = 93.14, |\mathcal{C}| = 27 \therefore N = \frac{93.14}{\log 27 - 1.2} \approx 26.2$; thus we expect to be able to find the key uniquely if the ciphertext is longer than 26 letters.

[I had stopped attending lectures in this course quite early on; at this point I stopped writing my notes, which are after all merely reworkings of the available online notes for the course]