ECE 587 / STA 563: Lecture 5 – Lossless Compression

Information Theory
Duke University, Fall 2023

Author: Galen Reeves
Last Modified: October 2, 2023

Outline of lecture:

5.1 Introduction to Lossless Source Coding

5.1.1 Motivating Example

- **Example:** Consider assigning binary phone numbers to your friends

friend	probability	code (i)	code (ii)	code (iii)	code (iv)	code (v)	code (vi)
Alice	1/4	0011	001101	0	00	0	10
Bob	1/2	0011	001110	1	11	11	0
Carol	1/4	1100	110000	10	10	10	11

- Analysis of codes:
  (i) Alice and Bob have same number. Does not work.
  (ii) Works, but phone numbers are too long
  (iii) Not decodable. '10' could mean Carol, or could mean 'Bob, Alice'
  (iv) Works, but why do we need two zeros for Alice? After first zero it is clear who we want.
  (v) Ok, but Alice has a shorter code than Bob
  (vi) This is the optimal code. Once you are finished dialing you can be connected immediately.

- Desirable properties of a code:

(1) Uniquely decodable.

(2) Efficient, i.e., minimize the average codeword length:

$$\mathbb{E}[\ell(X)] = \sum_{x \in \mathcal{X}} p(x)\ell(x)$$

where $\ell(x)$ is the length of the codeword associated with symbol x.

(3) Prefix-free, i.e., no codeword is the prefix of another code

5.1.2 Definitions

- A **source code** is a mapping C from a source alphabet $\mathcal{X}$ to D-ary sequences
  - $\mathcal{D}^*$ is set of finite-length strings of symbols from D-ary alphabet $\mathcal{D} = \{1, 2, \cdots, D\}$, i.e.

$$\mathcal{D}^* = \mathcal{D} \cup \mathcal{D}^2 \cup \mathcal{D}^3 \cup \cdots$$

  - $C(x) \in \mathcal{D}^*$ is the codeword for $x \in \mathcal{X}$
  - $\ell(x)$ is the length of $C(x)$

- A code is **nonsingular** if

$$x \neq \tilde{x} \Rightarrow C(x) \neq C(\tilde{x})$$

- The **extension** of the code C is the mapping from finite length strings of $\mathcal{X}$ to finite length strings of $\mathcal{D}$

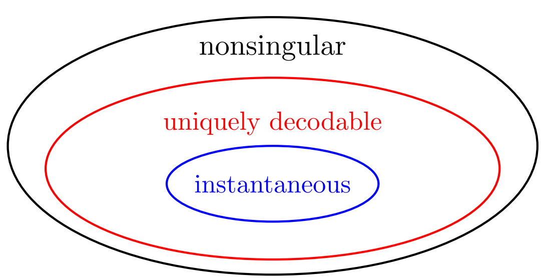
$$C(\underbrace{x_1 x_2 \cdots x_n}_{\text{input (source)}}) = \underbrace{C(x_1)C(x_2)\cdots C(x_n)}_{\text{output (code)}}$$

- A code C is **uniquely decodable** if its extension C^* is nonsingular, i.e., for all m, n,

$$x_1 x_2 \cdots x_m \neq \tilde{x}_1 \tilde{x}_2 \cdots \tilde{x}_n \quad \Longrightarrow \quad C(x_1)C(x_2)\cdots C(x_m) \neq C(\tilde{x}_1)C(\tilde{x}_2)\cdots C(\tilde{x}_n)$$

- A code is **prefix-free** if no codeword is prefixed by another codeword. Such codes are also known as "prefix" codes or instantaneous codes.

- Venn diagram of instantaneous, uniquely decodable, and nonsingular codes.



- Given a distribution $p(x)$ on the input symbol, the goal is to minimize the expected length per-symbol

$$\mathbb{E}[\ell(X)] = \sum_{x \in \mathcal{X}} \ell(x)p(x)$$

5.2 Fundamental Limits of Compression

- This section considers the limits of of lossless compression and proves the following result.
- **Theorem:** For any source distribution $p(x)$, the expected length $\mathbb{E}[\ell(X)]$ of the optimal uniquely decodable D-ary code obeys

$$\frac{H(X)}{\log D} \leq \mathbb{E}[\ell(X)] < \frac{H(X)}{\log D} + 1$$

  Furthermore, there exists a prefix-free code which is optimal.

5.2.1 Uniquely Decodable Codes & Kraft Inequality

- Let $\ell(x)$ be the length function associated with a code C. i.e.,

$$\ell(x) \text{ is length of codeword } C(x) \text{ for all } x \in \mathcal{X}$$

- A code C satisfies the **Kraft Inequality** if and only if

$$\sum_{x \in \mathcal{X}} D^{-\ell(x)} \leq 1 \qquad \text{(Kraft Inequality)}$$

- **Theorem:** Every uniquely decodable code satisfies the Kraft inequality, i.e.,

$$\text{Uniquely decodable} \quad \implies \quad \text{Kraft Inequality}$$

- **Proof:**
  - Let C be a uniquely decodable source code with length function $\ell(x)$ and let $\ell_{\max} = \max_{x \in \mathcal{X}} \ell(x)$ be the length of the longest codeword.
  - For a source sequence x^n, the length of the extended codeword $C(x^n)$ is given by

$$\ell(x^n) = \sum_{i=1}^{n} \ell(x_i) \leq n\ell_{\max}$$

  - Let A_k be the number of source sequences of length n for which $\ell(x^n) = k$, i.e.

$$A_k = \#\{x^n \in \mathcal{X}^n \,:\, \ell(x^n) = k\}$$

  - Since the code is uniquely decodable, the number of source sequences with codewords of length k cannot exceed the number of D-ary sequences of length k, an so

$$A_k \leq D^k$$

  - The extended codeword lengths must obey

$$\begin{aligned}
\sum_{x^n \in \mathcal{X}^n} D^{-\ell(x^n)} &= \sum_{k=1}^{n\ell_{\max}} A_k D^{-k} \\
&\leq \sum_{k=1}^{n\ell_{max}} D^k D^{-k} \qquad \text{(since uniquely decodable)} \\
&\leq n\ell_{max}
\end{aligned}$$

- The extended codeword lengths must also obey

$$\sum_{x^n \in \mathcal{X}^n} D^{-\ell(x^n)} = \sum_{x_1 \in \mathcal{X}} \sum_{x_2 \in \mathcal{X}} \cdots \sum_{x_n \in \mathcal{X}^n} D^{-\ell(x_1)} D^{-\ell(x_2)} \cdots D^{-\ell(x_n)}$$
$$= \sum_{x_1 \in \mathcal{X}} D^{-\ell(x_1)} \sum_{x_2 \in \mathcal{X}} D^{-\ell(x_2)} \times \cdots \times \sum_{x_n \in \mathcal{X}} D^{-\ell(x_n)} = \left[\sum_{x \in \mathcal{X}} D^{-\ell(x)} \right]^n$$

- Combining the above displays shows that

$$\left[\sum_{x \in \mathcal{X}} D^{-\ell(x)} \right]^n \le n \ell_{max} \qquad \text{for all } n$$

- If the code does not satisfy the Kraft inequality, then the left hand side will blow up exponentially as n becomes large, and this inequality will be violated. Thus, the code must satisfy the Kraft inequality.

- **Theorem:** For any source distribution $p(x)$, the expected codeword length of every D-ary uniquely decodable code obeys the lower bound

$$\mathbb{E}[\ell(X)] \ge \frac{H(X)}{\log D}$$

- **Proof:**

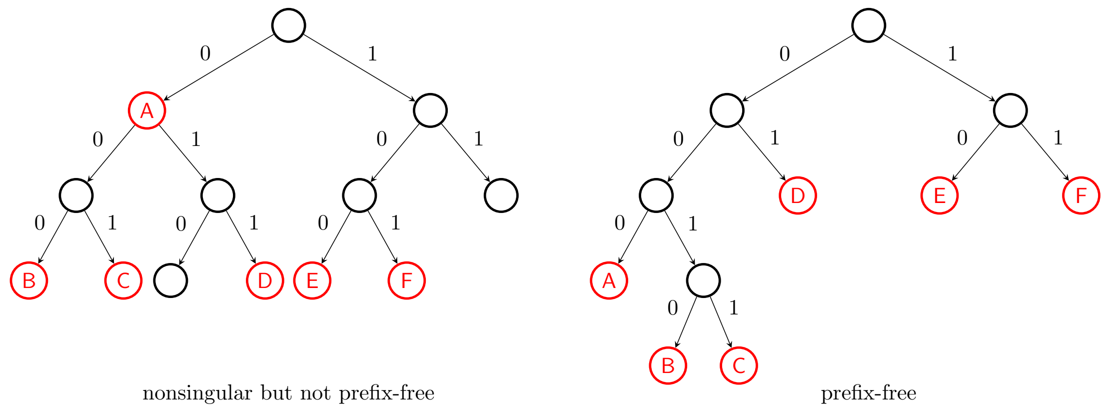
$$\begin{aligned}
\mathbb{E}[\ell(X)] - \frac{H(X)}{\log D} &= \sum_x p(x) \Big[\ell(x) + \log_D p(x) \Big] \\
&= \sum_x p(x) \Big[\log_D D^{\ell(x)} + \log_D p(x) \Big] \\
&= \sum_x p(x) \log_D \Big(D^{\ell(x)} p(x) \Big) \\
&\ge \sum_x p(x) \log_D(e) \left[1 - \frac{D^{-\ell(x)}}{p(x)} \right] && \text{(Fundamental Inq.)} \\
&= \log_D(e) \left(1 - \sum_x D^{-\ell(x)} \right) \\
&\ge 0 && \text{(Kraft Inq.)}
\end{aligned}$$

5.2.2 Codes on Trees

- Any D-ary code can be represented as a D-ary tree.

- A D-ary tree consists of a root with branches, nodes, and leaves. The root and every node has exactly D children.

- Examples of a binary trees



nonsingular but not prefix-free prefix-free

- The depth of a leaf (i.e., the number of steps it takes to reach the root) corresponds to the length of the codeword.
- **Lemma:** A code is prefix-free if and only if each of its codewords is a leaf.

$$\text{code is prefix-free} \quad \Longleftrightarrow \quad \text{every codeword is a leaf}$$

5.2.3 Prefix-Free Codes & Kraft Inequality

- **Theorem:** There exists a prefix-free code with length function $\ell(x)$ if and only if $\ell(x)$ satisfies the Kraft Inequality, i.e.

$$\ell(x) \text{ is the length function of a prefix-free code} \quad \Longleftrightarrow \quad \sum_x D^{-\ell(x)} \leq 1$$

- Proof of '$\Longrightarrow$'
  - This follows because a prefix-free code is uniquely decodable and the length function of a uniquely decodable code satisfies the Kraft inequality.
- Proof of '$\Longleftarrow$'
  - Let $\ell(x)$ be a length function that satisfies the Kraft inequality.
  - The goal is to create a D-ary tree where the depths of the leaves correspond to $\ell(x)$.
  - It suffices to show that, for each integer k, after all codewords of length $\ell(x) < k$ have been assigned, there remain enough unpruned nodes on level k to handle codewords with length $\ell(x) = k$.
  - That is, we need to show that for each k,

$$\underbrace{D^k - \sum_{x:\ell(x)<k} D^{k-\ell(x)}}_{\text{no. remaining nodes after assigning short codes}} \geq \underbrace{\#\{x : \ell(x) = k\}}_{\text{no. needed for codes of length } k}$$

  - The right-hand side can be written as

$$\#\{x : \ell(x) = k\} = \sum_{x \,:\, \ell(x)=k} D^{k-\ell(x)}$$

- So, to succeed on level k we need

$$D^k \geq \sum_{x\,:\,\ell(x)<k} D^{k-\ell(x)} + \sum_{x\,:\,\ell(x)=k} D^{k-\ell(x)}$$

- Dividing both sides by D^k yields

$$1 \geq \sum_{x\,:\,\ell(x)\leq k} D^{-\ell(x)}$$

- Since the lengths satisfy the Kraft inequality,

$$\sum_{x\,:\,\ell(x)\leq \ell} D^{-\ell(x)} \leq \sum_{x\in\mathcal{X}} D^{-\ell(x)} \leq 1$$

  And thus we have shown that there always exist enough remaining nodes to handle the codewords of length k.

- **Theorem:** For any source distribution $p(x)$, there exists a D-ary prefix-free code whose expected length satisfies the upper bound

$$\mathbb{E}[\ell(X)] < \frac{H(X)}{\log D} + 1$$

- **Proof** (This proof is nonintuitive, the next section gives an explicit construction)
  - By the previous theorem, is suffices to show that there exists a length function $\ell(x)$ that satisfies the Kraft inequality and the stated inequality.
  - Consider the length function

$$\ell(x) = \lceil -\log_D p(x) \rceil$$

    where $\lceil x \rceil$ denotes the ceiling function (i.e., round up to the nearest integer). Then

$$\log_D\left(\frac{1}{p(x)}\right) \leq \ell(x) < \log_D\left(\frac{1}{p(x)}\right) + 1$$

  - Since

$$\sum_{x\in\mathcal{X}} D^{-\ell(x)} \leq D^{\log_D p(x)} = \sum_{x\in\mathcal{X}} p(x) = 1$$

    this length function satisfies the Kraft inequality, and there exists a prefix-free code with length function $\ell(x)$.
  - The expected word length is given by

$$\mathbb{E}[\ell(X)] = \mathbb{E}[\lceil -\log_D p(X) \rceil] < \mathbb{E}\left[\log_D\left(\frac{1}{p(X)}\right) + 1\right] = \frac{H(X)}{\log D} + 1$$

5.3 Shannon Code

- We now investigate how to construct codes with nice properties. These include:
  - short expected code length $\Rightarrow$ better compression
  - prefix-free $\Rightarrow$ can decode instantaneously

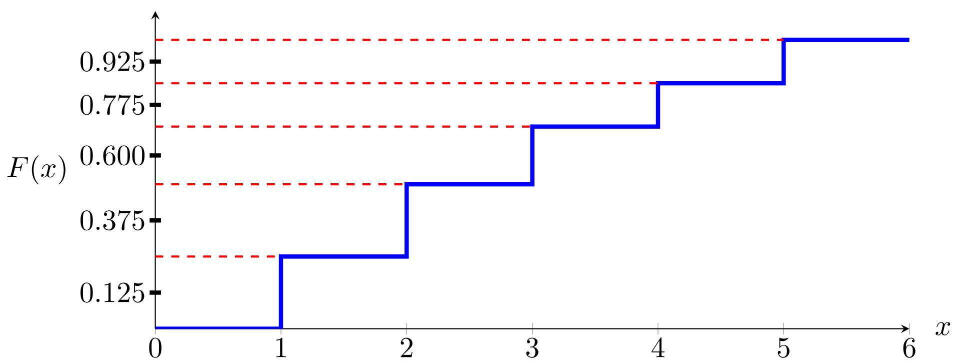
  - efficient representation $\Rightarrow$ don't need huge lookup table for encoding and decoding

- Intuitively, the key idea is to assign shorter codewords to more likely source symbols. The results of the previous section show that there exists a prefix-free code such that:
  - The length function $\ell(x)$ is given by:
$$\ell(x) = \left\lceil \log\left(\frac{1}{p(x)}\right)\right\rceil$$
  - The expected length obeys
$$\mathbb{E}[\ell(X)] < \frac{H(X)}{\log D} + 1$$

- In 1948, Shannon proposed a specific way to build this code. The resulting code is also known as the Shannon–Fano–Elias Code.

- Without loss of generality let the source alphabet be $\mathcal{X} = \{1, 2, \cdots, m\}$.

- The cumulative distribution function (cdf) of the source distribution is
$$F(x) = \sum_{k \le x} p(k)$$



- Construction of the Shannon Code
  - For $x \in \{1, 2, \cdots, m\}$, let $\overline{F}(x)$ be the midpoint of the interval $[F(x-1), F(x))$, i.e.
$$\overline{F}(x) = \frac{F(x-1) + F(x)}{2} = F(x-1) + \frac{p(x)}{2}$$
    Note that $\overline{F}(x)$ is a real number between zero and one that uniquely identifies x.
  - The codeword $C(x)$ corresponds to the D-ary expansion of the real number $\overline{F}(x)$, truncated at the point where the codeword is unique (i.e. cannot be confused with the midpoint of any other interval)
$$C(x) = D\text{-ary expansion of } \overline{F}(x) \text{ such that } |C(x) - \overline{F}(x)| < \frac{1}{2}p(x)\ .$$
    If $\ell(x)$ terms are retained then the codeword is given by
$$\overline{F}(x) = \overbrace{0.\underbrace{z_1 z_2 \cdots z_{\ell(x)}}_{C(x)} z_{\ell(x)+1} z_{\ell(x)+2} \cdots}^{D\text{-ary expansion}}$$

- It is sufficient to retain the first $\ell(x)$ terms where

$$\ell(x) = \left\lceil \log_D \left(\frac{1}{p(x)} \right) \right\rceil + 1$$

since this implies that

$$|C(x) - \overline{F}(x)| \leq D^{-\ell(x)} \leq \frac{p(x)}{D} \leq \frac{1}{2} p(x)$$

Thus, the expected length of the Shannon code obeys:

$$\mathbb{E}[\ell(X)] < \frac{H(X)}{\log D} + 2$$

- **Example:** Consider the following binary Shannon code. The entropy is $H(X) \approx 2.2855$ (bits) and the expected length is $\mathbb{E}[\ell(X)] = 3.5$

x	$p(x)$	$F(x)$	$\bar{F}(x)$	$\bar{F}(x)$ in binary	$\left\lceil \log \frac{1}{p(x)} \right\rceil + 1$	$C(x)$
1	0.25	0.25	0.125	0.001	3	001
2	0.25	0.5	0.375	0.011	3	011
3	0.2	0.7	0.6	$0.1\overline{0011}$	4	1001
4	0.15	0.85	0.775	$0.110\overline{0011}$	4	1100
5	0.15	1	0925	$0.111\overline{01100}$	4	1110

5.4 Huffman Code

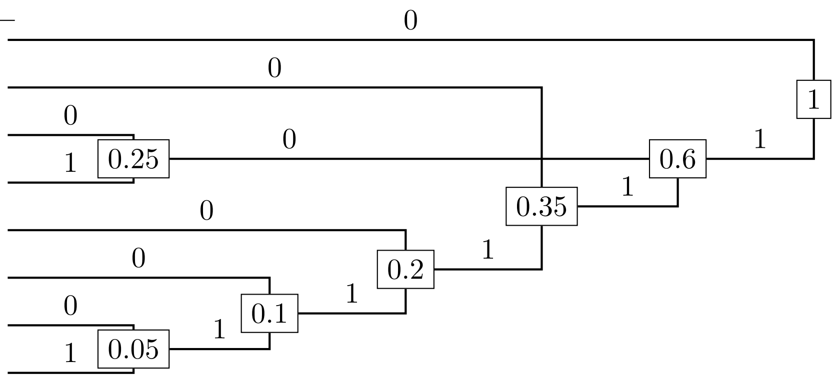
- The Shannon code described in the previous section is good, but it is not necessarily optimal.
- Recall that the Kraft inequality is a:
  - necessary condition for uniquely decodable
  - sufficient condition for the existence of a prefix-free code
- The search for the optimal code can be states as the following optimization problem. Given $p(x)$ find a length function $\ell(x)$ that minimizes the expected length and satisfies the Kraft inequality:

$$\min_{\ell(\cdot)} \sum_{x \in \mathcal{X}} p(x)\ell(x) \quad \text{s.t.} \quad \sum_{x \in \mathcal{X}} D^{-\ell(x)} \leq 1, \qquad \ell(x) \text{ is an integer}$$

- The optimal code was discovered by David Huffman, who was a graduate student in an information theory course (1952).
- Construction of the Huffman Code
  (1) Take the two least probable symbols. These are assigned the longest codewords which have equal length and differ only in the last digit.
  (2) Merge these two symbols into a new symbol with combined probability mass and repeat.

- **Example:** Consider the following source distribution.

code	x	p(x)
0	1	0.4
110	2	0.15
100	3	0.15
101	4	0.1
1110	5	0.1
11110	6	0.05
111110	7	0.04
111111	8	0.01



The entropy is $H(X) \approx 2.45$ bits and the expected length is $\mathbb{E}[\ell(X)] = 2.55$

5.4.1 Optimality of Huffman code

- Let $\mathcal{X} = \{1, 2, \cdots, m\}$ and let $\ell_i = \ell(i)$, $p_i = p(i)$, and $C_i = C(i)$. Without loss of generality, assume probabilities are in descending order

$$p_1 \geq p_2 \geq \cdots \geq p_m$$

- **Lemma 1:** In an optimal code, shorter codewords are assigned large probabilities, i.e.

$$p_i > p_j \quad \Longrightarrow \quad \ell_i \leq \ell_j$$

- **Proof:**
  - Assume otherwise, that is $\ell_i > \ell_j$ and $p_i > p_j$. Then, by exchanging these codewords the expected length will decrease, and thus the code is not optimal.

- **Lemma 2:** There exists an optimal code for which the codewords assigned to the smallest probabilities are siblings (i.e., they have the same length and differ only in the last symbol).

- **Proof:**
  - Consider any optimal code. By lemma 1, codeword C_m has the longest length. Assume for the sake of contradiction, its sibling is not a codeword. Then the expected length can be decreased by moving C_m to its parent. Thus, the code is not optimal and a contradiction is reached.
  - Now, we know the sibling of C_m is a codeword. If it is C_{m-1}, we are done.
  - Assume it is some C_i for $i \neq m-1$ and the code is optimal. By Lemma 1, this implies $p_i = p_{m-1}$. Therefore, C_i and C_{m-1} can be exchanged without changing expected length.

- **Theorem:** Huffman's algorithm produces an optimal code tree

- Proof of optimality of Huffman Code
  - Let $\ell(x)$ be the length function of the optimal code.
  - By lemmas 1 and 2, C_{m-1} and C_m are siblings and the longest codewords.

- Let $\tilde{p}_1 \geq \tilde{p}_2 \geq \cdots \geq \tilde{p}_{m-1}$ denote the ordered probabilities after merging p_{m-1} and p_m. Let $\tilde{\ell}(\tilde{x})$ be the length function of resulting code for this new distribution. (Note the new distribution has support of size $m-1$).
- Let $\mathbb{E}[\ell(X)]$ be the expected length of the original code and $\mathbb{E}\left[\tilde{\ell}(\tilde{X})\right]$ the expected length of the reduced code. Then

$$\mathbb{E}[\ell(X)] = \mathbb{E}\left[\tilde{\ell}(\tilde{X})\right] + \underbrace{\mathbb{P}\left[\tilde{\ell}(\tilde{X}) \neq \ell(X)\right]}_{\text{prob of merged symbol}} \times 1 = \mathbb{E}\left[\tilde{\ell}(\tilde{X})\right] + p_{m-1} + p_m$$

- Thus, $\ell(x)$ is the length function of an optimal code if an only if $\tilde{\ell}(\tilde{x})$ is the length function of an optimal code.
- Therefore, we have reduced the problem to finding and optimal code tree for $\tilde{p}_1, \cdots \tilde{p}_{m-1}$.
- Again, merge, and continue the process....

- Thus, the Huffman algorithm yields the optimal code in a greedy fashion (there may be other optimal codes).

5.5 Coding Over Blocks

- Let $X_1, X_2, \cdots$ be an iid source with finite alphabet $|\mathcal{X}|$. This is known as a **discrete memoryless source**
- One issue with symbol codes is that there is a penalty for using integer codeword lengths.
- **Example:** Suppose that $X_1, X_2, \cdots$ are $\sim$ iid Bernoulli(p) with p very small.
  - The optimal code is given by

$$C(x) = \begin{cases} 0, & x = 0 \\ 1, & x = 1 \end{cases}$$

  - The expected length is $\mathbb{E}[\ell(X)] = 1$ but the entropy obeys

$$H(X) = H_b(p) \sim p\log(1/p), \qquad p \to 0$$

- We can overcome the integer effects by coding over blocks of inputs symbols.
  - Group inputs into blocks of size n to create a new source $\tilde{X}_1, \tilde{X}_2, \cdots$ where

$$\begin{aligned} \tilde{X}_1 &= [X_1, X_2, \cdots, X_n] \\ \tilde{X}_2 &= [X_{n+1}, X_{n+2}, \cdots, X_{2n}] \\ &\vdots \\ \tilde{X}_i &= [X_{(i-1)n+1}, X_{(i-1)n+2}, \cdots, X_{in}] \end{aligned}$$

  - Each length-n vector can be viewed as a "symbol" from the alphabet $\tilde{\mathcal{X}} = \mathcal{X}^n$. This new source alphabet has size $|\mathcal{X}|^n$.
  - The new probabilities are given by

$$p(\tilde{x}) = \prod_{k=1}^{n} p(\tilde{x}_k)$$

- The entropy of the new source distribution is

$$H(\tilde{X}) = H(X_1, X_2, \cdots, X_n) = n\, H(X)$$

- The expected length of the optimal code for the source distribution $p(\tilde{x})$ obeys

$$\underbrace{nH(X)}_{H(\tilde{X})} \le \mathbb{E}\left[\ell(\tilde{X})\right] < \underbrace{nH(X)}_{H(\tilde{X})} + 1$$

- To encode the source $X_1, X_2, \dots$ it is sufficient to encode the new source $\tilde{X}_1, \tilde{X}_2, \dots$. If we use a prefix-free code, then once the codeword $C(\tilde{X}_1)$ is received, we can decode $\tilde{X}_1$, and thus recover the first n source symbols $X_1, \dots, X_n$.
  - The expected codeword length per source symbol is given by the expected codeword length $\mathbb{E}[\ell(\tilde{X})]$ per block, normalized by the block length. It obeys

$$H(X) \le \frac{1}{n}\mathbb{E}[\ell(\tilde{X})] < H(X) + \frac{1}{n}$$

    Thus, the integer effects are negligible as we increase the block length!.
  - However, by coding over an input block of length n we have introduced delay in the system.
  - Furthermore, we have increased the complexity of the code.

5.6 Coding with Unknown Distributions

5.6.1 Minimax Redundancy

- Suppose X is drawn according to a distribution $p_\theta(x)$ with unknown parameter θ belonging to set Θ.

- If θ is known, then we can construct a code that achieves the optimal expected length

$$\sum_x p_\theta(x)\ell(x) = H(p_\theta)$$

- The **redundancy** of coding a distribution p with the optimal code for a distribution q (i.e., $\ell(x) = -\log q(x)$) is given by

$$\begin{aligned}
R(p,q) &= \overbrace{\sum_x p(x)\ell(x)}^{\text{actual length}} - \overbrace{H(p)}^{\text{optimal length}} \\
&= \sum_x p(x)\left(\log\left(\frac{1}{q(x)}\right) - \log\left(\frac{1}{p(x)}\right)\right) \\
&= \sum_x p(x)\log\left(\frac{p(x)}{q(x)}\right) \\
&= D(p||q)
\end{aligned}$$

- The **minimax redundancy** is defined by

$$R^* = \min_q \max_{\theta\in\Theta} R(p_\theta, q) = \min_q \max_{\theta\in\Theta} D(p_\theta||q)$$

- Intuitively, the distribution q that leads to a code minimizing the minimax redundancy is the distribution at the center of the "information ball" of radius R^*.

- **Minimax Theorem:** Let $f(x, y)$ be a continuous function that is convex in x and concave in y, and let $\mathcal{X}$ and $\mathcal{Y}$ be compact convex sets. Then:

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} f(x, y) = \max_{y \in \mathcal{Y}} \min_{x \in \mathcal{X}} f(x, y)$$

  This is a classic result in game theory. There are many extensions, such as Sion's minimax theorem, which applies when $f(x, y)$ is quasi-convex-concave and at least one of the sets is compact.

- Recall that $D(p||q)$ is convex in the pair (p, q), i.e., for all $\lambda \in [0, 1]$,

$$D(\lambda p_1 + (1 - \lambda)p_2 \,||\, \lambda q_1 + (1 - \lambda)q_2) \leq \lambda D(p_1||q_1) + (1 - \lambda)D(p_2||q_2)$$

- Let Π be the set of all distributions on θ. Note that Π is a convex set, i.e., for all $\lambda \in [0, 1]$,

$$\pi_1, \pi_2 \in \Pi \implies \lambda\pi_1 + (1 - \lambda)\pi_2 \in \Pi$$

- **Lemma:** The maximum over $R(p_\theta, q)$ with respect to $\theta \in \Theta$ is equal to the maximum over $\pi \in \Pi$ of the expectation with respect to π.

$$\max_{\theta \in \Theta} D(p_\theta||q) = \max_{\pi \in \Pi} \underbrace{\sum_{\theta \in \Theta} \pi(\theta) D(p_\theta||q)}_{\text{expectation with respect to } \pi}$$

  This lemma follows from the fact that the maximum of a convex function over a convex set is attained at an extreme point of the set. We provide a simple proof below.

  - **Proof of less than or equal:** Let δ_{θ_0} denote the distribution that has probability one at θ_0 and note that

$$D(p_{\theta_0}||q) = \underbrace{\sum_{\theta \in \Theta} \delta_{\theta_0}(\theta) D(p_\theta||q)}_{\text{expectation with respect to } \delta_\theta}$$

    Therefore, maximizing over θ is equivalent to maximizing over the expectation with respect to distributions in the set $\tilde{\Pi} = \{\delta_\theta : \theta \in \Theta\}$. Hence,

$$\max_{\theta \in \Theta} D(p_\theta||q) = \max_{\pi \in \tilde{\Pi}} \sum_{\theta \in \Theta} \pi(\theta) D(p_\theta||q) \leq \max_{\pi \in \Pi} \sum_{\theta \in \Theta} \pi(\theta) D(p_\theta||q)$$

    where the inequality holds because $\tilde{\Pi}$ is a subset of Π.

- **Proof of greater than or equal:** Let θ^* be a value that attains the maximum of $D(p_\theta||q)$. Note that for every $\pi \in \Pi$ we have

$$\sum_{\theta\in\Theta} \pi(\theta) D(p_\theta||q) \le \sum_{\theta\in\Theta} \pi(\theta) D(p_{\theta^*}||q) = D(p_{\theta^*}||q) = \max_{\theta\in\Theta} D(p_\theta||q)$$

Taking the maximum of the left-hand side with respect to π in Π yields the stated inequality.

- This means that the minimax redundancy can be expressed equivalently as

$$R^* = \min_q \max_{\pi\in\Pi} \sum_{\theta\in\Theta} \pi(\theta) D(p_\theta||q)$$

Note that the objective is linear (and hence both convex and concave) in π and convex in q. Applying the minimax theorem yields:

$$R^* = \max_{\pi\in\Pi} \min_q \sum_{\theta\in\Theta} \pi(\theta) D(p_\theta||q)$$

- For each distribution π we want to find the optimal distribution q. As an educated guess, consider the distribution induced on x by p_θ when θ is drawn according to π i.e.

$$q_\pi(x) = \sum_{\theta\in\Theta} \pi(\theta) p_\theta(x)$$

To see that q_π achieves the minimum, observe that for any q, we can write

$$\begin{aligned}
\sum_\theta \pi(\theta) D(p_\theta||q) &= \sum_\theta \pi(\theta) D(p_\theta||q) - D(q_\pi||q) + D(q_\pi||q) \\
&= \sum_\theta \sum_x \pi(\theta) p_\theta(x) \log\left(\frac{p_\theta(x)}{q(x)}\right) - \sum_x \underbrace{\left(\sum_\theta \pi(\theta) p_\theta(x)\right)}_{q_\pi(x)} \log\left(\frac{q_\pi(x)}{q(x)}\right) + D(q_\pi||q) \\
&= \sum_\theta \sum_x \pi(\theta) p_\theta(x) \left[\log\left(\frac{p(x)}{q(x)}\right) - \log\left(\frac{q_\pi(x)}{q(x)}\right)\right] + D(q_\pi||q) \\
&= \sum_\theta \sum_x \pi(\theta) p_\theta(x) \log\left(\frac{p_\theta(x)}{q_\pi(x)}\right) + D(q_\pi||q)
\end{aligned}$$

Note that the first term on the right-hand side does not depend on q. Since $D(q_\pi||q)$ is nonnegative and equal to zero if and only if $q = q_\pi$, we see that q_π is the unique minimizer.

- To make the expression more interpretable, consider the notation

$$p(\theta) = \pi(\theta), \qquad p(x \mid \theta) = p_\theta(x), \qquad p(x) = q_\pi(x)$$

Then, we have shown that the minimax redundancy can be expressed as

$$\begin{aligned}
R^* &= \max_{p(\theta)} \sum_\theta \sum_x p(\theta) p(x|\theta) \log\left(\frac{p(x|\theta))}{p(x)}\right) \\
&= \max_{p(\theta)} I(\theta; X)
\end{aligned}$$

- **Theorem:** The minimax redundancy is equal to the maximum mutual information between the the parameter θ and the source X
- In other words, the code that minimizes the minimax redundancy has length function $\ell(x) = -\log p(x)$ where $p(x)$ is the distribution of $X \sim p(x|\theta)$ when θ is drawn according to the distribution that maximizes the mutual information $I(\theta; X)$.

5.6.2 Coding with Unknown Alphabet

- We want to compress integers $x \in \mathbb{N} = \{1, 2, 3, \dots\}$ without specifying a probability distribution.
$$c(x) = ??$$
- First consider the setting where we have an upper bound N on the integer, and thus $\mathcal{X} = \{1, 2, \cdots, N\}$. We can simply send $\lceil \log N \rceil$ bits. For example, $N = 8$, then we send three bits per integer:
$$3, 7 \quad \Longrightarrow \quad c(3)c(7) = \underbrace{011}_{3}\underbrace{111}_{7}$$
- To analyze minimax redundancy of this approach, consider the set of distributions:
$$p_\theta(x) = \begin{cases} 1, & x = \theta \\ 0, & x \neq \theta \end{cases}, \qquad \mathcal{X} = \Theta = \{1, 2, \cdots, N\},$$
The minimax redundancy is given by
$$R^* = \max_{p(\theta)} I(\theta; X) = \max_{p(x)} H(X) = \log N$$
since the uniform distribution maximizes entropy on a finite set.
- But we want the code to be *universal* and work for any integer.
- A **unary code** sends a sequence of $x-1$ '0's followed by a '1' to mark the end of the codeword. For example,
$$3, 7 \quad \Longrightarrow \quad c(3)c(7) = \underbrace{001}_{3}\underbrace{0000001}_{7}$$
- The unary code requires x bits to represent each symbol. This seems wasteful.
- Idea: First use a unary code to describe how many bits are needed for the binary code, and then send the binary code,
$$c_{\text{universal}}(x) = (c_{\text{unary}}(\ell_{\text{binary}}(x)), c_{\text{binary}}(x))$$
- For example, suppose we want to compress 9:
  - The binary code is $c_{\text{binary}}(9) = 1001$
  - The length of the binary code is $\ell_{\text{binary}}(9) = 4$
  - So the universal code is
$$c_{\text{universal}}(9) = \underbrace{0001}_{\text{header}}\ \underbrace{1001}_{\text{number}}$$
- This universal code requires $\lceil \log_2(x) \rceil + \lceil \log_2(x) \rceil = 2\lceil \log_2(x) \rceil$ bits.

- In fact, we can do better by repeating the process to first compress universal code using itself!

$$c^{(2)}_{\text{universal}}(x) = \left(c^{(1)}_{\text{univeral}}(\ell_{\text{binary}}(x)), c_{\text{binary}}(x)\right)$$

The number of bits this scheme requires obeys

$$\begin{aligned}\ell^{(2)}_{\text{universal}}(x) &= \lceil \log_2(x) \rceil + 2\lceil \log_2(\lceil \log_2(x) \rceil) \rceil \\ &\le \log_2(x) + 2\log_2(\log_2(x)) + 4\end{aligned}$$

- It is interesting to note that this length function obeys the Kraft inequality. Thus, the length function may be viewed as a universal prior

$$p_{\text{universal}}(x) = 2^{-\ell^{(2)}_{\text{universal}}(x)} \approx \frac{1}{x(\log(x))^2}$$

Recall that $\sum_{n\ge1} 1/(n\log n)^p$ diverges for $p = 1$ but converges for $p > 1$.

- It is also interesting to note that the entropy of this distribution is infinite,

$$H(p_{\text{universal}}) = \sum_{x=1}^{\infty} \frac{1}{x(\log(x))^2} \log(x \log(x)^2) = +\infty$$

- In this case, we have $\theta = X$ and so the minimax redundancy corresponds to a distribution which maximizes $I(X;X) = H(X)$.

5.6.3 Lempel-Ziv Code

- Lempel-Ziv (LZ) codes are a key component of many moderns data compression algorithms, including:
  - `compress`, `gzip`, `pkzip`, ZIP file format
  - Graphics Interchange Format (GIF)
  - Portable Document Format (PDF)
- Basic idea: Compress string using reference to its past. The more redundant the string, the better this process works.
- Roughly speaking, Lempel-Ziv codes are optimal in two senses:
  (1) They approach the entropy rate if the source is generated from a stationary and ergodic distribution.
  (2) They are competitive against all possible finite-state machines
- Two different variations, LZ 77 and LZ 78. The `gzip` algorithm using LZ '77 followed by a Huffman code.
- Construction of Lempel-Ziv code:
  - Input: a string of source symbols $x_1x_2x_3, \cdots$
  - Output: sequence of code words: $c(x_1)c(x_2)c(x_3)\cdots$

- Assume that we have compressed the string from x_1 to x_{i-1}. The goal is to find the longest possible match between the next symbols and a sequence in the previous sequence. In other words, we want to find the largest integer k such that

$$\underbrace{x_i x_{i+1} \dots x_{i+k}}_{\text{new bits}} = \underbrace{x_j x_{j+1} \dots x_{j+k}}_{\text{previous bits}} \qquad \text{for some } j \le i-1$$

- Thus, this matching phrase can be represented by a pointer to index $i-j$ and its length k. For convenience,
- If no match is found, we send the next symbol uncompressed. Use a flag to distinguish the two cases:
  * Find a match $\implies$ send (1, pointer, length)
  * No match $\implies$ send (0, x_i)

- **Example:** Compress the following sequence with window size $W = 4$

$$ABBABBBAABBBA$$

Parsed String:

$$A, B, B, ABB, BA, ABBBA$$

Output:

$$(0, A), (0, B), (1, 1, 1), (1, 3, 3), (1, 4, 2), (1, 5, 5)$$

- **Theorem:** If a process $X_1, X_2, \dots$ is stationary and ergodic, then the per-symbol expected codeword length of the Lempel-Ziv code asymptotically achieves the entropy rate of the source.

- **Proof sketch:**
  - Assume infinite window size.
  - Assume that we only consider matches of exactly length m, and that the sequence has been running long enough that all possible strings of length n have occurred previously.
  - Given a new sequence of length n, how far back in time must we look to find a match? The **return time** is defined by:

$$R_n(X_1, X_2, \dots, X_n) = \min\left\{ j \ge 1 \,:\, X_{1-j}, X_{2-j}, \dots, X_{n-j} = X_1, X_2, \dots, X_n \right\}$$

  - Using universal integer code, can describe R_n with $\log_2 R_n + 2\log_2\log_2 R_n + 4$ bits.
  - Thus, the expected per-symbol length of our code is given by

$$\frac{1}{n}\mathbb{E}[\log_2 R_n + 2\log_2\log_2 R_n + 4]$$

  - Observe that if the sequence is iid, then the return time of a sequence of x_1^n is geometrically distributed with probability $p(x_1^n)$, and thus the expected wait time is

$$\mathbb{E}[R_n(X_1^n) \mid X_1^n = x_1^n] = \frac{1}{p(x_1^n)}$$

- **Kac's Lemma:** If $X_1, X_2, \ldots$ is a stationary ergodic processes, then

$$\mathbb{E}[R_n(X_1^n) \mid X_1^n = x_1^n] = \frac{1}{p(x_1^n)}$$

- To conclude proof, we use Jensen's inequality:

$$\begin{aligned}
\mathbb{E}[\log R_n] &= \mathbb{E}_{X_1^n}[\mathbb{E}[\log(R_n(X_1^n)) \mid X_1^n]] \\
&\leq \mathbb{E}_{X_1^n}[\log(\mathbb{E}[\log(R_n(X_1^n)) \mid X_1^n])] \\
&= \mathbb{E}_{X_1^n}\left[\log\left(\frac{1}{p(X_1^n)}\right)\right] \\
&= H(X_1, \ldots, X_n)
\end{aligned}$$

- By the AEP for stationary ergodic processes,

$$\frac{H(X_1, \ldots, X_n)}{n} \to H(\mathcal{X})$$