LECTURE 7

Last time:

- Kraft's Inequality
- Shannon Code

Lecture outline

- A little more about Shannon code.
- Huffman code
- Elias code

Reading: Scts. 5.6-5.11

Quick Review

- Kraft's inequality, $\sum_{x \in \mathcal{X}} D^{-l(x)} \leq 1$.
- Optimal code, $\frac{1}{n} E[l(X)] \geq H(X)$.
- Shannon code $l(x) = \lceil - \log P_X(x) \rceil$.
- Achieve asymptotic optimality, as $n \to \infty$.

Questions

- Can we construct optimal code for finite n.
- How does information theory help us to construct code for finite n.

Competitive Optimality of Shannon Code

Question What do we know about the codeword length for a particular outcome?

Let $l(x) = \lceil -\log P_X(x) \rceil$ be the codeword length assignment for Shannon code, and $l'(x)$ be the codeword length of any other uniquely decodable code,

Claim

$$P(l(X) \geq l'(X) + c) \leq \frac{1}{2^{c-1}}$$

Proof

$$\begin{aligned}
& P(l(X) \geq l'(X) + c) \\
= \;& P\left(l'(X) + c \leq \lceil -\log P_X(x) \rceil\right) \\
\leq \;& P(l'(X) + c - 1 \leq -\log P_X(x)) \\
= \;& P(P_X(x) \leq 2^{-l'(x)-(c-1)}) \\
= \;& \sum_{x: P_X(x) \leq 2^{-l'(x)-(c-1)}} P_X(x) \\
\leq \;& \sum_{x: P_X(x) \leq 2^{-l'(x)+c-1}} 2^{-l'(x)-(c-1)} \\
\leq \;& \sum_x 2^{-l'(x)-(c-1)} \leq 2^{-(c-1)}
\end{aligned}$$

Theorem Idealize the Shannon code to have $l(x) = -\log P_X(x)$, and let $l'(x)$ be the codeword length of any other uniquely decodable code.

$$P(l(X) < l'(X)) \geq P(l(X) > l'(X))$$

Why this is not trivial?

Lemma $\text{sgn}(t) \leq 2^t - 1$ for any integer t.

Proof

$$\begin{aligned} & P(l(X) > l'(X)) - P(l(X) < l'(X)) \\ = & \sum_x P_X(x)\text{sgn}(l(x) - l'(x)) \\ \leq & \sum_x P_X(x)(2^{l(x)-l'(x)} - 1) \\ = & \sum_x 2^{-l(x)}(2^{l(x)-l'(x)} - 1) \\ = & \sum_x 2^{-l'(x)} - \sum_x 2^{-l(x)} \\ \leq & 1 - 1 = 0 \end{aligned}$$

Equality holds only if $l(x) = l'(x)$ for all x.

Constructing the Optimal Prefix Code

X has probability masses $p_1 \geq p_2 \ldots \geq p_m$, construct binary code to minimize $\sum_i p_i l_i$.

What should the optimal code look like?

- If $p_i > p_j$, then $l_i \leq l_j$.
- The two longest codewords have the same length.
- The two longest codewords differ only in the last bit.

Construction:
Take the two least likely symbols, merge them to get a size $m-1$ problem.

D-ary Huffman Code

Definition Complete tree: every leaf is assigned to a codeword. Every intermediate node has D branches stemming from it.

- A complete tree means Kraft's inequality holds with equality.
- Size of a D-ary complete tree: $1 + n(D - 1)$ for integer n.
- For an arbitrary $\mathcal{X}$, add 0 probability symbols to make it fit in a complete tree.

What can we say about Huffman Code

- Optimal prefix code for any source.
- Always equally good or better than Shannon code.
- $\frac{1}{n}E[l(X_1^n)] \to H(X)$ as $n \to \infty$.

Binary Huffman Coding and "Slice" Questions

- The realization of a random variable $X \in \mathcal{X}$ can be determined by asking a sequence of questions " Is X in A_i " for some subset $A \in \mathcal{X}$. How to choose a sequence of A's to minimize the number of questions?
- "Slice" questions represent X by a sequence of binary random variables.

Notation

- For an internal node k on the coding tree, *reach probability* p_k is the sum of the probability of all the codewords descending from k.
- Let m, n be the two children of k, then the *branching distribution* is $\left\{\frac{p_m}{p_k}, \frac{p_n}{p_k}\right\}$.
- The *conditional entropy* of node k is $h_k = H\left(\frac{p_m}{p_k}, \frac{p_n}{p_k}\right)$.

The redundancy of Huffman Code

Claim

$$H(X) = \sum_k p_k h_k$$

summing over all the internal nodes.

Proof by induction.

Reminder: This is how we define the entropy.

Claim

$$E[l(X)] = \sum_k p_k$$

At each node k, no matter what p_k is, we have to add 1 more bit to the codeword.

Definition The *Local Redundancy* at node k is

$$r_k = p_k(1 - h_k)$$

Redundancy of Huffman Code

Theorem

$$E[l(X)] - H(X) = \sum_k r_k$$

The entropy bound $E[l(X)] \geq H(X)$ is achieved with equality iff for any node, the local redundancy is 0.

Consider a codeword as revealing a random variable X one bit at a time. Achieving the entropy bound means each bit reveals precisely 1 bit of information, or equivalently, $h_k = 1$.

Elias Code

Idea: use the value of the cdf. $F(X)$ to indicate X.

- We can only use finite number of bits $l(x)$ to represent the real number $F(x)$.

$$
\begin{aligned}
F(x) &= \sum_{a \le x} P_X(a) \\
\overline{F}(X) &= \sum_{a < x} P_X(a) + \frac{1}{2} P_X(x)
\end{aligned}
$$

- Round off $\overline{F}(x)$ to $l(x)$ bits to get $\lfloor \overline{F}(x) \rfloor_{l(x)}$.
- Want $\lfloor \overline{F}(x) \rfloor_{l(x)} \in (F(x-1), \overline{F}(x))$.
- We need $l(x) = \lceil -\log P_X(x) \rceil + 1$:

$$
\begin{aligned}
\overline{F}(x) - \lfloor \overline{F}(x) \rfloor_{l(x)} &< \frac{1}{2^{l(x)}} \\
&\le \frac{P_X(x)}{2} \\
&= \overline{F}(x) - F(x-1)
\end{aligned}
$$

- Prefix free: the intervals

$$\left(\lfloor \overline{F}(x) \rfloor_{l(x)}, \lfloor \overline{F}(x) \rfloor_{l(x)} + 2^{-l(x)}\right)$$

  do not overlap for different x.

- Sufficient to have

$$\lfloor \overline{F}(x) \rfloor_{l(x)} + 2^{-l(x)} \leq F(x)$$

  which is true since $2^{-l(x)} \leq \frac{P_X(x)}{2}$.

- Average codeword length

$$\begin{aligned} E[l(X)] &= \sum_x P_X(x) \left(\lceil - \log P_X(x) \rceil + 1\right) \\ &\leq H(X) + 2 \end{aligned}$$

As coding over long sequence of source symbols, the optimal performance achieved.

- Advantage: can decode sequentially for i.i.d. source.