Last Video: Theoretical bounds for lossless compression with symbol codes

Lower bound (for any unique symbol code):

\[ L := E_p[L(X)] \geq H[p] \]

Upper bound for optimal symbol code:

\[ L_{\text{opt}} < H[p] + 1 \text{ bit (per symbol)} \]

“Shannon Coding”

Problem Set #2

Theoretical bounds beyond symbol codes:

\[ H[p] \leq L_{\text{opt}} < H[p] + \varepsilon \]

Implement Huffman coding in Python

Claim: M.C. is optimal

Complication: Breaking ties in M.C.

\[ p(x) = \begin{cases} \frac{1}{6} & \text{for } x = 00 \\ \frac{1}{6} & \text{for } x = 01 \\ \frac{1}{3} & \text{for } x = 10 \\ \frac{1}{2} & \text{for } x = 11 \end{cases} \]

\[ L = \frac{1}{2} \times \frac{1}{3} + \frac{1}{2} \times \frac{1}{3} + \frac{1}{2} \times \frac{1}{3} = 2 \]
\[ L = \sum_{x \in X} p(x) \ell(x) \]
\[ = \frac{1}{6} \times 3 + \frac{1}{6} \times 3 + \frac{1}{3} \times 2 + \frac{1}{3} \times 1 = 2 \]

**Theorem 1:** \( L = \mathbb{E}_p \{ \ell(x) \} \) does not depend on how you break ties in \( H.C. \)

**Remark:** Encoder & decoder still have to break ties in the same way.

**Theorem 2:** "Huffman Coding constructs an optimal syn code"

**Assumptions:**
\[ \{ \begin{align*}
\bullet & \text{ alphabet } X \text{ with } |X| \geq 2 \\
\bullet & p(x) > 0 \quad \forall x \in X
\end{align*} \]

**Then:** A unique decodable syn code \( C \) on \( X \) that are optimal w.r.t. \( p \) is a Huffman Code \( C_H \) with the same code word lengths \( \forall x \in X \)

(i.e.: \( |C(x)| = |C_H(x)| \quad \forall x \in X \))

**Reminder:** Problem 2.1 suffices to show that Theorem 2 holds for optimal prefix codes.

**Lemma 1:** Assume again \( \square \), and let \( C \) be an optimal prefix code; let's sort the symbols s.t.
\[ p(x_1) \leq p(x_2) \leq p(x_3) \leq \ldots \]
break ties by codeword lengths (lexicographically)

i.e., if \( p(x_i) = p(x_{i+1}) \) then \( \ell(x_i) \geq \ell(x_{i+1}) \)

(then break ties arbitrarily) \( |C(x_i)| \)
Then: (i) \( L(x_1) \geq L(x_2) \geq L(x_3) \geq \ldots \)

(ii) \( L(x_1) = L(x_2) \)

**Proof of Lemma 1:**

(i) assume \( i < j \) and \( L(x_i) < L(x_j) \)

\[ \Rightarrow p(x_i) \neq p(x_j) \]

\[ \Rightarrow p(x_i) < p(x_j) \]

Claim: thus, \( C \) is not optimal because we could swap \( C(x_i) \) & \( C(x_j) \)

\[ \Rightarrow \text{would reduce } L \]

(ii) assume \( L(x_1) > L(x_2) \)

(know from (i) that \( L(x_2) \geq L(x_1) \) \( \forall x' \neq x_1 \))

\[ \Rightarrow L(x_1) > L(x') \quad \forall x' \neq x_1 \]

Claim: thus, \( C \) is not an optimal prefix code, because we could drop the last bit of \( C(x_1) \); can't cleave \( C(x_1) \)

if \( C \) is a prefix code

\[ \Rightarrow \text{reduces } L \text{ by } p(x_1) > 0 \]

**Lemma 2:** Assume \( \otimes \) & \( C \) is optimal prefix code

Then: \( \exists x, x' \in \mathcal{X} \) with \( x \neq x' \) and \( L(x) = L(x') \forall x \in \mathcal{X} \)

s.t. \( C(x) \) & \( C(x') \) only differ on last bit.
Proof of Lemma 2: Assume that such a pair does not exist. But, from Lemma 1, we know \( \exists x \neq \hat{x} \) that satisfies \( \Delta \).

Claim: thus, \( C \) is not optimal because we can drop the last bit of \( C(x) \) without violating prefix code.

Proof: let's call \( C(x) \) with last bit dropped \( \hat{x} \). Then \( \forall x \neq \hat{x} \):

- \( C(x) \) is not prefix of \( C(x) \)
  \[ \Rightarrow (x) \text{ is not prefix of } \hat{x} \]
- if \( \hat{x} \) is prefix of \( C(x) \) \[ \Rightarrow |C(x)| \geq |\hat{x}| \]
  \[ \Rightarrow C(\hat{x}) \text{ is a longest code word} \]
  \[ \Rightarrow C(\hat{x}) \text{ & } C(x) \text{ are two longest code words that differ only on last bit (i.e., they satisfy } \Delta) \]
Reap:

Theorem 2: "Huffman Coding constructs an optimal prefix code"

Assumptions:
\[ \exists \text{ alphabet } X \text{ with } |X| \geq 2 \]
\[ p(x) > 0 \quad \forall x \in X \]

Then: Unique decodable prefix codes on X that are optimal w.r.t. \( p \) for Huffman Code \( C_H \) with the same code word lengths \( \forall x \in X \)

Lemma 1: Sort \( p(x_1) \leq p(x_2) \leq p(x_3) \leq ... \)

Then:
(i) \( L(x_1) \neq L(x_2) \neq L(x_3) \neq ... \)
(ii) \( L(x_1) = L(x_2) \)

Lemma 2: Assume \( \otimes \) & \( C \) is optimal prefix code \( \otimes \)

Then:
\[ \exists x \neq x' \text{ with } x \succ x' \text{ and } L(x) > L(x') \forall x \}

s.t.: \( C(x) & C(x') \) only differ on last bit

Proof of Theorem 2 ("optimality of H.C.")

by induction on \(|X|\)

• base case: \(|X|=2\)

\[ \Rightarrow \text{ only optimal prefix codes} \]
\[ c(\text{"a"}) = 0 \quad \text{and} \quad c(\text{"b"}) = 1 \]

there are H.C.

• induction step: \(|X| > 2\)

\[ \Rightarrow \text{ from Lemma 2: } \exists x \neq x' \text{ with longest code words} \]

that differ only on last bit.

\[ \Rightarrow \text{ if } p(x) \neq p(x') \text{ among the two lowest probs} \]

then apply Lemma 1: symbols \( x_1, x_2 \) with lowest probs and also longest code word length

\[ \begin{array}{c}
\text{"a"} \quad \text{"b"} \\
0 \quad 1
\end{array} \]
\[ \Rightarrow \text{construct prefix code } C' \text{ by swapping } \]
\[ (C(x_1), C(x_2)) \text{ with } (C(x_i), C(x_i)) \]
\[ \text{all have same (longest) length} \]
\[ \Rightarrow \text{swapping then doesn't change } L(x) \text{ for any } x \in X \]

\[ \Rightarrow \text{in } C': \ x_1, x_2 \text{ with lowest } p(x) \text{s} \]
\[ \bullet C(x_i), C(x_i) \text{ are longest} \]
\[ \& \text{only differ on last bit} \]

**Def.** \[ \tilde{X} := (X \setminus \{x_1, x_2\}) \cup \{\star\} \]
\[ \Rightarrow |\tilde{X}| = |X| - 1 \geq 2 \]

\[ \tilde{\rho}(\tilde{x}) = \begin{cases} 
\rho(\tilde{x}) & \text{if } \tilde{x} \in X \\
\rho(x_i)+\rho(x_i) & \text{if } \tilde{x} = \star 
\end{cases} \]

\[ \tilde{C}(\tilde{x}) = \begin{cases} 
C(x_i) & \text{if } \tilde{x} \in X \\
C'(x_i) \text{ with last bit dropped} & \text{if } \tilde{x} = \star 
\end{cases} \]

\[ \tilde{C} \text{ replaces } L \text{ by } p(x_i)+p(x_i) \]

**Claim:** \[ \tilde{C} \text{ is an optimal prefix code (w.r.t } \tilde{\rho}) \]

**Proof:** if \[ \tilde{C} \text{ weren't optimal then } \exists \text{ better prefix code } \tilde{C} \text{ on } \tilde{X} \]

\[ \Rightarrow \text{an constant symbol code on } X \text{ by invert above step (i.e., remove } \star, \text{ introduce } x_i \& x_2 \]

\[ \text{with } C''(x_i) = \tilde{C}(\star)10, \quad C''(x_2) = \tilde{C}(\star)112 \]

\[ \Rightarrow \text{increases } L \text{ by } p(x_i)+p(x_2) \]
or \( x \) →
\[
\begin{align*}
C' & \quad \text{dropped 2 bit from } C(x_1) \& C(x_2) \\
L' &= L \\
\text{appending 1 bit} & \quad C'(x_1) \& C'(x_2) \\
& \quad \text{on } X \\
C & \quad \text{on } X \\
\overline{C} & \quad \text{on } X \\
\overline{C} &= \overline{C} - (p(x_1) + p(x_2)) \\
\overline{C} &= \overline{L} + p(x_1) + p(x_2) \\
&= \overline{L} + p(x_1) + p(x_2) = L'
\end{align*}
\]

\[\Rightarrow L'' < L' \Rightarrow C \text{ is not optimal} \]

contradiction

\[\Rightarrow \overline{C} \text{ is optimal prefix code on alphabet } X \text{ of size } \vert X \vert - 1\]

\[\Rightarrow \text{Theorem 2 applies} \Rightarrow \text{H.C. on } X \text{ with same } \ell(x) \text{ as } \overline{C}\]

\[\Rightarrow C' \text{ has same code word lengths as a H.C. on } X\]

\[\Rightarrow C \text{ has same } \Rightarrow \]  

Next video: begin thinking about better probabilistic models of the data source 

\[\Rightarrow \text{correlations} \]

\[\Rightarrow \text{play back into source coding algorithms} \]