COMP2610 / COMP6261 Information Theory
Lecture 14: Source Coding Theorem for Symbol Codes

Quanling Deng

Computational Mathematics Group
School of Computing
College of Engineering & Computer Science
The Australian National University
Canberra, Australia

Australian National University

Announcements

Assignment 2

- Available via Wattle
- Worth 20% of Course total
- Due Monday 29 September 2022, 5:00 pm
- Answers could be typed or handwritten

Last Time

- Variable-length codes
- Uniquely decodable and prefix codes
    - Prefix codes as trees
- Kraft's inequality:
  Lengths $\{\ell_i\}_{i=1}^{I}$ can form a prefix code $\iff \Sigma_{i=1}^{I} 2^{-\ell_i} \leq 1$.
- How to generate prefix codes?

Prefix Codes (Recap)

A simple property of codes **guarantees** unique decodeability

Prefix property

A codeword $\mathbf{c} \in \{0,1\}^+$ is said to be a **prefix** of another codeword $\mathbf{c}' \in \{0,1\}^+$ if there exists a string $\mathbf{t} \in \{0,1\}^+$ such that $\mathbf{c}' = \mathbf{ct}$.

Can you create $\mathbf{c}'$ by gluing something to the end of $\mathbf{c}$?

- **Example**: 01101 has prefixes 0, 01, 011, 0110.

Prefix Codes

A code $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_I\}$ is a **prefix code** if for every codeword $\mathbf{c}_i \in C$ there is no prefix of $\mathbf{c}_i$ in C.

In a stream, no confusing one codeword with another

Prefix Codes as Trees (Recap)

$$C_2 = \{0, 10, 110, 111\}$$

<table>
<tr><td rowspan="8">0</td><td rowspan="4">00</td><td rowspan="2">000</td><td>0000</td></tr>
<tr><td>0001</td></tr>
<tr><td rowspan="2">001</td><td>0010</td></tr>
<tr><td>0011</td></tr>
<tr><td rowspan="4">01</td><td rowspan="2">010</td><td>0100</td></tr>
<tr><td>0101</td></tr>
<tr><td rowspan="2">011</td><td>0110</td></tr>
<tr><td>0111</td></tr>
<tr><td rowspan="8">1</td><td rowspan="4">10</td><td rowspan="2">100</td><td>1000</td></tr>
<tr><td>1001</td></tr>
<tr><td rowspan="2">101</td><td>1010</td></tr>
<tr><td>1011</td></tr>
<tr><td rowspan="4">11</td><td rowspan="2">110</td><td>1100</td></tr>
<tr><td>1101</td></tr>
<tr><td rowspan="2">111</td><td>1110</td></tr>
<tr><td>1111</td></tr>
</table>

This time

Bound on expected length for a prefix code

Shannon codes

Huffman coding

Expected Code Length

With uniform codes, the length of a message of N outcomes is trivial to compute

With variable-length codes, the length of a message of N outcomes will depend on the outcomes we observe

Outcomes we observe have some uncertainty

- On average, what length of message can we expect?

Expected Code Length

With uniform codes, the length of a message of N outcomes is trivial to compute

With variable-length codes, the length of a message of N outcomes will depend on the outcomes we observe

Outcomes we observe have some uncertainty

- On average, what length of message can we expect?

Expected Code Length

The **expected length** for a code C for ensemble X with $\mathcal{A}_X = \{a_1, \ldots, a_I\}$ and $\mathcal{P}_X = \{p_1, \ldots, p_I\}$ is

$$L(C, X) = \mathbb{E}\left[\ell(x)\right] = \sum_{x \in \mathcal{A}_X} p(x)\, \ell(x) = \sum_{i=1}^{I} p_i\, \ell_i$$

Expected Code Length: Examples

Example: X has $\mathcal{A}_X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$ and $\mathcal{P} = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\}$

1. The code $C_1 = \{0001, 0010, 0100, 1000\}$ has

$$L(C_1, X) = \sum_{i=1}^{4} p_i \, \ell_i = 4$$

Expected Code Length: Examples

Example: X has $\mathcal{A}_X = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\}$ and $\mathcal{P} = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\}$

1. The code $C_1 = \{0001, 0010, 0100, 1000\}$ has

$$L(C_1, X) = \sum_{i=1}^{4} p_i \, \ell_i = 4$$

2. The code $C_2 = \{0, 10, 110, 111\}$ has

$$L(C_2, X) = \sum_{i=1}^{4} p_i \, \ell_i = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{8} \times 3 = 1.75$$

Code Lengths and Probabilities

The *Kraft inequality* says that $\{\ell_1, \ldots, \ell_I\}$ are prefix code lengths **iff**

$$\sum_{i=1}^{I} 2^{-\ell_i} \leq 1$$

If it were true that

$$\sum_{i=1}^{I} 2^{-\ell_i} = 1$$

then we could interpret

$$\mathbf{q} = (2^{-\ell_1}, \ldots, 2^{-\ell_I})$$

as a probability vector over I outcomes

General lengths ℓ?

Code Lengths and Probabilities

Probabilities from Code Lengths

Given code lengths $\ell = \{\ell_1, \ldots, \ell_I\}$ such that $\sum_{i=1}^{I} 2^{-\ell_i} \leq 1$, we define $\mathbf{q} = \{q_1, \ldots, q_I\}$, the **probabilities for** ℓ, by

$$q_i = \frac{2^{-\ell_i}}{z}$$

where

$$z = \sum_i 2^{-\ell_i}$$

ensure that q_i satisfy $\sum_i q_i = 1$.

Note: this implies $\ell_i = \log_2 \frac{1}{zq_i}$

Code Lengths and Probabilities

Probabilities from Code Lengths

Given code lengths $\ell = \{\ell_1, \ldots, \ell_I\}$ such that $\sum_{i=1}^{I} 2^{-\ell_i} \leq 1$, we define $\mathbf{q} = \{q_1, \ldots, q_I\}$, the **probabilities for** ℓ, by

$$q_i = \frac{2^{-\ell_i}}{z}$$

where

$$z = \sum_i 2^{-\ell_i}$$

ensure that q_i satisfy $\sum_i q_i = 1$.

Note: this implies $\ell_i = \log_2 \frac{1}{zq_i}$

Examples:

1. Lengths $\{1, 2, 2\}$ give $z = 1$ so $q_1 = \frac{1}{2}$, $q_2 = \frac{1}{4}$, and $q_3 = \frac{1}{4}$

Code Lengths and Probabilities

Probabilities from Code Lengths

Given code lengths $\ell = \{\ell_1, \dots, \ell_I\}$ such that $\sum_{i=1}^{I} 2^{-\ell_i} \leq 1$, we define $\mathbf{q} = \{q_1, \dots, q_I\}$, the **probabilities for** ℓ, by

$$q_i = \frac{2^{-\ell_i}}{z}$$

where

$$z = \sum_i 2^{-\ell_i}$$

ensure that q_i satisfy $\sum_i q_i = 1$.

Note: this implies $\ell_i = \log_2 \frac{1}{zq_i}$

Examples:

1. Lengths $\{1, 2, 2\}$ give $z = 1$ so $q_1 = \frac{1}{2}$, $q_2 = \frac{1}{4}$, and $q_3 = \frac{1}{4}$
2. Lengths $\{2, 2, 3\}$ give $z = \frac{5}{8}$ so $q_1 = \frac{2}{5}$, $q_2 = \frac{2}{5}$, and $q_3 = \frac{1}{5}$

Minimising Expected Code Length

The probability view of lengths will be useful in answering:

Goal of compression

Given an ensemble X with probabilities $\mathcal{P}_X = \mathbf{p} = \{p_1, \ldots, p_I\}$ how can we minimise the expected code length?

Minimising Expected Code Length

The probability view of lengths will be useful in answering:

Goal of compression

Given an ensemble X with probabilities $\mathcal{P}_X = \mathbf{p} = \{p_1, \ldots, p_I\}$ how can we minimise the expected code length?

In particular, we can relate the expected code length to the **relative entropy** (KL divergence) between $\mathbf{p}, \mathbf{q}$:

Minimising Expected Code Length

The probability view of lengths will be useful in answering:

Goal of compression

Given an ensemble X with probabilities $\mathcal{P}_X = \mathbf{p} = \{p_1, \ldots, p_I\}$ how can we minimise the expected code length?

In particular, we can relate the expected code length to the **relative entropy** (KL divergence) between $\mathbf{p}, \mathbf{q}$:

Limits of compression

Given an ensemble X with probabilities $\mathbf{p}$, and prefix code C with codeword length probabilities $\mathbf{q}$ and normalisation z,

$$\begin{aligned} L(C, X) &= H(X) + D_{\mathrm{KL}}(\mathbf{p} \| \mathbf{q}) + \log_2 \frac{1}{z} \\ &\geq H(X), \end{aligned}$$

with equality only when $\ell_i = \log_2 \frac{1}{p_i}$.

Minimising Expected Code Length

Suppose we use code C with lengths $\ell = \{\ell_1, \ldots, \ell_I\}$ and corresponding probabilities $\mathbf{q} = \{q_1, \ldots, q_I\}$ with $q_i = \frac{1}{z} 2^{-\ell_i}$. Then,

$$L(C, X) = \sum_i p_i \ell_i$$

Minimising Expected Code Length

Suppose we use code C with lengths $\ell = \{\ell_1, \ldots, \ell_I\}$ and corresponding probabilities $\mathbf{q} = \{q_1, \ldots, q_I\}$ with $q_i = \frac{1}{z} 2^{-\ell_i}$. Then,

$$
\begin{aligned}
L(C, X) &= \sum_i p_i \ell_i \\
&= \sum_i p_i \log_2 \left(\frac{1}{z q_i} \right)
\end{aligned}
$$

Minimising Expected Code Length

Suppose we use code C with lengths $\ell = \{\ell_1, \ldots, \ell_I\}$ and corresponding probabilities $\mathbf{q} = \{q_1, \ldots, q_I\}$ with $q_i = \frac{1}{z} 2^{-\ell_i}$. Then,

$$
\begin{aligned}
L(C, X) &= \sum_i p_i \ell_i \\
&= \sum_i p_i \log_2 \left(\frac{1}{z q_i} \right) \\
&= \sum_i p_i \log_2 \left(\frac{1}{z p_i} \frac{p_i}{q_i} \right)
\end{aligned}
$$

Minimising Expected Code Length

Suppose we use code C with lengths $\ell = \{\ell_1, \ldots, \ell_I\}$ and corresponding probabilities $\mathbf{q} = \{q_1, \ldots, q_I\}$ with $q_i = \frac{1}{z} 2^{-\ell_i}$. Then,

$$
\begin{aligned}
L(C, X) &= \sum_i p_i \ell_i \\
&= \sum_i p_i \log_2 \left(\frac{1}{z q_i} \right) \\
&= \sum_i p_i \log_2 \left(\frac{1}{z p_i} \frac{p_i}{q_i} \right) \\
&= \sum_i p_i \left[\log_2 \left(\frac{1}{p_i} \right) + \log_2 \left(\frac{p_i}{q_i} \right) + \log_2 \left(\frac{1}{z} \right) \right]
\end{aligned}
$$

Minimising Expected Code Length

Suppose we use code C with lengths $\ell = \{\ell_1, \ldots, \ell_I\}$ and corresponding probabilities $\mathbf{q} = \{q_1, \ldots, q_I\}$ with $q_i = \frac{1}{z} 2^{-\ell_i}$. Then,

$$
\begin{aligned}
L(C, X) &= \sum_i p_i \ell_i \\
&= \sum_i p_i \log_2 \left(\frac{1}{z q_i} \right) \\
&= \sum_i p_i \log_2 \left(\frac{1}{z p_i} \frac{p_i}{q_i} \right) \\
&= \sum_i p_i \left[\log_2 \left(\frac{1}{p_i} \right) + \log_2 \left(\frac{p_i}{q_i} \right) + \log_2 \left(\frac{1}{z} \right) \right] \\
&= \sum_i p_i \log_2 \frac{1}{p_i} + \sum_i p_i \log_2 \frac{p_i}{q_i} + \log_2 \left(\frac{1}{z} \right) \sum_i p_i
\end{aligned}
$$

Minimising Expected Code Length

Suppose we use code C with lengths $\ell = \{\ell_1, \ldots, \ell_I\}$ and corresponding probabilities $\mathbf{q} = \{q_1, \ldots, q_I\}$ with $q_i = \frac{1}{z} 2^{-\ell_i}$. Then,

$$
\begin{aligned}
L(C, X) &= \sum_i p_i \ell_i \\
&= \sum_i p_i \log_2 \left(\frac{1}{z q_i} \right) \\
&= \sum_i p_i \log_2 \left(\frac{1}{z p_i} \frac{p_i}{q_i} \right) \\
&= \sum_i p_i \left[\log_2 \left(\frac{1}{p_i} \right) + \log_2 \left(\frac{p_i}{q_i} \right) + \log_2 \left(\frac{1}{z} \right) \right] \\
&= \sum_i p_i \log_2 \frac{1}{p_i} + \sum_i p_i \log_2 \frac{p_i}{q_i} + \log_2 \left(\frac{1}{z} \right) \sum_i p_i \\
&= \quad H(X) \quad + \quad D_{\mathrm{KL}}(\mathbf{p} \| \mathbf{q}) \quad + \quad \log_2(1/z) \quad \cdot \quad 1
\end{aligned}
$$

Minimising Expected Code Length

So if $\mathbf{q} = \{q_1, \ldots, q_I\}$ are the probabilities for the code lengths of C then under ensemble X with probabilities $\mathbf{p} = \{p_1, \ldots, p_I\}$

$$L(C, X) = H(X) + D_{\mathrm{KL}}(p \| q) + \log_2 \frac{1}{z}$$

Minimising Expected Code Length

So if $\mathbf{q} = \{q_1, \dots, q_I\}$ are the probabilities for the code lengths of C then under ensemble X with probabilities $\mathbf{p} = \{p_1, \dots, p_I\}$

$$L(C, X) = H(X) + D_{\text{KL}}(p \| q) + \log_2 \frac{1}{z}$$

Thus, $L(C, X)$ is minimal (and equal to the entropy $H(X)$) if we can choose code lengths so that $D_{\text{KL}}(\mathbf{p} \| \mathbf{q}) = 0$ and $\log_2 \frac{1}{z} = 0$

Minimising Expected Code Length

So if $\mathbf{q} = \{q_1, \ldots, q_I\}$ are the probabilities for the code lengths of C then under ensemble X with probabilities $\mathbf{p} = \{p_1, \ldots, p_I\}$

$$L(C, X) = H(X) + D_{\mathrm{KL}}(p \| q) + \log_2 \frac{1}{z}$$

Thus, $L(C, X)$ is minimal (and equal to the entropy $H(X)$) if we can choose code lengths so that $D_{\mathrm{KL}}(\mathbf{p} \| \mathbf{q}) = 0$ and $\log_2 \frac{1}{z} = 0$

But the *relative entropy* $D_{\mathrm{KL}}(\mathbf{p} \| \mathbf{q}) \geq 0$ with $D_{\mathrm{KL}}(\mathbf{p} \| \mathbf{q}) = 0$ iff $\mathbf{q} = \mathbf{p}$ (Gibb's inequality)

Minimising Expected Code Length

So if $\mathbf{q} = \{q_1, \ldots, q_I\}$ are the probabilities for the code lengths of C then under ensemble X with probabilities $\mathbf{p} = \{p_1, \ldots, p_I\}$

$$L(C, X) = H(X) + D_{\mathrm{KL}}(p \| q) + \log_2 \frac{1}{z}$$

Thus, $L(C, X)$ is minimal (and equal to the entropy $H(X)$) if we can choose code lengths so that $D_{\mathrm{KL}}(\mathbf{p} \| \mathbf{q}) = 0$ and $\log_2 \frac{1}{z} = 0$

But the *relative entropy* $D_{\mathrm{KL}}(\mathbf{p} \| \mathbf{q}) \geq 0$ with $D_{\mathrm{KL}}(\mathbf{p} \| \mathbf{q}) = 0$ iff $\mathbf{q} = \mathbf{p}$ (Gibb's inequality)

For $\mathbf{q} = \mathbf{p}$, we have $z \stackrel{\text{def}}{=} \sum_i q_i = \sum_i p_i = 1$ and so $\log_2 \frac{1}{z} = 0$

Entropy as a Lower Bound on Expected Length

We have shown that for a code C with lengths corresponding to $\mathbf{q}$,

$$L(C, X) \geq H(X)$$

with equality only when C has code lengths $\ell_i = \log_2 \frac{1}{p_i}$

Once again, the entropy determines a lower bound on how much compression is possible

- $L(C, X)$ refers to *average* compression
- Individual message length could be bigger than the entropy

Shannon Codes

If we pick lengths $\ell_i = \log_2 \frac{1}{p_i}$, we get optimal expected code lengths

But $\log_2 \frac{1}{p_i}$ is not always an integer—a problem for code lengths!

Shannon Codes

If we pick lengths $\ell_i = \log_2 \frac{1}{p_i}$, we get optimal expected code lengths

But $\log_2 \frac{1}{p_i}$ is not always an integer—a problem for code lengths!

Shannon Code

Given an ensemble X with $\mathcal{P}_X = \{p_1, \ldots, p_I\}$ define codelengths $\ell = \{\ell_1, \ldots, \ell_I\}$ by

$$\ell_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil \geq \log_2 \frac{1}{p_i}.$$

A code C is called a **Shannon code** if it has codelengths ℓ.

Here $\lceil x \rceil$ is "smallest integer not smaller than x". e.g., $\lceil 2.1 \rceil = 3$, $\lceil 5 \rceil = 5$.

This gives us code lengths that are "closest" to $\log_2 \frac{1}{p_i}$

Shannon Codes: Examples

Examples:

1. If $\mathcal{P}_X = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$ then $\ell = \{1, 2, 2\}$ so $C = \{0, 10, 11\}$ is a Shannon code (in fact, this code has *optimal* length)

Shannon Codes: Examples

Examples:

1. If $\mathcal{P}_X = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$ then $\ell = \{1, 2, 2\}$ so $C = \{0, 10, 11\}$ is a Shannon code (in fact, this code has *optimal* length)

2. If $\mathcal{P}_X = \{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$ then $\ell = \{2, 2, 2\}$ with Shannon code $C = \{00, 10, 11\}$ (or $C = \{01, 10, 11\} \ldots$)

Source Coding Theorem for Symbol Codes

Shannon codes let us prove the following:

Source Coding Theorem for Symbol Codes

Shannon codes let us prove the following:

Source Coding Theorem for Symbol Codes

For any ensemble X, there exists a prefix code C such that

$$H(X) \leq L(C, X) < H(X) + 1.$$

In particular, **Shannon codes** C — those with lengths $\ell_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil$ — have *expected code length within 1 bit of the entropy*.

Source Coding Theorem for Symbol Codes

Shannon codes let us prove the following:

Source Coding Theorem for Symbol Codes

For any ensemble X, there exists a prefix code C such that

$$H(X) \leq L(C, X) < H(X) + 1.$$

In particular, **Shannon codes** C — those with lengths $\ell_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil$ — have *expected code length within 1 bit of the entropy*.

Entropy also gives a guideline upper bound of compression

Shannon Codes

Since $\lceil x \rceil$ is the *smallest* integer bigger than or equal to x it must be the case that $x \leq \lceil x \rceil < x + 1$.

Shannon Codes

Since $\lceil x \rceil$ is the *smallest* integer bigger than or equal to x it must be the case that $x \leq \lceil x \rceil < x + 1$.

Therefore, if we create a Shannon code C for $\mathbf{p} = \{p_1, \ldots, p_I\}$ with $\ell_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil < \log_2 \frac{1}{p_i} + 1$ it will satisfy

$$
\begin{aligned}
L(C, X) = \sum_i p_i \ell_i < \sum_i p_i \log_2 \frac{1}{p_i} + 1 &= \sum_i p_i \log_2 \frac{1}{p_i} + \sum_i p_i \\
&= H(X) + 1
\end{aligned}
$$

Shannon Codes

Since $\lceil x \rceil$ is the *smallest* integer bigger than or equal to x it must be the case that $x \leq \lceil x \rceil < x + 1$.

Therefore, if we create a Shannon code C for $\mathbf{p} = \{p_1, \ldots, p_I\}$ with $\ell_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil < \log_2 \frac{1}{p_i} + 1$ it will satisfy

$$L(C, X) = \sum_i p_i \ell_i < \sum_i p_i \log_2 \frac{1}{p_i} + 1 = \sum_i p_i \log_2 \frac{1}{p_i} + \sum_i p_i$$
$$= H(X) + 1$$

Furthermore, since $\ell_i \geq -\log_2 p_i$ we have $2^{-\ell_i} \leq 2^{\log_2 p_i} = p_i$, so $\sum_i 2^{-\ell_i} \leq \sum_i p_i = 1$. By Kraft there is a *prefix code* with lengths ℓ_i

Shannon Codes

Since $\lceil x \rceil$ is the *smallest* integer bigger than or equal to x it must be the case that $x \leq \lceil x \rceil < x + 1$.

Therefore, if we create a Shannon code C for $\mathbf{p} = \{p_1, \ldots, p_I\}$ with $\ell_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil < \log_2 \frac{1}{p_i} + 1$ it will satisfy

$$L(C, X) = \textstyle\sum_i p_i \ell_i < \sum_i p_i \log_2 \frac{1}{p_i} + 1 = \sum_i p_i \log_2 \frac{1}{p_i} + \sum_i p_i$$
$$= H(X) + 1$$

Furthermore, since $\ell_i \geq -\log_2 p_i$ we have $2^{-\ell_i} \leq 2^{\log_2 p_i} = p_i$, so $\sum_i 2^{-\ell_i} \leq \sum_i p_i = 1$. By Kraft there is a *prefix code* with lengths ℓ_i

Examples:

1. If $\mathcal{P}_X = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$ then $\ell = \{1, 2, 2\}$ and $H(X) = \frac{3}{2} = L(C, X)$

Shannon Codes

Since $\lceil x \rceil$ is the *smallest* integer bigger than or equal to x it must be the case that $x \le \lceil x \rceil < x + 1$.

Therefore, if we create a Shannon code C for $\mathbf{p} = \{p_1, \ldots, p_I\}$ with $\ell_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil < \log_2 \frac{1}{p_i} + 1$ it will satisfy

$$L(C, X) = \sum_i p_i \ell_i < \sum_i p_i \log_2 \frac{1}{p_i} + 1 = \sum_i p_i \log_2 \frac{1}{p_i} + \sum_i p_i$$
$$= H(X) + 1$$

Furthermore, since $\ell_i \ge -\log_2 p_i$ we have $2^{-\ell_i} \le 2^{\log_2 p_i} = p_i$, so $\sum_i 2^{-\ell_i} \le \sum_i p_i = 1$. By Kraft there is a *prefix code* with lengths ℓ_i

Examples:

1. If $\mathcal{P}_X = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$ then $\ell = \{1, 2, 2\}$ and $H(X) = \frac{3}{2} = L(C, X)$
2. If $\mathcal{P}_X = \{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$ then $\ell = \{2, 2, 2\}$ and $H(X) = \log_2 3 \approx 1.58 \le L(C, X) = 2 \le 2.58 \approx H(X) + 1$

The Source Coding Theorem for Symbol Codes

The previous arguments have established:

Source Coding Theorem for Symbol Codes

For any ensemble X there exists a *prefix code* C such that

$$H(X) \leq L(C, X) < H(X) + 1.$$

In particular, **Shannon codes** C — those with lengths $\ell_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil$ — have *expected code length within 1 bit of the entropy*.

The Source Coding Theorem for Symbol Codes

The previous arguments have established:

Source Coding Theorem for Symbol Codes

For any ensemble X there exists a *prefix code* C such that

$$H(X) \leq L(C, X) < H(X) + 1.$$

In particular, **Shannon codes** C — those with lengths $\ell_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil$ — have *expected code length within 1 bit of the entropy*.

This is good, but is it **optimal**?

Shannon codes are suboptimal

Example: Consider $p_1 = 0.0001$ and $p_2 = 0.9999$. (Note $H(X) \approx 0.0013$)

- The Shannon code C has lengths $\ell_1 = \lceil \log_2 10000 \rceil = 14$ and $\ell_2 = \lceil \log_2 \frac{10000}{9999} \rceil = 1$
- The expected length is $L(C, X) = 14 \times 0.0001 + 1 \times 0.9999 = 1.0013$
- But clearly $C' = \{0, 1\}$ is a prefix code and $L(C', X) = 1$

Shannon codes are suboptimal

Example: Consider $p_1 = 0.0001$ and $p_2 = 0.9999$. (Note $H(X) \approx 0.0013$)

- The Shannon code C has lengths $\ell_1 = \lceil \log_2 10000 \rceil = 14$ and $\ell_2 = \lceil \log_2 \frac{10000}{9999} \rceil = 1$
- The expected length is $L(C, X) = 14 \times 0.0001 + 1 \times 0.9999 = 1.0013$
- But clearly $C' = \{0, 1\}$ is a prefix code and $L(C', X) = 1$

Shannon codes do not necessarily have **smallest** expected length

This is perhaps disappointing, as these codes were constructed very naturally from the theorem

- Fortunately, there is another simple code that **is** provably optimal

Constructing a Huffman Code

Huffman Coding is a procedure for making provably optimal prefix codes

It assigns the longest codewords to least probable symbols

Basic algorithm:

- Take the two least probable symbols in the alphabet
- Prepend bits 0 and 1 to current codewords of symbols
- Combine these two symbols into a single “meta-symbol”
- Repeat

Huffman Coding: Example 1

Start with $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and $\mathcal{P} = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$

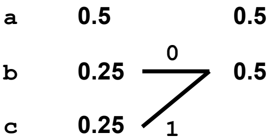
Step 1

a	**0.5**
b	**0.25**
c	**0.25**

Huffman Coding: Example 1

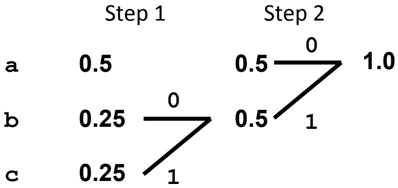
Start with $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and $\mathcal{P} = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$

Step 1



Huffman Coding: Example 1

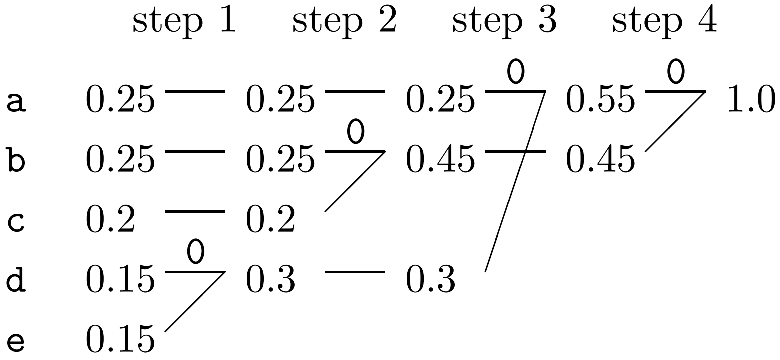
Start with $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and $\mathcal{P} = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$



Now we read off the labelling implied by path from the last meta-symbol to each of the original symbols: $C = \{0, 10, 11\}$

Huffman Coding: Example 2

$\mathcal{A}_X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$ and $\mathcal{P}_X = \{0.25, 0.25, 0.2, 0.15, 0.15\}$



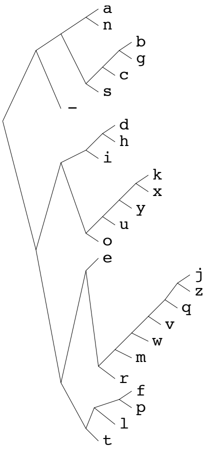
From Example 5.15 of MacKay

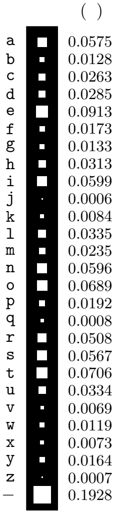
$C = \{00, 10, 11, 010, 011\}$

Huffman Coding: Example 3

English letters – Monogram statistics

i	i	$\log_2 \frac{1}{p_i}$	i	$(\ i)$
a	0.0575	4.1	4	0000
b	0.0128	6.3	6	001000
c	0.0263	5.2	5	00101
d	0.0285	5.1	5	10000
e	0.0913	3.5	4	1100
f	0.0173	5.9	6	111000
g	0.0133	6.2	6	001001
h	0.0313	5.0	5	10001
i	0.0599	4.1	4	1001
j	0.0006	10.7	10	1101000000
k	0.0084	6.9	7	1010000
l	0.0335	4.9	5	11101
m	0.0235	5.4	6	110101
n	0.0596	4.1	4	0001
o	0.0689	3.9	4	1011
p	0.0192	5.7	6	111001
q	0.0008	10.3	9	110100001
r	0.0508	4.3	5	11011
s	0.0567	4.1	4	0011
t	0.0706	3.8	4	1111
u	0.0334	4.9	5	10101
v	0.0069	7.2	8	11010001
w	0.0119	6.4	7	1101001
x	0.0073	7.1	7	1010001
y	0.0164	5.9	6	101001
z	0.0007	10.4	10	1101000001
–	0.1928	2.4	2	01





Huffman Coding: Formally

HUFFMAN($\mathcal{A}, \mathcal{P}$):

1. If $|\mathcal{A}| = 2$ return $C = \{0, 1\}$; else
2. Let $a, a' \in \mathcal{A}$ be *least probable* symbols.
3. Let $\mathcal{A}' = \mathcal{A} - \{a, a'\} \cup \{aa'\}$
4. Let $\mathcal{P}' = \mathcal{P} - \{p_a, p_{a'}\} \cup \{p_{aa'}\}$ where $p_{aa'} = p_a + p_{a'}$
5. Compute $C' =$ HUFFMAN($\mathcal{A}', \mathcal{P}'$)
6. Define C by
   - $c(a) = c'(aa')0$
   - $c(a') = c'(aa')1$
   - $c(x) = c'(x)$ for $x \in \mathcal{A}'$
7. Return C

Huffman Coding: Example 1

Start with $\mathcal{A} = \{\texttt{a}, \texttt{b}, \texttt{c}\}$ and $\mathcal{P} = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):
  - b and c are least probable with $p_{\texttt{a}} = p_{\texttt{b}} = \frac{1}{4}$

Huffman Coding: Example 1

Start with $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and $\mathcal{P} = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):
  - $\mathtt{b}$ and $\mathtt{c}$ are least probable with $p_{\mathtt{a}} = p_{\mathtt{b}} = \frac{1}{4}$
  - $\mathcal{A}' = \{\mathtt{a}, \mathbf{bc}\}$ and $\mathcal{P}' = \{\frac{1}{2}, \mathbf{\frac{1}{2}}\}$

Huffman Coding: Example 1

Start with $\mathcal{A} = \{\texttt{a}, \texttt{b}, \texttt{c}\}$ and $\mathcal{P} = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):
  - $\texttt{b}$ and $\texttt{c}$ are least probable with $p_{\texttt{a}} = p_{\texttt{b}} = \frac{1}{4}$
  - $\mathcal{A}' = \{\texttt{a}, \mathbf{bc}\}$ and $\mathcal{P}' = \{\frac{1}{2}, \mathbf{\frac{1}{2}}\}$
  - Call HUFFMAN($\mathcal{A}', \mathcal{P}'$):
    - $|\mathcal{A}| = |\{\texttt{a}, \texttt{bc}\}| = 2$
    - Return code with $c'(\texttt{a}) = 0$, $c'(\texttt{bc}) = 1$

Huffman Coding: Example 1

Start with $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and $\mathcal{P} = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):
  - $\mathtt{b}$ and $\mathtt{c}$ are least probable with $p_\mathtt{a} = p_\mathtt{b} = \frac{1}{4}$
  - $\mathcal{A}' = \{\mathtt{a}, \mathbf{bc}\}$ and $\mathcal{P}' = \{\frac{1}{2}, \mathbf{\frac{1}{2}}\}$
  - Call HUFFMAN($\mathcal{A}', \mathcal{P}'$):
    - $|\mathcal{A}| = |\{\mathtt{a}, \mathtt{bc}\}| = 2$
    - Return code with $c'(\mathtt{a}) = 0$, $c'(\mathtt{bc}) = 1$
  - Define
    - $c(\mathtt{b}) = c'(\mathtt{bc})0 = \mathbf{1}0$
    - $c(\mathtt{c}) = c'(\mathtt{bc})1 = \mathbf{1}1$
    - $c(\mathtt{a}) = c'(\mathtt{a}) = 0$

Huffman Coding: Example 1

Start with $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and $\mathcal{P} = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):
  - $\mathtt{b}$ and $\mathtt{c}$ are least probable with $p_\mathtt{a} = p_\mathtt{b} = \frac{1}{4}$
  - $\mathcal{A}' = \{\mathtt{a}, \mathbf{bc}\}$ and $\mathcal{P}' = \{\frac{1}{2}, \mathbf{\frac{1}{2}}\}$
  - Call HUFFMAN($\mathcal{A}', \mathcal{P}'$):
    - $|\mathcal{A}| = |\{\mathtt{a}, \mathtt{bc}\}| = 2$
    - Return code with $c'(\mathtt{a}) = 0$, $c'(\mathtt{bc}) = 1$
  - Define
    - $c(\mathtt{b}) = c'(\mathtt{bc})0 = \mathbf{1}0$
    - $c(\mathtt{c}) = c'(\mathtt{bc})1 = \mathbf{1}1$
    - $c(\mathtt{a}) = c'(\mathtt{a}) = 0$
  - Return $C = \{0, 10, 11\}$

Huffman Coding: Example 1

Start with $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and $\mathcal{P} = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):
  - $\mathtt{b}$ and $\mathtt{c}$ are least probable with $p_{\mathtt{a}} = p_{\mathtt{b}} = \frac{1}{4}$
  - $\mathcal{A}' = \{\mathtt{a}, \mathbf{bc}\}$ and $\mathcal{P}' = \{\frac{1}{2}, \mathbf{\frac{1}{2}}\}$
  - Call HUFFMAN($\mathcal{A}', \mathcal{P}'$):
    - $|\mathcal{A}| = |\{\mathtt{a}, \mathtt{bc}\}| = 2$
    - Return code with $c'(\mathtt{a}) = 0$, $c'(\mathtt{bc}) = 1$
  - Define
    - $c(\mathtt{b}) = c'(\mathtt{bc})0 = \mathbf{1}0$
    - $c(\mathtt{c}) = c'(\mathtt{bc})1 = \mathbf{1}1$
    - $c(\mathtt{a}) = c'(\mathtt{a}) = 0$
  - Return $C = \{0, 10, 11\}$

Huffman Coding: Example 1

Start with $\mathcal{A} = \{\texttt{a}, \texttt{b}, \texttt{c}\}$ and $\mathcal{P} = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):
  - $\texttt{b}$ and $\texttt{c}$ are least probable with $p_\texttt{a} = p_\texttt{b} = \frac{1}{4}$
  - $\mathcal{A}' = \{\texttt{a}, \mathbf{bc}\}$ and $\mathcal{P}' = \{\frac{1}{2}, \mathbf{\frac{1}{2}}\}$
  - Call HUFFMAN($\mathcal{A}', \mathcal{P}'$):
    - $|\mathcal{A}| = |\{\texttt{a}, \texttt{bc}\}| = 2$
    - Return code with $c'(\texttt{a}) = 0$, $c'(\texttt{bc}) = 1$
  - Define
    - $c(\texttt{b}) = c'(\texttt{bc})0 = \mathbf{1}0$
    - $c(\texttt{c}) = c'(\texttt{bc})1 = \mathbf{1}1$
    - $c(\texttt{a}) = c'(\texttt{a}) = 0$
  - Return $C = \{0, 10, 11\}$

The constructed code has $L(C, X) = \frac{1}{2} \times 1 + \frac{1}{4} \times (2 + 2) = 1.5$.
The entropy is $H(X) = 1.5$.

Huffman Coding: Example 2

Start with $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$ and $\mathcal{P} = \{0.25, 0.25, 0.2, 0.15, 0.15\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):

Huffman Coding: Example 2

Start with $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$ and $\mathcal{P} = \{0.25, 0.25, 0.2, 0.15, 0.15\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):
  - $\mathcal{A}' = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathbf{de}\}$ and $\mathcal{P}' = \{0.25, 0.25, 0.2, \mathbf{0.3}\}$

Huffman Coding: Example 2

Start with $\mathcal{A} = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}\}$ and $\mathcal{P} = \{0.25, 0.25, 0.2, 0.15, 0.15\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):
  - $\mathcal{A}' = \{\texttt{a}, \texttt{b}, \texttt{c}, \textbf{de}\}$ and $\mathcal{P}' = \{0.25, 0.25, 0.2, \mathbf{0.3}\}$
  - Call HUFFMAN($\mathcal{A}', \mathcal{P}'$):

Huffman Coding: Example 2

Start with $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$ and $\mathcal{P} = \{0.25, 0.25, 0.2, 0.15, 0.15\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):
  - $\mathcal{A}' = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathbf{de}\}$ and $\mathcal{P}' = \{0.25, 0.25, 0.2, \mathbf{0.3}\}$
  - Call HUFFMAN($\mathcal{A}', \mathcal{P}'$):
    - $\mathcal{A}'' = \{\mathtt{a}, \mathbf{bc}, \mathtt{de}\}$ and $\mathcal{P}'' = \{0.25, \mathbf{0.45}, 0.3\}$

Huffman Coding: Example 2

Start with $\mathcal{A} = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}\}$ and $\mathcal{P} = \{0.25, 0.25, 0.2, 0.15, 0.15\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):
  - $\mathcal{A}' = \{\texttt{a}, \texttt{b}, \texttt{c}, \textbf{de}\}$ and $\mathcal{P}' = \{0.25, 0.25, 0.2, \mathbf{0.3}\}$
  - Call HUFFMAN($\mathcal{A}', \mathcal{P}'$):
    - $\mathcal{A}'' = \{\texttt{a}, \textbf{bc}, \texttt{de}\}$ and $\mathcal{P}'' = \{0.25, \mathbf{0.45}, 0.3\}$
    - Call HUFFMAN($\mathcal{A}'', \mathcal{P}''$):
      - $\mathcal{A}''' = \{\textbf{ade}, \texttt{bc}\}$ and $\mathcal{P}''' = \{\mathbf{0.55}, 0.45\}$
      - Return $c'''(\texttt{ade}) = 0$, $c'''(\texttt{bc}) = 1$

Huffman Coding: Example 2

Start with $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$ and $\mathcal{P} = \{0.25, 0.25, 0.2, 0.15, 0.15\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):
  - $\mathcal{A}' = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathbf{de}\}$ and $\mathcal{P}' = \{0.25, 0.25, 0.2, \mathbf{0.3}\}$
  - Call HUFFMAN($\mathcal{A}', \mathcal{P}'$):
    - $\mathcal{A}'' = \{\mathtt{a}, \mathbf{bc}, \mathtt{de}\}$ and $\mathcal{P}'' = \{0.25, \mathbf{0.45}, 0.3\}$
    - Call HUFFMAN($\mathcal{A}'', \mathcal{P}''$):
      - $\mathcal{A}''' = \{\mathbf{ade}, \mathtt{bc}\}$ and $\mathcal{P}''' = \{\mathbf{0.55}, 0.45\}$
      - Return $c'''(\mathtt{ade}) = 0$, $c'''(\mathtt{bc}) = 1$
    - Return $c''(\mathtt{a}) = \mathbf{0}0$, $c''(\mathtt{bc}) = 1$, $c''(\mathtt{de}) = \mathbf{0}1$

Huffman Coding: Example 2

Start with $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$ and $\mathcal{P} = \{0.25, 0.25, 0.2, 0.15, 0.15\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):
  - $\mathcal{A}' = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathbf{de}\}$ and $\mathcal{P}' = \{0.25, 0.25, 0.2, \mathbf{0.3}\}$
  - Call HUFFMAN($\mathcal{A}', \mathcal{P}'$):
    - $\mathcal{A}'' = \{\mathtt{a}, \mathbf{bc}, \mathtt{de}\}$ and $\mathcal{P}'' = \{0.25, \mathbf{0.45}, 0.3\}$
    - Call HUFFMAN($\mathcal{A}'', \mathcal{P}''$):
      - $\mathcal{A}''' = \{\mathbf{ade}, \mathtt{bc}\}$ and $\mathcal{P}''' = \{\mathbf{0.55}, 0.45\}$
      - Return $c'''(\mathtt{ade}) = 0$, $c'''(\mathtt{bc}) = 1$
    - Return $c''(\mathtt{a}) = \mathbf{0}0$, $c''(\mathtt{bc}) = 1$, $c''(\mathtt{de}) = \mathbf{0}1$
  - Return $c'(\mathtt{a}) = 00$, $c'(\mathtt{b}) = 1\mathbf{0}$, $c'(\mathtt{c}) = 1\mathbf{1}$, $c'(\mathtt{de}) = 01$

Huffman Coding: Example 2

Start with $\mathcal{A} = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}\}$ and $\mathcal{P} = \{0.25, 0.25, 0.2, 0.15, 0.15\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):
  - $\mathcal{A}' = \{\texttt{a}, \texttt{b}, \texttt{c}, \textbf{de}\}$ and $\mathcal{P}' = \{0.25, 0.25, 0.2, \mathbf{0.3}\}$
  - Call HUFFMAN($\mathcal{A}', \mathcal{P}'$):
    - $\mathcal{A}'' = \{\texttt{a}, \textbf{bc}, \texttt{de}\}$ and $\mathcal{P}'' = \{0.25, \mathbf{0.45}, 0.3\}$
    - Call HUFFMAN($\mathcal{A}'', \mathcal{P}''$):
      - $\mathcal{A}''' = \{\textbf{ade}, \texttt{bc}\}$ and $\mathcal{P}''' = \{\mathbf{0.55}, 0.45\}$
      - Return $c'''(\texttt{ade}) = 0$, $c'''(\texttt{bc}) = 1$
    - Return $c''(\texttt{a}) = \mathbf{0}0$, $c''(\texttt{bc}) = 1$, $c''(\texttt{de}) = \mathbf{0}1$
  - Return $c'(\texttt{a}) = 00$, $c'(\texttt{b}) = 1\mathbf{0}$, $c'(\texttt{c}) = 1\mathbf{1}$, $c'(\texttt{de}) = 01$
- Return $c(\texttt{a}) = 00$, $c(\texttt{b}) = 10$, $c(\texttt{c}) = 11$, $c(\texttt{d}) = \mathbf{01}0$, $c(\texttt{e}) = \mathbf{01}1$

Huffman Coding: Example 2

Start with $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$ and $\mathcal{P} = \{0.25, 0.25, 0.2, 0.15, 0.15\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):
  - $\mathcal{A}' = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathbf{de}\}$ and $\mathcal{P}' = \{0.25, 0.25, 0.2, \mathbf{0.3}\}$
  - Call HUFFMAN($\mathcal{A}', \mathcal{P}'$):
    - $\mathcal{A}'' = \{\mathtt{a}, \mathbf{bc}, \mathtt{de}\}$ and $\mathcal{P}'' = \{0.25, \mathbf{0.45}, 0.3\}$
    - Call HUFFMAN($\mathcal{A}'', \mathcal{P}''$):
      - $\mathcal{A}''' = \{\mathbf{ade}, \mathtt{bc}\}$ and $\mathcal{P}''' = \{\mathbf{0.55}, 0.45\}$
      - Return $c'''(\mathtt{ade}) = 0$, $c'''(\mathtt{bc}) = 1$
    - Return $c''(\mathtt{a}) = \mathbf{0}0$, $c''(\mathtt{bc}) = 1$, $c''(\mathtt{de}) = \mathbf{0}1$
  - Return $c'(\mathtt{a}) = 00$, $c'(\mathtt{b}) = 1\mathbf{0}$, $c'(\mathtt{c}) = 1\mathbf{1}$, $c'(\mathtt{de}) = 01$
- Return $c(\mathtt{a}) = 00$, $c(\mathtt{b}) = 10$, $c(\mathtt{c}) = 11$, $c(\mathtt{d}) = 01\mathbf{0}$, $c(\mathtt{e}) = 01\mathbf{1}$

Huffman Coding: Example 2

Start with $\mathcal{A} = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}\}$ and $\mathcal{P} = \{0.25, 0.25, 0.2, 0.15, 0.15\}$

- HUFFMAN($\mathcal{A}, \mathcal{P}$):
  - $\mathcal{A}' = \{\texttt{a}, \texttt{b}, \texttt{c}, \textbf{de}\}$ and $\mathcal{P}' = \{0.25, 0.25, 0.2, \mathbf{0.3}\}$
  - Call HUFFMAN($\mathcal{A}', \mathcal{P}'$):
    - $\mathcal{A}'' = \{\texttt{a}, \textbf{bc}, \texttt{de}\}$ and $\mathcal{P}'' = \{0.25, \mathbf{0.45}, 0.3\}$
    - Call HUFFMAN($\mathcal{A}'', \mathcal{P}''$):
      - $\mathcal{A}''' = \{\textbf{ade}, \texttt{bc}\}$ and $\mathcal{P}''' = \{\mathbf{0.55}, 0.45\}$
      - Return $c'''(\texttt{ade}) = 0$, $c'''(\texttt{bc}) = 1$
    - Return $c''(\texttt{a}) = \mathbf{0}0$, $c''(\texttt{bc}) = 1$, $c''(\texttt{de}) = \mathbf{0}1$
  - Return $c'(\texttt{a}) = 00$, $c'(\texttt{b}) = \mathbf{1}0$, $c'(\texttt{c}) = \mathbf{1}1$, $c'(\texttt{de}) = 01$
- Return $c(\texttt{a}) = 00$, $c(\texttt{b}) = 10$, $c(\texttt{c}) = 11$, $c(\texttt{d}) = \mathbf{01}0$, $c(\texttt{e}) = \mathbf{01}1$

The constructed code is $C = \{00, 10, 11, 010, 011\}$.
It has $L(C, X) = 2 \times (0.25 + 0.25 + 0.2) + 3 \times (0.15 + 0.15) = 2.3$.
Note that $H(X) \approx 2.29$.

Huffman Coding in Python

See full example code with examples at:

`https://gist.github.com/mreid/fdf6353ec39d050e972b`

```python
def huffman(p):
    '''Return a Huffman code for an ensemble with distribution p.'''
    assert(sum(p.values()) == 1.0) # Ensure probabilities sum to 1

    # Base case of only two symbols, assign 0 or 1 arbitrarily
    if(len(p) == 2):
        return dict(zip(p.keys(), ['0', '1']))

    # Create a new distribution by merging lowest prob. pair
    p_prime = p.copy()
    a1, a2 = lowest_prob_pair(p)
    p1, p2 = p_prime.pop(a1), p_prime.pop(a2)
    p_prime[a1 + a2] = p1 + p2

    # Recurse and construct code on new distribution
    c = huffman(p_prime)
    ca1a2 = c.pop(a1 + a2)
    c[a1], c[a2] = ca1a2 + '0', ca1a2 + '1'

    return c
```

Advantages of Huffman coding

- Produces prefix codes automatically (by design)
- Algorithm is simple and efficient
- Huffman Codes are provably optimal [Exercise 5.16 (MacKay)]

Advantages of Huffman coding

- Produces prefix codes automatically (by design)
- Algorithm is simple and efficient
- Huffman Codes are provably optimal [Exercise 5.16 (MacKay)]

  If C_{Huff} is a Huffman code, then for any other uniquely decodable code C',

  $$L(C_{\text{Huff}}, X) \leq L(C', X)$$

  It follows that

  $$H(X) \leq L(C_{\text{Huff}}, X) < H(X) + 1$$

Disadvantages of Huffman coding

- Assumes a fixed distribution of symbols

- The extra bit in the SCT
  - If $H(X)$ is large – not a problem
  - If $H(X)$ is small (e.g., $\sim$ 1 bit for English) codes are $2\times$ optimal

Disadvantages of Huffman coding

- Assumes a fixed distribution of symbols
- The extra bit in the SCT
  - If $H(X)$ is large – not a problem
  - If $H(X)$ is small (e.g., $\sim$ 1 bit for English) codes are $2\times$ optimal

Huffman codes are the best possible symbol code
but symbol coding is not always the best type of code

Next Time: Stream Codes!

Summary

Key Concepts:

1. The expected code length $L(C, X) = \sum_i p_i \ell_i$
2. Probabilities and codelengths are interchangeable
   $q_i = 2^{-\ell_i} \iff \ell_i = \log_2 \frac{1}{q_i}$
3. Relative entropy $D_{\mathrm{KL}}(\mathbf{p}\|\mathbf{q})$ measures excess bits over the entropy $H(X)$ for using the wrong code $\mathbf{q}$ for probabilities $\mathbf{p}$
4. The Source Coding Theorem for symbol codes: There exists prefix (Shannon) code C for ensemble X with $\ell_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil$ so that

$$H(X) \leq L(C, X) \leq H(X) + 1$$

5. Huffman codes are optimal symbol codes

Reading:

- §5.3-5.7 of MacKay
- §5.3-5.4, §5.6 & §5.8 of Cover & Thomas

Acknowledgement

These slides were originally developed by Professor Robert C. Williamson.