Consider a BSC with error probability \( f < 1/2 \). This channel has capacity \( C = 1 - H_2(f) \).

For any desired closeness to capacity, \( \eta > 0 \), and for any desired limit on error probability, \( \epsilon > 0 \), there is a code of some length \( N \) whose rate, \( R \), is at least \( C - \eta \), and for which the probability that nearest neighbor decoding will decode a codeword incorrectly is less than \( \epsilon \).

Last class we started to give a proof of this, which more-or-less follows the proof for general channels in Chapter 10 of MacKay's book.

The idea is based on showing that a randomly chosen code performs quite well and hence that there must be specific codes which also perform quite well.

---

**Strategy for Proving the Theorem**

- Rather than showing how to construct a specific code for given values of \( f, \eta, \) and \( \epsilon \), we will consider choosing a code of a suitable length, \( N \), and rate \( \log_2(M)/N \), by picking \( M \) codewords at random from \( \mathbb{Z}_2^N \).

- We consider the following scenario:
  1. We randomly pick a code, \( C \), which we give to both the sender and the receiver.
  2. The sender randomly picks a codeword \( x \in C \), and transmits it through the channel.
  3. The channel randomly generates an error pattern, \( n \), and delivers \( y = x + n \) to the receiver.
  4. The receiver decodes \( y \) to a codeword, \( x^* \), that is nearest to \( y \) in Hamming distance.

- If the probability that this process leads to \( x^* \neq x \) is \( < \epsilon \), then there must be some specific code with error probability \( < \epsilon \).

---

**Rearranging the Order of Choices**

- It will be convenient to rearrange the order in which random choices are made, as follows:
  1. We randomly pick one codeword, \( x \), which is the one the sender transmits.
  2. The channel randomly generates an error pattern, \( n \), that is added to \( x \) to give the received data, \( y \). Let the number of transmission errors (ie, ones in \( n \)) be \( w \).
  3. We now randomly pick the other \( M-1 \) codewords. If the Hamming distance from \( y \) of all these codewords is greater than \( w \), nearest-neighbor decoding will make the correct choice.

- The probability of the decoder making the wrong choice here is the same as before.
• The probability that the codeword nearest to $y$ is the correct decoding will be at least as great as the probability that the following sub-optimal decoder decodes correctly:
  
  If there is exactly one codeword $x^*$ for which $n = y - x^*$ has a typical number of ones, then decode to $x^*$, otherwise declare that decoding has failed.

• This sub-optimal decoder can fail in two ways:
  
  - The correct decoding, $x$, may correspond to an error pattern, $n = y - x$, that is not typical.
  
  - Some other codeword, $x'$, may exist for which the error pattern $n' = y - x'$ is typical.

Bounding the Probability of Failure (I)

• The total probability of decoding failure is less than the sum of the probabilities of failing in these two ways.
  
  We will try to limit each of these to $\epsilon/2$.

• We can choose $N$ to be big enough that
  
  $$P(f - \beta < w/N < f + \beta) \geq 1 - \epsilon/2$$
  
  This ensures that the actual error pattern will be non-typical with probability less than $\epsilon/2$.

• We now need to limit the probability that some other codeword also corresponds to a typical error pattern.

Bounding the Probability of Failure (II)

• The number of typical error patterns is
  
  $$J < 2^{N(H_2(f) + \beta \log_2((1-f)/f))}$$

• For a random codeword, $x$, other than the one actually transmitted, the corresponding error pattern given $y$ will contain 0s and 1s that are independent and equally likely.

• The probability that one such codeword will produce a typical error pattern is therefore
  
  $$J/2^N < 2^{-N(1-H_2(f) - \beta \log_2((1-f)/f))}$$

• The probability that any of the other $M-1$ codewords will correspond to a typical error pattern is bounded by $M$ times this. We need this to be less than $\epsilon/2$, ie
  
  $$M 2^{-N(1-H_2(f) - \beta \log_2((1-f)/f))} < \epsilon/2$$

Finishing the Proof

• Finally, we need to pick $\beta$, $M$, and $N$ so that the two types of error have probabilities less than $\epsilon/2$, and the rate, $R$ is at least $C - \eta$.

• We will let $M = 2^{[(C-\eta)N]}$, and make sure $N$ is large enough that $R = [(C-\eta)N]/N < C$.

• With this value of $M$, we need
  
  $$2^{[(C-\eta)N]} 2^{-N(1-H_2(f) - \beta \log_2((1-f)/f))} < \epsilon/2$$
  
  $$\Rightarrow 2^{-N(1-H_2(f) - [(C-\eta)N]/N - \beta \log_2((1-f)/f))} < \epsilon/2$$

• The channel capacity is $C = 1 - H_2(f)$, so that
  
  $$1 - H_2(f) - [(C-\eta)N]/N = C - R$$

  is positive.

• For a sufficiently small value of $\beta$,
  
  $$1 - H_2(f) - [(C-\eta)N]/N - \beta \log_2((1-f)/f)$$

  will also be positive. With this $\beta$ and a large enough $N$, the probabilities of both types of error will be less than $\epsilon/2$, so the total error probability will be less than $\epsilon$. 

• Recall that for a code to be guaranteed to correct up to \( t \) errors, its minimum distance must be at least \( 2t + 1 \).
• What’s the minimum distance for the random codes used to prove the noisy coding theorem?
• A random \( N \)-bit code is very likely to have minimum distance \( d \approx N/2 \) — if we pick two codewords randomly, about half their bits will differ. So these codes are likely not guaranteed to correct patterns of \( N/4 \) or more errors.
• A BSC with error probability \( f \) will produce about \( Nf \) errors. So for \( f > 1/4 \), we expect to get more errors than the code is guaranteed to correct. Yet we know these codes are good!
• **Conclusion:** A code may be able to correct almost all patterns of \( t \) errors even if it can’t correct all such patterns.

### Dimensionality of Product Codes

- Suppose \( C_1 \) is an \([N_1, K_1]\) code and \( C_2 \) is an \([N_2, K_2]\) code. Then their product will be an \([N_1N_2, K_1K_2]\) code.
- Suppose \( C_1 \) and \( C_2 \) are in systematic form. Here’s a picture of a codeword of the product code:

  ![Product Code Picture]

  - 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 the first \( K_2 \) rows, according to \( C_1 \).
  3. Compute the check bits for the \( N_1 \) columns, according to \( C_2 \).
- After this, all the columns will be codewords of \( C_2 \), since they were given the right check bits in step (3). The first \( K_2 \) rows will be codewords of \( C_1 \), since they were given the right check bits in step (2). But are the last \( N_2 - K_2 \) rows codewords of \( C_1 \)?
- 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 \( C_1 \), their combinations are too.
Minimum Distance of Product Codes

- If $C_1$ has minimum distance $d_1$ and $C_2$ has minimum distance $d_2$, then the minimum distance of their product is $d_1 d_2$.

**Proof:**
Let $u_1$ be a codeword of $C_1$ of weight $d_1$ and $u_2$ be a codeword of $C_2$ of weight $d_2$. Build a codeword of the product code by putting $u_1$ in row $i$ of the array if $u_2$ has a 1 in position $i$. Put zeros elsewhere. This codeword has weight $d_1 d_2$.

The new codeword is the outer product of the vectors $u_1$ and $u_2$.

- 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 $C_1$ and $C_2$ can easily be decoded, we can decode the product code by first decoding the rows (replacing them with the decoding), then decoding the columns. (Or the other way around.)

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

Why use Products of Codes?

- The analysis above shows that for large $N$, these product codes are both unlikely to correct all errors, and also that they have a low rate (approaching zero)!

- So why would we ever use them?

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