CSC 310: Information Theory

University of Toronto, Fall 2011

Instructor: Radford M. Neal

Week 10

Extensions of Channels

The Nth extension of a channel consists of N independent copies of the channel, replicated in space or time.

The input alphabet for this extension, $\mathcal{A}_X$, consists of N-tuples $(a_{i_1}, \ldots, a_{i_N})$. The output alphabet, $\mathcal{A}_Y$, consists of N-tuples $(b_{j_1}, \ldots, b_{j_N})$

Assuming the N copies don't interact, the transition probabilities for the extension are

$$Q_{j_1,\ldots,j_N|i_1,\ldots,i_N} \quad = \quad Q_{j_1|i_1} \cdots Q_{j_N|i_N}$$

If we use input probabilities of

$$p_{i_1,\ldots,i_N} \quad = \quad p_{i_1} \cdots p_{i_N}$$

it is easy to show that the input and output entropies, the conditional entropies, and the mutual information are all N times those of the original channel.

Capacity of the Extension

We can maximize mutual information for the extension by using an input distribution in which

- each symbol is independent
- each symbol has the distribution that maximizes mutual information for the original channel.

It follows that the capacity of the extension is N times the capacity of the original channel.

But treating the N copies independently is uninteresting — we gain nothing over the original channel.

The strategy: We *don't* chose an input distribution that maximizes mutual information, but rather use one that *almost* maximizes mutual information, while allowing us to *correct almost all errors*.

Codes for the Extension

A *code*, $\mathcal{C}$, for the Nth extension is a subset of the set of all possible blocks of N input symbols — ie, $\mathcal{C} \subseteq \mathcal{A}_X^N$.

The elements of $\mathcal{C}$ are called the *codewords*. These are the only blocks that we transmit.

For example, one code for the third extension of a BSC is the "repetition code", in which there are two codewords, 000 and 111.

The Nth extension together with the code can be seen as a channel with $|\mathcal{C}|$ input symbols and $|\mathcal{A}_Y|^N$ output symbols.

Decoding to a Codeword

When the sender transmits a codeword in $\mathcal{C}$, the receiver might (in general) see any output block, $b_{j_1} \cdots b_{j_N} \in \mathcal{A}_Y^N$.

The receiver can try to *decode* this output in order to recover the codeword that was sent.

The optimal method of decoding is to choose a codeword, $w \in \mathcal{C}$, which maximizes

$$P(w \,|\, b_{j_1} \cdots b_{j_N}) \quad = \quad \frac{P(w)\, P(b_{j_1} \cdots b_{j_N} \,|\, w)}{P(b_{j_1} \cdots b_{j_N})}$$

In case of a tie, we pick one of the best w arbitrarily.

If $P(w)$ is the same for all $w \in \mathcal{C}$, this scheme is equivalent to choosing w to maximize the "likelihood", $P(b_{j_1} \cdots b_{j_N} \,|\, w)$.

Example: A Repetition Code

Suppose we use the three-symbol repetition code for a BSC with $f = 0.1$. Assume that the probability of 000 being sent is 0.7 and the probability of 111 being sent is 0.3.

What codeword should the decoder guess if the received symbols are 101?

$$P(w = 000 \,|\, b_1 = 1, b_2 = 0, b_3 = 1)$$

$$= \frac{P(w = 000)\, P(b_1 = 1, b_2 = 0, b_3 = 1 \,|\, w = 000)}{P(b_1 = 1, b_2 = 0, b_3 = 1)}$$

$$= \frac{0.7 \times 0.1 \times 0.9 \times 0.1}{0.7 \times 0.1 \times 0.9 \times 0.1 + 0.3 \times 0.9 \times 0.1 \times 0.9} = 0.206$$

$$P(w = 111 \,|\, b_1 = 1, b_2 = 0, b_3 = 1)$$

$$= \frac{P(w = 111)\, P(b_1 = 1, b_2 = 0, b_3 = 1 \,|\, w = 000)}{P(b_1 = 1, b_2 = 0, b_3 = 1)}$$

$$= \frac{0.3 \times 0.9 \times 0.1 \times 0.9}{0.7 \times 0.1 \times 0.9 \times 0.1 + 0.3 \times 0.9 \times 0.1 \times 0.9} = 0.794$$

The decoder should guess that 111 was sent.

Associating Codewords with Messages

Suppose our original message is a sequence of K bits. (Or we might break our message up into K-bit blocks.)

If we use a code with 2^K codewords, we can send this message (or block) as follows:

- The encoder maps the block of K message symbols to a codeword.
- The encoder transmits this codeword.
- The decoder guesses at the codeword sent.
- The decoder maps the guessed codeword back to a block of K message symbols.

We hope that the block of decoded message symbols is the same as the original block!

Example: To send binary messages through a BSC with the repetition code, we use blocks of size one, and the map $0 \leftrightarrow 000,\ 1 \leftrightarrow 111$.

Decoding for a BSC By Maximum Likelihood

For a BSC, if all codewords are equally likely, the optimal decoding is the codeword differing in the fewest bits from what was received.

The number of bits where two bit sequences, u and v, differ is called the *Hamming distance*, written $d(u, v)$. Example: $d(00110, 01101) = 3$.

The probability that a codeword w of length N will be received as a block b through a BSC with error probability f is

$$(1-f)^{N-d(w,b)} \, f^{\,d(w,b)} \;=\; (1-f)^N \left(\frac{f}{1-f}\right)^{d(w,b)}$$

If $f < 1/2$, and hence $f/(1-f) < 1$, choosing w to maximize this likelihood is the same as choosing w to minimize the Hamming distance between w and b.

An Example Code for the BSC

Here's a code with four 5-bit codewords:

$$00000, \quad 00111, \quad 11001, \quad 11110$$

We might decide to map between 2-bit blocks of message bits and these codewords as follows:

$$00 \leftrightarrow 00000 \qquad 01 \leftrightarrow 00111$$

$$10 \leftrightarrow 11001 \qquad 11 \leftrightarrow 11110$$

Suppose the sender encodes the message block 01 as 00111 and transmits it through a BSC, and that the receiver then sees the output 00101.

How should this be decoded? If all 2-bit messages are equally likely to be encoded, we just look at the Hamming distances to each codeword:

$$d(00000, 00101) = 2 \quad d(00111, 00101) = 1$$

$$d(11001, 00101) = 3 \quad d(11110, 00101) = 4$$

So the decoder picks the codeword 00111, corresponding to the message block 01. Here, this is correct — but the decoder won't always be right!

The Rate of a Code

A code $\mathcal{C}$ with $|\mathcal{C}|$ binary codewords of length N, is said to have *rate*

$$R \;=\; \frac{\log_2 |\mathcal{C}|}{N}$$

Suppose we are sending binary messages through a binary channel, using a code with 2^K codewords of length N. Then the rate will be

$$R \;=\; K/N$$

For example, if we encode message blocks of 100 bits into codewords consisting of 300 bits, the rate will be 1/3.

A Preview of the Noisy Coding Theorem

Shannon's noisy coding theorem states that:

> For any channel with capacity C, any desired error probability, $\epsilon > 0$, and any transmission rate, $R < C$, there exists a code with some length N having rate at least R such that the probability of error when decoding this code by maximum likelihood is less than ϵ.

In other words: We can transmit at a rate arbitrarily close to the channel capacity with arbitrarily small probability of error.

A near converse is also true: We *cannot* transmit with arbitrarily small error probability at a rate greater than the channel capacity.

Why We Can't Use a BSC Beyond Capacity With Vanishing Error

We'll see here that if we could transmit through a BSC beyond the capacity $C = 1 - H_2(f)$, with vanishingly small error probability, we could compress data to less than its entropy, which we know isn't possible.

In particular, suppose that we can encode blocks of length K into codewords of length N, with a very small probability of decoding error...

Continuing...

Here's how we use this code to compress data:

- Represent our data as two blocks: x is of length K and has bit probabilities of $1/2$, y is of length N and has bit probabilities of f and $1-f$. The total information in x and y is $K + NH_2(f)$.
- Encode x in a codeword w of length N, and compute $z = w + y$, with addition modulo 2. This z is the compressed form of the data.
- Apply the decoding algorithm to recover w from z, treating y as noise. We can then recover x from w and also $y = z - w$.
- The encoder checks whether the previous step would produce any errors, and if so transmits extra bits to identify corrections. This adds only a few bits, on average.

The result: We compressed a source with entropy $K + NH_2(f)$ into only slightly more than N bits.

This is possible only if $N \geq K + NH_2(f)$, which implies that

$$R = K/N \leq 1 - H_2(f) = C$$

The Finite Field Z_2

From now on, we look only at binary channels, whose input and output alphabets are both $\{0, 1\}$.

We will look at the symbols 0 and 1 as elements of Z_2, the integers considered modulo 2.

Z_2 (also called F_2 or $GF(2)$) is the smallest example of a "field" — a collection of "numbers" that behave like real and complex numbers. Specifically, in a field:

- Addition and multiplication are defined. They are commutative and associative. Multiplication is distributive over addition.
- There are numbers called 0 and 1, such that $z + 0 = z$ and $z \cdot 1 = z$ for all z.
- Subtraction and division (except by 0) can be done, and these operations are the inverses of addition and multiplication.

Arithmetic in Z_2

Addition and multiplication in Z_2 are defined as follows:

$$\begin{array}{ll} 0+0=0 & 0\cdot 0=0 \\ 0+1=1 & 0\cdot 1=0 \\ 1+0=1 & 1\cdot 0=0 \\ 1+1=0 & 1\cdot 1=1 \end{array}$$

This can also be seen as arithmetic modulo 2, in which we always take the remainder of the result after dividing by 2.

Viewed as logical operations, addition is the same as 'exclusive-or', and multiplication is the same as 'and'.

Note: In Z_2, $-a = a$, and hence $a - b = a + b$.

Vector Spaces Over Z_2

Just as we can define vectors over the reals, we can define vectors over any other field, including over Z_2. We get to add such vectors, and multiply them by a scalar from the field.

We can think of these vectors as N-tuples of field elements. For instance, with vectors of length five over Z_2:

$$\begin{aligned} (1,0,0,1,1) + (0,1,0,0,1) &= (1,1,0,1,0) \\ 1 \cdot (1,0,0,1,1) &= (1,0,0,1,1) \\ 0 \cdot (1,0,0,1,1) &= (0,0,0,0,0) \end{aligned}$$

Most properties of real vector spaces hold for vectors over Z_2 — eg, the existence of basis vectors.

We refer to the vector space of all N-tuples from Z_2 as Z_2^N. We will use boldface letters such as $\mathbf{u}$ and $\mathbf{v}$ to refer to such vectors.

Linear Codes

We can view Z_2^N as the input and output alphabet of the Nth extension of a binary channel.

A code, $\mathcal{C}$, for this extension of the channel is a subset of Z_2^N.

$\mathcal{C}$ is a *linear code* if the following conditions hold:

1) If $\mathbf{u}$ and $\mathbf{v}$ are codewords of $\mathcal{C}$, then $\mathbf{u} + \mathbf{v}$ is also a codeword of $\mathcal{C}$.

2) If $\mathbf{u}$ is a codeword of $\mathcal{C}$ and z is in $\mathcal{A}_X$, then $z\mathbf{u}$ is also a codeword of $\mathcal{C}$.

In other words, $\mathcal{C}$ must be a subspace of $\mathcal{A}_X^N$. Note that the all-zero codeword must be in $\mathcal{C}$, since $\mathbf{0} = 0\mathbf{u}$ for any $\mathbf{u}$.

Note: For binary codes, where $\mathcal{A}_X = Z_2$, condition (2) will always hold if condition (1) does, since $1\mathbf{u} = \mathbf{u}$ and $0\mathbf{u} = \mathbf{0} = \mathbf{u} + \mathbf{u}$.

Linear Codes From Basis Vectors

We can construct a linear code by choosing K linearly-independent *basis vectors* from Z_2^N.

We'll call the basis vectors $\mathbf{u}_1, \ldots, \mathbf{u}_K$. We define the set of codewords to be all those vectors that can be written in the form

$$a_1\mathbf{u}_1 + a_2\mathbf{u}_2 + \cdots + a_K\mathbf{u}_K$$

where $a_1, \ldots, a_K$ are elements of Z_2.

The codewords obtained with different $a_1, \ldots, a_K$ are all different. (Otherwise $\mathbf{u}_1, \ldots, \mathbf{u}_K$ wouldn't be linearly-independent.)

There are therefore 2^K codewords. We can encode a block consisting of K symbols, $a_1, \ldots, a_k$, from Z_2 as a codeword of length N using the formula above.

This is called an $[N, K]$ code. (MacKay's book uses (N, K), but that has another meaning in other books.)

Linear Codes From Linear Equations

Another way to define a linear code for Z_2^N is to provide a set of simultaneous equations that must be satisfied for $\mathbf{v}$ to be a codeword.

These equations have the form $\mathbf{c} \cdot \mathbf{v} = 0$, ie

$$c_1 v_1 + c_2 v_2 + \cdots + c_N v_N \;=\; 0$$

The set of solutions is a linear code because

1) $\mathbf{c} \cdot \mathbf{u} = 0$ and $\mathbf{c} \cdot \mathbf{v} = 0$ implies $\mathbf{c} \cdot (\mathbf{u} + \mathbf{v}) = 0$.
2) $\mathbf{c} \cdot \mathbf{v} = 0$ implies $\mathbf{c} \cdot (z\mathbf{v}) = 0$.

If we have $N - K$ such equations, and they are independent, the code will have 2^K codewords.

The Repetition Codes Over Z_2

A repetition code for Z_2^N has only two codewords — one has all 0s, the other all 1s.

This is a linear $[N, 1]$ code, with $(1, \ldots, 1)$ as the basis vector.

The code is also defined by the following $N - 1$ equations satisfied by a codeword $\mathbf{v}$:

$$v_1 + v_2 = 0, \quad v_2 + v_3 = 0, \ \cdots, \quad v_{N-1} + v_N = 0$$

The Single Parity-Check Codes

An $[N, N-1]$ code over Z_2 can be defined by the following single equation satisfied by a codeword $\mathbf{v}$:

$$v_1 + v_2 + \cdots + v_N \;=\; 0$$

In other words, the *parity* of all the bits in a codeword must be even.

This code can also be defined using $N-1$ basis vectors. One choice of basis vectors when $N = 5$ is as follows:

$$(1, 0, 0, 0, 1)$$

$$(0, 1, 0, 0, 1)$$

$$(0, 0, 1, 0, 1)$$

$$(0, 0, 0, 1, 1)$$

A $[5, 2]$ Binary Code

Recall the following code from earlier in the lecture:

$$\{\ 00000,\ \ 00111,\ \ 11001,\ \ 11110\ \}$$

Is this a linear code? We need to check that all sums of codewords are also codewords:

$$00111 + 11001 = 11110$$

$$00111 + 11110 = 11001$$

$$11001 + 11110 = 00111$$

We can generate this code using 00111 and 11001 as basis vectors. We then get the four codewords as follows:

$$0 \cdot 00111 + 0 \cdot 11001 = 00000$$

$$0 \cdot 00111 + 1 \cdot 11001 = 11001$$

$$1 \cdot 00111 + 0 \cdot 11001 = 00111$$

$$1 \cdot 00111 + 1 \cdot 11001 = 11110$$

The $[7, 4]$ Binary Hamming code

The $[7, 4]$ Hamming code is defined over Z_2 by the following equations that are satisfied by a codeword $\mathbf{u}$:

$$\begin{aligned} u_1 + u_2 + u_3 + u_5 &= 0 \\ u_2 + u_3 + u_4 + u_6 &= 0 \\ u_1 + u_3 + u_4 + u_7 &= 0 \end{aligned}$$

Since these equations are independent, there should be 16 codewords.

We can also define the code in terms of the following four basis vectors:

$$1000101, \quad 0100110, \quad 0010111, \quad 0001011$$

There are other sets equations and other sets of basis vectors that define the same code.

We will see later that this code is capable of correcting any single error.

Generator Matrices

We can arrange a set of basis vectors for a linear code in a *generator matrix*, each row of which is a basis vector.

A generator matrix for an $[N, K]$ code will have K rows and N columns.

Here's a generator matrix for the $[5, 2]$ code looked at earlier:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Note: Almost all codes have more than one generator matrix.

Encoding Blocks Using a Generator Matrix

We can use a generator matrix for an $[N, K]$ code to encode a block of K message bits as a block of N bits to send through the channel.

We regard the K message bits as a row vector, $\mathbf{s}$, and multiply by the generator matrix, G, to produce the channel input, $\mathbf{t}$:

$$\mathbf{t} \;=\; \mathbf{s}G$$

Every $\mathbf{t}$ that is a codeword will be produced by some $\mathbf{s}$. If the rows of G are linearly independent, each distinct $\mathbf{s}$ will produce a different $\mathbf{t}$.

Example: Encoding the message block $(1, 1)$ using the generator matrix for the $[5, 2]$ code given earlier:

$$\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Parity-Check Matrices

Suppose we have specified an $[N, K]$ code by a set of $M = N - K$ equations that any codeword, $\mathbf{v}$, must satisfy:

$$\begin{aligned} c_{1,1}\, v_1 + c_{1,2}\, v_2 + \cdots + c_{1,N}\, v_N &= 0 \\ c_{2,1}\, v_1 + c_{2,2}\, v_2 + \cdots + c_{2,N}\, v_N &= 0 \\ &\vdots \\ c_{M,1}\, v_1 + c_{M,2}\, v_2 + \cdots + c_{M,N}\, v_N &= 0 \end{aligned}$$

We can arrange the coefficients in these equations in a *parity-check matrix*, as follows:

$$\begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,N} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,N} \\ & & \vdots & \\ c_{M,1} & c_{M,2} & \cdots & c_{M,N} \end{bmatrix}$$

If $\mathcal{C}$ has parity-check matrix H, then $\mathbf{v}$ is in $\mathcal{C}$ if and only if $\mathbf{v}H^T = \mathbf{0}$.

Note: Almost all codes have more than one parity-check matrix.

Example: The $[5, 2]$ Code

Here is one parity-check matrix for the $[5, 2]$ code used earlier:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

We see that 11001 is a codeword as follows:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

But 10011 isn't a codeword, since

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}$$

Example: Repetition Codes & Single Parity-Check Codes

An $[N, 1]$ repetition code has the following generator matrix (for $N = 4$):

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

Here is a parity-check matrix for this code:

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

One generator matrix for the $[N, N-1]$ single parity-check code is this:

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Here is the parity-check matrix for this code:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

Manipulating the Parity-Check Matrix

There are usually many parity-check matrices for a given code. We can get one such matrix from another using the following "elementary row operations":

- Swapping two rows.
- Multipling a row by a non-zero constant (not useful for Z_2).
- Adding a row to a different row.

These operations don't alter the solutions to the equations the parity-check matrix represents.

Example: This parity-check matrix for the example $[5, 2]$ code:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

can be transformed into this alternative:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Manipulating the Generator Matrix

We can apply the same elementary row operations to a generator matrix for a code, in order to produce another generator matrix, since these operations just convert one set of basis vectors to another.

Example: Here is a generator matrix for the $[5, 2]$ code we have been looking at:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Here is another generator matrix, found by adding the first row to the second:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Note: These manipulations leave the set of codewords unchanged, but they *don't* leave the way we encode messages by computing $\mathbf{t} = \mathbf{s}G$ unchanged!

Equivalent Codes

Two codes are said to be *equivalent* if the codewords of one are just the codewords of the other with the order of symbols permuted.

Permuting the order of the columns of a generator matrix will produce a generator matrix for an equivalent code, and similarly for a parity-check matrix.

Example: Here is a generator matrix for the $[5, 2]$ code we have been looking at:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

We can get an equivalent code using the following generator matrix obtained by moving the last column to the middle:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Generator and Parity-Check Matrices In Systematic Form

Using elementary row operations and column permutations, we can convert any generator matrix to a generator matrix for an equivalent code that is in *systematic form*, in which the left end of the matrix is the identity matrix.

Similarly, we can convert to the systematic form for a parity-check matrix, which has an identity matrix in the right end.

For the $[5, 2]$ code, only permutations are needed. The generator matrix can be permuted by swapping columns 1 and 3:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

When we use a systematic generator matrix to encode a block $\mathbf{s}$ as $\mathbf{t} = \mathbf{s}G$, the first K bits will be the same as those in $\mathbf{s}$. The remaining $N - K$ bits can be seen as "check bits".

Relationship of Generator and Parity-Check Matrices

If G and H are generator and parity-check matrices for $\mathcal{C}$, then for every $\mathbf{s}$, we must have $(\mathbf{s}G)H^T = \mathbf{0}$ — since we should only generate valid codewords. It follows that

$$GH^T = \mathbf{0}$$

Furthermore, any H with $N - K$ independent rows that satisfies this is a valid parity-check matrix for $\mathcal{C}$.

Suppose G is in systematic form, so

$$G \;=\; [\, I_K \mid P \,]$$

for some P. Then we can find a parity-check matrix for $\mathcal{C}$ in systematic form as follows:

$$H \;=\; [\, -P^T \mid I_{N-K} \,]$$

since $GH^T = -I_K P + P I_{N-K} = \mathbf{0}$.