E2 201 Information Theory
Lecture 18 - Data Compression/Arithmetic Coding

Lecturer: Prof. Himanshu Tyagi  Scribe: Anamay Chaturvedi

September 16, 2015

1 Review

Last time we saw the following chain of inequalities -

$$L_0(X) \leq H(X) \leq L_0^\text{unique} = L_0^\text{prefix} \leq H(X) + 1$$

Here the first inequality follows from the structure of the optimal code, the second follows from Kraft’s inequality for UDCs, and the last one by considering a Shannon code with \(l(x) = \lceil \log(p(x)) \rceil\).

2 Huffman Code (Continued)

Last time we saw that construction of a Huffman code is by constructing a binary tree from the bottom up, assigning codeword symbols inductively.

- Step 1 Given \(p_1 \geq p_2 \geq \cdots \geq p_m\), construct a new node \([x_{m-1}, x_m]\) and let \(x_{m-1}\) and \(x_m\) be the left and right children of this node respectively.

- Step 2 - Replace \(x_m\) with \(x_{m-1, m} = \{x_{m-1}, x_m\}\). Let \(\tilde{p}_i = p_i\) for \(i \leq m - 2\), and \(\tilde{p}_{m-1} = p_{m-1} + p_m\). Let \(\tilde{p}'\) be the sorted version of \(\tilde{p}\).

Apply step 1 to \(\tilde{p}'\).

**Drawback** - Without running the algorithm, we cannot say what the length of the code will be, given just the probabilities.

**Theorem 2.1.** (see Cover-Thomas) The Huffman code has the least average length among all prefix-free codes (and therefore, among all uniquely decodable codes).

3 Arithmetic coding

Now we study the Shannon-Fano-Elias code. First, fix some arbitrary ordering on \(X\), say by an indexing set \(I\) (i.e. \(x_i < x_j\) if and only if \(i < j\)). Then, given a pmf \(P\) on \(X\), let
\[ F(x_i) := \sum_{j < i} P(x_j) \]

**Observation:** \( F(x) \) uniquely determines \( x \).

**Idea:** Represent \( F(x) \) by \( l(x) = \lceil \log p(x) \rceil + 1 \) most significant bits.

Then, the average length of this code is bounded as

\[
\mathbb{L} = \mathbb{E}l(X) \\
\leq \mathbb{E}[\log p(x)] + 1 \\
\leq \mathbb{E} - \log p(x) + 2 \\
= H(X) + 2
\]

Let \( F_l(x) = Fx + p(x)/2, x \in X \), and \( F_l(x) = \) approximation of \( F(x) \) to \( l(x) \) bits.

\[
F(x) - F_l(x) \leq 1/2^{l(x)}
\]

**Claim.** \( 1/2^{l(x)} \leq p(x)/2 = F(x) - F_l(x) \)

**Proof.**

\[
2^{l(x)} = 2^{\lceil -\log p(x) \rceil + 1} \\
\geq 2^{-\log p(x) + 1} \\
= \frac{2}{p(x)}
\]

Thus \( F_l(x) \in [F(x), F(x)] \) and therefore \( F_l(x) \) are distinct for \( x \in X \).

**Claim.** This code is prefix-free.

**Proof.** For each codeword \( F_l(x) = 0, y_1 y_2 \ldots y_l \) consider the interval \( [0, y_1 y_2 \ldots y_l, 0, y_1 y_2 \ldots y_l + 1/2^l] \).

A codeword \( c \) has \( F_l(x) \) as its prefix if \( c \in [0, y_1 y_2 \ldots y_l, 0, y_1 y_2 \ldots y_l + 1/2^l] = I(x) \). If they are unequal this cannot be true since \( I(x) \subseteq [F(x), F(x)] \) and \( F_l(x') \in [F(x'), F(x')] \).
Remark: The advantage of this algorithm is that it is adaptive.

Question: Given a prefix-free code with codeword lengths \( l'(x) \), how do the codeword lengths \( l(x) = \lfloor -\log p(x) \rfloor \) for a Shannon code compare with \( l'(x) \)?

Claim. \( \mathbb{P}\{l(x) > l'(X) + \lambda \} \leq 2^{1-\lambda} \)

Proof.

\[
\mathbb{P}\{l(x) > l'(X) + \lambda \} = \sum_{x: l(x) > l'(X) + \lambda} p(x) \leq \sum_{x: l(x) > l'(X) + \lambda} 2^{\log p(x)} \leq 2 \sum_{x: l(x) > l'(X) + \lambda} 2^{1-l(x)} = 2^{1-\lambda} \sum_x 2^{-l'(x)} \leq 2^{1-\lambda} \text{ (By Kraft's inequality)}
\]

Claim. Consider a pmf \( P \) such that \( l(x) = -\log p(x) \) are the positive integers \( \forall x \). Then,

\[
\mathbb{P}(l(X) > l'(X)) \leq \mathbb{P}(l(X) < l'(X))
\]

Proof.

\[
\mathbb{P}(l(X) > l'(X)) - \mathbb{P}(L(X) < l'(X)) = \sum_{x: l(x) > l'(x)} P(x) - \sum_{x: l(x) < l'(x)} P(x) = \sum_x P(x) - \text{sgn}(l(x) - l'(x)) \leq \sum_x p(x)[2^{l(x)} - 1 - l'(x)] \leq \sum_x p(x)2^{l(x)}2^{-l'(x)} - 1 = \sum_x 2^{-l(x)} - 1 \leq 0
\]
In general,
\[
P(l(X) > l'(X) + 1) - P(l(X) < l'(X) + 1) \leq \left[ \sum_x p(x) 2^{l(x) - l'(x) - 1} \right] - 1 \\
\leq \left[ \sum_x p(x) \cdot 2/p(x) \cdot 2^{-l'(x) - 1} \right] - 1 \text{(Using } 2^{l(x)} \leq 2/p(x) \text{)} \\
\leq 0
\]

\[\square\]

**Remark:** This situation is equivalent to the game of twenty questions. Consider a binary tree, in which we have to determine the identity of a particular node. We are allowed to ask the Oracle one question at a time, of the form, ”Is the target node a left child or right child of node X?” It is easy to see that the expected number of questions is in fact equal to the length of the codeword corresponding to that node, because each new level corresponds to one more symbol in the codeword. There are many variations to this question of determining the average number of questions, for example, dealing with an Oracle which lies with a small non-zero probability, or an Oracle with replies with a delay. All of these questions come within the ambit of 'Search Theory', in which information theoretic questions are converted into entirely combinatorial ones.

**Question:** Can we do better with non-singular codes?

We need to lower bound \( L_0(X) \). Consider a source code \((f; \phi)\) such that \( P(\phi(f(x))) = 1 \). Let \( Y = Y_1 Y_2 \ldots Y_N \) denote the random codeword \( f(X) \) of (random) length \( N \). Thus, the average length is given by \( \mathbb{E}N \). Then we have the following inequality -
\[
H(X) = H(\phi(f(X))) \\
\leq H(f(X)) \\
= H(Y_1 \ldots Y_N) \\
= H(Y_1 \ldots Y_N, N) \\
= H(N) + H(Y_1 \ldots Y_N | N) \\
\leq H(N) + \sum_n P_N(n)n \\
= H(N) + \mathbb{E}N
\]

(1)

Note that \( N \) takes values in positive integers. Using Q2 in Homework 3, we get

**Claim.**
\[
H(N) \leq \log(e\mathbb{E}N). \tag{2}
\]

**Proof.** As was shown in Q2 of HW3, for every \( a \in (0, 1) \),
\[
H(N) \leq \log \frac{a}{1-a} - \mathbb{E}N \log a.
\]
Choose

\[ a = \frac{(\mathbb{E}N - 1)}{\mathbb{E}N}. \]

Then,

\[ H(N) \leq \log(\mathbb{E}N - 1) + \mathbb{E}N \log \mathbb{E}N - \mathbb{E}N \log(\mathbb{E}N - 1) \]
\[ = (\mathbb{E}N - 1) \log \left( 1 + \frac{1}{\mathbb{E}N - 1} \right) + \log \mathbb{E}N \]
\[ \leq \log e + \log \mathbb{E}N, \]

since \( \log(1 + x) \leq x \log e. \)

Now, consider two cases:

Case 1: \( H(X) \geq \mathbb{E}N, \) which by (1) and (2) gives

\[ H(X) - \log(eH(X)) \leq \mathbb{E}N. \]

Case 2: \( H(X) < \mathbb{E}N, \) in which case the lower bound for \( \mathbb{E}N \) above holds trivially.

Thus, \( H(X) - \log(eH(X)) \leq \mathbb{E}N \) for every nonsingular variable-length code, which gives

\[ H(X) - \log(eH(X)) \leq \mathcal{T}_0(X). \]

To summarize, we now have the following chain of inequalities -

\[ H(X) - \log(eH(X)) \leq \mathcal{T}_0(X) \leq H(X) \leq \mathcal{T}_0^{\text{unique}} = \mathcal{T}_0^{\text{prefix}} \leq H(X) + 1 \]

**Remark:** Alternative proof of \( H(X) \leq \mathcal{T}_0^{\text{unique}}. \)

**Proof.** Given a UDC, consider its \( k \)th extension. Let \( \mathcal{T}_k \) be the average length of \( k \)th extension. Since the \( k \)th extension is non-singular, we have

\[ \mathcal{T}_k \geq H(X^k) - \log[eH(X^k)] \]

But,

\[ \mathcal{T}_k = k\mathcal{T}_1 \]

Then,

\[ \mathcal{T}_1 \geq \frac{1}{k} H(X^k) - \frac{1}{k} \log(eH(X^k)), \forall k \in \mathbb{N} \]

Taking limit \( k \to \infty \) (and assuming \( H(X) \) finite),

\[ \mathcal{T}_1 \geq H(X) \]

\[ \square \]
Consider \( L_\epsilon(X) \) now. Given \((\phi, f)\) such that

\[
P(\hat{x} \neq x) \leq \epsilon
\]

Where \( \hat{x} = \phi(f(x)) \). By Fano’s inequality,

\[
H(X) = H(X|\hat{X}) + H(\hat{X}) \leq \epsilon \log(|\mathcal{X}| - 1) + h(\epsilon) + H(\hat{X})
\]

Proceeding as before,

\[
L_\epsilon(X) \geq H(X) - \log eH(X) - \epsilon \log(|\mathcal{X}| - 1) - h(\epsilon)
\]

To summarize,

\[
H(X) - \log eH(X) - \epsilon \log |\mathcal{X}| - 1 \leq L_\epsilon(X) \leq L_0(X) \leq H(X) + 1
\]