How Good Are Simple Codes?

Shannon's noisy coding theorem says we can get the probability of error, P_E, 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, $P_E \to 0$, but $R \to 0$ as well.

For Hamming codes, as $n = 2^c - 1$ increases, $R \to 1$, but $P_E \to 1$ 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 with 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 Q, we expect that for large n, the proportion of erroneous bits in a block will be very close to Q. (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 54 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 $c = n-k$ parity-check equations:

$$\begin{aligned} b_{1,1}\,v_1 + b_{1,2}\,v_2 + \cdots + b_{1,n}\,v_n &= 0 \\ b_{2,1}\,v_1 + b_{2,2}\,v_2 + \cdots + b_{2,n}\,v_n &= 0 \\ &\vdots \\ b_{c,1}\,v_1 + b_{c,2}\,v_2 + \cdots + b_{c,n}\,v_n &= 0 \end{aligned}$$

For the BEC, any bit, v_i, 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 Q, and n is large, there will very likely be around nQ erasures in the received data (the Law of Large Numbers again).

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

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 c equations are independent.

Picking the Code at Random

Suppose we pick a code — specified by the parity-check coefficients, b_{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 c to be only slightly larger than U, which is likely to be no more than slightly larger than nQ.

So with a random code, we will be likely to correct all erasures when n is large as long as $Q < c/n = (n-k)/n = 1-R$. In other words, as long as $R < 1-Q$. As we saw in tutorial, the capacity of the BEC is equal to $1-Q$, 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.