ECS 122A Lecture 11 Jasper Lee

Huffman Coding

Goal: Encode messages (strings of symbols) in binary bit strings

Why? Compression (Lossless)
Transmission (High-voltage vs low-voltage)

Setting

Σ - Alphabet, i.e. set of possible message symbols

e.g. all English letters $\cup$ {",", ".", " "}

Binary code - Function mapping (injective)

$$\Sigma \to \{0,1\}^*$$

i.e. mapping each symbol to a binary bit string (codeword)

Fixed-length code - Every codeword has the same length for every symbol
Min length $= \lceil \log_2 |\Sigma| \rceil$

Variable-length code - Not fixed length

e.g. $\Sigma = \{a, b, c, d\}$

$$C = (a \mapsto 0,\ b \mapsto 10,\ c \mapsto 110,\ d \mapsto 111)$$

Why would we do this instead of (00, 01, 10, 11)?

If a is much much likely, the "average" or "in expectation" code length is shorter.

Will formalize problem in a bit.

Need to be careful w/ ambiguities for var-length codes

Prefix-free code — A (binary) code C such that for every pair of symbols $i, j \in \Sigma$

$C(i)$ is not a prefix of $C(j)$

e.g. - fixed length codes are always prefix-free

- example var-length code above

Optimal Prefix-Free Binary Code

Given: Alphabet Σ

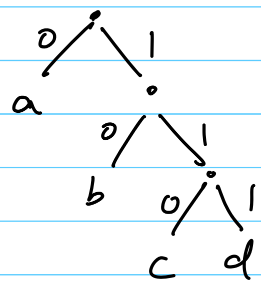
Probability p_σ for each symbol $\sigma \in \Sigma$

Compute prefix-free binary code C w/ min expected code length

$$\sum_{\sigma \in \Sigma} p_\sigma \cdot |C(\sigma)|$$ where $|C(\sigma)|$ is length of $C(\sigma)$

Observation

1. Prefix-Free binary codes are binary trees w/ symbols $\sigma \in \Sigma$ as leave nodes, and 0/1 label on edges (no shared label for both edges out of parent node)



2. Code length for σ is depth of leaf node.

3. Only tree structure matters, any labelling gives same code length.

Huffman Coding

Greedy Algorithm

Maintain a list of subtrees — subtree encodes a subset of symbols

Repeatedly merge pairs until single tree left:

Merge 2 subtrees w/ smallest total prob in subtree

Pseudocode

for $\sigma \in \Sigma$

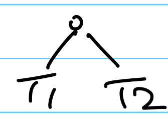
  $T_\sigma \leftarrow$ A single leaf node "σ"

  $P(T_\sigma) \leftarrow p_\sigma$

List $\leftarrow \{T_\sigma\}_{\sigma \in \Sigma}$ // init set of subtrees

while List size > 1

  $T_1, T_2 \leftarrow$ smallest, second smallest $P(T)$ in List

  $T_{merge} \leftarrow$ 

; $P(T_{merge}) \leftarrow P(T_1) + P(T_2)$

  Remove T_1, T_2 from List

  Add T_{merge} back to List

Return final single tree from List

Correctness (Slightly different presentation from textbook)

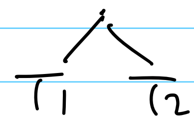
Induction hypothesis:

At the end of the i^{th} iter, there is an opt code with all trees in List as subtrees.

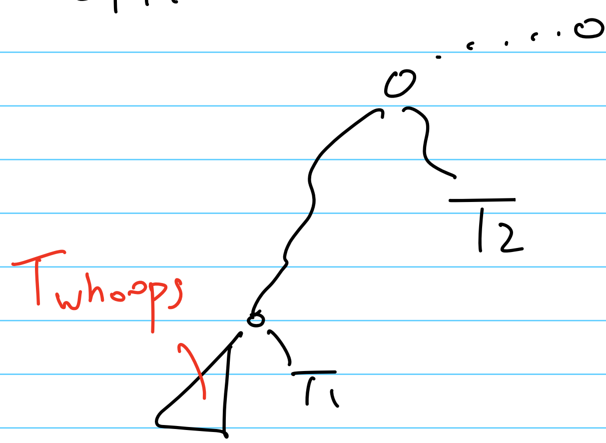
Base case: By defn of codes.

Induction step:

Let OPT_i be the opt code tree guaranteed by IH for i^{th} iter.

If  is subtree of OPT_i then we're done.

Otherwise, wlog say T_1 is deeper in OPT_i than T_2.



Twhoops contains at least one subtree in List (and it can't be T_1 or T_2).

So $P(T_{whoops}) \geq P(T_2)$.

So swap, and new tree has at most same expected code length.

Hence new tree is opt, and has  as subtree ☺

Huffman coding is the best possible, but what is its performance?

Def ((Discrete) Entropy)

Consider a discrete random variable X over the domain $\mathcal{X}$.

$$H(X) = -\sum_{x \in \mathcal{X}} p_x \log p_x$$

usually base 2, which we'll use.

Why is this relevant?

Thm (Shannon's Source Coding Theorem, informal)

No lossless coding scheme can encode X using fewer than $H(X)$ bits

on average, over n iid symbols as $n \to \infty$

Very much graduate level proof, needs some semi-advanced intuition for probability.

Thm Let $C(X)$ be a Huffman code of random variable X.

Then $\mathbb{E}_X(|C(X)|) \leq H(X) + 1$.

Proof is also annoying, afaik no direct analysis of Huffman Coding!

Need to show Shannon-Fano Codes achieve that bound.