Product Codes

A *product code* is formed from two other codes $\mathcal{C}_1$, of length N_1, and $\mathcal{C}_2$, of length N_2. The product code has length N_1N_2.

We can visualize the N_1N_2 symbols of the product code as a 2D array with N_1 columns and N_2 rows.

Definition of a product code: An array is a codeword of the product code if and only if

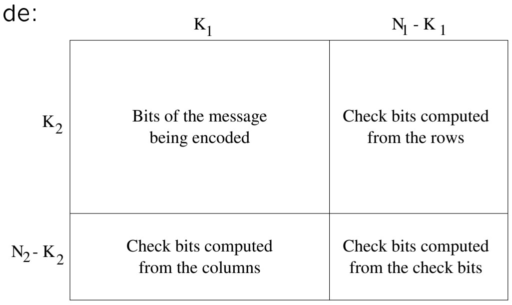
- all its rows are codewords of $\mathcal{C}_1$
- all its columns are codewords of $\mathcal{C}_2$

We will assume here that $\mathcal{C}_1$ and $\mathcal{C}_2$ are linear codes, in which case the product code is also linear. (Why?)

Dimensionality of Product Codes

Suppose $\mathcal{C}_1$ is an $[N_1, K_1]$ code and $\mathcal{C}_2$ is an $[N_2, K_2]$ code. Then their product will be an $[N_1N_2, K_1K_2]$ code.

Suppose $\mathcal{C}_1$ and $\mathcal{C}_2$ are in systematic form. Here's a picture a codeword of the product code:



The dimensionality of the product code is not more than K_1K_2, since the message bits in the upper-left determine the check bits. We'll see that the dimensionality equals K_1K_2 by showing how to find correct check bits for any message.

Encoding Product Codes

Here's a procedure for encoding messages with a product code:

1. Put K_1K_2 message bits into the upper-left K_2 by K_1 corner of the N_2 by N_1 array.
2. Compute the check bits for each of the first K_2 rows, according to $\mathcal{C}_1$.
3. Compute the check bits for each of the N_1 columns, according to $\mathcal{C}_2$.

After this, all the columns will be codewords of $\mathcal{C}_2$, since they were given the right check bits in step (3). The first K_2 rows will be codewords of $\mathcal{C}_1$, since they were given the right check bits in step (2). But are the *last* $N_2 - K_2$ rows codewords of $\mathcal{C}_1$?

Yes! Check bits are linear combinations of message bits. So the last $N_2 - K_2$ rows are linear combinations of earlier rows. Since these rows are in $\mathcal{C}_1$, their combinations are too.

Minimum Distance of Product Codes

If $\mathcal{C}_1$ has minimum distance d_1 and $\mathcal{C}_2$ has minimum distance d_2, then the minimum distance of their product is d_1d_2.

Proof:

Let $\mathbf{u}_1$ be a codeword of $\mathcal{C}_1$ of weight d_1 and $\mathbf{u}_2$ be a codeword of $\mathcal{C}_2$ of weight d_2. Build a codeword of the product code by putting $\mathbf{u}_1$ in row i of the array if $\mathbf{u}_2$ has a 1 in position i. Put zeros elsewhere. This codeword has weight d_1d_2.

Furthermore, any non-zero codeword must have at least this weight. It must have at least d_2 rows that aren't all zero, and each such row must have at least d_1 ones in it.

Decoding Product Codes

Products of even small codes (eg, [7, 4] Hamming codes) have lots of check bits, so decoding by building a syndrome table may be infeasible.

But if $\mathcal{C}_1$ and $\mathcal{C}_2$ can easily be decoded, we can decode the product code by first decoding the rows (replacing them with the decoding), then decoding the columns.

This will usually **not** be a nearest-neighbor decoder (and hence will be sub-optimal, assuming a BSC and equally-likely messages).

One advantage of product codes: They can correct some *burst errors* — errors that come together, rather than independently.

How Good Are Simple Codes?

Shannon's noisy coding theorem says we can get the probability of error in decoding a block, p_B, arbitrarily close to zero when transmitting at any rate, R, below the capacity, C — if we use good codes of large enough length, N.

For repetition codes, as N increases, $p_B \to 0$, but $R \to 0$ as well.

For Hamming codes, as $N = 2^c - 1$ increases, $R \to 1$, but $p_B \to 1$ as well, since there's bound to be more than one error in a really big block.

How Good are Products of Codes?

Let $\mathcal{C}$ be an $[N, K]$ code of minimum distance d (guaranteed to correct $t = \lfloor (d-1)/2 \rfloor$ errors).

How good is the code obtained by taking the product of $\mathcal{C}$ with itself p times?

Length: $N_p = N^p$
Rate: $R_p = K^p/N^p = (K/N)^p \to 0$
Distance: $d_p = d^p$
Relative distance: $\rho_p = d_p/N_p = (d/N)^p \to 0$

The code can correct up to about $d_p/2$ errors, corresponding to a proportion of errors of $\rho_p/2$.

For a BSC with error probability f, we expect that for large N, the proportion of erroneous bits in a block will be very close to f. (This is the "Law of Large Numbers".)

So for large N, these product codes are *unlikely* to correct all errors, and also have a low rate!

Good Codes Aren't Easy to Find

In the 56 years since Shannon's noisy coding theorem, many schemes for creating codes have been found, but most of them *don't* allow one to reach the performance promised by theorem.

They can still be useful. For example, error correction in computer memory necessarily works on fairly small blocks (eg, 64 bits). Performance on bigger blocks is irrelevant.

But in other applications — computer networks, communication with spacecraft, digital television — we could use quite big blocks if it would help with error correction.

How can we do this in practice?

Getting to Capacity for the BEC

We *can* get near-error-free transmission for the binary erasure channel, at any rate below capacity, using a practical method.

We use a linear $[N, K]$ code, defined by a set of $M = N-K$ parity-check equations:

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

For the BEC, any bit received as 0 or 1 is guaranteed to be correct. To decode, we fill in these known values in the equations above, and then try to solve for the unknown values, where the bit was received as an erasure.

When Will This BEC Decoding Method Succeed?

If the probability of an erasure is f, and N is large, there will very likely be around Nf erasures in the received data (the Law of Large Numbers again).

So the decoder will be solving M equations in U unknowns, where U is very likely to be near Nf

These equations will be *consistent*, since the correct decoding is certainly a solution.

The correct decoding will be the *unique* solution — which the decoder is guaranteed to find — as long as U out of the M equations are independent.

Picking the Code at Random

Suppose we pick a code — specified by the parity-check coefficients, c_{ij} — *at random*.

How likely is it that the equations that we need to solve to decode a transmission that has U erasures will have a unique solution?

Imagine randomly picking the parity-check equations *after* we receive the transmission with U erasures. How many equations would we expect to have to pick to get U independent equations?

Once we have i independent equations, the probability that the next equation picked will be dependent on these will be

$$\frac{2^i}{2^U} = \frac{1}{2^{U-i}}$$

since there are 2^i ways of combining the previous equations, and 2^U possible equations.

Picking the Code at Random (Continued)

The expected number of dependent equations picked before we get U independent ones is

$$\sum_{i=0}^{U-1} \frac{1}{2^{U-i}} \left(1 - \frac{1}{2^{U-i}}\right)^{-1} = \sum_{i=0}^{U-1} \frac{1}{2^{U-i}-1}$$

Reordering the terms, we can see that this is small:

$$1 + 1/3 + 1/7 + \cdots < 1 + 1/2 + 1/4 + \cdots < 2$$

Hence, we likely need M to be only slightly larger than U, which is likely to be no more than slightly larger than Nf.

So with a random code, we will be likely to correct all erasures when N is large as long as $f < M/N = (N-K)/N = 1 - R$. In other words, as long as $R < 1-f$. As we saw in tutorial, the capacity of the BEC is equal to $1-f$, so we've achieved the promise of Shannon's theorem.

What about the BSC?

A similar argument using randomly-chosen codes is used in the proof of Shannon's noisy coding theorem for the BSC. We'll look at a sketch of this proof.

But unlike the random codes for the BEC, the random codes used in this proof are completely impractical.

We'll then look briefly at random codes of a different kind, whose parity-check matrices are mostly zeros. These "Low Density Parity Check Codes" can be used in practice, and allow near-error-free transmission at close to capacity.