Information and Coding Theory (CO349)

Representation of Information

Herbert Wiklicky
herbert@doc.ic.ac.uk

Department of Computing
Imperial College London

Autumn 2018

Information

What is Information?

Information resolves or allows to resolve uncertainty. The larger the resolved uncertainty is the more information we get/have/need/obtain.

One could also refer to this as surprise. This can be seen as the "opposite" or reciprocal of what we expect. If p is the probability of an event, then $\frac{1}{p}$ could be a surprise measure.

If there are two pieces of independent information then we would like that they "add up", but $\Pr(A \wedge B) = \Pr(A) \cdot \Pr(B)$.

Using the logarithm we get an additive information measure with fulfills: $\log(\frac{1}{p_1} \times \frac{1}{p_2}) = \log(\frac{1}{p_1}) + \log(\frac{1}{p_2})$.

Information

What is Information?

Information resolves or allows to resolve uncertainty. The larger the resolved uncertainty is the more information we get/have/need/obtain.

One could also refer to this as surprise. This can be seen as the "opposite" or reciprocal of what we expect. If p is the probability of an event, then $\frac{1}{p}$ could be a surprise measure.

If there are two pieces of independent information then we would like that they "add up", but $\Pr(A \wedge B) = \Pr(A) \cdot \Pr(B)$.

Using the logarithm we get an additive information measure with fulfills: $\log(\frac{1}{p_1} \times \frac{1}{p_2}) = \log(\frac{1}{p_1}) + \log(\frac{1}{p_2})$.

Information

What is Information?

Information resolves or allows to resolve uncertainty. The larger the resolved uncertainty is the more information we get/have/need/obtain.

One could also refer to this as surprise. This can be seen as the "opposite" or reciprocal of what we expect. If p is the probability of an event, then $\frac{1}{p}$ could be a surprise measure.

If there are two pieces of independent information then we would like that they "add up", but $\Pr(A \wedge B) = \Pr(A) \cdot \Pr(B)$.

Using the logarithm we get an additive information measure with fulfills: $\log(\frac{1}{p_1} \times \frac{1}{p_2}) = \log(\frac{1}{p_1}) + \log(\frac{1}{p_2})$.

Information

What is Information?

Information resolves or allows to resolve uncertainty. The larger the resolved uncertainty is the more information we get/have/need/obtain.

One could also refer to this as surprise. This can be seen as the "opposite" or reciprocal of what we expect. If p is the probability of an event, then $\frac{1}{p}$ could be a surprise measure.

If there are two pieces of independent information then we would like that they "add up", but $\Pr(A \wedge B) = \Pr(A) \cdot \Pr(B)$.

Using the logarithm we get an additive information measure with fulfills: $\log(\frac{1}{p_1} \times \frac{1}{p_2}) = \log(\frac{1}{p_1}) + \log(\frac{1}{p_2})$.

Sources

Definition

Let S be an alphabet. A source $(S, \mathbf{p})$ with probability distribution(s) $\mathbf{p} = \mathbf{p}^{(k)} = (p_1^{(k)}, p_2^{(k)}, \cdots, p_n^{(k)})$ emits a stream (sequence) $\sigma_1\sigma_2\sigma_3\cdots$ of symbols with probability

$$\Pr(\sigma_k = s_i) = p_i^{(k)}$$

Assume: Identical $\mathbf{p}^{(k)} = \mathbf{p}$, but not necessarily independent.

Definition

Let S be an alphabet. A memoryless source $(S, \mathbf{p})$ emits a stream (sequence) $\sigma_1\sigma_2\sigma_3\cdots$ such that for all k and ℓ:

$$\Pr(\sigma_k = s_i \wedge \sigma_\ell = s_j) = \Pr(\sigma_k = s_i)\Pr(\sigma_\ell = s_j)$$

Sources

Definition

Let S be an alphabet. A source $(S, \mathbf{p})$ with probability distribution(s) $\mathbf{p} = \mathbf{p}^{(k)} = (p_1^{(k)}, p_2^{(k)}, \cdots, p_n^{(k)})$ emits a stream (sequence) $\sigma_1\sigma_2\sigma_3\cdots$ of symbols with probability

$$\Pr(\sigma_k = s_i) = p_i^{(k)}$$

Assume: Identical $\mathbf{p}^{(k)} = \mathbf{p}$, but not necessarily independent.

Definition

Let S be an alphabet. A memoryless source $(S, \mathbf{p})$ emits a stream (sequence) $\sigma_1\sigma_2\sigma_3\cdots$ such that for all k and ℓ:

$$\Pr(\sigma_k = s_i \wedge \sigma_\ell = s_j) = \Pr(\sigma_k = s_i)\Pr(\sigma_\ell = s_j)$$

Sources

Definition

Let S be an alphabet. A source $(S, \mathbf{p})$ with probability distribution(s) $\mathbf{p} = \mathbf{p}^{(k)} = (p_1^{(k)}, p_2^{(k)}, \cdots, p_n^{(k)})$ emits a stream (sequence) $\sigma_1\sigma_2\sigma_3\cdots$ of symbols with probability

$$\Pr(\sigma_k = s_i) = p_i^{(k)}$$

Assume: Identical $\mathbf{p}^{(k)} = \mathbf{p}$, but not necessarily independent.

Definition

Let S be an alphabet. A memoryless source $(S, \mathbf{p})$ emits a stream (sequence) $\sigma_1\sigma_2\sigma_3\cdots$ such that for all k and ℓ:

$$\Pr(\sigma_k = s_i \wedge \sigma_\ell = s_j) = \Pr(\sigma_k = s_i)\Pr(\sigma_\ell = s_j)$$

Stochastic Processes

Definition

A (discrete time) stochastic process on a (state) space S is a collection of S-valued random variables X_i with $i \in \mathbb{N}$ or $\mathbb{Z}$.

With this one can, for example, investigate the probability of (finite) paths, traces, steames as:

$$\begin{aligned}
&\Pr(X_0 = s_{i_0} \wedge X_1 = s_{i_1} \wedge \ldots \wedge X_N = s_{i_N}) = \\
&= \Pr(X_0 = s_{i_0}) \cdot \\
&\quad \Pr(X_1 = s_{i_1} \mid X_0 = s_{i_0}) \cdot \\
&\quad \Pr(X_2 = s_{i_2} \mid X_0 = s_{i_0} \wedge X_1 = s_{i_1}) \cdot \cdots \cdot \\
&\quad \Pr(X_N = s_{i_N} \mid X_0 = s_{i_0} \wedge X_1 = s_{i_1} \ldots \wedge X_{N-1} = s_{i_{N-1}})
\end{aligned}$$

The issue is not just what the probability of each event is at any moment (in time) but also how this depends on the past.

Stochastic Processes

Definition

A (discrete time) stochastic process on a (state) space S is a collection of S-valued random variables X_i with $i \in \mathbb{N}$ or $\mathbb{Z}$.

With this one can, for example, investigate the probability of (finite) paths, traces, steames as:

$$\begin{aligned}
&\Pr(X_0 = s_{i_0} \wedge X_1 = s_{i_1} \wedge \ldots \wedge X_N = s_{i_N}) = \\
&= \quad \Pr(X_0 = s_{i_0}) \cdot \\
&\qquad \Pr(X_1 = s_{i_1} \mid X_0 = s_{i_0}) \cdot \\
&\qquad \Pr(X_2 = s_{i_2} \mid X_0 = s_{i_0} \wedge X_1 = s_{i_1}) \cdot \ldots \cdot \\
&\qquad \Pr(X_N = s_{i_N} \mid X_0 = s_{i_0} \wedge X_1 = s_{i_1} \ldots \wedge X_{N-1} = s_{i_{N-1}})
\end{aligned}$$

The issue is not just what the probability of each event is at any moment (in time) but also how this depends on the past.

Stochastic Processes

Definition

A (discrete time) stochastic process on a (state) space S is a collection of S-valued random variables X_i with $i \in \mathbb{N}$ or $\mathbb{Z}$.

With this one can, for example, investigate the probability of (finite) paths, traces, steames as:

$$\begin{aligned}
&\Pr(X_0 = s_{i_0} \wedge X_1 = s_{i_1} \wedge \ldots \wedge X_N = s_{i_N}) = \\
&= \quad \Pr(X_0 = s_{i_0}) \cdot \\
&\qquad \Pr(X_1 = s_{i_1} \mid X_0 = s_{i_0}) \cdot \\
&\qquad \Pr(X_2 = s_{i_2} \mid X_0 = s_{i_0} \wedge X_1 = s_{i_1}) \cdot \ldots \cdot \\
&\qquad \Pr(X_N = s_{i_N} \mid X_0 = s_{i_0} \wedge X_1 = s_{i_1} \ldots \wedge X_{N-1} = s_{i_{N-1}})
\end{aligned}$$

The issue is not just what the probability of each event is at any moment (in time) but also how this depends on the past.

Markov Chains

Definition

A (Discrete Time) Markov Chain is a stochastic process with $\forall N$

$$\Pr(X_N = s_{i_N} \mid X_0 = s_{i_0} \wedge X_1 = s_{i_1} \wedge \ldots \wedge X_{N-1} = s_{i_{N-1}}) = \\ = \Pr(X_N = s_{i_N} \mid X_{N-1} = s_{i_{N-1}}).$$

A DTMC can be specified by a stochastic (square) matrix

$$(\mathbf{P}_{ij}) = \Pr(X_N = s_i \mid X_{N-1} = s_j) \text{ with } \sum_{s_i \in S} \mathbf{P}_{ij} = 1.$$

Remember Nothing: Memoryless Process.
Remember Last State: Markov Chain.
Remember Everything: Stochastic Process.

cf. Applebaum; Norris: Markov Chains, CUP, 1997; etc.

Markov Chains

Definition

A (Discrete Time) Markov Chain is a stochastic process with $\forall N$

$$\Pr(X_N = s_{i_N} \mid X_0 = s_{i_0} \wedge X_1 = s_{i_1} \wedge \ldots \wedge X_{N-1} = s_{i_{N-1}}) = \\ = \Pr(X_N = s_{i_N} \mid X_{N-1} = s_{i_{N-1}}).$$

A DTMC can be specified by a stochastic (square) matrix

$$(\mathbf{P}_{ij}) = \Pr(X_N = s_i \mid X_{N-1} = s_j) \text{ with } \sum_{s_i \in S} \mathbf{P}_{ij} = 1.$$

Remember Nothing: Memoryless Process.

Remember Last State: Markov Chain.

Remember Everything: Stochastic Process.

cf. Applebaum; Norris: Markov Chains, CUP, 1997; etc.

Markov Chains

Definition

A (Discrete Time) Markov Chain is a stochastic process with $\forall N$

$$\Pr(X_N = s_{i_N} \mid X_0 = s_{i_0} \wedge X_1 = s_{i_1} \wedge \ldots \wedge X_{N-1} = s_{i_{N-1}}) = \\ = \Pr(X_N = s_{i_N} \mid X_{N-1} = s_{i_{N-1}}).$$

A DTMC can be specified by a stochastic (square) matrix

$$(\mathbf{P}_{ij}) = \Pr(X_N = s_i \mid X_{N-1} = s_j) \text{ with } \sum_{s_i \in S} \mathbf{P}_{ij} = 1.$$

Remember Nothing: Memoryless Process.

Remember Last State: Markov Chain.

Remember Everything: Stochastic Process.

cf. Applebaum; Norris: Markov Chains, CUP, 1997; etc.

Markov Chains

Definition

A (Discrete Time) Markov Chain is a stochastic process with $\forall N$

$$\Pr(X_N = s_{i_N} \mid X_0 = s_{i_0} \wedge X_1 = s_{i_1} \wedge \ldots \wedge X_{N-1} = s_{i_{N-1}}) = \\ = \Pr(X_N = s_{i_N} \mid X_{N-1} = s_{i_{N-1}}).$$

A DTMC can be specified by a stochastic (square) matrix

$$(\mathbf{P}_{ij}) = \Pr(X_N = s_i \mid X_{N-1} = s_j) \text{ with } \sum_{s_i \in S} \mathbf{P}_{ij} = 1.$$

Remember Nothing: Memoryless Process.

Remember Last State: Markov Chain.

Remember Everything: Stochastic Process.

cf. Applebaum; Norris: Markov Chains, CUP, 1997; etc.

Markov Chains

Definition

A (Discrete Time) Markov Chain is a stochastic process with $\forall N$

$$\Pr(X_N = s_{i_N} \mid X_0 = s_{i_0} \wedge X_1 = s_{i_1} \wedge \ldots \wedge X_{N-1} = s_{i_{N-1}}) = \\ = \Pr(X_N = s_{i_N} \mid X_{N-1} = s_{i_{N-1}}).$$

A DTMC can be specified by a stochastic (square) matrix

$$(\mathbf{P}_{ij}) = \Pr(X_N = s_i \mid X_{N-1} = s_j) \text{ with } \sum_{s_i \in S} \mathbf{P}_{ij} = 1.$$

Remember Nothing: Memoryless Process.

Remember Last State: Markov Chain.

Remember Everything: Stochastic Process.

cf. Applebaum; Norris: Markov Chains, CUP, 1997; etc.

Markov Chains

Definition

A (Discrete Time) Markov Chain is a stochastic process with $\forall N$

$$\Pr(X_N = s_{i_N} \mid X_0 = s_{i_0} \wedge X_1 = s_{i_1} \wedge \ldots \wedge X_{N-1} = s_{i_{N-1}}) = \\ = \Pr(X_N = s_{i_N} \mid X_{N-1} = s_{i_{N-1}}).$$

A DTMC can be specified by a stochastic (square) matrix

$$(\mathbf{P}_{ij}) = \Pr(X_N = s_i \mid X_{N-1} = s_j) \text{ with } \sum_{s_i \in S} \mathbf{P}_{ij} = 1.$$

Remember Nothing: Memoryless Process.

Remember Last State: Markov Chain.

Remember Everything: Stochastic Process.

cf. Applebaum; Norris: Markov Chains, CUP, 1997; etc.

Average Word Length

Consider a source $(S, \mathbf{p})$ and a code $c : S \to T^*$. Consider rthe stream of symbols we obtain by encoding $\{s_1, \cdots, s_m\}$ via c.

Definition

The average word-length L of a code $c : S \to T^*$ for a source $(S, \mathbf{p})$ is

$$L = p_1 l_1 + p_2 l_2 + \cdots p_m l_m = \sum_{i=1}^{m} p_i l_i = \mathbf{E}(l)$$

with $l_i = |c(s_i)|$ the length of codewords in T^* for each $s_i \in S$.

The question is what is the average "blow-up" of a message using a code c. The aim is to keep this as small as possible.

Average Word Length

Consider a source $(S, \mathbf{p})$ and a code $c : S \to T^*$. Consider rthe stream of symbols we obtain by encoding $\{s_1, \cdots, s_m\}$ via c.

Definition

The average word-length L of a code $c : S \to T^*$ for a source $(S, \mathbf{p})$ is

$$L = p_1 l_1 + p_2 l_2 + \cdots p_m l_m = \sum_{i=1}^{m} p_i l_i = \mathbf{E}(l)$$

with $l_i = |c(s_i)|$ the length of codewords in T^* for each $s_i \in S$.

The question is what is the average "blow-up" of a message using a code c. The aim is to keep this as small as possible.

Average Word Length

Consider a source $(S, \mathbf{p})$ and a code $c : S \to T^*$. Consider rthe stream of symbols we obtain by encoding $\{s_1, \cdots, s_m\}$ via c.

Definition

The average word-length L of a code $c : S \to T^*$ for a source $(S, \mathbf{p})$ is

$$L = p_1 l_1 + p_2 l_2 + \cdots p_m l_m = \sum_{i=1}^{m} p_i l_i = \mathbf{E}(l)$$

with $l_i = |c(s_i)|$ the length of codewords in T^* for each $s_i \in S$.

The question is what is the average "blow-up" of a message using a code c. The aim is to keep this as small as possible.

An Example

Example

With $S = \{s_1, s_2, s_3\}$ and for a (memoryless) source with $\mathbf{p} = (\frac{1}{5}, \frac{3}{5}, \frac{1}{5})$ consider the binary code $c : S \to \mathbb{B}^*$:

$$s_1 \mapsto 0, \quad s_2 \mapsto 10, \quad s_3 \mapsto 01$$

For this code, the average word-length is

$$L = 1 \times \frac{1}{5} + 2 \times \frac{3}{5} + 2 \times \frac{1}{5} = 1\frac{4}{5}.$$

Another binary code with shorter average word-length is:

$$s_1 \mapsto 10, \quad s_2 \mapsto 0, \quad s_3 \mapsto 01$$

$$L = 2 \times \frac{1}{5} + 1 \times \frac{3}{5} + 2 \times \frac{1}{5} = 1\frac{2}{5}.$$

An Example

Example

With $S = \{s_1, s_2, s_3\}$ and for a (memoryless) source with $\mathbf{p} = (\frac{1}{5}, \frac{3}{5}, \frac{1}{5})$ consider the binary code $c : S \to \mathbb{B}^*$:

$$s_1 \mapsto 0, \quad s_2 \mapsto 10, \quad s_3 \mapsto 01$$

For this code, the average word-length is

$$L = 1 \times \frac{1}{5} + 2 \times \frac{3}{5} + 2 \times \frac{1}{5} = 1\frac{4}{5}.$$

Another binary code with shorter average word-length is:

$$s_1 \mapsto 10, \quad s_2 \mapsto 0, \quad s_3 \mapsto 01$$

$$L = 2 \times \frac{1}{5} + 1 \times \frac{3}{5} + 2 \times \frac{1}{5} = 1\frac{2}{5}.$$

An Example

Example

With $S = \{s_1, s_2, s_3\}$ and for a (memoryless) source with $\mathbf{p} = (\frac{1}{5}, \frac{3}{5}, \frac{1}{5})$ consider the binary code $c : S \to \mathbb{B}^*$:

$$s_1 \mapsto 0, \quad s_2 \mapsto 10, \quad s_3 \mapsto 01$$

For this code, the average word-length is

$$L = 1 \times \frac{1}{5} + 2 \times \frac{3}{5} + 2 \times \frac{1}{5} = 1\frac{4}{5}.$$

Another binary code with shorter average word-length is:

$$s_1 \mapsto 10, \quad s_2 \mapsto 0, \quad s_3 \mapsto 01$$

$$L = 2 \times \frac{1}{5} + 1 \times \frac{3}{5} + 2 \times \frac{1}{5} = 1\frac{2}{5}.$$

An Example

Example

With $S = \{s_1, s_2, s_3\}$ and for a (memoryless) source with $\mathbf{p} = (\frac{1}{5}, \frac{3}{5}, \frac{1}{5})$ consider the binary code $c : S \to \mathbb{B}^*$:

$$s_1 \mapsto 0, \quad s_2 \mapsto 10, \quad s_3 \mapsto 01$$

For this code, the average word-length is

$$L = 1 \times \frac{1}{5} + 2 \times \frac{3}{5} + 2 \times \frac{1}{5} = 1\frac{4}{5}.$$

Another binary code with shorter average word-length is:

$$s_1 \mapsto 10, \quad s_2 \mapsto 0, \quad s_3 \mapsto 01$$

$$L = 2 \times \frac{1}{5} + 1 \times \frac{3}{5} + 2 \times \frac{1}{5} = 1\frac{2}{5}.$$

Optimal Code

Definition

Given a source $(S, \mathbf{p})$ and an alphabet T. A uniquely decodable code $c : S \to T^*$ is optimal if there is no other code $c' : S \to T^*$ with smaller average word-length.

As an optimisation problem we can formulate this as follows: Given $b \in \mathbb{N}$ and $p_1, p_2, \ldots, p_m \in [0, 1]$ with $\sum_{i=1}^{m} p_i = 1$; find (positive) integers $y_1, y_2, \ldots, y_m$ such that

$$\text{minimise } L = \sum_{i=1}^{m} p_i y_i \qquad \text{subject to } K = \sum_{i=1}^{m} \frac{1}{b^{y_i}} \leq 1$$

where we use $y_i = |c(s_i)| = l_i$ to express K (with $m = |S|$) as:

$$K = \sum_{i=1}^{M} \frac{n_i}{b^i} = \sum_{i=1}^{M} \left(\sum_{j=1}^{n_i} \frac{1}{b^i} \right) = \sum_{i=1}^{m} \frac{1}{b^{y_i}}$$

Optimal Code

Definition

Given a source $(S, \mathbf{p})$ and an alphabet T. A uniquely decodable code $c : S \to T^*$ is optimal if there is no other code $c' : S \to T^*$ with smaller average word-length.

As an optimisation problem we can formulate this as follows:
Given $b \in \mathbb{N}$ and $p_1, p_2, \ldots, p_m \in [0, 1]$ with $\sum_{i=1}^{m} p_i = 1$; find (positive) integers $y_1, y_2, \ldots, y_m$ such that

$$\text{minimise } L = \sum_{i=1}^{m} p_i y_i \qquad \text{subject to } K = \sum_{i=1}^{m} \frac{1}{b^{y_i}} \leq 1$$

where we use $y_i = |c(s_i)| = l_i$ to express K (with $m = |S|$) as:

$$K = \sum_{i=1}^{M} \frac{n_i}{b^i} = \sum_{i=1}^{M} \left(\sum_{j=1}^{n_i} \frac{1}{b^i} \right) = \sum_{i=1}^{m} \frac{1}{b^{y_i}}$$

Optimal Code

Definition

Given a source $(S, \mathbf{p})$ and an alphabet T. A uniquely decodable code $c : S \to T^*$ is optimal if there is no other code $c' : S \to T^*$ with smaller average word-length.

As an optimisation problem we can formulate this as follows: Given $b \in \mathbb{N}$ and $p_1, p_2, \ldots, p_m \in [0, 1]$ with $\sum_{i=1}^{m} p_i = 1$; find (positive) integers $y_1, y_2, \ldots, y_m$ such that

$$\text{minimise } L = \sum_{i=1}^{m} p_i y_i \qquad \text{subject to } K = \sum_{i=1}^{m} \frac{1}{b^{y_i}} \leq 1$$

where we use $y_i = |c(s_i)| = l_i$ to express K (with $m = |S|$) as:

$$K = \sum_{i=1}^{M} \frac{n_i}{b^i} = \sum_{i=1}^{M} \left(\sum_{j=1}^{n_i} \frac{1}{b^i} \right) = \sum_{i=1}^{m} \frac{1}{b^{y_i}}$$

Optimal Code

Definition

Given a source $(S, \mathbf{p})$ and an alphabet T. A uniquely decodable code $c : S \to T^*$ is optimal if there is no other code $c' : S \to T^*$ with smaller average word-length.

As an optimisation problem we can formulate this as follows: Given $b \in \mathbb{N}$ and $p_1, p_2, \ldots, p_m \in [0, 1]$ with $\sum_{i=1}^{m} p_i = 1$; find (positive) integers $y_1, y_2, \ldots, y_m$ such that

$$\text{minimise } L = \sum_{i=1}^{m} p_i y_i \qquad \text{subject to } K = \sum_{i=1}^{m} \frac{1}{b^{y_i}} \leq 1$$

where we use $y_i = |c(s_i)| = l_i$ to express K (with $m = |S|$) as:

$$K = \sum_{i=1}^{M} \frac{n_i}{b^i} = \sum_{i=1}^{M} \left(\sum_{j=1}^{n_i} \frac{1}{b^i} \right) = \sum_{i=1}^{m} \frac{1}{b^{y_i}}$$

Entropy of a Distribution

Definition

Given a distribution $\mathbf{p} = (p_1, p_2, \cdots, p_m)$. The entropy of $\mathbf{p}$ (to base b) is given by

$$\mathbf{H}_b(\mathbf{p}) = \sum_{i=1}^{m} p_i \log_b(\frac{1}{p_i})$$

For $p_i = 0$ we set $p_i \log_b(\frac{1}{p_i}) = 0$. Note: $\log_b(p_i) = -\log_b(\frac{1}{p_i})$.

Example

For $\mathbf{p} = (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ we have the (binary) entropy:

$$\begin{aligned} \mathbf{H}_2(\mathbf{p}) &= \frac{1}{2}\log_2(2) + \frac{1}{4}\log_2(4) + \frac{1}{4}\log_2(4) \\ &= 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 2 \cdot \frac{1}{4} = \frac{1}{2} + \frac{2}{4} + \frac{2}{4} = \frac{3}{2} \end{aligned}$$

Entropy of a Distribution

Definition

Given a distribution $\mathbf{p} = (p_1, p_2, \cdots, p_m)$. The entropy of $\mathbf{p}$ (to base b) is given by

$$\mathbf{H}_b(\mathbf{p}) = \sum_{i=1}^{m} p_i \log_b(\frac{1}{p_i})$$

For $p_i = 0$ we set $p_i \log_b(\frac{1}{p_i}) = 0$. Note: $\log_b(p_i) = -\log_b(\frac{1}{p_i})$.

Example

For $\mathbf{p} = (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ we have the (binary) entropy:

$$\begin{aligned} \mathbf{H}_2(\mathbf{p}) &= \frac{1}{2}\log_2(2) + \frac{1}{4}\log_2(4) + \frac{1}{4}\log_2(4) \\ &= 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 2 \cdot \frac{1}{4} = \frac{1}{2} + \frac{2}{4} + \frac{2}{4} = \frac{3}{2} \end{aligned}$$

Entropy of a Distribution

Definition

Given a distribution $\mathbf{p} = (p_1, p_2, \cdots, p_m)$. The entropy of $\mathbf{p}$ (to base b) is given by

$$\mathbf{H}_b(\mathbf{p}) = \sum_{i=1}^{m} p_i \log_b(\frac{1}{p_i})$$

For $p_i = 0$ we set $p_i \log_b(\frac{1}{p_i}) = 0$. Note: $\log_b(p_i) = -\log_b(\frac{1}{p_i})$.

Example

For $\mathbf{p} = (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ we have the (binary) entropy:

$$\begin{aligned} \mathbf{H}_2(\mathbf{p}) &= \frac{1}{2}\log_2(2) + \frac{1}{4}\log_2(4) + \frac{1}{4}\log_2(4) \\ &= 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 2 \cdot \frac{1}{4} = \frac{1}{2} + \frac{2}{4} + \frac{2}{4} = \frac{3}{2} \end{aligned}$$

Properties of Entropy

For source $(\mathbb{B}, (p, 1-p))$ the (binary) entropy is given by:

$$\mathbf{h}(p) = p \cdot \log_b\left(\frac{1}{p}\right) + (1-p) \cdot \log_b\left(\frac{1}{(1-p)}\right).$$

We have: $\mathbf{h}(0) = 0 = \mathbf{h}(1)$ and maximal for $\mathbf{h}(\frac{1}{2}) = 1$.

We can change the base simply via: $\mathbf{H}_a(\mathbf{p}) = \log_a(b)\mathbf{H}_b(\mathbf{p})$.

Except for the base, entropy is uniquely defined if we require (see Applebaum, Section 6.6) a continuous function with:

- $\mathbf{H}(\mathbf{p})$ is maximal for uniform distribution $\mathbf{p}$,
- $\mathbf{H}(\mathbf{p} \otimes \mathbf{q}) = \mathbf{H}(\mathbf{p}) + \mathbf{H}(\mathbf{q} \mid \mathbf{p})$,
- $\mathbf{H}((p_1, p_2, \ldots, p_n, 0)) = \mathbf{H}((p_1, p_2, \ldots, p_n))$,

with conditional entropy $\mathbf{H}(\mathbf{p} \mid \mathbf{q}) = \sum_j q_j \cdot \mathbf{H}(\mathbf{p} \mid j)$, cf. later (for independent distributions: $\mathbf{H}(\mathbf{p} \mid \mathbf{q}) = \mathbf{H}(\mathbf{p})$).

Properties of Entropy

For source $(\mathbb{B}, (p, 1-p))$ the (binary) entropy is given by:

$$\mathbf{h}(p) = p \cdot \log_b\left(\frac{1}{p}\right) + (1-p) \cdot \log_b\left(\frac{1}{(1-p)}\right).$$

We have: $\mathbf{h}(0) = 0 = \mathbf{h}(1)$ and maximal for $\mathbf{h}(\frac{1}{2}) = 1$.

We can change the base simply via: $\mathbf{H}_a(\mathbf{p}) = \log_a(b)\mathbf{H}_b(\mathbf{p})$.

Except for the base, entropy is uniquely defined if we require (see Applebaum, Section 6.6) a continuous function with:

- $\mathbf{H}(\mathbf{p})$ is maximal for uniform distribution $\mathbf{p}$,
- $\mathbf{H}(\mathbf{p} \otimes \mathbf{q}) = \mathbf{H}(\mathbf{p}) + \mathbf{H}(\mathbf{q} \mid \mathbf{p})$,
- $\mathbf{H}((p_1, p_2, \ldots, p_n, 0)) = \mathbf{H}((p_1, p_2, \ldots, p_n))$,

with conditional entropy $\mathbf{H}(\mathbf{p} \mid \mathbf{q}) = \sum_j q_j \cdot \mathbf{H}(\mathbf{p} \mid j)$, cf. later (for independent distributions: $\mathbf{H}(\mathbf{p} \mid \mathbf{q}) = \mathbf{H}(\mathbf{p})$).

Properties of Entropy

For source $(\mathbb{B}, (p, 1-p))$ the (binary) entropy is given by:

$$\mathbf{h}(p) = p \cdot \log_b\left(\frac{1}{p}\right) + (1-p) \cdot \log_b\left(\frac{1}{(1-p)}\right).$$

We have: $\mathbf{h}(0) = 0 = \mathbf{h}(1)$ and maximal for $\mathbf{h}(\frac{1}{2}) = 1$.

We can change the base simply via: $\mathbf{H}_a(\mathbf{p}) = \log_a(b)\mathbf{H}_b(\mathbf{p})$.

Except for the base, entropy is uniquely defined if we require (see Applebaum, Section 6.6) a continuous function with:

- $\mathbf{H}(\mathbf{p})$ is maximal for uniform distribution $\mathbf{p}$,
- $\mathbf{H}(\mathbf{p} \otimes \mathbf{q}) = \mathbf{H}(\mathbf{p}) + \mathbf{H}(\mathbf{q} \mid \mathbf{p})$,
- $\mathbf{H}((p_1, p_2, \ldots, p_n, 0)) = \mathbf{H}((p_1, p_2, \ldots, p_n))$,

with conditional entropy $\mathbf{H}(\mathbf{p} \mid \mathbf{q}) = \sum_j q_j \cdot \mathbf{H}(\mathbf{p} \mid j)$, cf. later (for independent distributions: $\mathbf{H}(\mathbf{p} \mid \mathbf{q}) = \mathbf{H}(\mathbf{p})$).

Properties of Entropy

For source $(\mathbb{B}, (p, 1-p))$ the (binary) entropy is given by:

$$\mathbf{h}(p) = p \cdot \log_b\left(\frac{1}{p}\right) + (1-p) \cdot \log_b\left(\frac{1}{(1-p)}\right).$$

We have: $\mathbf{h}(0) = 0 = \mathbf{h}(1)$ and maximal for $\mathbf{h}(\frac{1}{2}) = 1$.

We can change the base simply via: $\mathbf{H}_a(\mathbf{p}) = \log_a(b)\mathbf{H}_b(\mathbf{p})$.

Except for the base, entropy is uniquely defined if we require (see Applebaum, Section 6.6) a continuous function with:

- $\mathbf{H}(\mathbf{p})$ is maximal for uniform distribution $\mathbf{p}$,
- $\mathbf{H}(\mathbf{p} \otimes \mathbf{q}) = \mathbf{H}(\mathbf{p}) + \mathbf{H}(\mathbf{q} \mid \mathbf{p})$,
- $\mathbf{H}((p_1, p_2, \ldots, p_n, 0)) = \mathbf{H}((p_1, p_2, \ldots, p_n))$,

with conditional entropy $\mathbf{H}(\mathbf{p} \mid \mathbf{q}) = \sum_j q_j \cdot \mathbf{H}(\mathbf{p} \mid j)$, cf. later (for independent distributions: $\mathbf{H}(\mathbf{p} \mid \mathbf{q}) = \mathbf{H}(\mathbf{p})$).

Properties of Entropy

For source $(\mathbb{B}, (p, 1-p))$ the (binary) entropy is given by:

$$\mathbf{h}(p) = p \cdot \log_b\left(\frac{1}{p}\right) + (1-p) \cdot \log_b\left(\frac{1}{(1-p)}\right).$$

We have: $\mathbf{h}(0) = 0 = \mathbf{h}(1)$ and maximal for $\mathbf{h}(\frac{1}{2}) = 1$.

We can change the base simply via: $\mathbf{H}_a(\mathbf{p}) = \log_a(b)\mathbf{H}_b(\mathbf{p})$.

Except for the base, entropy is uniquely defined if we require (see Applebaum, Section 6.6) a continuous function with:

- $\mathbf{H}(\mathbf{p})$ is maximal for uniform distribution $\mathbf{p}$,
- $\mathbf{H}(\mathbf{p} \otimes \mathbf{q}) = \mathbf{H}(\mathbf{p}) + \mathbf{H}(\mathbf{q} \mid \mathbf{p})$,
- $\mathbf{H}((p_1, p_2, \ldots, p_n, 0)) = \mathbf{H}((p_1, p_2, \ldots, p_n))$,

with conditional entropy $\mathbf{H}(\mathbf{p} \mid \mathbf{q}) = \sum_j q_j \cdot \mathbf{H}(\mathbf{p} \mid j)$, cf. later (for independent distributions: $\mathbf{H}(\mathbf{p} \mid \mathbf{q}) = \mathbf{H}(\mathbf{p})$).

Properties of Entropy

For source $(\mathbb{B}, (p, 1-p))$ the (binary) entropy is given by:

$$\mathbf{h}(p) = p \cdot \log_b \left(\frac{1}{p}\right) + (1-p) \cdot \log_b \left(\frac{1}{(1-p)}\right).$$

We have: $\mathbf{h}(0) = 0 = \mathbf{h}(1)$ and maximal for $\mathbf{h}(\frac{1}{2}) = 1$.

We can change the base simply via: $\mathbf{H}_a(\mathbf{p}) = \log_a(b)\mathbf{H}_b(\mathbf{p})$.

Except for the base, entropy is uniquely defined if we require (see Applebaum, Section 6.6) a continuous function with:

- $\mathbf{H}(\mathbf{p})$ is maximal for uniform distribution $\mathbf{p}$,
- $\mathbf{H}(\mathbf{p} \otimes \mathbf{q}) = \mathbf{H}(\mathbf{p}) + \mathbf{H}(\mathbf{q} \mid \mathbf{p})$,
- $\mathbf{H}((p_1, p_2, \ldots, p_n, 0)) = \mathbf{H}((p_1, p_2, \ldots, p_n))$,

with conditional entropy $\mathbf{H}(\mathbf{p} \mid \mathbf{q}) = \sum_j q_j \cdot \mathbf{H}(\mathbf{p} \mid j)$, cf. later (for independent distributions: $\mathbf{H}(\mathbf{p} \mid \mathbf{q}) = \mathbf{H}(\mathbf{p})$).

Properties of Entropy

For source $(\mathbb{B}, (p, 1-p))$ the (binary) entropy is given by:

$$\mathbf{h}(p) = p \cdot \log_b\left(\frac{1}{p}\right) + (1-p) \cdot \log_b\left(\frac{1}{(1-p)}\right).$$

We have: $\mathbf{h}(0) = 0 = \mathbf{h}(1)$ and maximal for $\mathbf{h}(\frac{1}{2}) = 1$.

We can change the base simply via: $\mathbf{H}_a(\mathbf{p}) = \log_a(b)\mathbf{H}_b(\mathbf{p})$.

Except for the base, entropy is uniquely defined if we require (see Applebaum, Section 6.6) a continuous function with:

- $\mathbf{H}(\mathbf{p})$ is maximal for uniform distribution $\mathbf{p}$,
- $\mathbf{H}(\mathbf{p} \otimes \mathbf{q}) = \mathbf{H}(\mathbf{p}) + \mathbf{H}(\mathbf{q} \mid \mathbf{p})$,
- $\mathbf{H}((p_1, p_2, \ldots, p_n, 0)) = \mathbf{H}((p_1, p_2, \ldots, p_n))$,

with conditional entropy $\mathbf{H}(\mathbf{p} \mid \mathbf{q}) = \sum_j q_j \cdot \mathbf{H}(\mathbf{p} \mid j)$, cf. later (for independent distributions: $\mathbf{H}(\mathbf{p} \mid \mathbf{q}) = \mathbf{H}(\mathbf{p})$).

Comparison Theorem

Theorem

Given probability distributions $\mathbf{p} = (p_1, p_2, \cdots, p_m)$ *and* $\mathbf{q} = (q_1, q_2, \cdots, q_m)$*, then*

$$\mathbf{H}_b(\mathbf{p}) = \sum_{i=1}^{m} p_i \cdot \log_b(\frac{1}{p_i}) \leq \sum_{i=1}^{m} p_i \cdot \log_b(\frac{1}{q_i})$$

There is equality if and only if $p_i = q_i$ *for all* $1 \leq i \leq m$.

Lemma (Convexity)

For all $x > 0$ *we have* $\ln(x) \leq x - 1$ *with equality iff* $x = 1$.

Proof in elementary calculus or Biggs: Lemma 3.9 (p36).
Note: “iff” means “if and only if”.

Comparison Theorem

Theorem

Given probability distributions $\mathbf{p} = (p_1, p_2, \cdots, p_m)$ *and* $\mathbf{q} = (q_1, q_2, \cdots, q_m)$*, then*

$$\mathbf{H}_b(\mathbf{p}) = \sum_{i=1}^{m} p_i \cdot \log_b(\frac{1}{p_i}) \leq \sum_{i=1}^{m} p_i \cdot \log_b(\frac{1}{q_i})$$

There is equality if and only if $p_i = q_i$ *for all* $1 \leq i \leq m$.

Lemma (Convexity)

For all $x > 0$ *we have* $\ln(x) \leq x - 1$ *with equality iff* $x = 1$.

Proof in elementary calculus or Biggs: Lemma 3.9 (p36).
Note: "iff" means "if and only if".

Comparison Theorem

Theorem

Given probability distributions $\mathbf{p} = (p_1, p_2, \cdots, p_m)$ *and* $\mathbf{q} = (q_1, q_2, \cdots, q_m)$*, then*

$$\mathbf{H}_b(\mathbf{p}) = \sum_{i=1}^{m} p_i \cdot \log_b(\frac{1}{p_i}) \leq \sum_{i=1}^{m} p_i \cdot \log_b(\frac{1}{q_i})$$

There is equality if and only if $p_i = q_i$ *for all* $1 \leq i \leq m$.

Lemma (Convexity)

For all $x > 0$ *we have* $\ln(x) \leq x - 1$ *with equality iff* $x = 1$.

Proof in elementary calculus or Biggs: Lemma 3.9 (p36).
Note: “iff” means “if and only if”.

Proof of Comparison Theorem

Proof.

W.l.o.g take $b = e$, i.e. the natural logarithm. By previous lemma we have $\ln(\frac{q_i}{p_i}) \leq \frac{q_i}{p_i} - 1$, with equality iff $p_i = q_i$.

We also have $\ln(\frac{q_i}{p_i}) = \ln(\frac{1}{p_i}) - \ln(\frac{1}{q_i})$.

$$\sum_{i=1}^{m} p_i \cdot \ln(\frac{1}{p_i}) - \sum_{i=1}^{m} p_i \cdot \ln(\frac{1}{q_i}) = \sum_{i=1}^{m} p_i \cdot \ln(\frac{q_i}{p_i}) \leq$$

$$\leq \sum_{i=1}^{m} p_i \cdot \left(\frac{q_i}{p_i} - 1\right) = \sum_{i=1}^{m} q_i - \sum_{i=1}^{m} p_i = 1 - 1 = 0$$

Quality holds only iff $p_i = q_i$ for all i. □

Proof of Comparison Theorem

Proof.

W.l.o.g take $b = e$, i.e. the natural logarithm. By previous lemma we have $\ln(\frac{q_i}{p_i}) \leq \frac{q_i}{p_i} - 1$, with equality iff $p_i = q_i$.

We also have $\ln(\frac{q_i}{p_i}) = \ln(\frac{1}{p_i}) - \ln(\frac{1}{q_i})$.

$$\sum_{i=1}^{m} p_i \cdot \ln(\frac{1}{p_i}) - \sum_{i=1}^{m} p_i \cdot \ln(\frac{1}{q_i}) = \sum_{i=1}^{m} p_i \cdot \ln(\frac{q_i}{p_i}) \leq$$

$$\leq \sum_{i=1}^{m} p_i \cdot \left(\frac{q_i}{p_i} - 1\right) = \sum_{i=1}^{m} q_i - \sum_{i=1}^{m} p_i = 1 - 1 = 0$$

Quality holds only iff $p_i = q_i$ for all i. □

Proof of Comparison Theorem

Proof.

W.l.o.g take $b = e$, i.e. the natural logarithm. By previous lemma we have $\ln(\frac{q_i}{p_i}) \leq \frac{q_i}{p_i} - 1$, with equality iff $p_i = q_i$.

We also have $\ln(\frac{q_i}{p_i}) = \ln(\frac{1}{p_i}) - \ln(\frac{1}{q_i})$.

$$\sum_{i=1}^{m} p_i \cdot \ln(\frac{1}{p_i}) - \sum_{i=1}^{m} p_i \cdot \ln(\frac{1}{q_i}) = \sum_{i=1}^{m} p_i \cdot \ln(\frac{q_i}{p_i}) \leq$$

$$\leq \sum_{i=1}^{m} p_i \cdot \left(\frac{q_i}{p_i} - 1\right) = \sum_{i=1}^{m} q_i - \sum_{i=1}^{m} p_i = 1 - 1 = 0$$

Quality holds only iff $p_i = q_i$ for all i. □

Proof of Comparison Theorem

Proof.

W.l.o.g take $b = e$, i.e. the natural logarithm. By previous lemma we have $\ln(\frac{q_i}{p_i}) \leq \frac{q_i}{p_i} - 1$, with equality iff $p_i = q_i$.

We also have $\ln(\frac{q_i}{p_i}) = \ln(\frac{1}{p_i}) - \ln(\frac{1}{q_i})$.

$$\sum_{i=1}^{m} p_i \cdot \ln(\frac{1}{p_i}) - \sum_{i=1}^{m} p_i \cdot \ln(\frac{1}{q_i}) = \sum_{i=1}^{m} p_i \cdot \ln(\frac{q_i}{p_i}) \leq$$

$$\leq \sum_{i=1}^{m} p_i \cdot \left(\frac{q_i}{p_i} - 1\right) = \sum_{i=1}^{m} q_i - \sum_{i=1}^{m} p_i = 1 - 1 = 0$$

Quality holds only iff $p_i = q_i$ for all i. □

Uniform Distribution

Theorem

The entropy of a probability distribution $\mathbf{p}$ *on* m *symbols is at most* $\log_b(m)$*, i.e.*

$$\mathbf{H}_b(\mathbf{p}) \leq \log_b(m)$$

There is equality if and only if $p_i = \frac{1}{m}$ *for all symbols.*

Proof.

Given probability distributions $\mathbf{p} = (p_1, p_2, \cdots, p_m)$ and take $\mathbf{q} = (q_1, q_2, \cdots, q_m)$ with $q_i = \frac{1}{m}$ for all $1 \leq i \leq m$, then we have (by Comparison Theorem):

$$\mathbf{H}_b(\mathbf{p}) \leq \sum_{i=1}^{m} p_i \cdot \log_b(m) = \log_b(m)$$

There is equality if and only if $p_i = q_i = \frac{1}{m}$ for all $1 \leq i \leq m$. □

Uniform Distribution

Theorem

The entropy of a probability distribution $\mathbf{p}$ *on* m *symbols is at most* $\log_b(m)$*, i.e.*

$$\mathbf{H}_b(\mathbf{p}) \leq \log_b(m)$$

There is equality if and only if $p_i = \frac{1}{m}$ *for all symbols.*

Proof.

Given probability distributions $\mathbf{p} = (p_1, p_2, \cdots, p_m)$ and take $\mathbf{q} = (q_1, q_2, \cdots, q_m)$ with $q_i = \frac{1}{m}$ for all $1 \leq i \leq m$, then we have (by Comparison Theorem):

$$\mathbf{H}_b(\mathbf{p}) \leq \sum_{i=1}^{m} p_i \cdot \log_b(m) = \log_b(m)$$

There is equality if and only if $p_i = q_i = \frac{1}{m}$ for all $1 \leq i \leq m$. □

Fundamental Theorem

Theorem

The average word-length L of any uniquely decodable code $c : S \to T^*$ with $|T| = b$ for the source $(S, \mathbf{p})$ satisfies

$$L \geq \mathbf{H}_b(\mathbf{p}).$$

Proof.

With $S = \{s_1, s_2, \ldots, s_m\}$ denote by y_i the length of the codeword $s_i \in S$. Then the Kraft-McMillan number is

$$K = \sum_{i=1}^{m} \frac{1}{b^{y_i}} = \frac{1}{b^{y_1}} + \frac{1}{b^{y_2}} + \cdots \frac{1}{b^{y_m}} :$$

Take probability distribution $\mathbf{q} = (q_1, \ldots, q_m)$ with $q_i = \frac{1}{Kb^{y_i}}$.
[Note that $\sum_i 1/Kb^{y_i} = 1/K \sum_i 1/b^{y_i} = K/K = 1$.]

Fundamental Theorem

Theorem

The average word-length L of any uniquely decodable code $c : S \to T^*$ with $|T| = b$ for the source $(S, \mathbf{p})$ satisfies

$$L \geq \mathbf{H}_b(\mathbf{p}).$$

Proof.

With $S = \{s_1, s_2, \ldots, s_m\}$ denote by y_i the length of the codeword $s_i \in S$. Then the Kraft-McMillan number is

$$K = \sum_{i=1}^{m} \frac{1}{b^{y_i}} = \frac{1}{b^{y_1}} + \frac{1}{b^{y_2}} + \cdots \frac{1}{b^{y_m}} :$$

Take probability distribution $\mathbf{q} = (q_1, \ldots, q_m)$ with $q_i = \frac{1}{Kb^{y_i}}$.
[Note that $\sum_i 1/Kb^{y_i} = 1/K \sum_i 1/b^{y_i} = K/K = 1$.]

Fundamental Theorem

Theorem

The average word-length L of any uniquely decodable code $c : S \to T^*$ with $|T| = b$ for the source $(S, \mathbf{p})$ satisfies

$$L \geq \mathbf{H}_b(\mathbf{p}).$$

Proof.

With $S = \{s_1, s_2, \ldots, s_m\}$ denote by y_i the length of the codeword $s_i \in S$. Then the Kraft-McMillan number is

$$K = \sum_{i=1}^{m} \frac{1}{b^{y_i}} = \frac{1}{b^{y_1}} + \frac{1}{b^{y_2}} + \cdots \frac{1}{b^{y_m}} :$$

Take probability distribution $\mathbf{q} = (q_1, \ldots, q_m)$ with $q_i = \frac{1}{Kb^{y_i}}$.
[Note that $\sum_i 1/Kb^{y_i} = 1/K \sum_i 1/b^{y_i} = K/K = 1$.]

Proof of Fundamental Theorem (cont.)

Proof (cont).

Apply Comparison Theorem to **p** and **q** so that

$$\mathbf{H}_b(\mathbf{p}) = \sum_{i=1}^{m} p_i \cdot \log_b(\frac{1}{p_i}) \leq \sum_{i=1}^{m} p_i \cdot \log_b(\frac{1}{q_i})$$

which (with the **q** we have chosen) gives

$$\begin{aligned} \mathbf{H}_b(\mathbf{p}) &\leq \sum_{i=1}^{m} p_i \cdot \log_b(Kb^{y_i}) = \sum_{i=1}^{m} p_i \cdot (\log_b(K) + y_i) \\ &= \log_b(K) + \sum_{i=1}^{m} p_i \cdot y_i = \log_b(K) + L \leq L. \end{aligned}$$

as for an UD code $K \leq 1$ it follows that $\log_b(K) \leq 0$. □

Proof of Fundamental Theorem (cont.)

Proof (cont).

Apply Comparison Theorem to **p** and **q** so that

$$\mathbf{H}_b(\mathbf{p}) = \sum_{i=1}^{m} p_i \cdot \log_b(\frac{1}{p_i}) \leq \sum_{i=1}^{m} p_i \cdot \log_b(\frac{1}{q_i})$$

which (with the **q** we have chosen) gives

$$\begin{aligned} \mathbf{H}_b(\mathbf{p}) &\leq \sum_{i=1}^{m} p_i \cdot \log_b(Kb^{y_i}) = \sum_{i=1}^{m} p_i \cdot (\log_b(K) + y_i) \\ &= \log_b(K) + \sum_{i=1}^{m} p_i \cdot y_i = \log_b(K) + L \leq L. \end{aligned}$$

as for an UD code $K \leq 1$ it follows that $\log_b(K) \leq 0$. □

Proof of Fundamental Theorem (cont.)

Proof (cont).

Apply Comparison Theorem to **p** and **q** so that

$$\mathbf{H}_b(\mathbf{p}) = \sum_{i=1}^{m} p_i \cdot \log_b(\frac{1}{p_i}) \leq \sum_{i=1}^{m} p_i \cdot \log_b(\frac{1}{q_i})$$

which (with the **q** we have chosen) gives

$$\begin{aligned} \mathbf{H}_b(\mathbf{p}) &\leq \sum_{i=1}^{m} p_i \cdot \log_b(Kb^{y_i}) = \sum_{i=1}^{m} p_i \cdot (\log_b(K) + y_i) \\ &= \log_b(K) + \sum_{i=1}^{m} p_i \cdot y_i = \log_b(K) + L \leq L. \end{aligned}$$

as for an UD code $K \leq 1$ it follows that $\log_b(K) \leq 0$. □

Shannon-Fano Rule

To get $L = \mathbf{H}_b(\mathbf{p})$ we would need $b^{y_i} = \frac{1}{p_i}$ for all $1 \leq i \leq m$.

Shannon-Fano (SF) Rule:

Select the word length y_i for symbol s_i the least positive integer such that $b^{y_i} \geq \frac{1}{p_i}$.

Theorem

There exists a prefix-free code $c : S \to T^*$ with $|T| = b$ for the source $(S, \mathbf{p})$ which satisfies: $L < \mathbf{H}_b(\mathbf{p}) + 1$.

Proof.

We chose y_i as the least positive integer with $b^{y_i} \geq \frac{1}{p_i}$ so we have $b^{y_i - 1} < \frac{1}{p_i}$, thus $y_i - 1 < \log_b(\frac{1}{p_i})$ or $y_i < 1 + \log_b(\frac{1}{p_i})$,.

$$L = \sum p_i y_i < \sum p_i \cdot (1 + \log_b(\frac{1}{p_i})) = 1 + \mathbf{H}_b(\mathbf{p}).$$

Shannon-Fano Rule

To get $L = \mathbf{H}_b(\mathbf{p})$ we would need $b^{y_i} = \frac{1}{p_i}$ for all $1 \leq i \leq m$.

Shannon-Fano (SF) Rule:
Select the word length y_i for symbol s_i the least positive integer such that $b^{y_i} \geq \frac{1}{p_i}$.

Theorem

There exists a prefix-free code $c : S \to T^*$ with $|T| = b$ for the source $(S, \mathbf{p})$ which satisfies: $L < \mathbf{H}_b(\mathbf{p}) + 1$.

Proof.

We chose y_i as the least positive integer with $b^{y_i} \geq \frac{1}{p_i}$ so we have $b^{y_i - 1} < \frac{1}{p_i}$, thus $y_i - 1 < \log_b(\frac{1}{p_i})$ or $y_i < 1 + \log_b(\frac{1}{p_i})$,.

$$L = \sum p_i y_i < \sum p_i \cdot (1 + \log_b(\frac{1}{p_i})) = 1 + \mathbf{H}_b(\mathbf{p}).$$

Shannon-Fano Rule

To get $L = \mathbf{H}_b(\mathbf{p})$ we would need $b^{y_i} = \frac{1}{p_i}$ for all $1 \leq i \leq m$.

Shannon-Fano (SF) Rule:
Select the word length y_i for symbol s_i the least positive integer such that $b^{y_i} \geq \frac{1}{p_i}$.

Theorem

There exists a prefix-free code $c : S \to T^*$ with $|T| = b$ for the source $(S, \mathbf{p})$ which satisfies: $L < \mathbf{H}_b(\mathbf{p}) + 1$.

Proof.

We chose y_i as the least positive integer with $b^{y_i} \geq \frac{1}{p_i}$ so we have $b^{y_i - 1} < \frac{1}{p_i}$, thus $y_i - 1 < \log_b(\frac{1}{p_i})$ or $y_i < 1 + \log_b(\frac{1}{p_i})$,.

$$L = \sum p_i y_i < \sum p_i \cdot (1 + \log_b(\frac{1}{p_i})) = 1 + \mathbf{H}_b(\mathbf{p}).$$

Shannon-Fano Rule

To get $L = \mathbf{H}_b(\mathbf{p})$ we would need $b^{y_i} = \frac{1}{p_i}$ for all $1 \leq i \leq m$.

Shannon-Fano (SF) Rule:
Select the word length y_i for symbol s_i the least positive integer such that $b^{y_i} \geq \frac{1}{p_i}$.

Theorem

There exists a prefix-free code $c : S \to T^*$ with $|T| = b$ for the source $(S, \mathbf{p})$ which satisfies: $L < \mathbf{H}_b(\mathbf{p}) + 1$.

Proof.

We chose y_i as the least positive integer with $b^{y_i} \geq \frac{1}{p_i}$ so we have $b^{y_i - 1} < \frac{1}{p_i}$, thus $y_i - 1 < \log_b(\frac{1}{p_i})$ or $y_i < 1 + \log_b(\frac{1}{p_i})$,.

$$L = \sum p_i y_i < \sum p_i \cdot (1 + \log_b(\frac{1}{p_i})) = 1 + \mathbf{H}_b(\mathbf{p}).$$

Example SF Code

Example

Consider a source with $\mathbf{p} = (\frac{2}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{10}, \frac{1}{10})$ with $\mathbf{H}_2(\mathbf{p}) \approx 2.12$.

With $T = \mathbb{B}$ take for the first symbol a codeword of lengthy y_1 s.t. $2^{y_1} \geq \frac{5}{2}$, i.e. $y_1 = 2$, then $2^2 = 4 \geq 2.5$. Similarly, take y_2 (and y_3) s.t. $2^{y_2} \geq 5$, i.e. $y_2 = 3 = y_3$, and for y_4 and y_5 s.t. $2^{y_4} \geq 10$, i.e. $y_4 = 4 = y_5$.

Thus the parameters are $n_2 = 1, n_3 = 2, n_4 = 2$ (and $K \leq 1$) and we can construct a PF code, e.g.

$$s_1 \mapsto 00, s_2 \mapsto 010, s_3 \mapsto 011, s_4 \mapsto 1000, s_5 \mapsto 1001.$$

The average codeword length is:

$$L = 2 \cdot \frac{2}{5} + 3 \cdot \frac{1}{5} + 3 \cdot \frac{1}{5} + 4 \cdot \frac{1}{10} + 4 \cdot \frac{1}{10} + = 2.8 \leq 1 + \mathbf{H}_2(\mathbf{p}) \approx 3.12$$

Example SF Code

Example

Consider a source with $\mathbf{p} = (\frac{2}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{10}, \frac{1}{10})$ with $\mathbf{H}_2(\mathbf{p}) \approx 2.12$.

With $T = \mathbb{B}$ take for the first symbol a codeword of lengthy y_1 s.t. $2^{y_1} \geq \frac{5}{2}$, i.e. $y_1 = 2$, then $2^2 = 4 \geq 2.5$. Similarly, take y_2 (and y_3) s.t. $2^{y_2} \geq 5$, i.e. $y_2 = 3 = y_3$, and for y_4 and y_5 s.t. $2^{y_4} \geq 10$, i.e. $y_4 = 4 = y_5$.

Thus the parameters are $n_2 = 1, n_3 = 2, n_4 = 2$ (and $K \leq 1$) and we can construct a PF code, e.g.

$$s_1 \mapsto 00, s_2 \mapsto 010, s_3 \mapsto 011, s_4 \mapsto 1000, s_5 \mapsto 1001.$$

The average codeword length is:

$$L = 2 \cdot \frac{2}{5} + 3 \cdot \frac{1}{5} + 3 \cdot \frac{1}{5} + 4 \cdot \frac{1}{10} + 4 \cdot \frac{1}{10} + = 2.8 \leq 1 + \mathbf{H}_2(\mathbf{p}) \approx 3.12$$

Example SF Code

Example

Consider a source with $\mathbf{p} = (\frac{2}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{10}, \frac{1}{10})$ with $\mathbf{H}_2(\mathbf{p}) \approx 2.12$.

With $T = \mathbb{B}$ take for the first symbol a codeword of lengthy y_1 s.t. $2^{y_1} \geq \frac{5}{2}$, i.e. $y_1 = 2$, then $2^2 = 4 \geq 2.5$. Similarly, take y_2 (and y_3) s.t. $2^{y_2} \geq 5$, i.e. $y_2 = 3 = y_3$, and for y_4 and y_5 s.t. $2^{y_4} \geq 10$, i.e. $y_4 = 4 = y_5$.

Thus the parameters are $n_2 = 1, n_3 = 2, n_4 = 2$ (and $K \leq 1$) and we can construct a PF code, e.g.

$$s_1 \mapsto 00, s_2 \mapsto 010, s_3 \mapsto 011, s_4 \mapsto 1000, s_5 \mapsto 1001.$$

The average codeword length is:

$$L = 2 \cdot \frac{2}{5} + 3 \cdot \frac{1}{5} + 3 \cdot \frac{1}{5} + 4 \cdot \frac{1}{10} + 4 \cdot \frac{1}{10} + = 2.8 \leq 1 + \mathbf{H}_2(\mathbf{p}) \approx 3.12$$

Example SF Code

Example

Consider a source with $\mathbf{p} = (\frac{2}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{10}, \frac{1}{10})$ with $\mathbf{H}_2(\mathbf{p}) \approx 2.12$.

With $T = \mathbb{B}$ take for the first symbol a codeword of lengthy y_1 s.t. $2^{y_1} \geq \frac{5}{2}$, i.e. $y_1 = 2$, then $2^2 = 4 \geq 2.5$. Similarly, take y_2 (and y_3) s.t. $2^{y_2} \geq 5$, i.e. $y_2 = 3 = y_3$, and for y_4 and y_5 s.t. $2^{y_4} \geq 10$, i.e. $y_4 = 4 = y_5$.

Thus the parameters are $n_2 = 1, n_3 = 2, n_4 = 2$ (and $K \leq 1$) and we can construct a PF code, e.g.

$$s_1 \mapsto 00, s_2 \mapsto 010, s_3 \mapsto 011, s_4 \mapsto 1000, s_5 \mapsto 1001.$$

The average codeword length is:

$$L = 2 \cdot \frac{2}{5} + 3 \cdot \frac{1}{5} + 3 \cdot \frac{1}{5} + 4 \cdot \frac{1}{10} + 4 \cdot \frac{1}{10} + = 2.8 \leq 1 + \mathbf{H}_2(\mathbf{p}) \approx 3.12$$

Example SF Code

Example

Consider a source with $\mathbf{p} = (\frac{2}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{10}, \frac{1}{10})$ with $\mathbf{H}_2(\mathbf{p}) \approx 2.12$.

With $T = \mathbb{B}$ take for the first symbol a codeword of lengthy y_1 s.t. $2^{y_1} \geq \frac{5}{2}$, i.e. $y_1 = 2$, then $2^2 = 4 \geq 2.5$. Similarly, take y_2 (and y_3) s.t. $2^{y_2} \geq 5$, i.e. $y_2 = 3 = y_3$, and for y_4 and y_5 s.t. $2^{y_4} \geq 10$, i.e. $y_4 = 4 = y_5$.

Thus the parameters are $n_2 = 1, n_3 = 2, n_4 = 2$ (and $K \leq 1$) and we can construct a PF code, e.g.

$$s_1 \mapsto 00, s_2 \mapsto 010, s_3 \mapsto 011, s_4 \mapsto 1000, s_5 \mapsto 1001.$$

The average codeword length is:

$$L = 2 \cdot \frac{2}{5} + 3 \cdot \frac{1}{5} + 3 \cdot \frac{1}{5} + 4 \cdot \frac{1}{10} + 4 \cdot \frac{1}{10} + = 2.8 \leq 1 + \mathbf{H}_2(\mathbf{p}) \approx 3.12$$

Optimal PF Codes

Using SF we can obtain good but not optimal codes, it holds:

$$\mathbf{H} \leq L_{opt} \leq L_{SF} \leq \mathbf{H} + 1$$

Consider from now only binary codes and $\log = \log_2$.

Lemma

An optimal prefix-free code $c : S \to \mathbb{B}^*$ for a source $(S, \mathbf{p})$ has the following properties:

- *if the codeword $c(s')$ is longer than $c(s)$ then $p_s \geq p_{s'}$.*
- *among the codewords of maximal length there are two of the form $w0$ and $w1$ for some $w \in \mathbb{B}^*$.*

Optimal PF Codes

Using SF we can obtain good but not optimal codes, it holds:

$$\mathbf{H} \leq L_{opt} \leq L_{SF} \leq \mathbf{H} + 1$$

Consider from now only binary codes and $\log = \log_2$.

Lemma

An optimal prefix-free code $c : S \to \mathbb{B}^*$ for a source $(S, \mathbf{p})$ has the following properties:

- *if the codeword $c(s')$ is longer than $c(s)$ then $p_s \geq p_{s'}$.*
- *among the codewords of maximal length there are two of the form $w0$ and $w1$ for some $w \in \mathbb{B}^*$.*

Optimal PF Codes

Using SF we can obtain good but not optimal codes, it holds:

$$\mathbf{H} \leq L_{opt} \leq L_{SF} \leq \mathbf{H} + 1$$

Consider from now only binary codes and $\log = \log_2$.

Lemma

An optimal prefix-free code $c : S \to \mathbb{B}^*$ for a source $(S, \mathbf{p})$ has the following properties:

- *if the codeword $c(s')$ is longer than $c(s)$ then $p_s \geq p_{s'}$,*
- *among the codewords of maximal length there are two of the form $w0$ and $w1$ for some $w \in \mathbb{B}^*$.*

Optimal PF Codes

Using SF we can obtain good but not optimal codes, it holds:

$$\mathbf{H} \leq L_{opt} \leq L_{SF} \leq \mathbf{H} + 1$$

Consider from now only binary codes and $\log = \log_2$.

Lemma

An optimal prefix-free code $c : S \to \mathbb{B}^*$ for a source $(S, \mathbf{p})$ has the following properties:

- *if the codeword $c(s')$ is longer than $c(s)$ then $p_s \geq p_{s'}$,*
- *among the codewords of maximal length there are two of the form $w0$ and $w1$ for some $w \in \mathbb{B}^*$.*

Optimal PF Codes

Using SF we can obtain good but not optimal codes, it holds:

$$\mathbf{H} \leq L_{opt} \leq L_{SF} \leq \mathbf{H} + 1$$

Consider from now only binary codes and $\log = \log_2$.

Lemma

An optimal prefix-free code $c : S \to \mathbb{B}^*$ for a source $(S, \mathbf{p})$ has the following properties:

- *if the codeword $c(s')$ is longer than $c(s)$ then $p_s \geq p_{s'}$,*
- *among the codewords of maximal length there are two of the form $w0$ and $w1$ for some $w \in \mathbb{B}^*$.*

Proof

Proof.

- We have $|c(s')| > |c(s)|$ and $p_s \geq p_{s'}$.
  Define a new code which swaps the encodings of s and s', i.e. $\tilde{c}(s) = c(s')$, $\tilde{c}(s') = c(s)$.
  The difference between the average word-lengths is:

$$\begin{aligned} L(\tilde{c}) - L(c) &= (p_s|\tilde{c}(s')| + p_{s'}|\tilde{c}(s)|) \\ &- (p_s|c(s)| + p_{s'}|c(s')|) \\ &= (p_s - p_{s'})(|c(s')| - |c(s)|) \geq 0. \end{aligned}$$

- If there are no maximal codeword of the form $w0$ and $w1$ (i.e. they are always either of the form $w0$ or $w1$, but not both), then delete the last bit and obtain a better code.

□

Proof

Proof.

- We have $|c(s')| > |c(s)|$ and $p_s \geq p_{s'}$.
  Define a new code which swaps the encodings of s and s', i.e. $\tilde{c}(s) = c(s')$, $\tilde{c}(s') = c(s)$.
  The difference between the average word-lengths is:

$$\begin{aligned} L(\tilde{c}) - L(c) &= (p_s|\tilde{c}(s')| + p_{s'}|\tilde{c}(s)|) \\ &- (p_s|c(s)| + p_{s'}|c(s')|) \\ &= (p_s - p_{s'})(|c(s')| - |c(s)|) \geq 0. \end{aligned}$$

- If there are no maximal codeword of the form $w0$ and $w1$ (i.e. they are always either of the form $w0$ or $w1$, but not both), then delete the last bit and obtain a better code.

□

Huffman's Rules

Huffman's Rules to construct optimal codes (1952):

H1 Given a source $(S, \mathbf{p})$ and be s' and s'' be symbols with smallest probability. Construct a new source $(S^*, \mathbf{p}^*)$ by replacing s' and s'' with a new symbol s^* with probability $p_{s^*} = p_{s'} + p_{s''}$.
Nothing changes for other symbols.

H2 If we are given a PF binary code h^* for $(S^*, \mathbf{p}^*)$ with $h^* : s^* \mapsto w$ then define a binary code h for $(S, \mathbf{p})$ with $h : s' \mapsto w0$ and $h : s'' \mapsto w1$.
Nothing changes for other symbols.

The rule **H1** constructs a new, abstract source $(S^*, \mathbf{p}^*)$ s.t. $|S^*| = |S| - 1$ until we have a trivial source $(\{s^*\}, \mathbf{p}^* = (1))$.

The rule **H2** constructs a new, more concrete code h from h^* for a larger S by "splitting" a symbol s^* into s' and s''.

Huffman's Rules

Huffman's Rules to construct optimal codes (1952):

H1 Given a source $(S, \mathbf{p})$ and be s' and s'' be symbols with smallest probability. Construct a new source $(S^*, \mathbf{p}^*)$ by replacing s' and s'' with a new symbol s^* with probability $p_{s^*} = p_{s'} + p_{s''}$.
Nothing changes for other symbols.

H2 If we are given a PF binary code h^* for $(S^*, \mathbf{p}^*)$ with $h^* : s^* \mapsto w$ then define a binary code h for $(S, \mathbf{p})$ with $h : s' \mapsto w0$ and $h : s'' \mapsto w1$.
Nothing changes for other symbols.

The rule **H1** constructs a new, abstract source $(S^*, \mathbf{p}^*)$ s.t. $|S^*| = |S| - 1$ until we have a trivial source $(\{s^*\}, \mathbf{p}^* = (1))$.

The rule **H2** constructs a new, more concrete code h from h^* for a larger S by "splitting" a symbol s^* into s' and s''.

Huffman's Rules

Huffman's Rules to construct optimal codes (1952):

H1 Given a source $(S, \mathbf{p})$ and be s' and s'' be symbols with smallest probability. Construct a new source $(S^*, \mathbf{p}^*)$ by replacing s' and s'' with a new symbol s^* with probability $p_{s^*} = p_{s'} + p_{s''}$.
Nothing changes for other symbols.

H2 If we are given a PF binary code h^* for $(S^*, \mathbf{p}^*)$ with $h^* : s^* \mapsto w$ then define a binary code h for $(S, \mathbf{p})$ with $h : s' \mapsto w0$ and $h : s'' \mapsto w1$.
Nothing changes for other symbols.

The rule **H1** constructs a new, abstract source $(S^*, \mathbf{p}^*)$ s.t. $|S^*| = |S| - 1$ until we have a trivial source $(\{s^*\}, \mathbf{p}^* = (1))$.

The rule **H2** constructs a new, more concrete code h from h^* for a larger S by "splitting" a symbol s^* into s' and s''.

Huffman's Rules

Huffman's Rules to construct optimal codes (1952):

H1 Given a source $(S, \mathbf{p})$ and be s' and s'' be symbols with smallest probability. Construct a new source $(S^*, \mathbf{p}^*)$ by replacing s' and s'' with a new symbol s^* with probability $p_{s^*} = p_{s'} + p_{s''}$.
Nothing changes for other symbols.

H2 If we are given a PF binary code h^* for $(S^*, \mathbf{p}^*)$ with $h^* : s^* \mapsto w$ then define a binary code h for $(S, \mathbf{p})$ with $h : s' \mapsto w0$ and $h : s'' \mapsto w1$.
Nothing changes for other symbols.

The rule **H1** constructs a new, abstract source $(S^*, \mathbf{p}^*)$ s.t. $|S^*| = |S| - 1$ until we have a trivial source $(\{s^*\}, \mathbf{p}^* = (1))$.

The rule **H2** constructs a new, more concrete code h from h^* for a larger S by "splitting" a symbol s^* into s' and s''.

Huffman's Rules

Huffman's Rules to construct optimal codes (1952):

H1 Given a source $(S, \mathbf{p})$ and be s' and s'' be symbols with smallest probability. Construct a new source $(S^*, \mathbf{p}^*)$ by replacing s' and s'' with a new symbol s^* with probability $p_{s^*} = p_{s'} + p_{s''}$.
Nothing changes for other symbols.

H2 If we are given a PF binary code h^* for $(S^*, \mathbf{p}^*)$ with $h^* : s^* \mapsto w$ then define a binary code h for $(S, \mathbf{p})$ with $h : s' \mapsto w0$ and $h : s'' \mapsto w1$.
Nothing changes for other symbols.

The rule **H1** constructs a new, abstract source $(S^*, \mathbf{p}^*)$ s.t. $|S^*| = |S| - 1$ until we have a trivial source $(\{s^*\}, \mathbf{p}^* = (1))$.

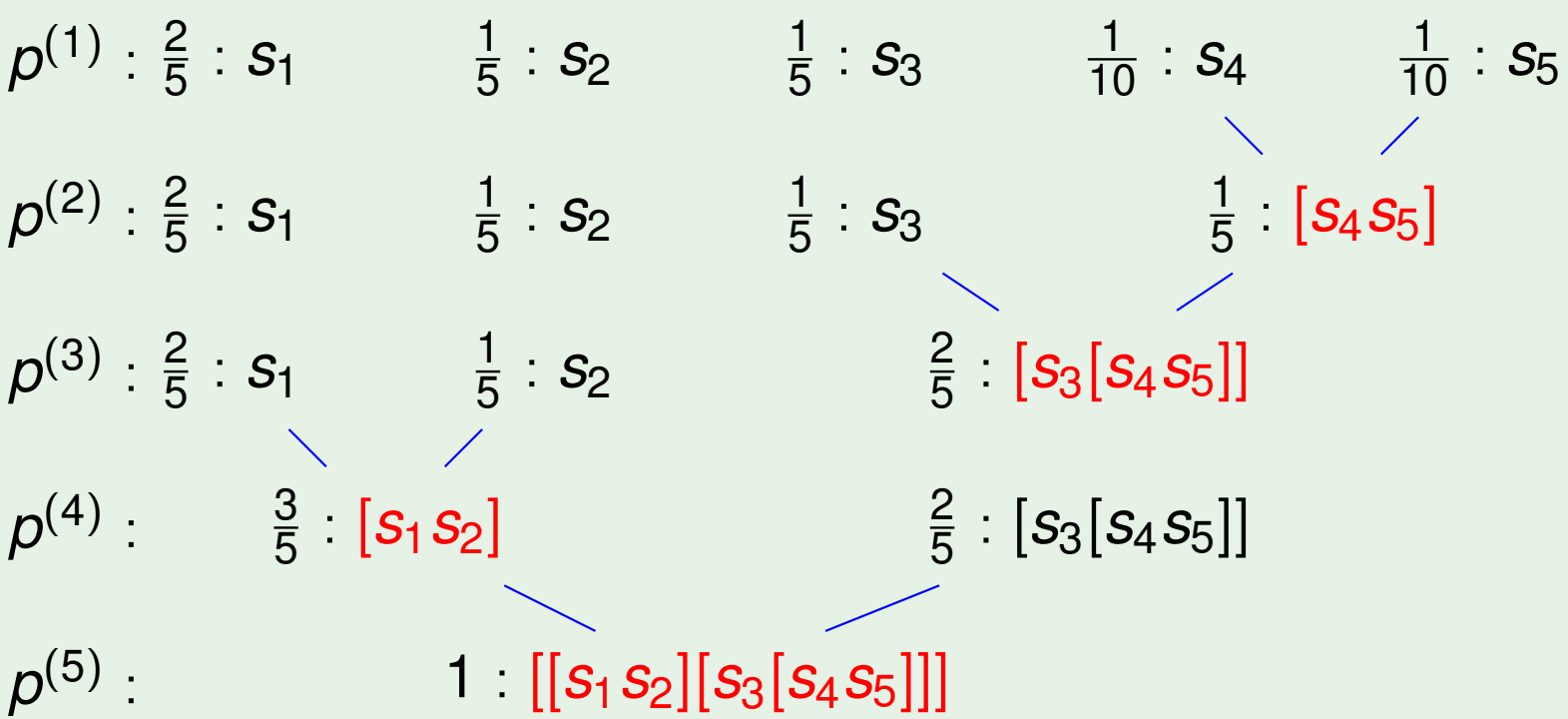
The rule **H2** constructs a new, more concrete code h from h^* for a larger S by "splitting" a symbol s^* into s' and s''.

Example

Example

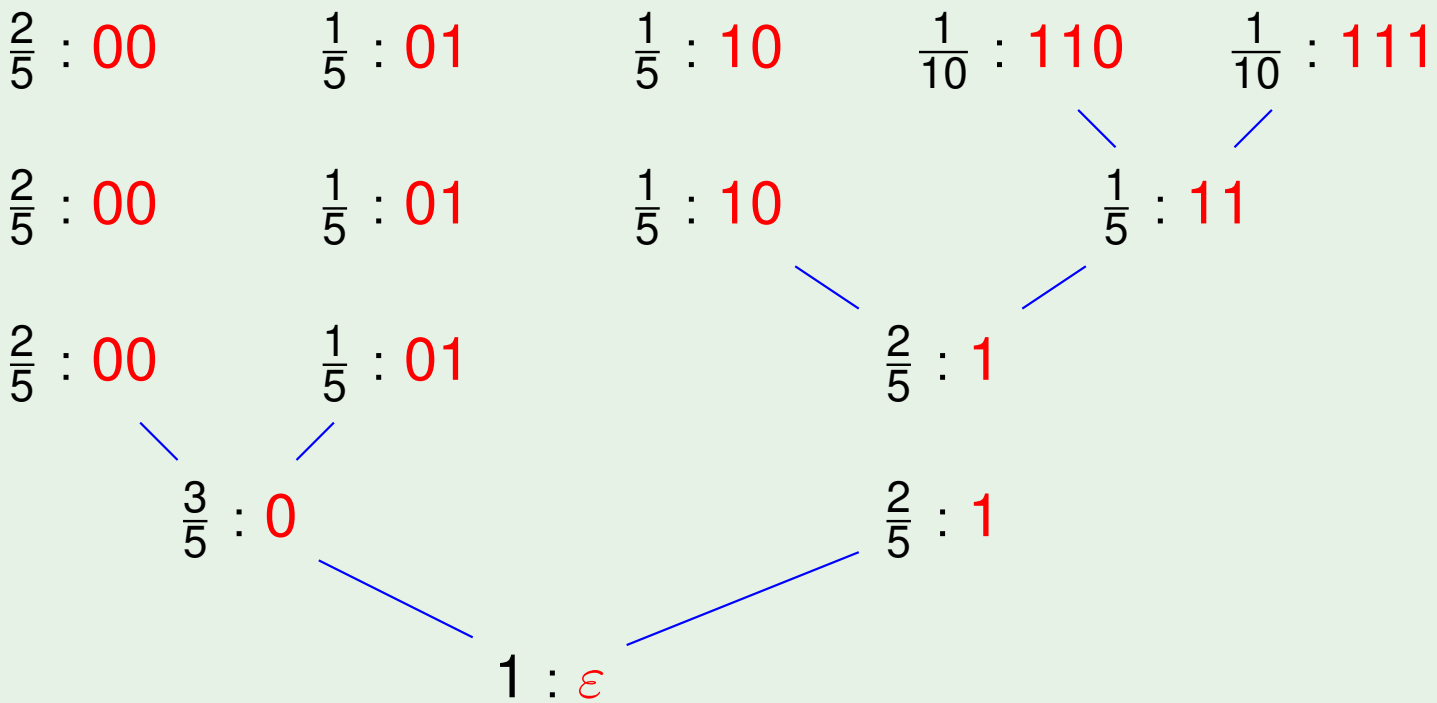
Given $\mathbf{p} = (\frac{2}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{10}, \frac{1}{10})$ as before.

Aggregating probabilities using Huffman's rule **H1**:



Example (cont.)

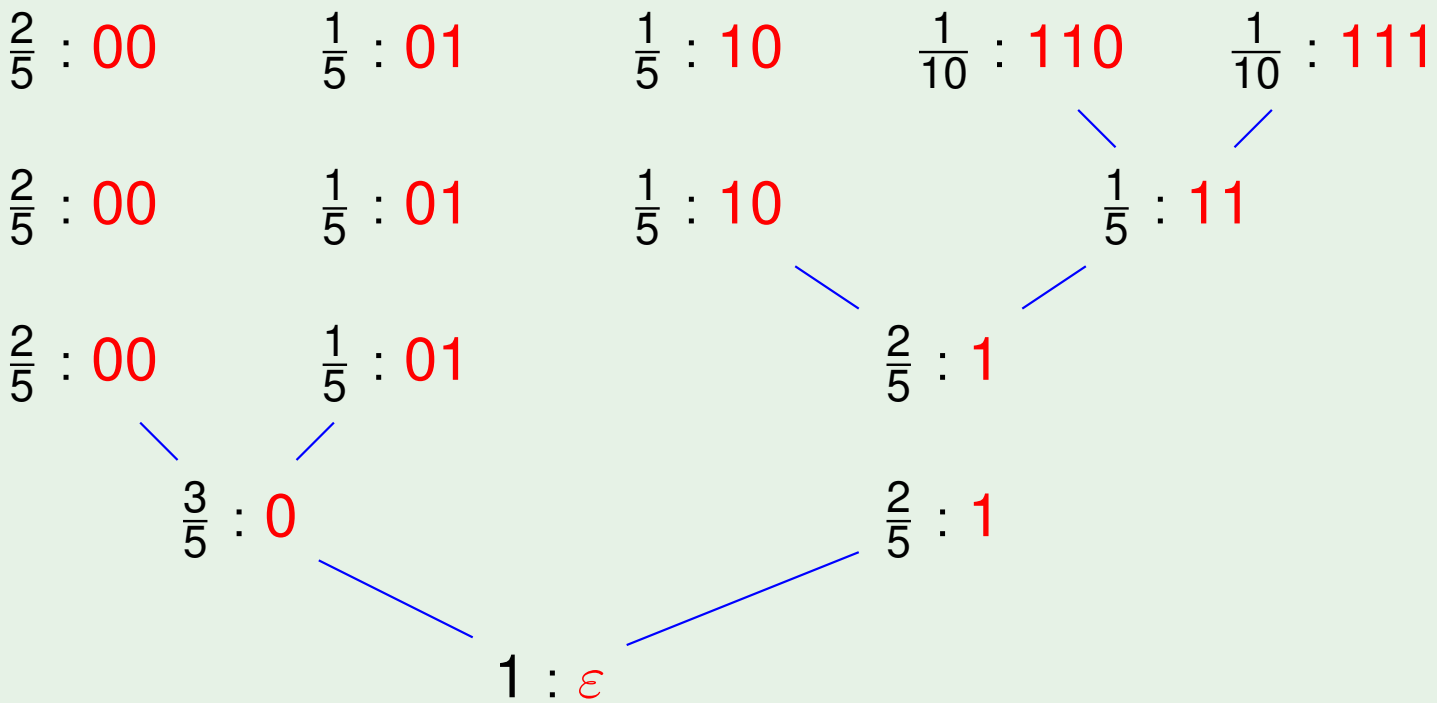
Splitting symbols using Huffman's rule **H2**:



Average codeword-length is $L = 2.2 \geq \mathbf{H}(\mathbf{p}) \approx 2.12$, quite close to theoretical optimum and better than $L_{SF} = 2.8$.

Example (cont.)

Splitting symbols using Huffman's rule **H2**:



Average codeword-length is $L = 2.2 \geq \mathbf{H}(\mathbf{p}) \approx 2.12$, quite close to theoretical optimum and better than $L_{SF} = 2.8$.

Quantifying the Increase in L

Lemma

Let codes h and h^* be defined as required by **H2**. *Then the average word-lengths of h and h^* satisfy:*

$$L(h) = L(h^*) + p^*_{s^*}$$

Proof.

Consider $|h^*(s^*)|$. Then $|h(s')| = |h(s'')| = |h^*(s^*)| + 1$. Hence

$$L(h) - L(h^*) = (p_{s'} + p_{s''})(|h^*(s^*)| + 1) - p^*_{s^*}|h^*(s^*)|$$

But since $p^*_{s^*} = p_{s'} + p_{p''}$ we have $L(h) - L(h^*) = p^*_{s^*}$. □

From trivial code ε for trivial source $(\{s^*\}, (1))$ we can control the change of L until we have a code for the original source.

Quantifying the Increase in L

Lemma

Let codes h and h^* be defined as required by **H2**. *Then the average word-lengths of h and h^* satisfy:*

$$L(h) = L(h^*) + p^*_{s^*}$$

Proof.

Consider $|h^*(s^*)|$. Then $|h(s')| = |h(s'')| = |h^*(s^*)| + 1$. Hence

$$L(h) - L(h^*) = (p_{s'} + p_{s''})(|h^*(s^*)| + 1) - p^*_{s^*}|h^*(s^*)|$$

But since $p^*_{s^*} = p_{s'} + p_{p''}$ we have $L(h) - L(h^*) = p^*_{s^*}$. □

From trivial code ε for trivial source $(\{s^*\}, (1))$ we can control the change of L until we have a code for the original source.

Quantifying the Increase in L

Lemma

Let codes h and h^* be defined as required by **H2**. *Then the average word-lengths of h and h^* satisfy:*

$$L(h) = L(h^*) + p^*_{s^*}$$

Proof.

Consider $|h^*(s^*)|$. Then $|h(s')| = |h(s'')| = |h^*(s^*)| + 1$. Hence

$$L(h) - L(h^*) = (p_{s'} + p_{s''})(|h^*(s^*)| + 1) - p^*_{s^*}|h^*(s^*)|$$

But since $p^*_{s^*} = p_{s'} + p_{p''}$ we have $L(h) - L(h^*) = p^*_{s^*}$. □

From trivial code ε for trivial source $(\{s^*\}, (1))$ we can control the change of L until we have a code for the original source.

Optimality of Huffman Code

Theorem

Using Huffman's rules: If the code h^* is optimal for $(S^*, \mathbf{p}^*)$ then the code h is optimal for $(S, \mathbf{p})$.

Proof $*$.

Show actually: if h is not optimal then h^* is not optimal.
Suppose: h is not optimal, but c is an optimal code for $(S, \mathbf{p})$.

We know that c assigns to some $t', t'' \in S$ codewords $w0$ and $w1$ of maximal length. Take another disjoint pair of symbols $s', s'' \in S$ involved in the construction of h. Then

$$c(t') = w0, \quad c(t'') = w1, \quad c(s') = u, \quad c(s'') = v.$$

Define a new code c^* for $(S^*, \mathbf{p}^*)$ via:

$$c^*(t') = u, \quad c^*(t'') = v, \quad c^*(s^*) = w.$$

Optimality of Huffman Code

Theorem

Using Huffman's rules: If the code h^* is optimal for $(S^*, \mathbf{p}^*)$ then the code h is optimal for $(S, \mathbf{p})$.

Proof $*$.

Show actually: if h is not optimal then h^* is not optimal.
Suppose: h is not optimal, but c is an optimal code for $(S, \mathbf{p})$.

We know that c assigns to some $t', t'' \in S$ codewords $w0$ and $w1$ of maximal length. Take another disjoint pair of symbols $s', s'' \in S$ involved in the construction of h. Then

$$c(t') = w0, \quad c(t'') = w1, \quad c(s') = u, \quad c(s'') = v.$$

Define a new code c^* for $(S^*, \mathbf{p}^*)$ via:

$$c^*(t') = u, \quad c^*(t'') = v, \quad c^*(s^*) = w.$$

Optimality of Huffman Code

Theorem

Using Huffman's rules: If the code h^* is optimal for $(S^*, \mathbf{p}^*)$ then the code h is optimal for $(S, \mathbf{p})$.

Proof $*$.

Show actually: if h is not optimal then h^* is not optimal.
Suppose: h is not optimal, but c is an optimal code for $(S, \mathbf{p})$.

We know that c assigns to some $t', t'' \in S$ codewords $w0$ and $w1$ of maximal length. Take another disjoint pair of symbols $s', s'' \in S$ involved in the construction of h. Then

$$c(t') = w0, \quad c(t'') = w1, \quad c(s') = u, \quad c(s'') = v.$$

Define a new code c^* for $(S^*, \mathbf{p}^*)$ via:

$$c^*(t') = u, \quad c^*(t'') = v, \quad c^*(s^*) = w.$$

Optimality of Huffman Code

Theorem

Using Huffman's rules: If the code h^* is optimal for $(S^*, \mathbf{p}^*)$ then the code h is optimal for $(S, \mathbf{p})$.

Proof $*$.

Show actually: if h is not optimal then h^* is not optimal.
Suppose: h is not optimal, but c is an optimal code for $(S, \mathbf{p})$.

We know that c assigns to some $t', t'' \in S$ codewords $w0$ and $w1$ of maximal length. Take another disjoint pair of symbols $s', s'' \in S$ involved in the construction of h. Then

$$c(t') = w0, \quad c(t'') = w1, \quad c(s') = u, \quad c(s'') = v.$$

Define a new code c^* for $(S^*, \mathbf{p}^*)$ via:

$$c^*(t') = u, \quad c^*(t'') = v, \quad c^*(s^*) = w.$$

Proof of Optimality (cont.)

Proof (cont).

With this we can give the difference (using $p^*_{s*} = p_{s'} + p_{s''}$):

$$\begin{aligned} L(c) - L(c^*) &= (p_{t'} + p_{t''})|c(t')| + p_{s'}|c(s')| + p_{s''}|c(s'')| - \\ &- p_{t'}|c(s')| - p_{t''}|c(s'')| - p^*_{s*}(|c(t')| - 1) = \\ &= p^*_{s*} + (p_{t'} - p_{s'})(|c(t')| - |c(s')|) + \\ &\quad + (p_{t''} - p_{s''})(|c(t'')| - |c(s'')|). \end{aligned}$$

Maximality for t, t'' in c. i.e. $|c(t')| > |c(s')|$ and $|c(t'')| > |c(s'')|$, and $p_{t'} \geq p_{s'}$ and $p_{t''} \geq p_{s''}$ imply that $L(c) - L(c^*) \geq p^*_{s*}$.
The previous lemma states that $L(h) = L(h^*) + p^*_{s*}$ and we assumed $L(c) < L(h)$, therefore:

$$L(c^*) \leq L(c) - p^*_{s*} < L(h) - p^*_{s*} = L(h^*)$$

so h^* is not optimal (generalise for non-disjoint s', s'', t', t'').

Proof of Optimality (cont.)

Proof (cont).

With this we can give the difference (using $p^*_{s*} = p_{s'} + p_{s''}$):

$$\begin{aligned} L(c) - L(c^*) &= (p_{t'} + p_{t''})|c(t')| + p_{s'}|c(s')| + p_{s''}|c(s'')| - \\ &- p_{t'}|c(s')| - p_{t''}|c(s'')| - p^*_{s*}(|c(t')| - 1) = \\ &= p^*_{s*} + (p_{t'} - p_{s'})(|c(t')| - |c(s')|) + \\ &\quad + (p_{t''} - p_{s''})(|c(t'')| - |c(s'')|). \end{aligned}$$

Maximality for t, t'' in c. i.e. $|c(t')| > |c(s')|$ and $|c(t'')| > |c(s'')|$, and $p_{t'} \geq p_{s'}$ and $p_{t''} \geq p_{s''}$ imply that $L(c) - L(c^*) \geq p^*_{s*}$.

The previous lemma states that $L(h) = L(h^*) + p^*_{s*}$ and we assumed $L(c) < L(h)$, therefore:

$$L(c^*) \leq L(c) - p^*_{s*} < L(h) - p^*_{s*} = L(h^*)$$

so h^* is not optimal (generalise for non-disjoint s', s'', t', t'').

Proof of Optimality (cont.)

Proof (cont).

With this we can give the difference (using $p^*_{s*} = p_{s'} + p_{s''}$):

$$\begin{aligned} L(c) - L(c^*) &= (p_{t'} + p_{t''})|c(t')| + p_{s'}|c(s')| + p_{s''}|c(s'')| - \\ &- p_{t'}|c(s')| - p_{t''}|c(s'')| - p^*_{s*}(|c(t')| - 1) = \\ &= p^*_{s*} + (p_{t'} - p_{s'})(|c(t')| - |c(s')|) + \\ &\quad + (p_{t''} - p_{s''})(|c(t'')| - |c(s'')|). \end{aligned}$$

Maximality for t, t'' in c. i.e. $|c(t')| > |c(s')|$ and $|c(t'')| > |c(s'')|$, and $p_{t'} \geq p_{s'}$ and $p_{t''} \geq p_{s''}$ imply that $L(c) - L(c^*) \geq p^*_{s*}$.
The previous lemma states that $L(h) = L(h^*) + p^*_{s*}$ and we assumed $L(c) < L(h)$, therefore:

$$L(c^*) \leq L(c) - p^*_{s*} < L(h) - p^*_{s*} = L(h^*)$$

so h^* is not optimal (generalise for non-disjoint s', s'', t', t'').

Proof of Optimality (cont.)

Proof (cont).

With this we can give the difference (using $p^*_{s*} = p_{s'} + p_{s''}$):

$$\begin{aligned} L(c) - L(c^*) &= (p_{t'} + p_{t''})|c(t')| + p_{s'}|c(s')| + p_{s''}|c(s'')| - \\ &- p_{t'}|c(s')| - p_{t''}|c(s'')| - p^*_{s*}(|c(t')| - 1) = \\ &= p^*_{s*} + (p_{t'} - p_{s'})(|c(t')| - |c(s')|) + \\ &\quad + (p_{t''} - p_{s''})(|c(t'')| - |c(s'')|). \end{aligned}$$

Maximality for t, t'' in c. i.e. $|c(t')| > |c(s')|$ and $|c(t'')| > |c(s'')|$, and $p_{t'} \geq p_{s'}$ and $p_{t''} \geq p_{s''}$ imply that $L(c) - L(c^*) \geq p^*_{s*}$.
The previous lemma states that $L(h) = L(h^*) + p^*_{s*}$ and we assumed $L(c) < L(h)$, therefore:

$$L(c^*) \leq L(c) - p^*_{s*} < L(h) - p^*_{s*} = L(h^*)$$

so h^* is not optimal (generalise for non-disjoint s', s'', t', t'').

Coding in Blocks

Previously we considered single symbol encoding $c : S \to T^*$.
We now consider block coding, $c : S^2 \to \mathbb{B}^*$, $c : S^3 \to \mathbb{B}^*$, etc.

Example (Biggs, Ex 4.1)

Consider $c : \mathbb{B} \to \mathbb{B}^*$ with $\mathbf{p} = (0.9, 0.1)$, it has entropy $\mathbf{H}(\mathbf{p}) \approx 0.469$ average codeword-length $L_1 = 1$ (at best).

Consider $c : \mathbb{B}^2 \to \mathbb{B}^*$ with $\mathbf{p} = (0.81, 0.09, 0.09, 0.01)$, which allows a Huffman code $00 \mapsto 0, 01 \mapsto 10, 10 \mapsto 110, 11 \mapsto 111$. Now, $L_2 = 1.29$ or $\frac{L_2}{2} = 0.645$.

Consider $c : \mathbb{B}^3 \to \mathbb{B}^*$ with $\mathbf{p} = (0.729, 0.081, 0.081, 0.009, 0.081, 0.009, 0.009, 0.001)$ with optimal Huffman code we get $L_3 = 1.598$, or $\frac{L_3}{3} = 0.533$.

This way we approach the theoretical minimum $\mathbf{H}(\mathbf{p})$.

Coding in Blocks

Previously we considered single symbol encoding $c : S \to T^*$.
We now consider block coding, $c : S^2 \to \mathbb{B}^*$, $c : S^3 \to \mathbb{B}^*$, etc.

Example (Biggs, Ex 4.1)

Consider $c : \mathbb{B} \to \mathbb{B}^*$ with $\mathbf{p} = (0.9, 0.1)$, it has entropy $\mathbf{H}(\mathbf{p}) \approx 0.469$ average codeword-length $L_1 = 1$ (at best).

Consider $c : \mathbb{B}^2 \to \mathbb{B}^*$ with $\mathbf{p} = (0.81, 0.09, 0.09, 0.01)$, which allows a Huffman code $00 \mapsto 0, 01 \mapsto 10, 10 \mapsto 110, 11 \mapsto 111$. Now, $L_2 = 1.29$ or $\frac{L_2}{2} = 0.645$.

Consider $c : \mathbb{B}^3 \to \mathbb{B}^*$ with $\mathbf{p} = (0.729, 0.081, 0.081, 0.009, 0.081, 0.009, 0.009, 0.001)$ with optimal Huffman code we get $L_3 = 1.598$, or $\frac{L_3}{3} = 0.533$.

This way we approach the theoretical minimum $\mathbf{H}(\mathbf{p})$.

Coding in Blocks

Previously we considered single symbol encoding $c : S \to T^*$.
We now consider block coding, $c : S^2 \to \mathbb{B}^*$, $c : S^3 \to \mathbb{B}^*$, etc.

Example (Biggs, Ex 4.1)

Consider $c : \mathbb{B} \to \mathbb{B}^*$ with $\mathbf{p} = (0.9, 0.1)$, it has entropy $\mathbf{H}(\mathbf{p}) \approx 0.469$ average codeword-length $L_1 = 1$ (at best).

Consider $c : \mathbb{B}^2 \to \mathbb{B}^*$ with $\mathbf{p} = (0.81, 0.09, 0.09, 0.01)$, which allows a Huffman code $00 \mapsto 0, 01 \mapsto 10, 10 \mapsto 110, 11 \mapsto 111$.
Now, $L_2 = 1.29$ or $\frac{L_2}{2} = 0.645$.

Consider $c : \mathbb{B}^3 \to \mathbb{B}^*$ with $\mathbf{p} = (0.729, 0.081, 0.081, 0.009, 0.081, 0.009, 0.009, 0.001)$ with optimal Huffman code we get $L_3 = 1.598$, or $\frac{L_3}{3} = 0.533$.

This way we approach the theoretical minimum $\mathbf{H}(\mathbf{p})$.

Coding in Blocks

Previously we considered single symbol encoding $c : S \to T^*$.
We now consider block coding, $c : S^2 \to \mathbb{B}^*$, $c : S^3 \to \mathbb{B}^*$, etc.

Example (Biggs, Ex 4.1)

Consider $c : \mathbb{B} \to \mathbb{B}^*$ with $\mathbf{p} = (0.9, 0.1)$, it has entropy $\mathbf{H}(\mathbf{p}) \approx 0.469$ average codeword-length $L_1 = 1$ (at best).

Consider $c : \mathbb{B}^2 \to \mathbb{B}^*$ with $\mathbf{p} = (0.81, 0.09, 0.09, 0.01)$, which allows a Huffman code $00 \mapsto 0, 01 \mapsto 10, 10 \mapsto 110, 11 \mapsto 111$. Now, $L_2 = 1.29$ or $\frac{L_2}{2} = 0.645$.

Consider $c : \mathbb{B}^3 \to \mathbb{B}^*$ with $\mathbf{p} = (0.729, 0.081, 0.081, 0.009, 0.081, 0.009, 0.009, 0.001)$ with optimal Huffman code we get $L_3 = 1.598$, or $\frac{L_3}{3} = 0.533$.

This way we approach the theoretical minimum $\mathbf{H}(\mathbf{p})$.

Coding in Blocks

Previously we considered single symbol encoding $c : S \to T^*$.
We now consider block coding, $c : S^2 \to \mathbb{B}^*$, $c : S^3 \to \mathbb{B}^*$, etc.

Example (Biggs, Ex 4.1)

Consider $c : \mathbb{B} \to \mathbb{B}^*$ with $\mathbf{p} = (0.9, 0.1)$, it has entropy $\mathbf{H}(\mathbf{p}) \approx 0.469$ average codeword-length $L_1 = 1$ (at best).

Consider $c : \mathbb{B}^2 \to \mathbb{B}^*$ with $\mathbf{p} = (0.81, 0.09, 0.09, 0.01)$, which allows a Huffman code $00 \mapsto 0, 01 \mapsto 10, 10 \mapsto 110, 11 \mapsto 111$. Now, $L_2 = 1.29$ or $\frac{L_2}{2} = 0.645$.

Consider $c : \mathbb{B}^3 \to \mathbb{B}^*$ with $\mathbf{p} = (0.729, 0.081, 0.081, 0.009, 0.081, 0.009, 0.009, 0.001)$ with optimal Huffman code we get $L_3 = 1.598$, or $\frac{L_3}{3} = 0.533$.

This way we approach the theoretical minimum $\mathbf{H}(\mathbf{p})$.

Products and Distributions

Consider two sets/alphabets

$$S' = \{s'_1, s'_2, \cdots, s'_m\} \text{ and } S'' = \{s''_1, s''_2, \cdots, s''_n\}$$

and a probability distribution $\mathbf{p}$ on their cartesian product S:

$$S' \times S'' = \{\langle s', s'' \rangle \mid s' \in S', s'' \in S''\} = \{s's'' \mid s' \in S', s'' \in S''\}.$$

Definition

Given a distribution $\mathbf{p} = (p_{ij})_{i,j}$ on $S = S' \times S''$. The marginal distributions $\mathbf{p}'$ on S' and $\mathbf{p}''$ on S'' are given by

$$p'_i = \sum_{j=1}^{n} p_{ij} \ (i = 1, \ldots, m) \text{ and } p''_j = \sum_{i=1}^{m} p_{ij} \ (j = 1, \ldots, n)$$

$\mathbf{p}'$ and $\mathbf{p}''$ are independent if $p_{ij} = p'_i p''_j$ or $\mathbf{p} = \mathbf{p}' \otimes \mathbf{p}''$.

Products and Distributions

Consider two sets/alphabets

$$S' = \{s'_1, s'_2, \cdots, s'_m\} \text{ and } S'' = \{s''_1, s''_2, \cdots, s''_n\}$$

and a probability distribution $\mathbf{p}$ on their cartesian product S:

$$S' \times S'' = \{\langle s', s'' \rangle \mid s' \in S', s'' \in S''\} = \{s's'' \mid s' \in S', s'' \in S''\}.$$

Definition

Given a distribution $\mathbf{p} = (p_{ij})_{i,j}$ on $S = S' \times S''$. The marginal distributions $\mathbf{p}'$ on S' and $\mathbf{p}''$ on S'' are given by

$$p'_i = \sum_{j=1}^{n} p_{ij} \ \ (i = 1, \ldots, m) \text{ and } p''_j = \sum_{i=1}^{m} p_{ij} \ \ (j = 1, \ldots, n)$$

$\mathbf{p}'$ and $\mathbf{p}''$ are independent if $p_{ij} = p'_i p''_j$ or $\mathbf{p} = \mathbf{p}' \otimes \mathbf{p}''$.

Products and Distributions

Consider two sets/alphabets

$$S' = \{s'_1, s'_2, \cdots, s'_m\} \text{ and } S'' = \{s''_1, s''_2, \cdots, s''_n\}$$

and a probability distribution $\mathbf{p}$ on their cartesian product S:

$$S' \times S'' = \{\langle s', s'' \rangle \mid s' \in S', s'' \in S''\} = \{s's'' \mid s' \in S', s'' \in S''\}.$$

Definition

Given a distribution $\mathbf{p} = (p_{ij})_{i,j}$ on $S = S' \times S''$. The marginal distributions $\mathbf{p}'$ on S' and $\mathbf{p}''$ on S'' are given by

$$p'_i = \sum_{j=1}^{n} p_{ij} \ \ (i = 1, \ldots, m) \text{ and } p''_j = \sum_{i=1}^{m} p_{ij} \ \ (j = 1, \ldots, n)$$

$\mathbf{p}'$ and $\mathbf{p}''$ are independent if $p_{ij} = p'_i p''_j$ or $\mathbf{p} = \mathbf{p}' \otimes \mathbf{p}''$.

Entropies of Products

Theorem

For a distributions $\mathbf{p}$ *on* $S' \times S''$ *and its marginals distribution* $\mathbf{p}'$ *and* $\mathbf{p}''$ *we have:*

$$\mathbf{H}(\mathbf{p}) \leq \mathbf{H}(\mathbf{p}') + \mathbf{H}(\mathbf{p}'')$$

Equality holds if and only if $\mathbf{p}'$ *and* $\mathbf{p}''$ *are independent.*

Proof.

We can describe the right hand side by

$$\begin{aligned}
\mathbf{H}(\mathbf{p}') + \mathbf{H}(\mathbf{p}'') &= \sum_i p'_i \log\left(\frac{1}{p'_i}\right) + \sum_j p''_j \log\left(\frac{1}{p''_j}\right) \\
&= \sum_i \sum_j p_{ij} \log\left(\frac{1}{p'_i}\right) + \sum_j \sum_i p_{ij} \log\left(\frac{1}{p''_j}\right)
\end{aligned}$$

Entropies of Products

Theorem

For a distributions $\mathbf{p}$ *on* $S' \times S''$ *and its marginals distribution* $\mathbf{p}'$ *and* $\mathbf{p}''$ *we have:*

$$\mathbf{H}(\mathbf{p}) \leq \mathbf{H}(\mathbf{p}') + \mathbf{H}(\mathbf{p}'')$$

Equality holds if and only if $\mathbf{p}'$ *and* $\mathbf{p}''$ *are independent.*

Proof.

We can describe the right hand side by

$$\begin{aligned}
\mathbf{H}(\mathbf{p}') + \mathbf{H}(\mathbf{p}'') &= \sum_i p'_i \log\left(\frac{1}{p'_i}\right) + \sum_j p''_j \log\left(\frac{1}{p''_j}\right) \\
&= \sum_i \sum_j p_{ij} \log\left(\frac{1}{p'_i}\right) + \sum_j \sum_i p_{ij} \log\left(\frac{1}{p''_j}\right)
\end{aligned}$$

Proof of Product Property (cont.)

Proof (cont).

$$\mathbf{H}(\mathbf{p}') + \mathbf{H}(\mathbf{p}'') = \sum_i \sum_j p_{ij} \log\left(\frac{1}{p'_i p''_j}\right) = \sum_{ij} p_{ij} \log\left(\frac{1}{p'_i p''_j}\right)$$

If we consider the left hand side, $\mathbf{H}(\mathbf{p})$, we define first a probability distribution $\mathbf{q} = \mathbf{p}' \otimes \mathbf{p}''$ on $S' \times S''$ via $q_{ij} = p'_i p''_j$.

Applying the Comparison Theorem, we can conclude:

$$\mathbf{H}(\mathbf{p}) \leq \sum_{ij} p_{ij} \log\left(\frac{1}{q_{ij}}\right) = \sum_{ij} p_{ij} \log\left(\frac{1}{p'_i p''_j}\right) = \mathbf{H}(\mathbf{p}') + \mathbf{H}(\mathbf{p}'').$$

Equality iff $\mathbf{q} = \mathbf{p}$, i.e. $\mathbf{p}'$ and $\mathbf{p}''$ are independent. □

Proof of Product Property (cont.)

Proof (cont).

$$\mathbf{H}(\mathbf{p}') + \mathbf{H}(\mathbf{p}'') = \sum_i \sum_j p_{ij} \log\left(\frac{1}{p'_i p''_j}\right) = \sum_{ij} p_{ij} \log\left(\frac{1}{p'_i p''_j}\right)$$

If we consider the left hand side, $\mathbf{H}(\mathbf{p})$, we define first a probability distribution $\mathbf{q} = \mathbf{p}' \otimes \mathbf{p}''$ on $S' \times S''$ via $q_{ij} = p'_i p''_j$.

Applying the Comparison Theorem, we can conclude:

$$\mathbf{H}(\mathbf{p}) \leq \sum_{ij} p_{ij} \log\left(\frac{1}{q_{ij}}\right) = \sum_{ij} p_{ij} \log\left(\frac{1}{p'_i p''_j}\right) = \mathbf{H}(\mathbf{p}') + \mathbf{H}(\mathbf{p}'').$$

Equality iff $\mathbf{q} = \mathbf{p}$, i.e. $\mathbf{p}'$ and $\mathbf{p}''$ are independent. □

Proof of Product Property (cont.)

Proof (cont).

$$\mathbf{H}(\mathbf{p}') + \mathbf{H}(\mathbf{p}'') = \sum_i \sum_j p_{ij} \log\left(\frac{1}{p'_i p''_j}\right) = \sum_{ij} p_{ij} \log\left(\frac{1}{p'_i p''_j}\right)$$

If we consider the left hand side, $\mathbf{H}(\mathbf{p})$, we define first a probability distribution $\mathbf{q} = \mathbf{p}' \otimes \mathbf{p}''$ on $S' \times S''$ via $q_{ij} = p'_i p''_j$.

Applying the Comparison Theorem, we can conclude:

$$\mathbf{H}(\mathbf{p}) \leq \sum_{ij} p_{ij} \log\left(\frac{1}{q_{ij}}\right) = \sum_{ij} p_{ij} \log\left(\frac{1}{p'_i p''_j}\right) = \mathbf{H}(\mathbf{p}') + \mathbf{H}(\mathbf{p}'').$$

Equality iff $\mathbf{q} = \mathbf{p}$, i.e. $\mathbf{p}'$ and $\mathbf{p}''$ are independent. □

Proof of Product Property (cont.)

Proof (cont).

$$\mathbf{H}(\mathbf{p}') + \mathbf{H}(\mathbf{p}'') = \sum_i \sum_j p_{ij} \log\left(\frac{1}{p'_i p''_j}\right) = \sum_{ij} p_{ij} \log\left(\frac{1}{p'_i p''_j}\right)$$

If we consider the left hand side, $\mathbf{H}(\mathbf{p})$, we define first a probability distribution $\mathbf{q} = \mathbf{p}' \otimes \mathbf{p}''$ on $S' \times S''$ via $q_{ij} = p'_i p''_j$.

Applying the Comparison Theorem, we can conclude:

$$\mathbf{H}(\mathbf{p}) \leq \sum_{ij} p_{ij} \log\left(\frac{1}{q_{ij}}\right) = \sum_{ij} p_{ij} \log\left(\frac{1}{p'_i p''_j}\right) = \mathbf{H}(\mathbf{p}') + \mathbf{H}(\mathbf{p}'').$$

Equality iff $\mathbf{q} = \mathbf{p}$, i.e. $\mathbf{p}'$ and $\mathbf{p}''$ are independent. □

Stationary Sources

Definition

A source emitting a stream $\sigma_1\sigma_2\sigma_3\cdots$ is called stationary if, for any positive integers $\ell_1, \ell_2, \ldots, \ell_r$, the probabilities

$$\Pr(\sigma_{k+\ell_1} = s_1, \sigma_{k+\ell_2} = s_2, \ldots, \sigma_{k+\ell_r} = s_r)$$

depend only on the string $s_1 s_2 \cdots s_r$ and not on k.

Typically, $\ell_1 = 1, \ell_2 = 2, \ldots, \ell_r = r$. With the notation:

$$\mathbf{p}^r(s_1 s_2 \cdots s_r) = \Pr(\sigma_{k+1} = s_1, \sigma_{k+2} = s_2, \ldots, \sigma_{k+r} = s_r)$$

distribution on S^r, we have for any stationary source that:

$$\mathbf{p}^{r-1}(s_1 s_2 \cdots s_{r-1}) = \sum_{s \in S} \mathbf{p}^r(s_1 s_2 \cdots s_{r-1} s) = \sum_{s \in S} \mathbf{p}^r(s s_1 s_2 \cdots s_{r-1})$$

A memoryless source is a stationary source with

$$\mathbf{p}^r(s_1 s_2 \cdots s_r) = \mathbf{p}^1(s_1)\mathbf{p}^1(s_2)\cdots\mathbf{p}^1(s_r).$$

Stationary Sources

Definition

A source emitting a stream $\sigma_1\sigma_2\sigma_3\cdots$ is called stationary if, for any positive integers $\ell_1, \ell_2, \ldots, \ell_r$, the probabilities

$$\Pr(\sigma_{k+\ell_1} = s_1, \sigma_{k+\ell_2} = s_2, \ldots, \sigma_{k+\ell_r} = s_r)$$

depend only on the string $s_1 s_2 \cdots s_r$ and not on k.

Typically, $\ell_1 = 1, \ell_2 = 2, \ldots, \ell_r = r$. With the notation:

$$\mathbf{p}^r(s_1 s_2 \cdots s_r) = \Pr(\sigma_{k+1} = s_1, \sigma_{k+2} = s_2, \ldots, \sigma_{k+r} = s_r)$$

distribution on S^r, we have for any stationary source that:

$$\mathbf{p}^{r-1}(s_1 s_2 \cdots s_{r-1}) = \sum_{s \in S} \mathbf{p}^r(s_1 s_2 \cdots s_{r-1} s) = \sum_{s \in S} \mathbf{p}^r(s s_1 s_2 \cdots s_{r-1})$$

A memoryless source is a stationary source with

$$\mathbf{p}^r(s_1 s_2 \cdots s_r) = \mathbf{p}^1(s_1)\mathbf{p}^1(s_2)\cdots\mathbf{p}^1(s_r).$$

Stationary Sources

Definition

A source emitting a stream $\sigma_1\sigma_2\sigma_3\cdots$ is called stationary if, for any positive integers $\ell_1, \ell_2, \ldots, \ell_r$, the probabilities

$$\Pr(\sigma_{k+\ell_1} = s_1, \sigma_{k+\ell_2} = s_2, \ldots, \sigma_{k+\ell_r} = s_r)$$

depend only on the string $s_1 s_2 \cdots s_r$ and not on k.

Typically, $\ell_1 = 1, \ell_2 = 2, \ldots, \ell_r = r$. With the notation:

$$\mathbf{p}^r(s_1 s_2 \cdots s_r) = \Pr(\sigma_{k+1} = s_1, \sigma_{k+2} = s_2, \ldots, \sigma_{k+r} = s_r)$$

distribution on S^r, we have for any stationary source that:

$$\mathbf{p}^{r-1}(s_1 s_2 \cdots s_{r-1}) = \sum_{s \in S} \mathbf{p}^r(s_1 s_2 \cdots s_{r-1} s) = \sum_{s \in S} \mathbf{p}^r(s s_1 s_2 \cdots s_{r-1})$$

A memoryless source is a stationary source with

$$\mathbf{p}^r(s_1 s_2 \cdots s_r) = \mathbf{p}^1(s_1)\mathbf{p}^1(s_2)\cdots\mathbf{p}^1(s_r).$$

Stationary Sources

Definition

A source emitting a stream $\sigma_1\sigma_2\sigma_3\cdots$ is called stationary if, for any positive integers $\ell_1, \ell_2, \ldots, \ell_r$, the probabilities

$$\Pr(\sigma_{k+\ell_1} = s_1, \sigma_{k+\ell_2} = s_2, \ldots, \sigma_{k+\ell_r} = s_r)$$

depend only on the string $s_1 s_2 \cdots s_r$ and not on k.

Typically, $\ell_1 = 1, \ell_2 = 2, \ldots, \ell_r = r$. With the notation:

$$\mathbf{p}^r(s_1 s_2 \cdots s_r) = \Pr(\sigma_{k+1} = s_1, \sigma_{k+2} = s_2, \ldots, \sigma_{k+r} = s_r)$$

distribution on S^r, we have for any stationary source that:

$$\mathbf{p}^{r-1}(s_1 s_2 \cdots s_{r-1}) = \sum_{s \in S} \mathbf{p}^r(s_1 s_2 \cdots s_{r-1} s) = \sum_{s \in S} \mathbf{p}^r(s s_1 s_2 \cdots s_{r-1})$$

A memoryless source is a stationary source with

$$\mathbf{p}^r(s_1 s_2 \cdots s_r) = \mathbf{p}^1(s_1)\mathbf{p}^1(s_2)\cdots\mathbf{p}^1(s_r).$$

Example of a Stationary Source

Example

Take the alphabet $S = \{a, b, c\}$. Define a probability $\mathbf{p}^2$ distribution on $S \times S$ via (row and columns correspond to elements in S):

$$\mathbf{p}^2 = \begin{matrix} 0.39 & 0.17 & 0.04 \\ 0.15 & 0.11 & 0.04 \\ 0.06 & 0.02 & 0.02 \end{matrix}$$

The distribution $\mathbf{p}^1$ on S is given by

$$\begin{aligned} \mathbf{p}^1(a) &= \mathbf{p}^2(aa) + \mathbf{p}^2(ab) + \mathbf{p}^2(ac) &= 0.6 \\ \mathbf{p}^1(b) &= \mathbf{p}^2(ba) + \mathbf{p}^2(bb) + \mathbf{p}^2(bc) &= 0.3 \\ \mathbf{p}^1(c) &= \mathbf{p}^2(ca) + \mathbf{p}^2(cb) + \mathbf{p}^2(cc) &= 0.1 \end{aligned}$$

It is not memoryless as $\mathbf{p}^2(aa) = 0.39$ but $\mathbf{p}^1(a)\mathbf{p}^1(a) = 0.36$. $\mathbf{H}(\mathbf{p}^1) \approx 1.295$ and $\mathbf{H}(\mathbf{p}^2) \approx 2.520 < \mathbf{H}(\mathbf{p}^1) + \mathbf{H}(\mathbf{p}^1)$.

Example of a Stationary Source

Example

Take the alphabet $S = \{a, b, c\}$. Define a probability $\mathbf{p}^2$ distribution on $S \times S$ via (row and columns correspond to elements in S):

$$\mathbf{p}^2 = \begin{matrix} 0.39 & 0.17 & 0.04 \\ 0.15 & 0.11 & 0.04 \\ 0.06 & 0.02 & 0.02 \end{matrix}$$

The distribution $\mathbf{p}^1$ on S is given by

$$\begin{aligned} \mathbf{p}^1(a) &= \mathbf{p}^2(aa) + \mathbf{p}^2(ab) + \mathbf{p}^2(ac) = 0.6 \\ \mathbf{p}^1(b) &= \mathbf{p}^2(ba) + \mathbf{p}^2(bb) + \mathbf{p}^2(bc) = 0.3 \\ \mathbf{p}^1(c) &= \mathbf{p}^2(ca) + \mathbf{p}^2(cb) + \mathbf{p}^2(cc) = 0.1 \end{aligned}$$

It is not memoryless as $\mathbf{p}^2(aa) = 0.39$ but $\mathbf{p}^1(a)\mathbf{p}^1(a) = 0.36$. $\mathbf{H}(\mathbf{p}^1) \approx 1.295$ and $\mathbf{H}(\mathbf{p}^2) \approx 2.520 < \mathbf{H}(\mathbf{p}^1) + \mathbf{H}(\mathbf{p}^1)$.

Example of a Stationary Source

Example

Take the alphabet $S = \{a, b, c\}$. Define a probability $\mathbf{p}^2$ distribution on $S \times S$ via (row and columns correspond to elements in S):

$$\mathbf{p}^2 = \begin{matrix} 0.39 & 0.17 & 0.04 \\ 0.15 & 0.11 & 0.04 \\ 0.06 & 0.02 & 0.02 \end{matrix}$$

The distribution $\mathbf{p}^1$ on S is given by

$$\begin{aligned} \mathbf{p}^1(a) &= \mathbf{p}^2(aa) + \mathbf{p}^2(ab) + \mathbf{p}^2(ac) &= 0.6 \\ \mathbf{p}^1(b) &= \mathbf{p}^2(ba) + \mathbf{p}^2(bb) + \mathbf{p}^2(bc) &= 0.3 \\ \mathbf{p}^1(c) &= \mathbf{p}^2(ca) + \mathbf{p}^2(cb) + \mathbf{p}^2(cc) &= 0.1 \end{aligned}$$

It is not memoryless as $\mathbf{p}^2(aa) = 0.39$ but $\mathbf{p}^1(a)\mathbf{p}^1(a) = 0.36$. $\mathbf{H}(\mathbf{p}^1) \approx 1.295$ and $\mathbf{H}(\mathbf{p}^2) \approx 2.520 < \mathbf{H}(\mathbf{p}^1) + \mathbf{H}(\mathbf{p}^1)$.

Entropy of a Stationary Source

Definition

The entropy **H** of a stationary source with probability distribution $\mathbf{p}^r$ is the infimum over the numbers

$$\mathbf{H} = \inf \frac{\mathbf{H}(\mathbf{p}^r)}{r} \quad r = 1, 2, 3, \ldots.$$

Theorem

For a memoryless stationary source its entropy $\mathbf{H} = \mathbf{H}(\mathbf{p}^1)$.

Proof.

For a memoryless sources $\mathbf{H}(\mathbf{p}^2) = 2\mathbf{H}(\mathbf{p}^1)$ (Product Entropy for independent distributions). By induction we have in general $\mathbf{H}(\mathbf{p}^r) = r\mathbf{H}(\mathbf{p}^1)$, thus $\inf \frac{\mathbf{H}(\mathbf{p}^r)}{r} = \frac{r\mathbf{H}(\mathbf{p}^1)}{r} = \mathbf{H}(\mathbf{p}^1)$. □

Entropy of a Stationary Source

Definition

The entropy **H** of a stationary source with probability distribution $\mathbf{p}^r$ is the infimum over the numbers

$$\mathbf{H} = \inf \frac{\mathbf{H}(\mathbf{p}^r)}{r} \quad r = 1, 2, 3, \ldots.$$

Theorem

For a memoryless stationary source its entropy $\mathbf{H} = \mathbf{H}(\mathbf{p}^1)$.

Proof.

For a memoryless sources $\mathbf{H}(\mathbf{p}^2) = 2\mathbf{H}(\mathbf{p}^1)$ (Product Entropy for independent distributions). By induction we have in general $\mathbf{H}(\mathbf{p}^r) = r\mathbf{H}(\mathbf{p}^1)$, thus $\inf \frac{\mathbf{H}(\mathbf{p}^r)}{r} = \frac{r\mathbf{H}(\mathbf{p}^1)}{r} = \mathbf{H}(\mathbf{p}^1)$. □

Entropy of a Stationary Source

Definition

The entropy **H** of a stationary source with probability distribution $\mathbf{p}^r$ is the infimum over the numbers

$$\mathbf{H} = \inf \frac{\mathbf{H}(\mathbf{p}^r)}{r} \quad r = 1, 2, 3, \ldots.$$

Theorem

For a memoryless stationary source its entropy $\mathbf{H} = \mathbf{H}(\mathbf{p}^1)$.

Proof.

For a memoryless sources $\mathbf{H}(\mathbf{p}^2) = 2\mathbf{H}(\mathbf{p}^1)$ (Product Entropy for independent distributions). By induction we have in general $\mathbf{H}(\mathbf{p}^r) = r\mathbf{H}(\mathbf{p}^1)$, thus $\inf \frac{\mathbf{H}(\mathbf{p}^r)}{r} = \frac{r\mathbf{H}(\mathbf{p}^1)}{r} = \mathbf{H}(\mathbf{p}^1)$. □

Sub-Block Coding

Lemma

Suppose n is a multiple of r then

$$\frac{\mathbf{H}(\mathbf{p}^n)}{n} \leq \frac{\mathbf{H}(\mathbf{p}^r)}{r}$$

Proof $*$.

Obvious for $r = n$. Assume $n = 2r$ then consider the distribution $\mathbf{p}^r$ on $\sigma_{\ell+1}\sigma_{\ell+2}\cdots\sigma_{\ell+r}$ and $\sigma_{\ell+r+1}\sigma_{\ell+r+2}\cdots\sigma_{\ell+2r}$ and also the distribution $\mathbf{p}^{2r}$ on $\sigma_{\ell+1}\sigma_{\ell+2}\cdots\sigma_{\ell+r}\sigma_{\ell+r+1}\sigma_{\ell+r+2}\cdots\sigma_{\ell+2r}$.
By previous theorem on Product Entropy we have:

$$\frac{\mathbf{H}(\mathbf{p}^{2r})}{2r} \leq \frac{1}{2r}(\mathbf{H}(\mathbf{p}^r) + \mathbf{H}(\mathbf{p}^r)) = \frac{\mathbf{H}(\mathbf{p}^r)}{r}.$$

Formally, by induction: $\mathbf{H}(\mathbf{p}^{qr}) \leq \mathbf{H}(\mathbf{p}^{(q-1)r}) + \mathbf{H}(\mathbf{p}^r)$. □

Sub-Block Coding

Lemma

Suppose n is a multiple of r then

$$\frac{\mathbf{H}(\mathbf{p}^n)}{n} \leq \frac{\mathbf{H}(\mathbf{p}^r)}{r}$$

Proof $*$.

Obvious for $r = n$. Assume $n = 2r$ then consider the distribution $\mathbf{p}^r$ on $\sigma_{\ell+1}\sigma_{\ell+2}\cdots\sigma_{\ell+r}$ and $\sigma_{\ell+r+1}\sigma_{\ell+r+2}\cdots\sigma_{\ell+2r}$ and also the distribution $\mathbf{p}^{2r}$ on $\sigma_{\ell+1}\sigma_{\ell+2}\cdots\sigma_{\ell+r}\sigma_{\ell+r+1}\sigma_{\ell+r+2}\cdots\sigma_{\ell+2r}$.
By previous theorem on Product Entropy we have:

$$\frac{\mathbf{H}(\mathbf{p}^{2r})}{2r} \leq \frac{1}{2r}(\mathbf{H}(\mathbf{p}^r) + \mathbf{H}(\mathbf{p}^r)) = \frac{\mathbf{H}(\mathbf{p}^r)}{r}.$$

Formally, by induction: $\mathbf{H}(\mathbf{p}^{qr}) \leq \mathbf{H}(\mathbf{p}^{(q-1)r}) + \mathbf{H}(\mathbf{p}^r)$. □

Coding Theorem: Stationary Sources

Theorem

For a stationary source on S and entropy $\mathbf{H}$. Given $\varepsilon > 0$ there exists $n > 0$ and a PF binary code $(S^n, \mathbf{p}^n)$ with L_n satisfying:

$$\frac{L_n}{n} < \mathbf{H} + \varepsilon.$$

Proof $*$.

By definition $\exists r$ s.t. $\frac{\mathbf{H}(\mathbf{p}^r)}{r} < \mathbf{H} + \frac{\varepsilon}{2}$. Choose $q \in \mathbb{Z}$ s.t. $q > \frac{2}{\varepsilon r}$ and $n = qr$, then $\frac{1}{n} < \frac{\varepsilon}{2}$. From last lemma: $\frac{\mathbf{H}(\mathbf{p}^n)}{n} \leq \frac{\mathbf{H}(\mathbf{p}^r)}{r}$. There exists a PF binary code for $\mathbf{p}^n$ with $L_n < \mathbf{H}(\mathbf{p}^n) + 1$. Hence:

$$\frac{L_n}{n} < \frac{\mathbf{H}(\mathbf{p}^n)}{n} + \frac{1}{n} < \frac{\mathbf{H}(\mathbf{p}^r)}{r} + \frac{\varepsilon}{2} < \mathbf{H} + \varepsilon \qquad \square$$

Coding Theorem: Stationary Sources

Theorem

For a stationary source on S and entropy $\mathbf{H}$. Given $\varepsilon > 0$ there exists $n > 0$ and a PF binary code $(S^n, \mathbf{p}^n)$ with L_n satisfying:

$$\frac{L_n}{n} < \mathbf{H} + \varepsilon.$$

Proof $*$.

By definition $\exists r$ s.t. $\frac{\mathbf{H}(\mathbf{p}^r)}{r} < \mathbf{H} + \frac{\varepsilon}{2}$. Choose $q \in \mathbb{Z}$ s.t. $q > \frac{2}{\varepsilon r}$ and $n = qr$, then $\frac{1}{n} < \frac{\varepsilon}{2}$. From last lemma: $\frac{\mathbf{H}(\mathbf{p}^n)}{n} \leq \frac{\mathbf{H}(\mathbf{p}^r)}{r}$. There exists a PF binary code for $\mathbf{p}^n$ with $L_n < \mathbf{H}(\mathbf{p}^n) + 1$. Hence:

$$\frac{L_n}{n} < \frac{\mathbf{H}(\mathbf{p}^n)}{n} + \frac{1}{n} < \frac{\mathbf{H}(\mathbf{p}^r)}{r} + \frac{\varepsilon}{2} < \mathbf{H} + \varepsilon \qquad \square$$

Algorithms for Data Compression

Question: Can we "recode" a file (seen as a source) such that we need less space to represent it.

Huffman's rule is not sufficient for data compression as we need to know the probabilities of symbols or blocks in advance, i.e. a priori.

Data Compression makes particular sense if based on blocks (not symbols). There are two possible approaches:

- The encoding of a block is calculated in isolation, i.e. we do not consider codes for other blocks.
- The codewords of blocks are updated 'online' which lead to adaptive coding.

Algorithms for Data Compression

Question: Can we "recode" a file (seen as a source) such that we need less space to represent it.

Huffman's rule is not sufficient for data compression as we need to know the probabilities of symbols or blocks in advance, i.e. a priori.

Data Compression makes particular sense if based on blocks (not symbols). There are two possible approaches:

- The encoding of a block is calculated in isolation, i.e. we do not consider codes for other blocks.
- The codewords of blocks are updated 'online' which lead to adaptive coding.

Algorithms for Data Compression

Question: Can we "recode" a file (seen as a source) such that we need less space to represent it.

Huffman's rule is not sufficient for data compression as we need to know the probabilities of symbols or blocks in advance, i.e. a priori.

Data Compression makes particular sense if based on blocks (not symbols). There are two possible approaches:

- The encoding of a block is calculated in isolation, i.e. we do not consider codes for other blocks.
- The codewords of blocks are updated 'online' which lead to adaptive coding.

Algorithms for Data Compression

Question: Can we "recode" a file (seen as a source) such that we need less space to represent it.

Huffman's rule is not sufficient for data compression as we need to know the probabilities of symbols or blocks in advance, i.e. a priori.

Data Compression makes particular sense if based on blocks (not symbols). There are two possible approaches:

- The encoding of a block is calculated in isolation, i.e. we do not consider codes for other blocks.
- The codewords of blocks are updated 'online' which lead to adaptive coding.

Algorithms for Data Compression

Question: Can we "recode" a file (seen as a source) such that we need less space to represent it.

Huffman's rule is not sufficient for data compression as we need to know the probabilities of symbols or blocks in advance, i.e. a priori.

Data Compression makes particular sense if based on blocks (not symbols). There are two possible approaches:

- The encoding of a block is calculated in isolation, i.e. we do not consider codes for other blocks.
- The codewords of blocks are updated 'online' which lead to adaptive coding.

Dynamic Dictionaries

Definition

A dictionary D based on an alphabet S is a (finite) sequence of (distinct) words in S^*:

$$D = d_1, d_2, d_3, \ldots, d_N.$$

We say that i is the (dictionary) index of d_i.

We can use a dictionary for a codebook, i.e. to record the encoding of symbols and blocks. There are two options:

Static Dictionary: Construct dictionary (at the start) which remains the unchanged (Arithmetic Coding).

Dynamic Dictionary: Update dictionary during data compression to adapt to frequency of blocks.

Dynamic Dictionaries

Definition

A dictionary D based on an alphabet S is a (finite) sequence of (distinct) words in S^*:

$$D = d_1, d_2, d_3, \ldots, d_N.$$

We say that i is the (dictionary) index of d_i.

We can use a dictionary for a codebook, i.e. to record the encoding of symbols and blocks. There are two options:

Static Dictionary: Construct dictionary (at the start) which remains the unchanged (Arithmetic Coding).

Dynamic Dictionary: Update dictionary during data compression to adapt to frequency of blocks.

Dynamic Dictionaries

Definition

A dictionary D based on an alphabet S is a (finite) sequence of (distinct) words in S^*:

$$D = d_1, d_2, d_3, \ldots, d_N.$$

We say that i is the (dictionary) index of d_i.

We can use a dictionary for a codebook, i.e. to record the encoding of symbols and blocks. There are two options:

Static Dictionary: Construct dictionary (at the start) which remains the unchanged (Arithmetic Coding).

Dynamic Dictionary: Update dictionary during data compression to adapt to frequency of blocks.

Dynamic Dictionaries

Definition

A dictionary D based on an alphabet S is a (finite) sequence of (distinct) words in S^*:

$$D = d_1, d_2, d_3, \ldots, d_N.$$

We say that i is the (dictionary) index of d_i.

We can use a dictionary for a codebook, i.e. to record the encoding of symbols and blocks. There are two options:

Static Dictionary: Construct dictionary (at the start) which remains the unchanged (Arithmetic Coding).

Dynamic Dictionary: Update dictionary during data compression to adapt to frequency of blocks.

Lempel, Ziv and Welch

Definition (LZW Code (1984))

Suppose we have a message $X = x_1 x_2 \cdots x_n$ in the alphabet $S = \{s_1, s_2, \ldots, s_m\}$, and $D_0 = d_1, d_2, \ldots, d_m$ with $d_i = s_i$.
The LZW coding rules construct $c(X) = c_1 c_2 c_3 \cdots$ as follows:

Step 1: The first symbol x_1 (let's say $x_1 = s_p$) is an entry d_p in D_0. Encode x_1 via

$$c_1 = p.$$

The string $x_1 x_2$ is not in D_0, but we define

$$d_{m+1} = x_1 x_2 \quad \text{and} \quad D_1 = D_0, d_{m+1}.$$

Lempel, Ziv and Welch

Definition (LZW Code (1984))

Suppose we have a message $X = x_1 x_2 \cdots x_n$ in the alphabet $S = \{s_1, s_2, \ldots, s_m\}$, and $D_0 = d_1, d_2, \ldots, d_m$ with $d_i = s_i$.
The LZW coding rules construct $c(X) = c_1 c_2 c_3 \cdots$ as follows:

Step 1: The first symbol x_1 (let's say $x_1 = s_p$) is an entry d_p in D_0. Encode x_1 via

$$c_1 = p.$$

The string $x_1 x_2$ is not in D_0, but we define

$$d_{m+1} = x_1 x_2 \quad \text{and} \quad D_1 = D_0, d_{m+1}.$$

Lempel, Ziv and Welch (cont.)

Definition (LZW Code, cont)

Step k: ($k > 1$) Suppose Step 1 to Step $k-1$ have been completed. We thus have the code $c_1 c_2 \cdots c_{k-1}$ for the initial $x_1 x_2 \cdots x_i$ and $D_{k-1} = d_1, d_2, \ldots, d_{m+k-1}$.

Find the longest string $w = x_{i+1} \cdots x_j = d_t$ in D_{k-1}. By definition wx_{j+1} is not in D_{k-1}. Encode

$$c_k = t$$

and extend D_{k-1}:

$$d_{m+k} = wx_{j+1} \quad \text{and} \quad D_k = D_{k-1}, d_{m+k}.$$

Repeat until the end of the message X.

Lempel, Ziv and Welch (cont.)

Definition (LZW Code, cont)

Step k: ($k > 1$) Suppose Step 1 to Step $k-1$ have been completed. We thus have the code $c_1 c_2 \cdots c_{k-1}$ for the initial $x_1 x_2 \cdots x_i$ and $D_{k-1} = d_1, d_2, \ldots, d_{m+k-1}$.

Find the longest string $w = x_{i+1} \cdots x_j = d_t$ in D_{k-1}. By definition wx_{j+1} is not in D_{k-1}. Encode

$$c_k = t$$

and extend D_{k-1}:

$$d_{m+k} = wx_{j+1} \quad \text{and} \quad D_k = D_{k-1}, d_{m+k}.$$

Repeat until the end of the message X.

An LZW Encoding Example

Example

Take alphabet $S = \{A, B, C, D, R\}$ and LZW encode the string:

$$X = ABRACADABRA$$

The initial dictionary is $D_0 = A, B, C, D, R$

Step 1: We need to encode A with index 1 and add AB to the dictionary, i.e.

$$c(A) = 1, \quad D_1 = A, B, C, D, R, AB.$$

Step 2: We need to encode B with index 2 in D_1 and add BR to the dictionary, i.e.

$$c(AB) = 1\ 2, \quad D_2 = A, B, C, D, R, AB, BR$$

An LZW Encoding Example

Example

Take alphabet $S = \{A, B, C, D, R\}$ and LZW encode the string:

$$X = ABRACADABRA$$

The initial dictionary is $D_0 = A, B, C, D, R$

Step 1: We need to encode A with index 1 and add AB to the dictionary, i.e.

$$c(A) = 1, \quad D_1 = A, B, C, D, R, AB.$$

Step 2: We need to encode B with index 2 in D_1 and add BR to the dictionary, i.e.

$$c(AB) = 1\ 2, \quad D_2 = A, B, C, D, R, AB, BR$$

An LZW Encoding Example

Example

Take alphabet $S = \{A, B, C, D, R\}$ and LZW encode the string:

$$X = ABRACADABRA$$

The initial dictionary is $D_0 = A, B, C, D, R$

Step 1: We need to encode A with index 1 and add AB to the dictionary, i.e.

$$c(A) = 1, \quad D_1 = A, B, C, D, R, AB.$$

Step 2: We need to encode B with index 2 in D_1 and add BR to the dictionary, i.e.

$$c(AB) = 1\ 2, \quad D_2 = A, B, C, D, R, AB, BR$$

An LZW Encoding Example (cont.)

Example (cont.)

LW encoding of $X = ABRACADABRA$.

Steps 3, 4, 5, 6, 7: We get in succesive steps for single symbols:

$$c(ABRACAD) = 1\ 2\ 5\ 1\ 3\ 1\ 4$$
$$D_7 = A, B, C, D, R, AB, BR, RA, AC, CA, AD, DA$$

Step 8: AB has index 6 in D_7 so we have

$$c(ABRACADAB) = 1\ 2\ 5\ 1\ 3\ 1\ 4\ 6$$
$$D_8 = A, B, C, D, R, AB, BR, RA, AC, CA, AD, DA, ABR$$

Step 9: RA has index 8 in D_8 so finally we get

$$c(ABRACADABRA) = 1\ 2\ 5\ 1\ 3\ 1\ 4\ 6\ 8$$

An LZW Encoding Example (cont.)

Example (cont.)

LW encoding of $X = ABRACADABRA$.

Steps $3, 4, 5, 6, 7$: We get in succesive steps for single symbols:

$$c(ABRACAD) = 1\ 2\ 5\ 1\ 3\ 1\ 4$$
$$D_7 = A, B, C, D, R, AB, BR, RA, AC, CA, AD, DA$$

Step 8: AB has index 6 in D_7 so we have

$$c(ABRACADAB) = 1\ 2\ 5\ 1\ 3\ 1\ 4\ 6$$
$$D_8 = A, B, C, D, R, AB, BR, RA, AC, CA, AD, DA, ABR$$

Step 9: RA has index 8 in D_8 so finally we get

$$c(ABRACADABRA) = 1\ 2\ 5\ 1\ 3\ 1\ 4\ 6\ 8$$

An LZW Encoding Example (cont.)

Example (cont.)

LW encoding of $X = ABRACADABRA$.

Steps $3, 4, 5, 6, 7$: We get in succesive steps for single symbols:

$$c(ABRACAD) = 1\ 2\ 5\ 1\ 3\ 1\ 4$$
$$D_7 = A, B, C, D, R, AB, BR, RA, AC, CA, AD, DA$$

Step 8: AB has index 6 in D_7 so we have

$$c(ABRACADAB) = 1\ 2\ 5\ 1\ 3\ 1\ 4\ 6$$
$$D_8 = A, B, C, D, R, AB, BR, RA, AC, CA, AD, DA, ABR$$

Step 9: RA has index 8 in D_8 so finally we get

$$c(ABRACADABRA) = 1\ 2\ 5\ 1\ 3\ 1\ 4\ 6\ 8$$

An LZW Encoding Example (cont.)

Example (cont.)

LW encoding of $X = ABRACADABRA$.

Steps $3, 4, 5, 6, 7$: We get in succesive steps for single symbols:

$$c(ABRACAD) = 1\ 2\ 5\ 1\ 3\ 1\ 4$$
$$D_7 = A, B, C, D, R, AB, BR, RA, AC, CA, AD, DA$$

Step 8: AB has index 6 in D_7 so we have

$$c(ABRACADAB) = 1\ 2\ 5\ 1\ 3\ 1\ 4\ 6$$
$$D_8 = A, B, C, D, R, AB, BR, RA, AC, CA, AD, DA, ABR$$

Step 9: RA has index 8 in D_8 so finally we get

$$c(ABRACADABRA) = 1\ 2\ 5\ 1\ 3\ 1\ 4\ 6\ 8$$

Unique Decodability for LZW

Theorem

An LZW code constructed as above is uniquely decodable.

Proof.

Formally by induction: Suppose we have partially decoded $c_1 c_2 c_3 \cdots c_{k-1}$ as $x = x_1 x_2 \ldots x_i$, and we constructed D_{k-1} by adding $k-1$ blocks to D_0 with $k \geq 2$.

Use this to construct c_k and $D_k = D_{k-1}, d_{m+k}$. It must be that c_k is the index of a string $s_u \cdots s_v$ in D_{k-1}. So the decoded string extends x by $x_{i+1} = s_u, \ldots, x_j = s_v$.

This leaves us to construct D_k or more concretely d_{m+k}.

Unique Decodability for LZW

Theorem

An LZW code constructed as above is uniquely decodable.

Proof.

Formally by induction: Suppose we have partially decoded $c_1 c_2 c_3 \cdots c_{k-1}$ as $x = x_1 x_2 \ldots x_i$, and we constructed D_{k-1} by adding $k - 1$ blocks to D_0 with $k \geq 2$.

Use this to construct c_k and $D_k = D_{k-1}, d_{m+k}$. It must be that c_k is the index of a string $s_u \cdots s_v$ in D_{k-1}. So the decoded string extends x by $x_{i+1} = s_u, \ldots, x_j = s_v$.

This leaves us to construct D_k or more concretely d_{m+k}.

Unique Decodability for LZW

Theorem

An LZW code constructed as above is uniquely decodable.

Proof.

Formally by induction: Suppose we have partially decoded $c_1 c_2 c_3 \cdots c_{k-1}$ as $x = x_1 x_2 \ldots x_i$, and we constructed D_{k-1} by adding $k-1$ blocks to D_0 with $k \geq 2$.

Use this to construct c_k and $D_k = D_{k-1}, d_{m+k}$. It must be that c_k is the index of a string $s_u \cdots s_v$ in D_{k-1}. So the decoded string extends x by $x_{i+1} = s_u, \ldots, x_j = s_v$.

This leaves us to construct D_k or more concretely d_{m+k}.

Unique Decodability for LZW

Theorem

An LZW code constructed as above is uniquely decodable.

Proof.

Formally by induction: Suppose we have partially decoded $c_1 c_2 c_3 \cdots c_{k-1}$ as $x = x_1 x_2 \ldots x_i$, and we constructed D_{k-1} by adding $k-1$ blocks to D_0 with $k \geq 2$.

Use this to construct c_k and $D_k = D_{k-1}, d_{m+k}$. It must be that c_k is the index of a string $s_u \cdots s_v$ in D_{k-1}. So the decoded string extends x by $x_{i+1} = s_u, \ldots, x_j = s_v$.

This leaves us to construct D_k or more concretely d_{m+k}.

Proof of UD for LZW (cont.)

Proof (cont).

If $c_{k+1} = r \leq m + k - 1$ then d_r is already in D_{k-1}, e.g. $d_r = s_a \cdots$. Thus $x_{i+1} = s_a$ and the new dictionary entry is $d_{m+k} = s_u \cdots s_v s_a$.

The other possibility is that c_{k+1} is exactly the new entry in D_k, i.e. d_{m+k}, or that $c_{k+1} = m + k$. Then d_{m+k} is a string of the form $s_u \cdots s_v s_z$ with $s_z = x_{j+1}$. Thus $c_k c_{k+1}$ is the encoded form of $s_u \cdots s_v s_u \cdots s_v s_z$ and in fact $x_{j+1} = s_u$ or $d_{m+k} = s_u \cdots s_v s_u$.

We thus have also reconstructed D_k. □

Proof of UD for LZW (cont.)

Proof (cont).

If $c_{k+1} = r \leq m + k - 1$ then d_r is already in D_{k-1}, e.g. $d_r = s_a \cdots$. Thus $x_{i+1} = s_a$ and the new dictionary entry is $d_{m+k} = s_u \cdots s_v s_a$.

The other possibility is that c_{k+1} is exactly the new entry in D_k, i.e. d_{m+k}, or that $c_{k+1} = m + k$. Then d_{m+k} is a string of the form $s_u \cdots s_v s_z$ with $s_z = x_{j+1}$. Thus $c_k c_{k+1}$ is the encoded form of $s_u \cdots s_v s_u \cdots s_v s_z$ and in fact $x_{j+1} = s_u$ or $d_{m+k} = s_u \cdots s_v s_u$.

We thus have also reconstructed D_k. □

Proof of UD for LZW (cont.)

Proof (cont).

If $c_{k+1} = r \leq m + k - 1$ then d_r is already in D_{k-1}, e.g. $d_r = s_a \cdots$. Thus $x_{i+1} = s_a$ and the new dictionary entry is $d_{m+k} = s_u \cdots s_v s_a$.

The other possibility is that c_{k+1} is exactly the new entry in D_k, i.e. d_{m+k}, or that $c_{k+1} = m + k$. Then d_{m+k} is a string of the form $s_u \cdots s_v s_z$ with $s_z = x_{j+1}$. Thus $c_k c_{k+1}$ is the encoded form of $s_u \cdots s_v s_u \cdots s_v s_z$ and in fact $x_{j+1} = s_u$ or $d_{m+k} = s_u \cdots s_v s_u$.

We thus have also reconstructed D_k. □

An LZW Decoding Example

Example

Consider the initial dictionary $D_0 = I, M, P, S$ and LZW decode

2 1 4 4 6 8 3 3 1

Step 1: $c_1 = 2$ and $d_2 = M$ so $x_1 = M$. Furthermore $c_2 = 1$ and $d_1 = I$, so the new dictionary entry is MI. Thus:

$$2 \mapsto M \quad D_1 = I, M, P, S, MI$$

Step 2: $c_2 = 1$ and $d_1 = I$. Also $c_3 = 4$ and $d_4 = S$ so the new entry is IS. Thus

$$2\ 1 \mapsto MI \quad D_2 = I, M, P, S, MI, IS$$

An LZW Decoding Example

Example

Consider the initial dictionary $D_0 = I, M, P, S$ and LZW decode

$$2\ 1\ 4\ 4\ 6\ 8\ 3\ 3\ 1$$

Step 1: $c_1 = 2$ and $d_2 = M$ so $x_1 = M$. Furthermore $c_2 = 1$ and $d_1 = I$, so the new dictionary entry is MI. Thus:

$$2 \mapsto M \quad D_1 = I, M, P, S, MI$$

Step 2: $c_2 = 1$ and $d_1 = I$. Also $c_3 = 4$ and $d_4 = S$ so the new entry is IS. Thus

$$2\ 1 \mapsto MI \quad D_2 = I, M, P, S, MI, IS$$

An LZW Decoding Example

Example

Consider the initial dictionary $D_0 = I, M, P, S$ and LZW decode

$$2\ 1\ 4\ 4\ 6\ 8\ 3\ 3\ 1$$

Step 1: $c_1 = 2$ and $d_2 = M$ so $x_1 = M$. Furthermore $c_2 = 1$ and $d_1 = I$, so the new dictionary entry is MI. Thus:

$$2 \mapsto M \quad D_1 = I, M, P, S, MI$$

Step 2: $c_2 = 1$ and $d_1 = I$. Also $c_3 = 4$ and $d_4 = S$ so the new entry is IS. Thus

$$2\ 1 \mapsto MI \quad D_2 = I, M, P, S, MI, IS$$

An LZW Decoding Example (cont.)

Example (cont.)

$$2\ 1\ 4\ 4\ 6\ 8\ 3\ 3\ 1, \quad D_2 = I, M, P, S, MI, IS$$

Steps 3&4: gives the decoding $2\ 1\ 4\ 4 \mapsto MISS$ and

$$D_4 = I, M, P, S, MI, IS, SS, SI$$

Step 5: $c_5 = 6$ and $d_6 = IS$ so $x_5 = I$ and $x_6 = S$. Also $c_6 = 8$ and $d_8 = SI$ so new entry is ISS. Thus

$$2\ 1\ 4\ 4\ 6 \mapsto MISSIS \quad D_5 = I, M, P, S, MI, IS, SS, SI, ISS$$

Steps 7, 8, 9: are similar and we get

$$2\ 1\ 4\ 4\ 6\ 8\ 3\ 3\ 1 \mapsto MISSISSIPPI$$

An LZW Decoding Example (cont.)

Example (cont.)

2 1 4 4 6 8 3 3 1, $D_2 = I, M, P, S, MI, IS$

Steps 3&4: gives the decoding 2 1 4 4 $\mapsto$ *MISS* and

$$D_4 = I, M, P, S, MI, IS, SS, SI$$

Step 5: $c_5 = 6$ and $d_6 = IS$ so $x_5 = I$ and $x_6 = S$. Also $c_6 = 8$ and $d_8 = SI$ so new entry is *ISS*. Thus

2 1 4 4 6 $\mapsto$ *MISSIS* $D_5 = I, M, P, S, MI, IS, SS, SI, ISS$

Steps 7, 8, 9: are similar and we get

2 1 4 4 6 8 3 3 1 $\mapsto$ *MISSISSIPPI*

An LZW Decoding Example (cont.)

Example (cont.)

$$2\ 1\ 4\ 4\ 6\ 8\ 3\ 3\ 1, \quad D_2 = I, M, P, S, MI, IS$$

Steps 3&4: gives the decoding $2\ 1\ 4\ 4 \mapsto MISS$ and

$$D_4 = I, M, P, S, MI, IS, SS, SI$$

Step 5: $c_5 = 6$ and $d_6 = IS$ so $x_5 = I$ and $x_6 = S$. Also $c_6 = 8$ and $d_8 = SI$ so new entry is ISS. Thus

$$2\ 1\ 4\ 4\ 6 \mapsto MISSIS \quad D_5 = I, M, P, S, MI, IS, SS, SI, ISS$$

Steps 7, 8, 9: are similar and we get

$$2\ 1\ 4\ 4\ 6\ 8\ 3\ 3\ 1 \mapsto MISSISSIPPI$$