Chapter 3: Compression Coding

Lectures 13-14

The Huffman Code Theorem

For any given source S and corresponding probabilities,
the HUFFMAN ALGORITHM yields an instantaneous minimum UD-code.

Proof

We proceed by induction on $q = |S|$.
For $q = 2$, each Huffman code is an instantaneous UD-code with minimum average length $L = 1$.

Now assume that each Huffman code on $q-1$ symbols is an instantaneous UD-code with minimum average length.
Let C be a Huffman code on q symbols with average length L and let C^* be any UD-code q symbols with minimum average length L^*.
Denote the codeword lengths of C and C^* by $\ell_1, \ldots, \ell_q$ and $\ell_1^*, \ldots, \ell_q^*$.
By construction, $\mathbf{c}_q$ and $\mathbf{c}_{q-1}$ in C differ only in last place.
By minimality, C^* has codewords $\mathbf{c}_q^*$, $\mathbf{c}_{q-1}^*$ differing only in the last place.
Combine $\mathbf{c}_q$ and $\mathbf{c}_{q-1}$ in C to get a Huffman code on $q-1$ symbols and combine $\mathbf{c}_q^*$ and $\mathbf{c}_{q-1}^*$ in C^* to get a UD-code on $q-1$ symbols.

Now assume that each Huffman code on $q-1$ symbols is an instantaneous UD-code with minimum average length.
Let C be a Huffman code on q symbols with average length L and let C^* be any UD-code q symbols with minimum average length L^*.
Denote the codeword lengths of C and C^* by $\ell_1, \ldots, \ell_q$ and $\ell_1^*, \ldots, \ell_q^*$.
By construction, $\mathbf{c}_q$ and $\mathbf{c}_{q-1}$ in C differ only in last place.
By minimality, C^* has codewords $\mathbf{c}_q^*$, $\mathbf{c}_{q-1}^*$ differing only in the last place.
Combine $\mathbf{c}_q$ and $\mathbf{c}_{q-1}$ in C to get a Huffman code on $q-1$ symbols and combine $\mathbf{c}_q^*$ and $\mathbf{c}_{q-1}^*$ in C^* to get a UD-code on $q-1$ symbols.
Denote the average lengths of these codes by M and M^*, respectively.

By the induction hypothesis $M \leq M^*$, so

$$
\begin{aligned}
L - L^* &= \sum_{i=1}^{q} \ell_i p_i - \sum_{i=1}^{q} \ell_i^* p_i \\
&= \left(\left(\sum_{i=1}^{q-2} \ell_i p_i \right) + \ell_{q-1} p_{q-1} + \ell_q p_q \right) - \left(\left(\sum_{i=1}^{q-2} \ell_i^* p_i \right) + \ell_{q-1}^* p_{q-1} + \ell_q^* p_q \right) \\
&= \left(\left(\sum_{i=1}^{q-2} \ell_i p_i \right) + \ell_q (p_{q-1} + p_q) \right) - \left(\left(\sum_{i=1}^{q-2} \ell_i^* p_i \right) + \ell_q^* (p_{q-1} + p_q) \right) \\
&= \left(\left(\sum_{i=1}^{q-2} \ell_i p_i \right) + (\ell_q - 1)(p_{q-1} + p_q) \right) - \left(\left(\sum_{i=1}^{q-2} \ell_i^* p_i \right) + (\ell_q^* - 1)(p_{q-1} + p_q) \right) \\
&= M - M^* \\
&\leq 0
\end{aligned}
$$

Thus $L \leq L^*$, so the Huffman code has minimum average length.
The code is created using a decision tree, so it is instantaneous.
The proof follows by induction. □

The Huffman Code Theorem

For any given source S and corresponding probabilities,
the HUFFMAN ALGORITHM yields an instantaneous minimum UD-code.

Theorem (Knuth)

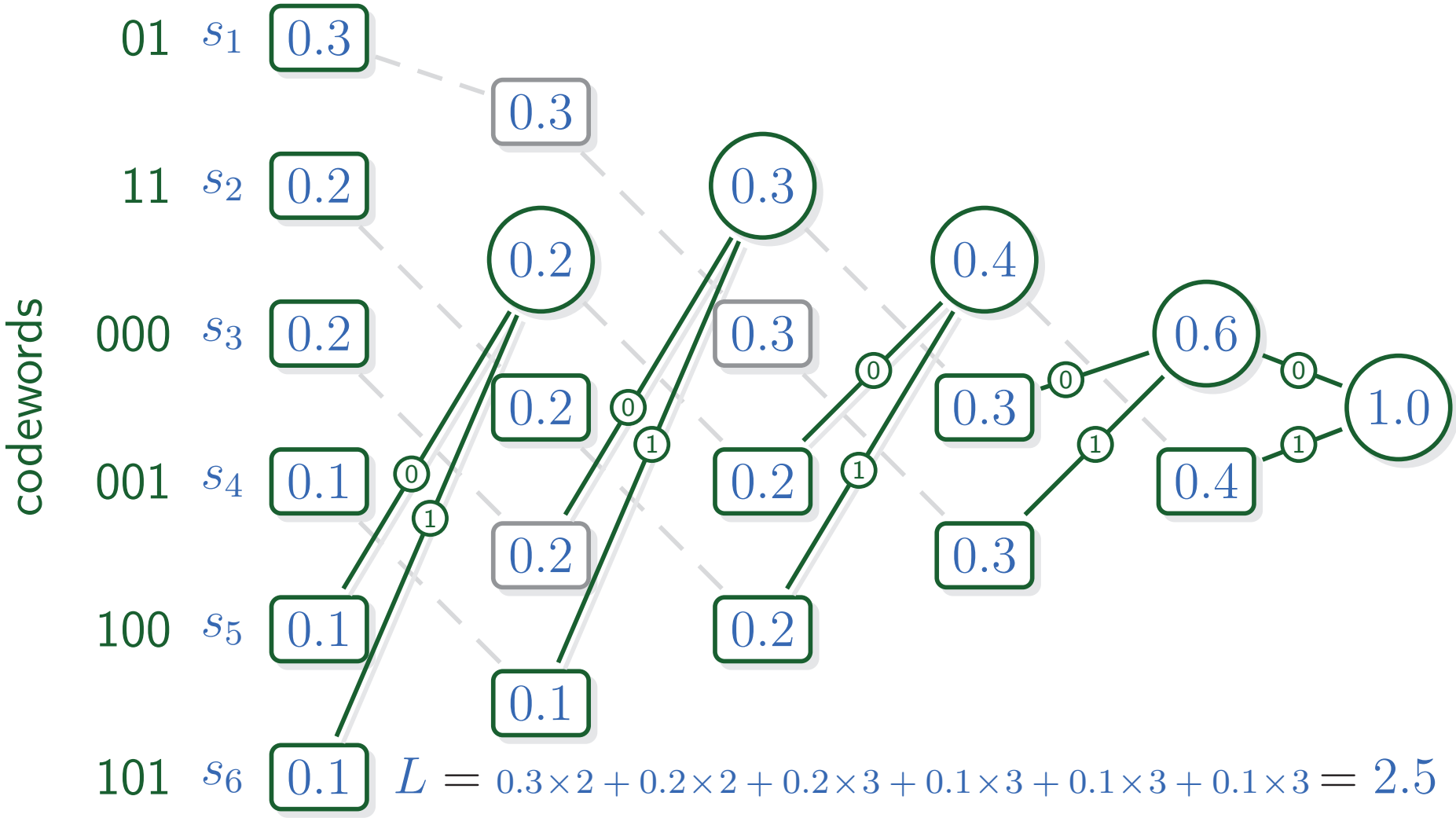
The average codeword length L of each Huffman code is
the sum of all child node probabilities.

Proof

The tree-path for symbol s_i will pass through exactly ℓ_i child nodes.
But p_i will occur as part of the sum in each of the these child nodes.
So, adding all child node probabilities adds ℓ_i copies of p_i for each s_i;
this is L. $\square$

Theorem (Knuth)

The average codeword length L of each Huffman code is the sum of all child node probabilities.



$$L = 0.3\times 2 + 0.2\times 2 + 0.2\times 3 + 0.1\times 3 + 0.1\times 3 + 0.1\times 3 = 2.5$$

$$L = 1.0 + 0.6 + 0.4 + 0.3 + 0.2 = 2.5$$

Huffman's Algorithm (radix r)

Huffman's Algorithm also works for radix r:
just combine r symbols at each step instead of 2.

However, there are at least two ways to do this:

1. Combine the last r symbols at each combining step.
2. First add dummy symbols; then combine the last r symbols at each step.

It turns out that 2 is the best strategy.

If there are k combining steps, then we need

$$|S| = k(r-1) + r = (k+1)(r-1) + 1$$

initial symbols. In other words, we must have $|S| \equiv 1 \pmod{r-1}$.
We can ensure this by adding dummy symbols.

Huffman's Algorithm (radix r)

Input: a source $S = \{s_1, \ldots, s_q\}$ and probabilities $p_1, \ldots, p_q$
Output: a code C for S, given by a decision tree

Add dummy symbols until $q = |S| \equiv 1 \pmod{r-1}$.

Combining phase

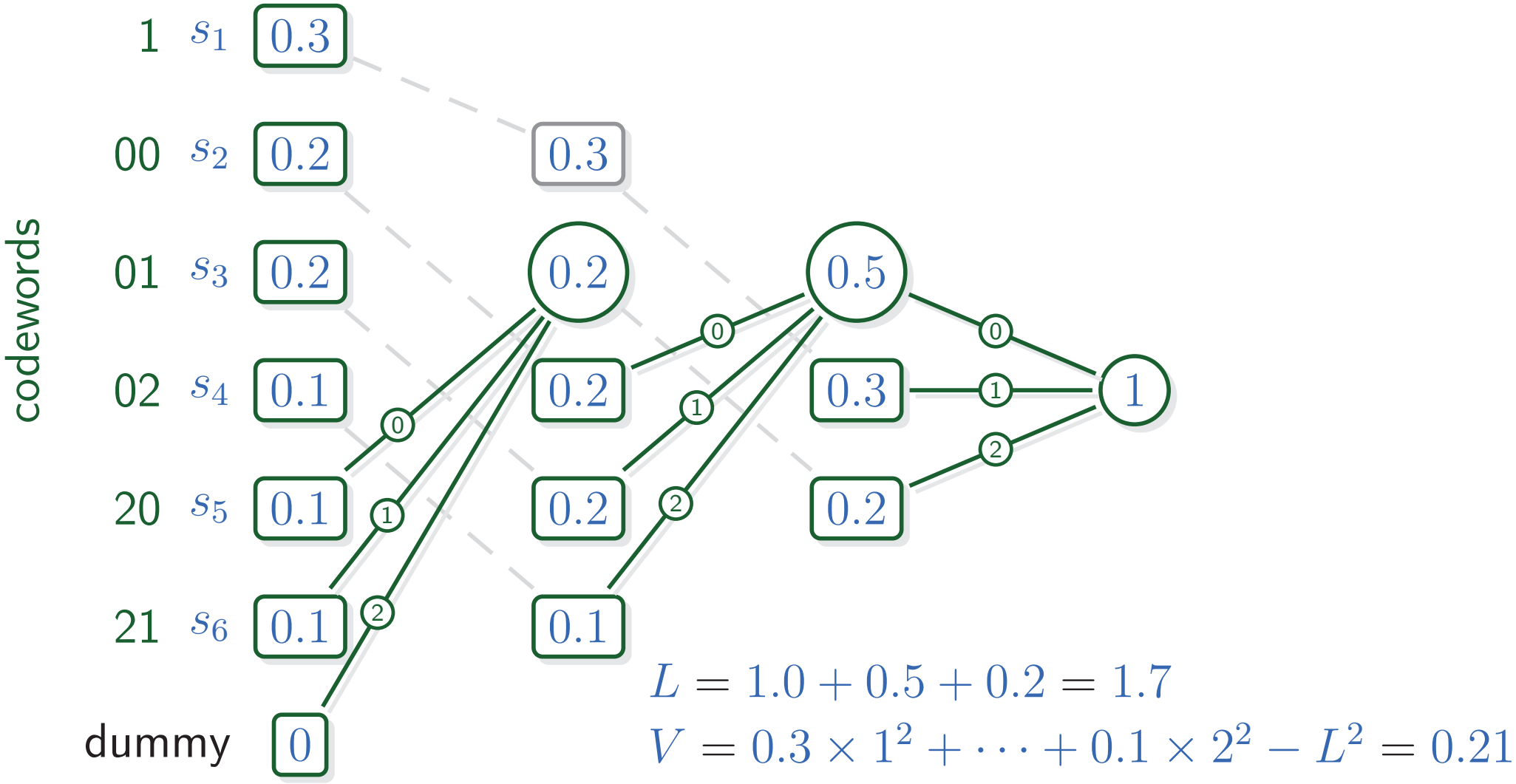
- Replace the last r symbols $s_{q-r+1}, \ldots, s_q$
  by a new symbol with probability $p_{q-r+1} + \cdots + p_q$.
- Reorder the symbols by their probabilities.
- Repeat until there is only one symbol left.

Splitting phase

- Root-label this symbol.
- Draw edges from each child node to the r preceding nodes.
- Label these edges from top to bottom by $0, \ldots, q-1$.

Example

Consider a source $s_1, \ldots, s_6$ with probabilities $0.3,\ 0.2,\ 0.2,\ 0.1,\ 0.1,\ 0.1$.
With radix $r = 3$, there are $6 \pmod{r-1} = 0$ symbols.
We need to add 1 dummy symbol.



$$L = 1.0 + 0.5 + 0.2 = 1.7$$

$$V = 0.3 \times 1^2 + \cdots + 0.1 \times 2^2 - L^2 = 0.21$$

Extensions

Given a source $S = \{s_1, \ldots, s_q\}$ with associated probabilities $p_1, \ldots, p_q$, the nth extension is the Cartesian product

$$S^n = \underbrace{S \times \cdots \times S}_{n} = \{s'_1 \cdots s'_n \ : \ s'_1, \ldots, s'_n \in S\} = \{\sigma_1, \ldots, \sigma_{q^n}\}$$

together with the following probability for each new symbol $\sigma_i \in S^n$:

$$p_i^{(n)} = P(\sigma_i) = P(s'_1 \cdots s'_n) = P(s'_1) \cdots P(s'_n)$$

- Note that we implictly assume that $p_1, \ldots, p_q$ are independent.
- We usually order the symbols σ_i so that $p_1^{(n)}, \ldots, p_{q^n}^{(n)}$ are non-increasing.

Example

Consider source $S = \{a, b\}$ with associated probabilities $\frac{3}{4}, \frac{1}{4}$.

We apply the (binary) HUFFMAN ALGORITHM:

$S^1 = S$	p_i	$\mathbf{c}_i$
a	$\frac{3}{4}$	0
b	$\frac{1}{4}$	1

S^2	$p_i^{(2)}$	$\mathbf{c}_i$
aa	$\frac{9}{16}$	0
ab	$\frac{3}{16}$	11
ba	$\frac{3}{16}$	100
bb	$\frac{1}{16}$	101

S^3	$p_i^{(3)}$	$\mathbf{c}_i$
aaa	$\frac{27}{64}$	1
aab	$\frac{9}{64}$	001
aba	$\frac{9}{64}$	010
baa	$\frac{9}{64}$	011
abb	$\frac{3}{64}$	00000
bab	$\frac{3}{64}$	00001
bba	$\frac{3}{64}$	00010
bbb	$\frac{1}{64}$	00011

We apply the (binary) HUFFMAN ALGORITHM:

$S^1 = S$	p_i	$\mathbf{c}_i$
a	$\frac{3}{4}$	0
b	$\frac{1}{4}$	1

S^2	$p_i^{(2)}$	$\mathbf{c}_i$
aa	$\frac{9}{16}$	0
ab	$\frac{3}{16}$	11
ba	$\frac{3}{16}$	100
bb	$\frac{1}{16}$	101

S^3	$p_i^{(3)}$	$\mathbf{c}_i$
aaa	$\frac{27}{64}$	1
aab	$\frac{9}{64}$	001
aba	$\frac{9}{64}$	010
baa	$\frac{9}{64}$	011
abb	$\frac{3}{64}$	00000
bab	$\frac{3}{64}$	00001
bba	$\frac{3}{64}$	00010
bbb	$\frac{1}{64}$	00011

$L^{(1)} = L = 1$

$L^{(2)} = \frac{27}{16}$

$L^{(3)} = \frac{158}{64}$

Average length per S-symbol for S: 1

Average length per S-symbol for S^2: $\frac{L^{(2)}}{2} = \frac{27}{32} \approx 0.84$

Average length per S-symbol for S^3: $\frac{L^{(3)}}{3} = \frac{158}{192} \approx 0.82$

Markov sources

A k-memory source S is one whose symbols each depend on the previous k.

- If $k = 0$, then no symbol depends on any other, and S is memoryless.
- If $k = 1$, then S is a Markov source.
- $p_{ij} = P(s_i|s_j)$ is the probability of s_i occurring right after a given s_j.
- The matrix $M = (p_{ij})$ is the transition matrix.
- Entry p_{ij} is the probability of getting from state s_j to state s_i.

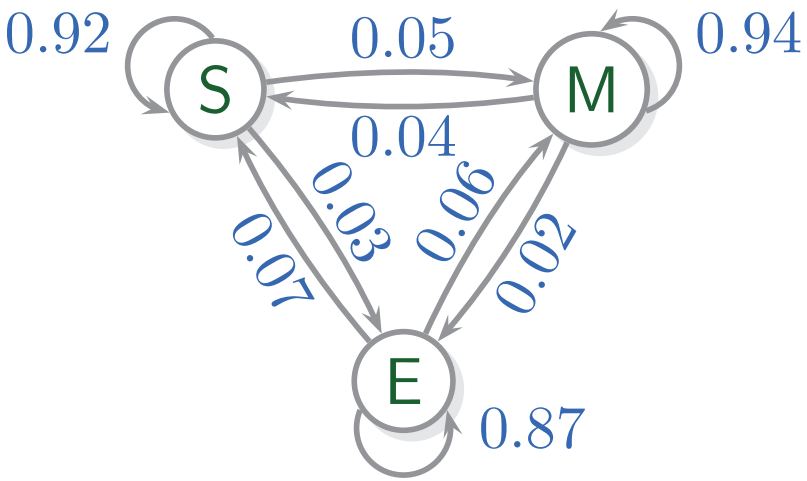
A Markov process is a set of states (the source S)
and probabilities $p_{ij} = P(s_i|s_j)$ of getting from state s_j to state s_i.

Example

Consider Sydney, Melbourne, and Elsewhere in Australia.
A simple Markov model for the populations of these is that, each year,

- population growth by births, deaths, and emmi-/immigration is 0%
- of people living in Sydney, 5% move to Melbourne; 3% move Elsewhere
- of people living in Melbourne, 4% move to Sydney; 2% move Elsewhere
- of people living Elsewhere, 7% move to Sydney; 6% move to Melbourne.

$S = \{\text{Sydney, Melbourne, Elsewhere}\}$

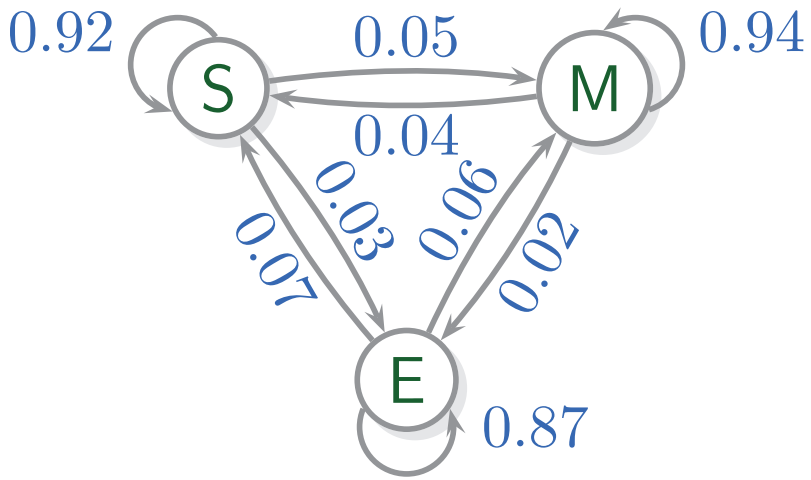


$$M = \begin{array}{c} \\ \text{To} \end{array} \begin{array}{c} \text{From} \\ \begin{array}{ccc} \text{S} & \text{M} & \text{E} \end{array} \\ \begin{array}{c} \text{S} \\ \text{M} \\ \text{E} \end{array} \begin{bmatrix} 0.92 & 0.04 & 0.07 \\ 0.05 & 0.94 & 0.06 \\ 0.03 & 0.02 & 0.87 \end{bmatrix} \end{array}$$

Example

Consider Sydney, Melbourne, and Elsewhere in Australia.

$$S = \{\text{Sydney, Melbourne, Elsewhere}\}$$



$$M = \begin{array}{c} \\ \text{To} \end{array} \begin{array}{c} \text{From} \\ \begin{array}{ccc} \text{S} & \text{M} & \text{E} \end{array} \\ \begin{array}{c} \text{S} \\ \text{M} \\ \text{E} \end{array} \begin{bmatrix} 0.92 & 0.04 & 0.07 \\ 0.05 & 0.94 & 0.06 \\ 0.03 & 0.02 & 0.87 \end{bmatrix} \end{array}$$

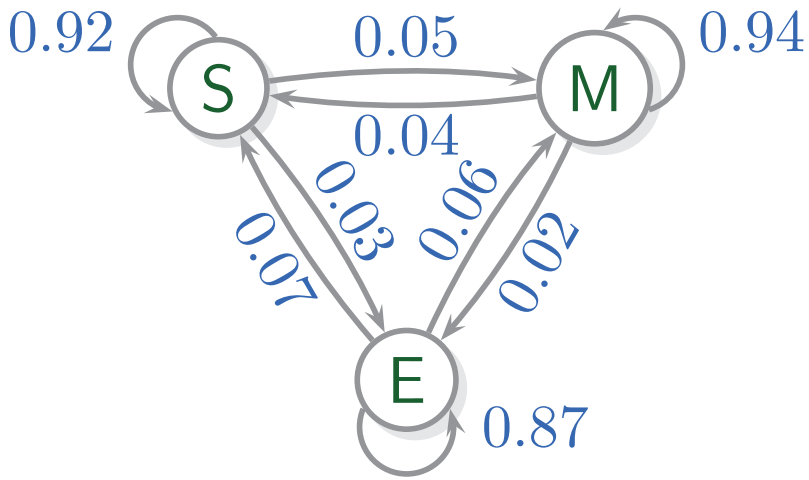
Lemma

The sum of entries in any column of M is 1.

Example

Consider Sydney, Melbourne, and Elsewhere in Australia.

$$S = \{\text{Sydney, Melbourne, Elsewhere}\}$$



$$M = \begin{array}{c} \\ \text{To} \\ \\ \\ \end{array} \begin{array}{c} \text{From} \\ \begin{array}{ccc} \text{S} & \text{M} & \text{E} \end{array} \\ \begin{bmatrix} 0.92 & 0.04 & 0.07 \\ 0.05 & 0.94 & 0.06 \\ 0.03 & 0.02 & 0.87 \end{bmatrix} \end{array}$$

Let $\mathbf{x}_k = \begin{pmatrix} s_k \\ m_k \\ e_k \end{pmatrix}$ denote the population distribution after k years.

Then

$$\mathbf{x}_{k+1} = M\mathbf{x}_k \quad \text{and} \quad \mathbf{x}_k = M^k\mathbf{x}_0$$

Let $\mathbf{x}_k = \begin{pmatrix} s_k \\ m_k \\ e_k \end{pmatrix}$ denote the population distribution after k years.
Then

$$\mathbf{x}_{k+1} = M\mathbf{x}_k \quad \text{and} \quad \mathbf{x}_k = M^k\mathbf{x}_0$$

Suppose that the initial population distribution is $\mathbf{x}_0 = \begin{pmatrix} 4.5M \\ 4M \\ 14M \end{pmatrix}$.

After $k = 20$ years, the population distribution is then

$$\mathbf{x}_{20} = M^{20}\mathbf{x}_0 = \begin{pmatrix} 0.41 & 0.34 & 0.38 \\ 0.42 & 0.52 & 0.44 \\ 0.16 & 0.15 & 0.19 \end{pmatrix} \begin{pmatrix} 4.5M \\ 4M \\ 14M \end{pmatrix} = \begin{pmatrix} 9.5M \\ 10.5M \\ 4M \end{pmatrix}$$

Note that S and M^{20} also form a Markov chain.
Eg., after 20 years, most people will have left Sydney (41% remain), whereas most people will have stayed in Melbourne (52%).

Let $\mathbf{x}_k = \begin{pmatrix} s_k \\ m_k \\ e_k \end{pmatrix}$ denote the population distribution after k years.
Then

$$\mathbf{x}_{k+1} = M\mathbf{x}_k \quad \text{and} \quad \mathbf{x}_k = M^k\mathbf{x}_0$$

Suppose that the initial population distribution is $\mathbf{x}_0 = \begin{pmatrix} 4.5M \\ 4M \\ 14M \end{pmatrix}$.

To find a stable population distribution, we need to find
a state $\mathbf{x}_0$ for which $\mathbf{x}_k = \mathbf{x}_{k-1} = \cdots = \mathbf{x}_1 = \mathbf{x}_0$; that is, $\mathbf{M}\mathbf{x}_0 = \mathbf{x}_0$.
In other words, we need an eigenvector $\mathbf{x}_0$ of M for the eigenvalue 1;
for instance, $\mathbf{x}_0 = \begin{pmatrix} 0.6 \\ 0.76 \\ 0.26 \end{pmatrix}$.

If we want an eigenvector that represents actual population numbers,
then we must scale $\mathbf{x}_0$ by $\frac{4.5M+4M+14M}{0.6+0.76+0.26}$: $\mathbf{x}_0 = \begin{pmatrix} 8.3M \\ 10.6M \\ 3.6M \end{pmatrix}$

A Markov process M is in equilibrium $\mathbf{p}$ if $\mathbf{p} = M\mathbf{p}$.

- In this case, $\mathbf{p}$ is an eigenvector for M and the eigenvalue 1.

We will assume that

- M is ergodic: we can get from any state j to any state i.
- M is aperiodic: the $\gcd$ of cycle lengths is 1.

Theorem

Under the above assumptions, M has a non-zero equilibrium state.

We will only consider equilibriums $\mathbf{p}$ with $|\mathbf{p}| = 1$.

Huffman coding for stationary Markov sources

Consider a Markov source $S = \{s_1, \ldots, s_q\}$ with probabilities $p_1, \ldots, p_q$, transition matrix M and equilibrium $\mathbf{p}$.

Define

- Huff_E: the binary Huffman code on $\mathbf{p}$ (ordered)
- $\text{Huff}_{(i)}$: the binary Huffman code on the (ordered) ith column of M.
- Huff_M: s_1 is encoded by Huff_E; for $i > 1$, s_i is encoded by $\text{Huff}_{(\mathbf{i-1})}$

This gives average lengths

- L_E
- $L_{(1)}, \ldots, L_{(q)}$
- $L_M \approx p_1 L_{(1)} + \cdots + p_q L_{(q)}$.

In general, $L_M \leq L_E$.