Today

- Optimality of Huffman codes
- "Universal Coding"
- Universal Coding for i.i.d. sources
- Universal Coding (theoretically) for Markovian Channels

- Lempel-Ziv Coding

Review of last lecture

Kraft's Inequality

DMC: $X = \{ 1, \ldots, n \}$

\[ p_1 \ldots p_n \]

$C: \{ 1, \ldots, n \} \rightarrow D^*$ with $C(i) \in D_i$
thus \[ \sum_{i=1}^{D} d_i^2 \leq 1 \] [for prefix free/uniquly decodable]

Furthermore if \( \exists h_1, \ldots, h_n \) s.t.
\[ \sum_{i=1}^{D} d_i^2 \leq 1 \]

then \( \exists C \) prefix free s.t.
\[ C(i) \subset d^2_i \]

---

1. Entropy lb.

\( \forall \) prefix/uniquly dec. code
\[ E[L] \geq H(x) \]

\( \forall \) non-singular code
\[ E[L] > H(x) - O(\sqrt{H(x)}) \]

2. Shannon Code: \( e_i = \lceil \log p_i \rceil \) QED
\[ \text{yields} \quad E[L] \leq H(x) + 1 \]

(3) Huffman Code: (Binary Version)

Code (\( p_1, \ldots, p_n \)):

- Sort so that \( p_1 \geq p_2, \ldots, p_n \);
- Let \( C' = \text{Code}(p_1, \ldots, p_{n-2}, p_{n-1} + p_n) \);
- \[ C[i] = \begin{cases} C'[i] & i \leq n-2 \\ C'[n-1].0 & i = n-1 \\ C'[n-1].1 & i = n \end{cases} \]

Return \( C \);

Huffman Code is optimal

\[ \forall \text{prefix code} \ C : \{1, \ldots, n\} \rightarrow \{0, 1\}^* \]

\[ \sum_{i=1}^{n} p_i |C(i)| \geq \sum_{i=1}^{n} p_i |C_{\text{Huffman}}(i)| . \]
Proof (by picture):

1. Prefix code looks like leaves of binary tree.

2. For optimal if \( p_i < p_j \) then \( l_i \geq l_j \).

3. Tree is "full" (every node has two/zero children).

4. W.l.o.g. \( A, B \) have lowest probabilities.
$3 \implies$ length of $C$ is optimal

$\implies$ length of $C'$ is optimal.

---

Pro: a Coroll of Shannon/Huffman Coding

New:

- Optimal given what we know
- Should apply to $X^{(k)} = (X_1, \ldots, X_k)$
  \[ \uparrow \quad \text{i.i.d} \sim X. \]

Then the "+1" is amortized over $k$ blocks.

- Need to know the $P_i$'s!
- Can we avoid this? (seems essential)?
- If we code $q_1, \ldots, q_n$ according to $q_1, \ldots, q_n$, then we have $H(X) + D(p \| q)$. 
Universal Encoders

\[ C : \mathbb{R}^* \to \{0, 1\}^* \]

is universal if

\[ \exists \text{ distribution } p \text{ on } \mathbb{R}^n, \epsilon > 0 \]

\[ \exists \text{ sequence } x_1, \ldots, x_n \text{ i.i.d. } \sim \text{ according to } p \]

\[ |C(x_1, \ldots, x_n)| \sim \epsilon \cdot \mathcal{H}(x) \cdot (1 + \epsilon) \]

Can such a \( C \) exist?

\[ \text{E.g.:} \]

- \( C \) first counts \( \#_i = |\{ j \mid x_j = i \}| \)

- Outputs \( (\#_1, \ldots, \#_n) \);

  Outputs which pattern it sees.
Won't prove conversely... but obvious it works.

Can: is Discrete Memoryless for real?
   - Huge "Dictionaries" - Can we do better? \[ \mathcal{L}_1 \]

[Fax anecdote here]

Lempel-Ziv \[ [\text{LZ76, ZL77, ZL78}] \]
- Nice, natural, efficient algorithm
- Universal coding for Markovian sources.
Markovian Source

\[ p_1 = \frac{1}{3} \]
\[ p_2 = \frac{1}{6} \]
\[ p_3 = \frac{1}{2} \]

"Models natural language reasonably [Shannon]."

Compress efficiently?

Use AEP ....
A Shabby Algorithm

How many sequences do we expect to see?

\[ H(x) \cdot L \]

2

- \[ H(x) = \text{Entropy rate of Hidden Markov Chain}. \]

- Other sequences very unlikely ... happen with \[ \leq \epsilon = \exp(-\text{rel}) \].

Encoding strategy. [SHABBY]

- Count # diff. patterns that appear more than \( k \) times. [\( k \) = parameter]
- Build dictionary of such patterns & send dictionary to receiver.

- For each block do:
  1. Send "typical" or "not typical";
  2. If typical, send index in dictionary;
  3. Else send entire sequence.

\[
\text{cost} = E \cdot (L \cdot \lceil \log n \rceil + 1) \cdot t \\
\text{cost} \approx H(x) \cdot L \cdot t \\
\text{cost} = 2^{H(x)} \cdot L \cdot \lceil \log n \rceil \quad \text{but no } t
\]

- vanishes as \( L \to \infty \)

- vanishes as \( b \to \infty \)
- But how to choose L, k?

- Try all possibilities; we are one which minimizes absolute length!

**Conclusion:** SHABBY is a universal encoder, but it is SHABBY

How can we make this more elegant?

What is inelegant?

1. "Explicit choices of L, k".
2. "Explicit a huge dictionary".
\[ Z \rightarrow \varepsilon \]

\[ \sigma_1 \sigma_2 \ldots \sigma_i \ldots \]

\[ \text{Parse} \]

\[ B_1, B_2, \ldots, B_i, \ldots \]

\[ B_i = \text{sequence over } \Sigma \]

has property that:

1. \( B_i \neq B_j \) for \( i < j \)
2. \( B_i = B_j \cdot \sigma \) for \( i < j \)
   \[ \sigma \in \Sigma \]

Encoding: \( B_1, \ldots, B_i, \ldots \)
\[ z_i \rightarrow (i, s) \]