

Information Theory and Applications

Information sources

Prof. Giorgio Taricco

Politecnico di Torino – DET

Program of the course (prof. Taricco)

- Time allocation:
 - 30 hours = 20 slots (of 1.5-hours)
- Information sources and source codes (lossless)
 - THEORY (5 slots)
 - ASSIGNMENT (5 slots)
- Lossy source coding
 - THEORY (5 slots)
 - ASSIGNMENT (5 slots)

Tentative schedule, subject to minor modifications

Information sources

- Information sources are statistical devices generating a data stream representing any kind of information
- They emit a sequence of random variables X_n , where $n \in \mathbb{Z}$, the set of integer numbers (n may be a discrete time index)
- The complete description requires all the possible probabilities

$$P(X_{\mathcal{S}} = x_{\mathcal{S}})$$

- The symbol \mathcal{S} represents any subset of \mathbb{Z}
- The event $X_{\mathcal{S}} = x_{\mathcal{S}}$ corresponds to $X_n = x_n \forall n \in \mathcal{S}$
- While the X_n are random symbols generated by the source, the x_n are deterministic values they take on, extracted from a source alphabet

$$\mathcal{X} = \{\xi_1, \dots, \xi_M\} \quad |\mathcal{X}| = M$$

Information sources

- Example 1

- Binary source, $\mathcal{X} = \{0,1\}$
- Sample subset: $\mathcal{S} = \{0,2,4,10\}$
- Sample probability

$$P(X_{\mathcal{S}} = \underbrace{\{0,1,0,1\}}_{x_{\mathcal{S}}}) = P(X_0 = 0, X_2 = 1, X_4 = 0, X_{10} = 1)$$

- Example 2

- Ternary information source, $\mathcal{X} = \{0,1,2\}$
- Sample subset: $\mathcal{S} = \{-2,0,2\}$
- Sample probability

$$P(X_{\mathcal{S}} = \{2,1,0\}) = P(X_{-2} = 2, X_0 = 1, X_2 = 0)$$

Stationary information sources

- Stationarity
 - For a fixed deterministic vector $x_{\mathcal{S}}$ of $|\mathcal{S}|$ values,

$$P(X_{\mathcal{S}} = x_{\mathcal{S}}) = P(X_{\mathcal{S} + \Delta} = x_{\mathcal{S}})$$

- $\Delta \in \mathbb{Z}$
- $\mathcal{S} + \Delta \stackrel{\text{def}}{=} \{n + \Delta, n \in \mathcal{S}\}$

- For example:

$$P(X_{1,3,5} = (a, b, c)) = P(X_{4,6,8} = (a, b, c))$$

$\Delta = 3$

Here, we have any
possible sequence

Stationary information sources

$$\mathcal{S} = \{n\}$$

- For example, if $|\mathcal{S}| = 1$, stationarity implies that

$$P(X_n = x) = P(X_0 = x) \stackrel{\text{def}}{=} p_X(x)$$

- The property holds for every $n \in \mathbb{Z}$ and every $x \in \mathcal{X}$
- The function $p_X(x)$ is the probability distribution function (a probability mass function in this case)
- In this case we can also say that the X_n are iid (*independent and identically distributed*)
- The most interesting class of stationary information sources are Markov sources

Markov sources

- Markov sources are an important class of information sources *with memory*
- The information source X_1, X_2, X_3, \dots is called a Markov source with memory L if the following property holds for every $n > L$:
- $$P(X_n = x_n | \underbrace{X_{1:n-1} = x_{1:n-1}}) = P(X_n = x_n | X_{n-L:n-1} = x_{n-L:n-1})$$
- The effective part of the conditioning clause is in the last L symbols
 - The previous symbols are irrelevant, so that we say there is memory L
- Among the applications of Markov sources there is modeling written text for data compression

The notation $a:b$ corresponds to the integer sequence from a to b

Markov sources

A finite n. of conditional probabilities
describes completely the T.I. info. source

- We consider **time-invariant*** Markov sources such that, for all $n > L$,

$$P(X_n = x_n | X_{n-L:n-1} = x_{n-L:n-1}) = P(X_{L+1} = x_{L+1} | X_{1:L} = x_{1:L})$$

- The symbols $\Sigma_n \stackrel{\text{def}}{=} X_{n-L:n-1}$ represent the state of the source at time $n (> L)$
- If the symbol alphabet is \mathcal{X} , the state alphabet is \mathcal{X}^L
The n. of states is $|\mathcal{X}|^L$
- We characterize time-invariant Markov sources by the distribution of the initial state $\Sigma_{L+1} = X_{1:L}$ and by the conditional probabilities

$$P(\Sigma_{L+2} = X_{2:L+1} = \sigma_2 | \Sigma_{L+1} = X_{1:L} = \sigma_1)$$

\uparrow
 $1 : |\mathcal{X}|^L$

*Time-invariance is a restricted form of stationarity

Markov sources

- The conditional probabilities $P(\Sigma_{L+2} = \sigma_2 | \Sigma_{L+1} = \sigma_1)$ form a transition probability matrix \mathbf{P} whose entries are defined as

$$(\mathbf{P})_{\sigma_1, \sigma_2} = P(\Sigma_{L+2} = \sigma_2 | \Sigma_{L+1} = \sigma_1)$$

- Usually, the states are represented by integer numbers:

$$(\mathbf{P})_{i,j}, \quad i, j \in \mathbb{S} \text{ (the } state \text{ space)}$$

- Typically, $\mathbb{S} = \{1, 2, \dots, |\mathbb{S}|\}$
- The sum of the row elements of \mathbf{P} is equal to 1: $\sum_{j \in \mathbb{S}} (\mathbf{P})_{i,j} = 1$
- State probability vectors \mathbf{p}_n* are defined by

$$(\mathbf{p}_n)_i = P(\Sigma_n = i), \quad i \in \mathbb{S}$$

*we use
row-vectors
for $\underline{\mathbf{p}_n}$*

Why $\sum_j (P)_{ij} = 1$?

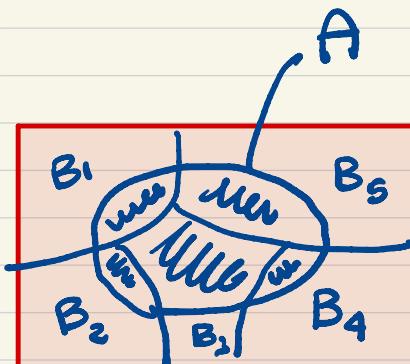
$$(P)_{ij} = P(\Sigma_{n+1}=j \mid \Sigma_n=i) = \frac{P(\Sigma_n=i, \Sigma_{n+1}=j)}{P(\Sigma_n=i)}$$

$$\sum_{j \in S} P(\Sigma_n=i, \Sigma_{n+1}=j) = P(\Sigma_n=i)$$

Total Probability Law

$$\sum_i P(A, B_i) = P(A)$$

partition
of a prob. space



$$P = \begin{pmatrix} 0.1 & 0.2 & 0.7 \\ 0.2 & 0.4 & 0.4 \\ 0.5 & 0.5 & 0 \end{pmatrix} \quad S = \{1, 2, 3\}$$

Example 1

x_n	x_{n-1}	$P(x_n=x_n x_{n-1}=x_{n-1})$
0	0	0.7
0	1	0.1
1	0	0.1
1	1	0.7

Find the matrix P

State definition $\{0\} \rightarrow 1$ $\{1\} \rightarrow 2$

$$P = \begin{pmatrix} 0.3 & 0.7 \\ \cancel{0.7} & \cancel{0.1} \\ 0.1 & 0.9 \end{pmatrix}$$

Example 2

Find P

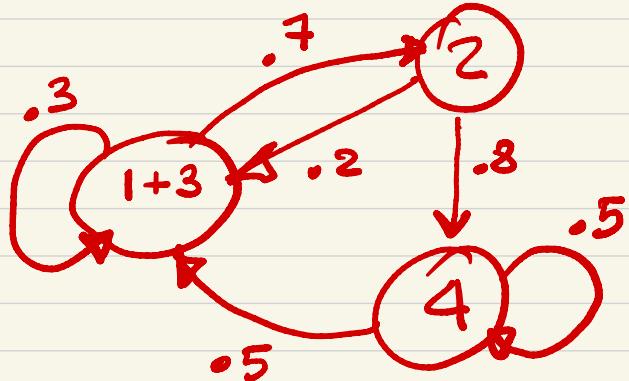
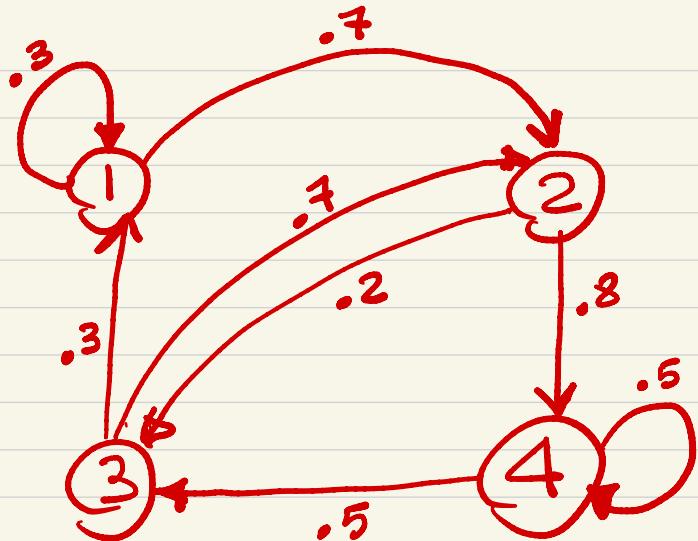
x_n	x_{n-1}	x_{n-2}	$P(x_n x_{n-1}, x_{n-2})$
0	0	*	.3
0	1	*	.7
1	0	0	.1
1	0	1	.2
1	1	0	.3
1	1	1	.4

x_{n-2}	x_{n-1}	x_n	$P(x_n x_{n-1}, x_{n-2})$
*	0	0	.3
*	0	1	.7
0	1	0	.2
0	1	1	.8
1	1	0	.5
1	1	1	.5

x_{n-2}	x_{n-1}	x_n	$P(x_n x_{n-1}, x_{n-2})$
x	0	0	.3
x	0	-1	.7
0	-1	0	.2
0	-1	-1	.8
1	-1	0	.5
1	-1	-1	.5

State	bits
1	00
2	01
3	10
4	11

$$P = \begin{bmatrix} .3 & .7 & 0 & 0 \\ 0 & 0 & .2 & .8 \\ .3 & .7 & 0 & 0 \\ 0 & 0 & .5 & .5 \end{bmatrix}$$



We have an equivalent description with 3 states instead of 4 \rightarrow the Markov chain can be compacted in this case

Markov chains

- The evolution of the state probability vectors is a *Markov chain*
- The evolution of the state probability distribution is governed by the equations

$$\mathbf{p}_{n+1} = \mathbf{p}_n \mathbf{P}$$

- This is a consequence of the **total probability law**:

$$(\mathbf{p}_{n+1})_j = P(\Sigma_{n+1} = j) = \sum_{i \in \mathbb{S}} P(\Sigma_n = i, \Sigma_{n+1} = j)$$

$$\begin{aligned} &= \sum_{i \in \mathbb{S}} P(\Sigma_n = i) P(\Sigma_{n+1} = j | \Sigma_n = i) \\ &= \sum_{i \in \mathbb{S}} (\mathbf{p}_n)_i (\mathbf{P})_{i,j} \end{aligned}$$

$$\begin{aligned} p_2 &= p_1 P \\ p_3 &= p_2 P = p_1 \cdot P \cdot P \\ &= p_1 P^2 \end{aligned}$$

Markov chains

- The equation $\mathbf{p}_{n+1} = \mathbf{p}_n \mathbf{P}$ can be applied repeatedly:

$$\mathbf{p}_{n+m} = \mathbf{p}_{n+m-1} \mathbf{P} = \mathbf{p}_{n+m-2} \mathbf{P}^2 = \cdots = \mathbf{p}_n \mathbf{P}^m$$

- The matrix \mathbf{P}^m corresponds to m consecutive transition steps in the Markov chain:

$$(\mathbf{P}^m)_{i,j} = P(\Sigma_{n+m} = j \mid \Sigma_n = i)$$

- In some cases, as $m \rightarrow \infty$, \mathbf{P}^m converges to a matrix whose rows are all equal
- To characterize Markov chains, we need some additional characterizations

Markov chains – basic definitions

- **Definition 1:** accessibility

A state $j \in \mathbb{S}$ is accessible from $i \in \mathbb{S}$ if $(P^m)_{i,j} > 0$ for some $m \geq 0$

- This definition means that “*it is possible to travel from i to j , with positive probability, in a certain number of steps*”
- A state is always considered accessible from itself since $P^0 = I$ (the identity matrix) even when $(P)_{i,i} = 0$
- If both “ j is accessible from i ” and “ i is accessible from j ,” we say that i and j *communicate* and write $i \leftrightarrow j$
- This is an *equivalence relation* since
 - $i \leftrightarrow i$ (reflexivity)
 - $i \leftrightarrow j$ implies $j \leftrightarrow i$ (symmetry)
 - $i \leftrightarrow j$ and $j \leftrightarrow k$ imply $i \leftrightarrow k$ (transitivity)

Markov chains – basic definitions

- The communication equivalence relation $i \leftrightarrow j$ induces a partition on the space state \mathbb{S} into disjoint sets A_p such that
 - $i \leftrightarrow j$ for all $i, j \in A_p$
 - $i \not\leftrightarrow j$ for all $i \in A_p$ and $j \in A_q$ with $p \neq q$
- The sets are called *communicating classes*
- State properties are called *class properties* if they are shared by all the states in a class

Markov chains – basic definitions

- **Definition 2: irreducibility**

A Markov chain whose state space \mathbb{S} has *a single communicating class* is said *irreducible*, otherwise it is *reducible*

- If a Markov chain is irreducible, then *all its states communicate*

- **Definition 3a: first return time**

For each state $i \in \mathbb{S}$, the first return time is defined by

$$T_i^r = \inf \{n \geq 1, \Sigma_n = i\}$$

- **Definition 3b: mean return time**

For each pair of states $i, j \in \mathbb{S}$, the mean return time from i to j is

$$\mu_{i \rightarrow j} = E[T_j^r | \Sigma_0 = i]$$

State classification

- The states of a Markov chain have the following characteristics:
- *Recurrence*: a state $i \in \mathbb{S}$ is *recurrent* if, starting from i , the chain will return to i in a finite time, with probability 1:

$$P(\Sigma_n = i \text{ for some } n \geq 1 \mid \Sigma_0 = i) = 1 \quad (\mathbf{P}^n)_{i,i} > 0$$

for some $n > 0$

- A state is *positive recurrent* if $\mu_{i \rightarrow i} < \infty$
- A state is *null recurrent* if $\mu_{i \rightarrow i} \rightarrow \infty$
- *Transience*: a state is *transient* when it is *not recurrent*
- Recurrence and transience are *class properties*
- *Absorbency*: a state is *absorbing* if $(\mathbf{P})_{i,i} = 1$

State classification

- Consider the state $i \in \mathbb{S}$ and define the sequence

$$S_i = \{m \geq 1: (\mathbf{P}^m)_{i,i} > 0\}$$

- The period of the state i is the *greatest common divisor* of the S_i
- If the period is equal to 1, the state is called *aperiodic*
- If *all states* are aperiodic, the Markov chain is called aperiodic
- If $(\mathbf{P})_{i,i} > 0$, the state i is aperiodic
- Periodicity is a **class property**

Provide reference for book on Markov chains

Calculation of $\mu_{i \rightarrow j}$

- The calculation can be done recursively:

$$\mu_{i \rightarrow j} = \underbrace{1 \times (\mathbf{P})_{i,j}}_{\text{Arrives at } j \text{ in one step}} + \underbrace{\sum_{k \neq j} (\mathbf{P})_{i,k} \times \{1 + E[T_k^r | \Sigma_0 = k]\}}_{\text{Arrives at } j \text{ in one step plus the average number of steps to reach } j \text{ from } k}$$

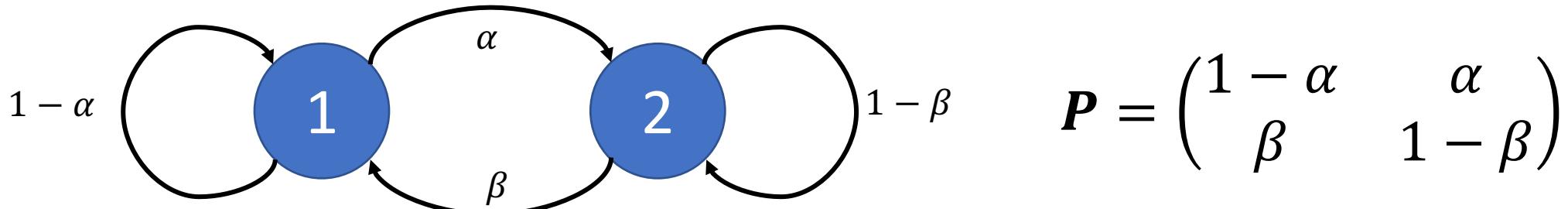
- Since $\sum_{k \in \mathbb{S}} (\mathbf{P})_{i,k} = 1$ and $\mu_{i \rightarrow j} = E[T_j^r | \Sigma_0 = i]$, the previous equation yields:

$$\mu_{i \rightarrow j} = 1 + \sum_{k \neq j} (\mathbf{P})_{i,k} \mu_{k \rightarrow j}$$

recursive
evaluation

Calculation of $\mu_{i \rightarrow j}$ for the two state Markov chain

- Take, for example, the two state Markov chain:



- In this case,

$$\mu_{1 \rightarrow 1} = 1 + \alpha \mu_{2 \rightarrow 1}, \quad \mu_{2 \rightarrow 1} = 1 + (1 - \beta) \mu_{2 \rightarrow 1}$$

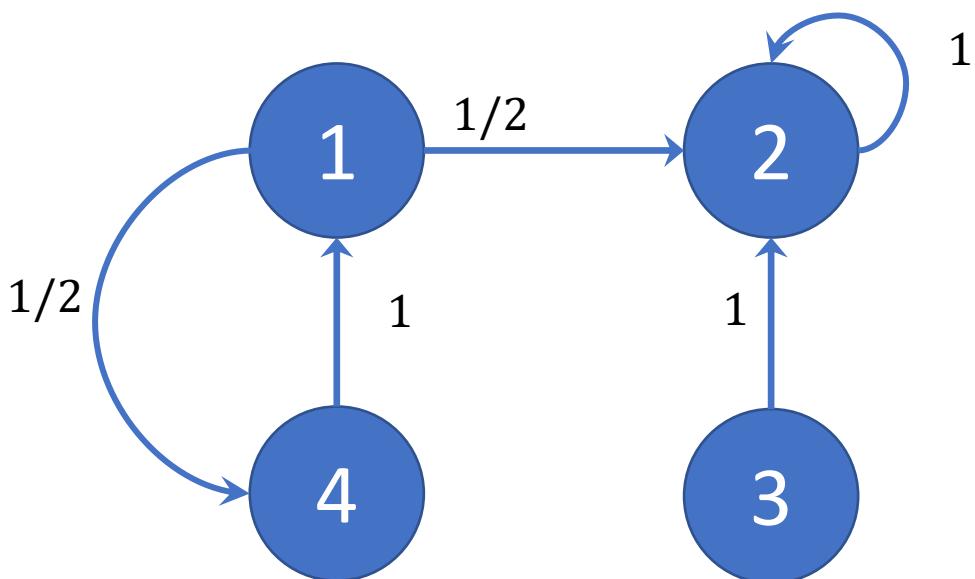
$$\mu_{2 \rightarrow 2} = 1 + \beta \mu_{1 \rightarrow 2}, \quad \mu_{1 \rightarrow 2} = 1 + (1 - \alpha) \mu_{1 \rightarrow 2}$$

- Solving the linear equations, one gets:

$$\mu_{1 \rightarrow 1} = 1 + \frac{\alpha}{\beta}, \quad \mu_{2 \rightarrow 1} = \frac{1}{\beta}, \quad \mu_{2 \rightarrow 2} = 1 + \frac{\beta}{\alpha}, \quad \mu_{1 \rightarrow 2} = \frac{1}{\alpha},$$

Reducible aperiodic Markov chain

- If P has one all-zero column, the associated Markov chain is reducible



$$P = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

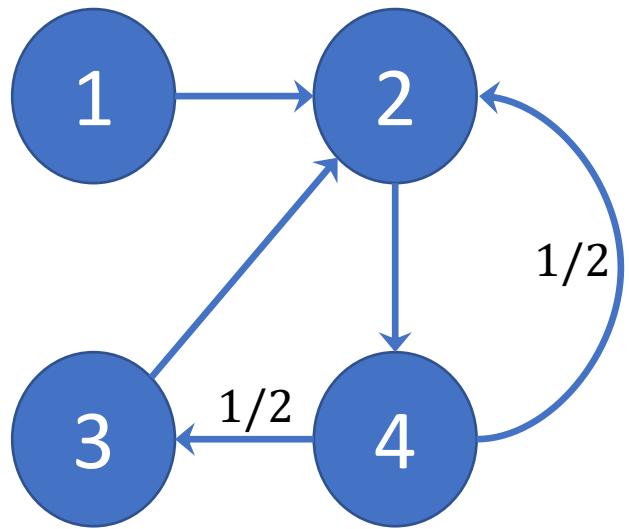
States 1,4 are recurrent.
State 2 is absorbing since $(P)_{2,2} = 1$.
State 3 is transient since $(P)_{i,3} = 0$.
The communication classes are:
 $\{1,4\}, \{2\}, \{3\}$

State evolution

- On the right, we have a sequence of powers of the transition matrix \mathbf{P}
- Whatever the initial state probability distribution \mathbf{p}_0 , the stationary state distribution is $\mathbf{p}_\infty = (0,1,0,0)$

$$\mathbf{P} = \begin{bmatrix} 0 & 0.5000 & 0 & 0.5000 \\ 0 & 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 \\ 1.0000 & 0 & 0 & 0 \end{bmatrix}$$
$$\mathbf{P}^2 = \begin{bmatrix} 0.5000 & 0.5000 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 \\ 0 & 0.5000 & 0 & 0.5000 \end{bmatrix}$$
$$\mathbf{P}^3 = \begin{bmatrix} 0 & 0.7500 & 0 & 0.2500 \\ 0 & 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 \\ 0.5000 & 0.5000 & 0 & 0 \end{bmatrix}$$
$$\mathbf{P}^5 = \begin{bmatrix} 0 & 0.8750 & 0 & 0.1250 \\ 0 & 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 \\ 0.2500 & 0.7500 & 0 & 0 \end{bmatrix}$$
$$\mathbf{P}^{10} = \begin{bmatrix} 0.0312 & 0.9688 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 \\ 0 & 0.9688 & 0 & 0.0312 \end{bmatrix}$$
$$\mathbf{P}^\infty = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Reducible aperiodic Markov chain



State 1 is transient so that the chain is reducible.
However, a stationary state distribution exists
since the irreducibility condition is not necessary.

$$P =$$

0	1.0000	0	0
0	0	0	1.0000
0	1.0000	0	0
0	0.5000	0.5000	0
0	0	0	1.0000
0	0.5000	0.5000	0
0	0	0	1.0000
0	0.5000	0	0.5000
0	0.5000	0.5000	0
0	0.5000	0	0.5000
0	0.2500	0.2500	0.5000
0	0.2500	0.2500	0.5000
0	0.5000	0.2500	0.2500
0	0.2500	0.2500	0.5000
0	0.3750	0.1250	0.5000
0	0.3750	0.1875	0.4375
0	0.4062	0.2188	0.3750
0	0.3750	0.1875	0.4375
0	0.4062	0.1875	0.4062
0	0.4000	0.2000	0.4000
0	0.4000	0.2000	0.4000
0	0.4000	0.2000	0.4000
0	0.4000	0.2000	0.4000

$$P^2 =$$

$$P^3 =$$

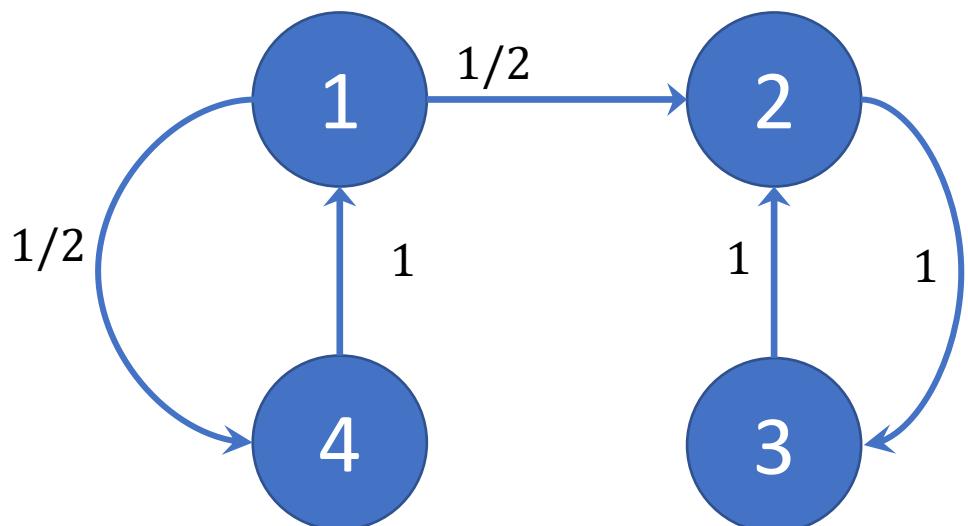
$$P^5 =$$

$$P^{10} =$$

$$P^\infty =$$

Reducible periodic Markov chain

- Another example of reducible Markov chain obtained by modifying the previous one:



$$P = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

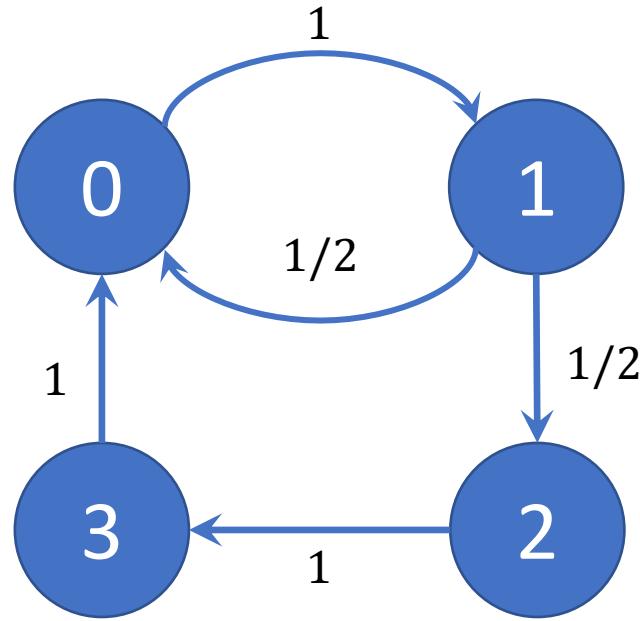
The communication classes are:
 $\{1,4\}, \{2,3\}$

State evolution

- On the right, we have a sequence of powers of the transition matrix \mathbf{P}
- The powers of \mathbf{P} do not converge
- Therefore, \mathbf{p}_∞ does not exist

$\mathbf{P} =$	0	0.5000	0	0.5000
	0	0	1.0000	0
	0	1.0000	0	0
	1.0000	0	0	0
$\mathbf{P}^2 =$	0.5000	0	0.5000	0
	0	1.0000	0	0
	0	0	1.0000	0
	0	0.5000	0	0.5000
$\mathbf{P}^3 =$	0	0.7500	0	0.2500
	0	0	1.0000	0
	0	1.0000	0	0
	0.5000	0	0.5000	0
$\mathbf{P}^5 =$	0	0.8750	0	0.1250
	0	0	1.0000	0
	0	1.0000	0	0
	0.2500	0	0.7500	0
$\mathbf{P}^{2n} =$	0	0	1	0
	0	1	0	0
	0	0	1	0
$n \rightarrow \infty$	0	1	0	0
$\mathbf{P}^{2n+1} =$	0	1	0	0
	0	0	1	0
	0	1	0	0
	0	0	1	0

Irreducible periodic Markov chain



$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

- Build the S_i sequences:
- All states have period 2

$$\begin{aligned} S_0 &= \{2, 4, 6, 8, 10, \dots\} \\ S_1 &= \{2, 4, 6, 8, 10, \dots\} \\ S_2 &= \{4, 6, 8, 10, 12, \dots\} \\ S_3 &= \{4, 6, 8, 10, 12, \dots\} \end{aligned}$$

State evolution

- On the right the sequence of powers of the transition matrix P
- We can see that P^n does not converge but oscillates between two different matrices
- For $p_0 = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right)$, we have
 $p_\infty = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6}\right)$
- But for $p_0 = (1,0,0,0)$, p_∞ does not exist

$n \rightarrow$

P^{2n}

$P^{1000} =$

P^{2n+1}

$P^{1001} =$

$P =$

$P^2 =$

$P^3 =$

$P^5 =$

0	1.0000	0	0
0.5000	0	0.5000	0
0	0	0	1.0000
1.0000	0	0	0
0.5000	0	0.5000	0
0	0.5000	0	0.5000
1.0000	0	0	0
0	1.0000	0	0
0	0.5000	0	0.5000
0.7500	0	0.2500	0
0	1.0000	0	0
0.5000	0	0.5000	0
0	0.7500	0	0.2500
0.6250	0	0.3750	0
0	0.5000	0	0.5000
0.7500	0	0.2500	0
0.6667	0	0.3333	0
0	0.6667	0	0.3333
0.6667	0	0.3333	0
0	0.6667	0	0.3333
0	0.6667	0	0.3333
0.6667	0	0.3333	0
0	0.6667	0	0.3333
0.6667	0	0.3333	0

Markov chain: convergence of p_n

- Summarizing the previous cases, we have seen that the state distribution doesn't converge when the Markov chain is periodic
- However, it may converge even if the Markov chain is reducible and aperiodic
- The following result provides a sufficient convergence condition:

If a finite-state Markov chain is irreducible and aperiodic, there exists a unique stationary state probability distribution p_∞ such that

$$p_\infty = \lim_{n \rightarrow \infty} p_n = \lim_{n \rightarrow \infty} p_0 P^n \quad \forall p_0$$

↑
prob. vector

How to calculate \mathbf{p}_∞

- The calculation of the successive powers of P converges to a matrix whose rows are \mathbf{p}_∞ when convergence is granted
- If a finite-state Markov chain is irreducible and aperiodic, the stationary state probability distribution \mathbf{p}_∞ can be obtained by solving the equation

$$\mathbf{p}_\infty = \mathbf{p}_\infty \cdot P \Rightarrow \mathbf{p}_\infty(I - P) = \mathbf{0}$$

- This equation system can be solved since 1 is an eigenvalue of P , so that $\det(I - P) = 0$
- In fact, $\cancel{\mathbf{1}}P\cancel{\mathbf{1}} = \mathbf{1}$ since the sum of the row elements of P is always 1

all-1 row-vector
~~column~~

$$P\underline{x} = \lambda \underline{x}$$

$$\det(P - \lambda I) = 0$$
$$\det(P - I) = 0$$

P

$$A = \text{eye}(n) - P;$$

$$A = A';$$

$$A = [A(1:n-1, :); \\ \text{ones}(1, n)];$$

$$(P_1, P_2, P_3) \begin{bmatrix} x & x \\ x & x \\ x & x \end{bmatrix} = (0, 0)$$

$$(P_1, P_2, P_3) \begin{bmatrix} x & x & 1 \\ x & x & 1 \\ x & x & 1 \end{bmatrix} = (0, 0, 1) \quad p = A \setminus [zeros(n-1, 1) \\ ; 1];$$

$$\begin{bmatrix} x & x & x \\ x & x & x \\ 1 & 1 & 1 \end{bmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

check that
 $P * p = p$

How to calculate \mathbf{p}_∞

- To solve the homogeneous linear equation system

$$\mathbf{p}_\infty(\mathbf{I} - \mathbf{P}) = \mathbf{0}$$

- We proceed as follows:
 - We discard one of the equations because the coefficient matrix $(\mathbf{I} - \mathbf{P})$ has not full rank
 - We replace this equation by the normalization condition that requires \mathbf{p}_∞ to be a probability vector, so that $\mathbf{p}_\infty \mathbf{1}^T = 1$
 - The vector \mathbf{p}_∞ is a left-eigenvector of \mathbf{P} corresponding to the eigenvalue 1
 - The vector \mathbf{p}_∞ obtained by solving this linear equation system is a probability vector (nonnegative components adding to 1) by the Perron-Frobenius Theorem

Nicolas Privault

Understanding Markov Chains

Examples and Applications

Second Edition

L
N
S

Graduate Texts
in Mathematics

Daniel W. Stroock

An Introduction to
Markov Processes

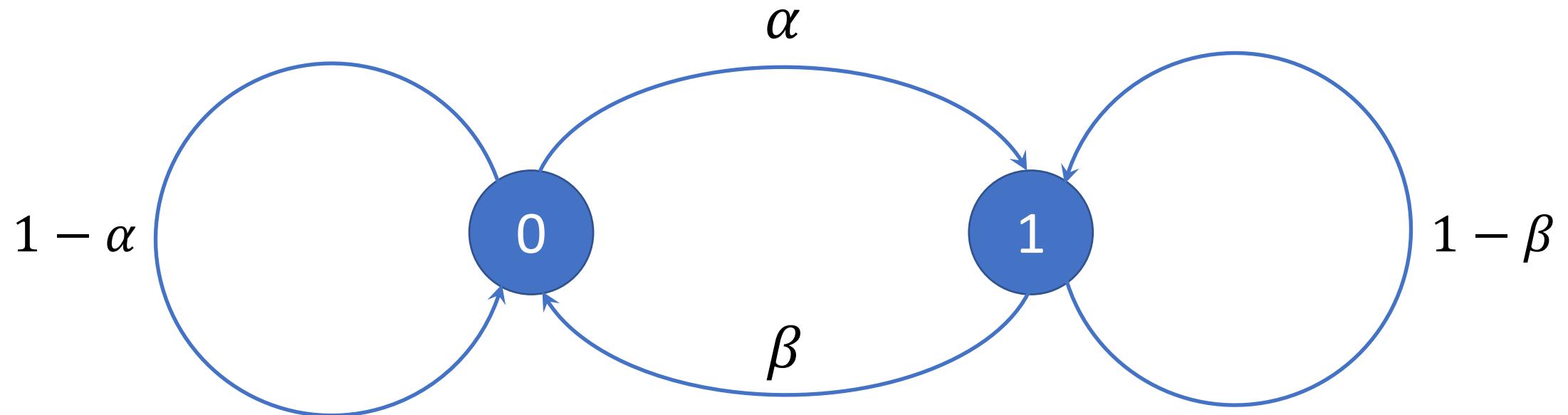
 Springer

Two-state Markov chain

- Probability matrix

$$P = \begin{pmatrix} 1 - \alpha & \cancel{\beta} \\ \cancel{\alpha} & 1 - \beta \end{pmatrix}, \quad \alpha, \beta \in [0,1]$$

- Graph:



Two-state Markov chain

- The stationary state probability distribution $\mathbf{p}_\infty = (p_0, p_1)$ can be obtained by solving the equations:

$$p_0 = \frac{\beta}{\alpha + \beta}$$

$$p_1 = \frac{\alpha}{\beta} p_0$$

$$(1 - \alpha)p_0 + \beta p_1 = p_0$$

$$\alpha p_0 + (1 - \beta)p_1 = p_1$$

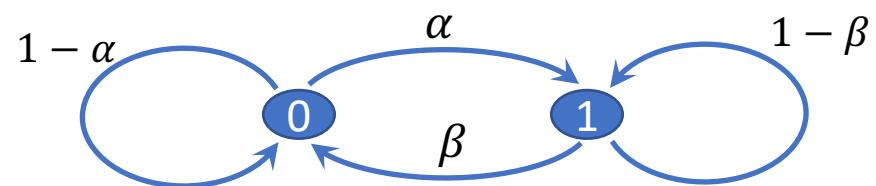
$$p_0 + p_1 = 1$$

- The equation can be found by following the paths leading to each state in the graphic representation
- The solution is

$$p_0 = \frac{\beta}{\alpha + \beta}, \quad p_1 = \frac{\alpha}{\alpha + \beta}$$

$$\begin{aligned} -\alpha p_0 + \beta p_1 &= 0 \\ \alpha p_0 - \beta p_1 &= 0 \end{aligned} \quad \left. \begin{array}{l} \text{linearly} \\ \text{dependent} \end{array} \right.$$

$$(p_0, p_1) \begin{pmatrix} 1 - \alpha & \alpha \\ \beta & 1 - \beta \end{pmatrix} = (p_0, p_1)$$



$$\underline{p} = (p_1, p_2, \dots, p_n) \quad p_i \geq 0 \quad \sum_i p_i = 1$$

$$H(\underline{p}) = - \sum_i p_i \log_2 \frac{1}{p_i} = - \sum_i p_i \log_2 p_i \quad [\text{bit}]$$

Entropy rate \rightarrow generalization of the entropy for information sources with memory (not independent)

Entropy rate of a Markov source

- General definition:

$$\bar{H} \stackrel{\text{def}}{=} \lim_{n \rightarrow \infty} \frac{H(X_{1:n})}{n}$$

$$H(X_{1:n}) = \sum_{i=1}^n H(x_i)$$
$$= n \cdot H(X)$$

- Stationary independent source:

not independent

$$\bar{H} = H(X)$$

Time index dropped
for stationarity

- For a stationary source,

$$\bar{H} \stackrel{\text{def}}{=} \lim_{n \rightarrow \infty} H(X_n | \underbrace{X_{1:n-1}}_{\text{represents the state of the source}})$$

- For a Markov source,

$$\bar{H} = H(X|\Sigma) = - \sum_{\sigma \in \mathcal{X}^L} p_\infty(\sigma) \sum_{x \in \mathcal{X}} p(x|\sigma) \log_2 p(x|\sigma)$$

Time index dropped
for stationarity

Stationary state distribution

Conditional distribution

$$H(y|x) = \sum_x \sum_y p(x,y) \log_2 \frac{1}{p(y|x)}$$

$$= - \sum_x p(x) \sum_y p(y|x) \log_2 p(y|x)$$

↓ ↓
 is the state Σ is the current
 of the source symbol X

$$H(x|\Sigma) = - \sum_{\sigma} p(\sigma) \sum_x p(x|\sigma) \log_2 p(x|\sigma)$$

↑
 stationary state
 dist₂.

Markov source: example 1

- The first example is a two-state Markov source
- In state 0, the source emits symbol s with probability distribution vector $\alpha_0 \ p_0$
- In state 1, the source emits symbol s with probability distribution vector $\alpha_1 \ p_1$
- The entropy rate is

$$\bar{H} = \frac{\beta}{\alpha + \beta} H(p_0) + \frac{\alpha}{\alpha + \beta} H(p_1)$$

$p_{\infty}(0) \qquad \qquad p_{\infty}(1)$

Markov source: example 2

- We consider a binary source

- With memory $L = 2$

- Defined by the conditional symbol probabilities

$$P(X_n = 0 | X_{n-1} + X_{n-2} = 0) = p_0 \rightarrow P(1 | \dots = 0) = 1 - p_0$$

$$P(X_n = 0 | X_{n-1} + X_{n-2} = 1) = p_1$$

$$P(X_n = 0 | X_{n-1} + X_{n-2} = 2) = p_2$$

- To find the stationary state distribution we must find the matrix P

- We define

- Starting state: X_{n-2}, X_{n-1}

- Ending state: X_{n-1}, X_n

conditional symbol probabilities

$$\begin{aligned} P(X_n = 0 | X_{n-1} + X_{n-2} = 0) &= p_0 \\ P(X_n = 0 | X_{n-1} + X_{n-2} = 1) &= p_1 \\ P(X_n = 0 | X_{n-1} + X_{n-2} = 2) &= p_2 \end{aligned}$$

$$\sigma_k = (x_{n-2}, x_{n-1}), x_{n-2} + x_{n-1} = k, k=0,1,2$$

	x_{n-2}	x_{n-1}	x_n
σ_0	0	0	0
	0	0	1
σ_1	0	1	0
	0	1	1
σ_2	1	0	0
	1	0	1
σ_3	1	1	0
	1	1	1

$$\begin{aligned} \sigma_0 &\rightarrow \sigma_0 \\ \sigma_0 &\rightarrow \sigma_1 \\ \sigma_1 &\rightarrow \sigma_1 \quad \swarrow \\ \sigma_1 &\rightarrow \sigma_2 \\ \sigma_1 &\rightarrow \sigma_0 \quad \swarrow \\ \sigma_1 &\rightarrow \sigma_1 \\ \sigma_2 &\rightarrow \sigma_1 \\ \sigma_2 &\rightarrow \sigma_2 \end{aligned}$$

If we used σ_1 , there would be ambiguity on the arrival state correspond. to $x_n = 0$: $\sigma_1 \rightarrow \sigma_1$ or σ_0 ?

↓
4 states necessary in this case

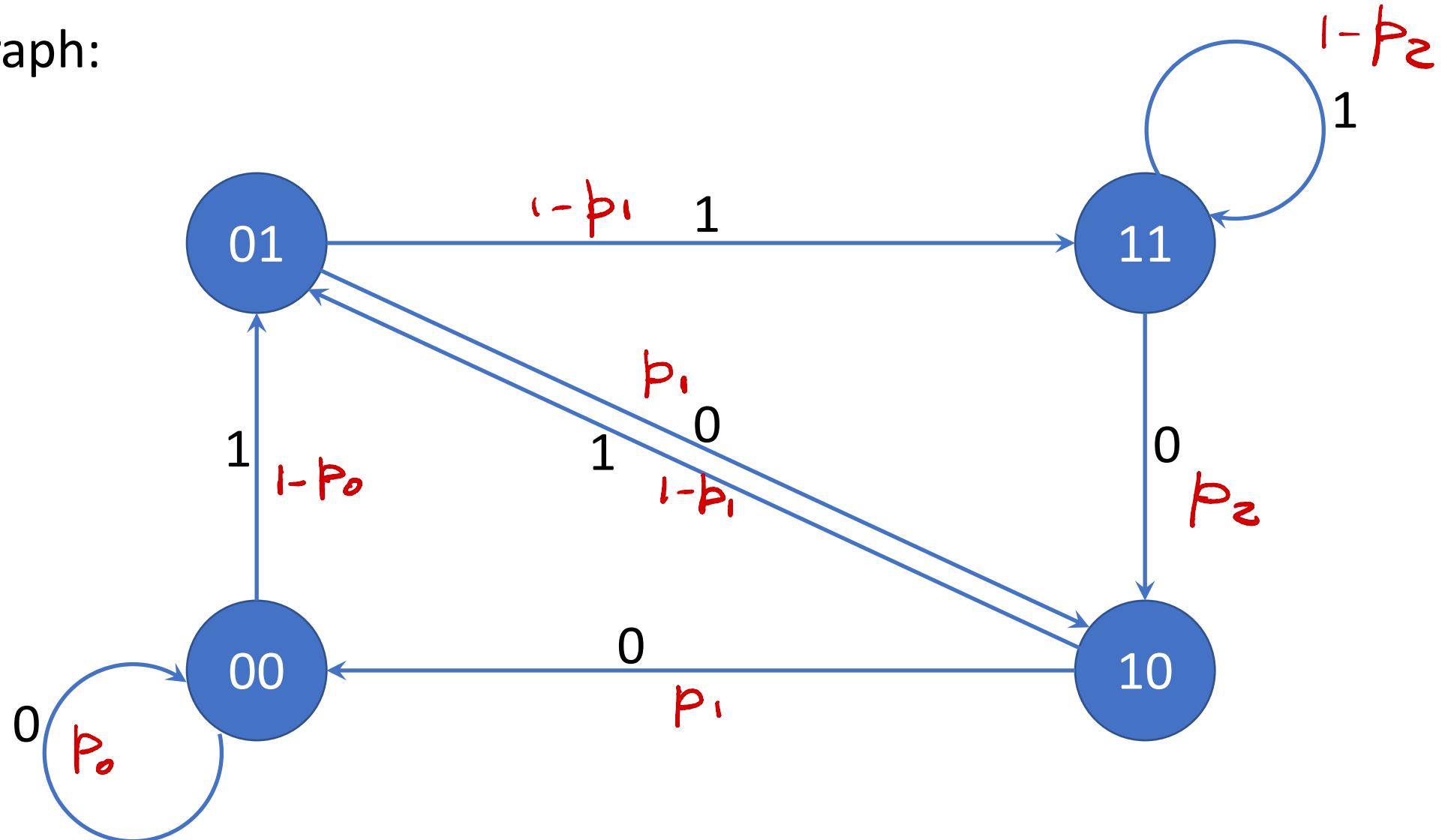
Markov source: example 2

- The state evolution is summarized by the following table:

Starting state	Current symbol	Ending state	Probability
X_{n-2}, X_{n-1}	X_n	X_{n-1}, X_n	
00	0 	0 	p_0
00	1	01	$1 - p_0$
01	0	10	p_1
01	1	11	$1 - p_1$
10	0	00	p_1
10	1	01	$1 - p_1$
11	0	10	p_2
11	1	11	$1 - p_2$

Markov source: example 2

- Graph:



Markov source: example 2

- Matrix:

$$P = \begin{pmatrix} p_0 & 0 & p_1 & 0 \\ 1 - p_0 & 0 & 1 - p_1 & 0 \\ 0 & p_1 & 0 & p_2 \\ 0 & 1 - p_1 & 0 & 1 - p_2 \end{pmatrix}$$

Transpose

- The states are listed as

1. 00
2. 01
3. 10
4. 11

$$P = \begin{pmatrix} p_0 & 1 - p_0 & 0 & 0 \\ 0 & 0 & p_1 & 1 - p_1 \\ p_1 & 1 - p_1 & 0 & 0 \\ 0 & 0 & p_2 & 1 - p_2 \end{pmatrix}$$

$$(p_{00}, p_{01}, p_{10}, p_{11}) P = (p_{00}, p_{01}, p_{10}, p_{11})$$

Markov source: example 2

- Finally, we solve the stationary state probability distribution equations:

$$p_{00} = p_0 p_{00} + p_1 p_{10}$$

$$p_{01} = (1 - p_0) p_{00} + (1 - p_1) p_{10}$$

$$p_{10} = p_1 p_{01} + p_2 p_{11}$$

$$1 = p_{00} + p_{01} + p_{10} + p_{11}$$

p_{ij} is the value
of p_∞ corresp.-to
the state "ij"

- The solution is

$$p_{00} = \alpha p_1 p_2$$

$$p_{01} = p_{10} = \alpha(1 - p_0)p_2$$

$$p_{11} = \alpha(1 - p_0)(1 - p_1)$$

$$\alpha = \frac{1}{p_1 p_2 + 2(1 - p_0)p_2 + (1 - p_0)(1 - p_1)}$$

Entropy rate

- Collecting the previous results, we get the entropy rate:

$$\bar{H} = \frac{p_1 p_2 H_b(p_0) + 2(1 - p_0)p_2 H_b(p_1) + (1 - p_0)(1 - p_1)H_b(p_2)}{p_1 p_2 + 2(1 - p_0)p_2 + (1 - p_0)(1 - p_1)}$$

$$H_b(p) \stackrel{\text{def}}{=} -p \log_2 p - (1 - p) \log_2(1 - p)$$

$$= H(p, 1-p)$$

Information Theory and Applications

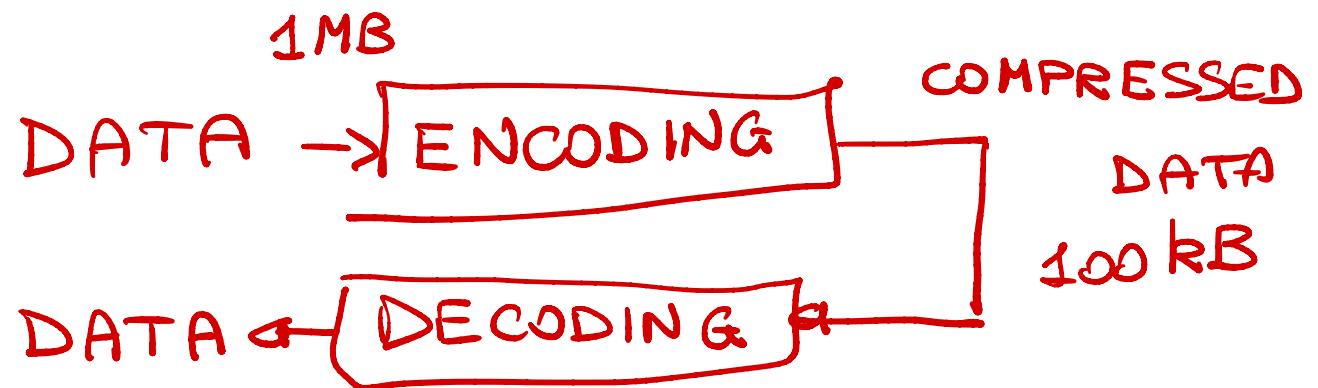
Source coding

Prof. Giorgio Taricco

Politecnico di Torino – DET

Source coding

- The goal of source coding is reducing the amount of data necessary to store a large set of symbols on a storage system
- This is sometimes referred to as **data compression**
- Source coding must be an **invertible** operation: from encoded data one must be able to retrieve the original data
- Depending on the length of the source and encoded blocks, we have three types of source coding algorithms:
 - Fixed-to-fixed
 - Fixed-to-variable
 - Variable-to-fixed



Fixed-to-fixed source coding

- We consider a block of N source symbols from an alphabet \mathcal{X} with cardinality $M = |\mathcal{X}|$
- There are M^N possible blocks so that a number from 0 to $M^N - 1$ identifies uniquely the source block
- Since n bits are sufficient to represent all integers between 0 and $2^n - 1$, we must have $2^n \geq M^N$ or $n \geq N \log_2 M$
- Since n must be an integer, its minimum value is given by
$$n = \lceil N \log_2 M \rceil$$
- The corresponding number of bits per encoded symbol is

$$\bar{n} = \frac{\nu}{N} = \frac{\lceil N \log_2 M \rceil}{N} \approx \log_2 M$$

Example

- Consider the English alphabet plus some punctuation characters $\mathcal{X} = \{A, B, C, \dots, Z, ., ', ',', ;, :, !\}$ with $M = |\mathcal{X}| = 32$ characters
- \mathcal{X} is encoded as follows:

A	00000	B	00001	C	00010	D	00011	E	00100
F	00101	G	00110	H	00111	I	01000	J	01001
K	01010	L	01011	M	01100	N	01101	O	01110
P	01111	Q	10000	R	10001	S	10010	T	10011
U	10100	V	10101	W	10110	X	10111	Y	11000
Z	11001	.	11010	,	11011	"	11100	;	11101
:	11110	!	11111						

Example

- Encode the sentence “GOOD NIGHT!”

00110|1110|01110|00011|11100|01010|10000|0011000111100111111

A	00000	B	00001	C	00010	D	00011	E	00100
F	00101	G	00110	H	00111	I	01000	J	01001
K	01010	L	01011	M	01100	N	01101	O	01110
P	01111	Q	10000	R	10001	S	10010	T	10011
U	10100	V	10101	W	10110	X	10111	Y	11000
Z	11001	.	11010	,	11011	‘’	11100	;	11101
:	11110	!	11111						

Decoding

- Decoding fixed-to-fixed source codes is very simple
 - We know how many bits (ν) form one encoded character
 - We process the bit stream extracting blocks of ν bits and use a look-up table to obtain the source symbols
 - For example
 - “OPEN BOOK” →
01110 01111 00100 01101 11100 00001 01110 01110 01010

O P E N B O O K

Fixed-to-variable source coding

- The concept is that a fixed number of symbols is encoded into a variable number of bits
- The approach is advantageous, on average, if the source symbols have different probabilities
 - Most likely source symbols are assigned shorter codewords
 - On the other hand, longer codewords are assigned to less likely source symbols but their impact is limited on the average codeword length

Fixed-to-variable source coding

- Assume a stationary source generating symbols from an alphabet \mathcal{X} , encoded individually

- Let the probability distribution be

$$p_i \stackrel{\text{def}}{=} P(X = \xi_i), \quad i = 1, \dots, (M \stackrel{\text{def}}{=} |\mathcal{X}|)$$

$$\frac{N_i}{N} \rightarrow p_i$$

- Let the codeword lengths be v_i (bit/symbol)

- The average number of bit per symbol for an encoded string is

$$\langle v \rangle = \frac{1}{N} \sum_{i=1}^N N_i v_i$$

$$\mathcal{X} = \{A, B, C\}$$

$$AABAC \rightarrow N = 5$$

- N : n. of symbols in the string
- N_i : n. of occurrences of ξ_i

$$\langle v \rangle = \frac{3 \cdot 1 + 1 \cdot 2 + 1 \cdot 2}{5} = 1.4$$

$N_1 = 3$	$v_1 = 1$
$N_2 = 1$	$v_2 = 2$
$N_3 = 1$	$v_3 = 2$

Fixed-to-variable source coding

- For a very long string, the frequencies of the symbols are approximately the probabilities of their occurrences:

$$\frac{N_i}{N} \approx p_i, \quad i = 1, \dots, M$$

- Accordingly, the average number of bit per symbol for an encoded string is approximated by

$$\bar{v} = \sum_{i=1}^N p_i v_i$$

- \bar{v} is the expected average number of bit per symbol of the source code and represents a quality measure for the source code
- The lower \bar{v} , the better the source code

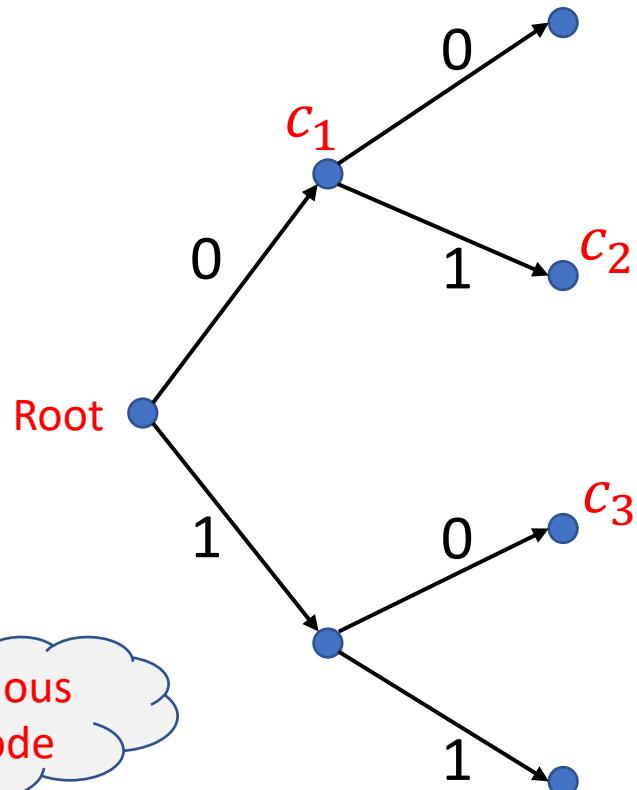
Decoding fixed-to-variable source codes

- Decoding is complicated by the fact that one doesn't know where encoded symbols are separated
- This may generate **decoding ambiguity**
- For example, consider the code
$$c(\xi_1) = 0, \quad c(\xi_2) = 01, \quad c(\xi_3) = 10$$
- Apply this code to the sequence symbols
 - $\xi_1 \xi_3 \xi_2 \xi_1$
 - $\xi_2 \xi_1 \xi_2 \xi_1$
- In both cases we obtain 010010

Decoding fixed-to-variable source codes

- To resolve decoding ambiguity codes must have a tree structure

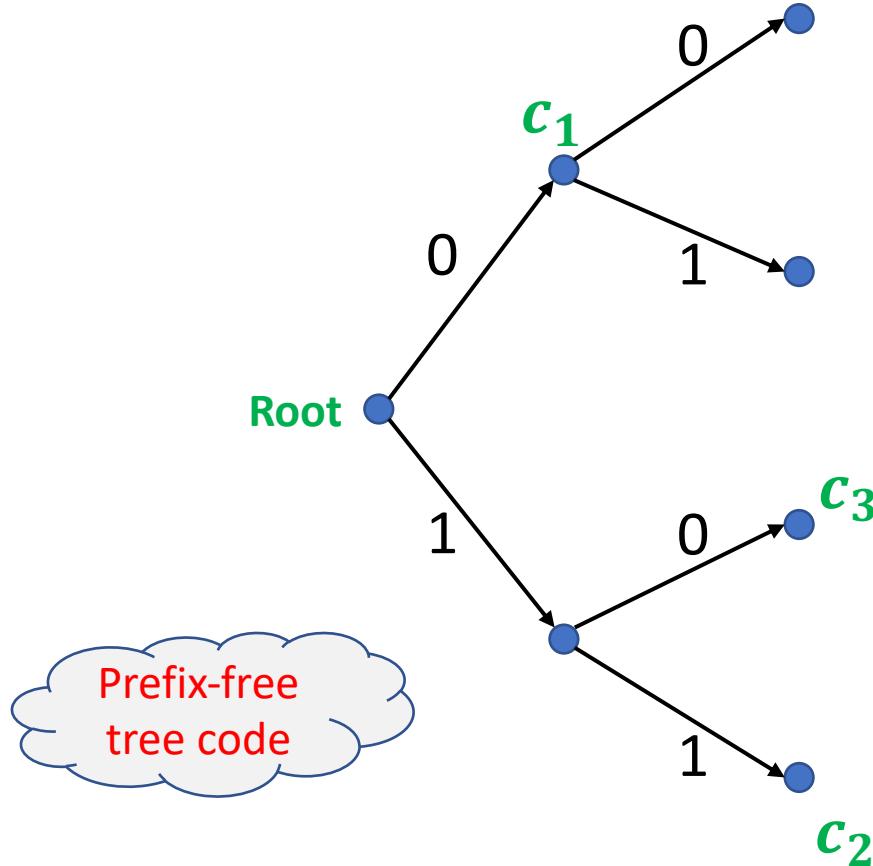
{but this
is not
enough!}



- The code words (bit sequences representing each symbol) are located at tree nodes
- The bits of each code word are read off traveling the tree from the root to the node
- For example, c_2 is read passing through a path labelled 0 and another labelled 1
- If one code word is at a node inside the path to another codeword, the former is prefix of the latter
- Fixed-to-variables source codes are uniquely decodable if they satisfy the **prefix-free** condition

Decoding fixed-to-variable source codes

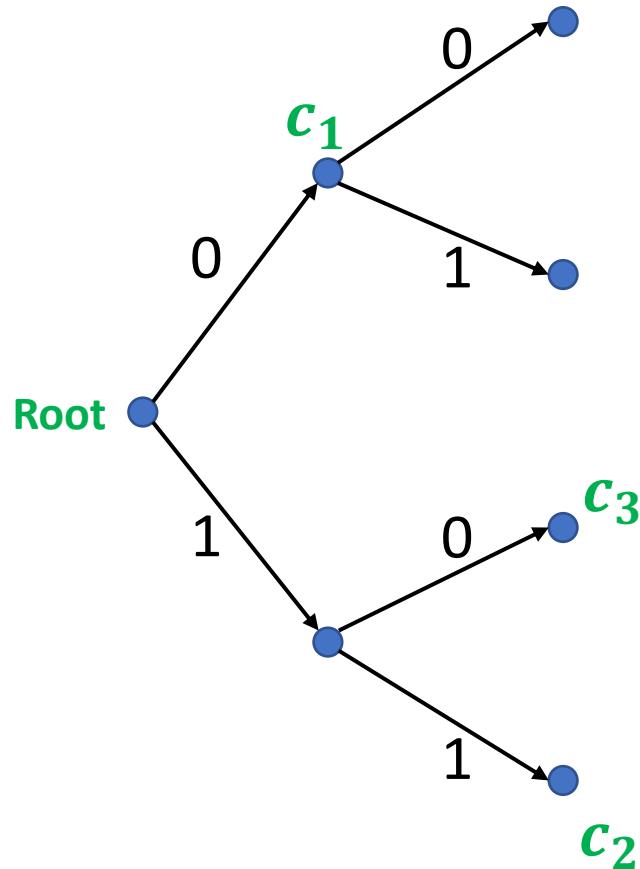
- The following code satisfies the no-prefix condition:



- Apply this code to the sequence symbols
 - $\xi_1 \xi_3 \xi_2 \xi_1$
 - $\xi_2 \xi_1 \xi_2 \xi_1$
- In the first case we get
 - 010110
- In the second case:
 - 110110

Decoding fixed-to-variable source codes

- If a code satisfies the no-prefix condition, decoding is always uniquely possible



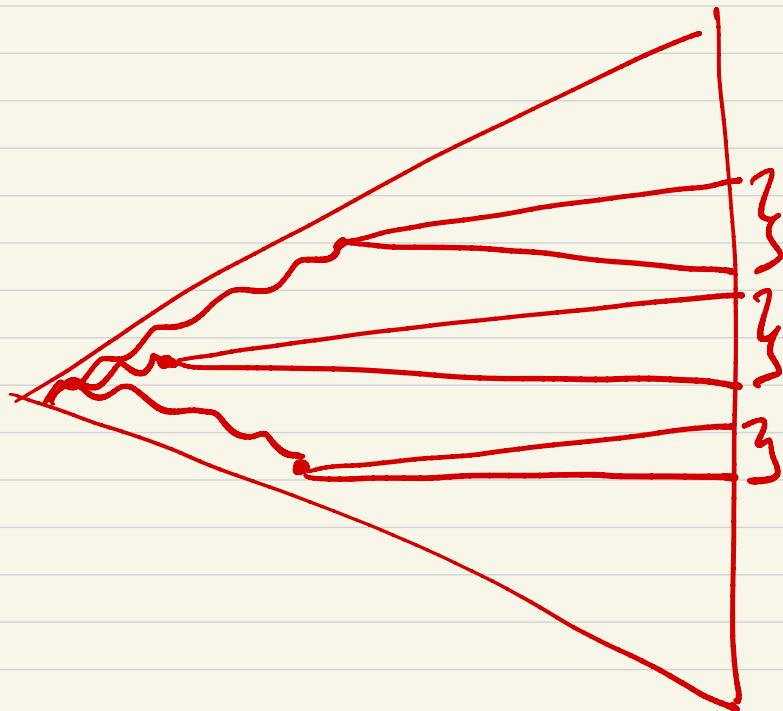
- Decode the bit sequence 0111001110

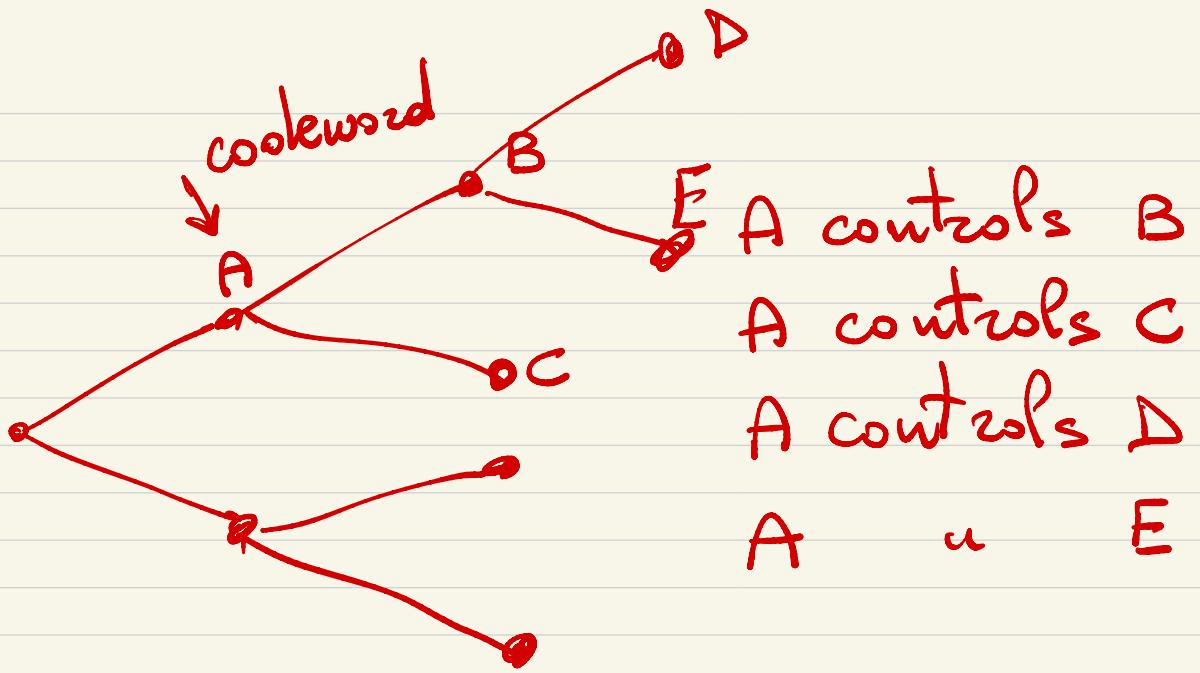
\downarrow
 $c_1 | c_2 | c_3 | c_1 | c_2 | c_3$
 $\xi_1 \xi_2 \xi_3 \xi_1 \xi_2 \xi_3$

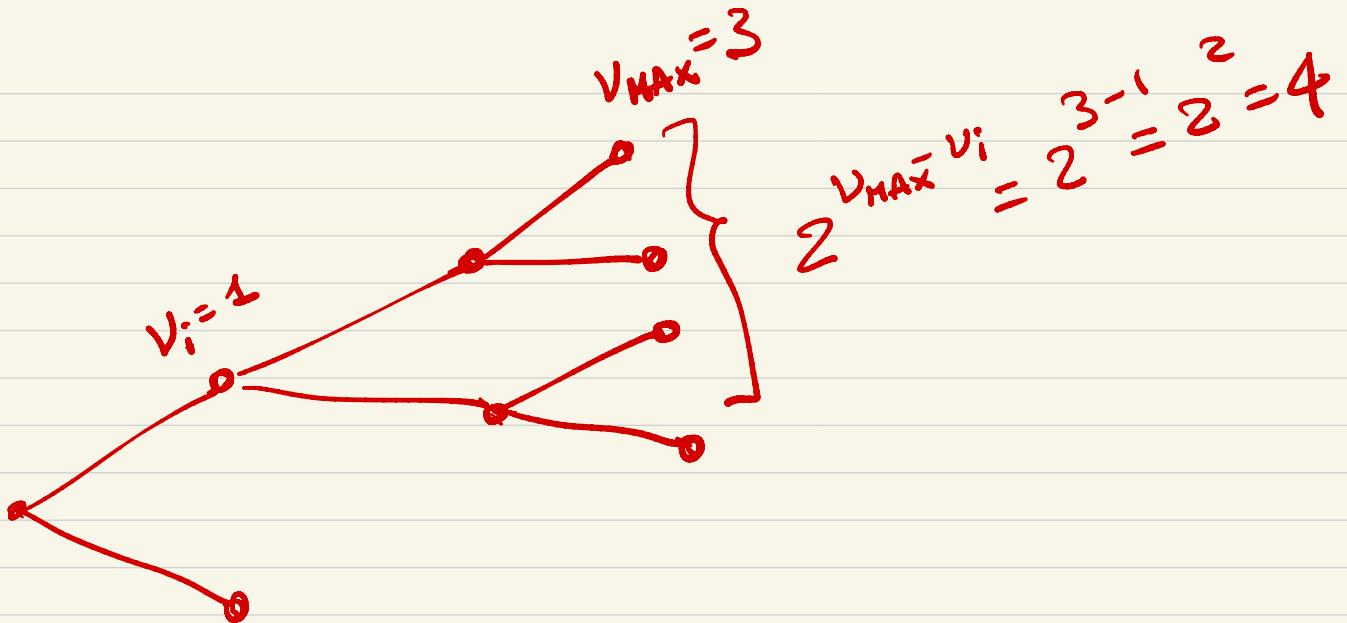
Kraft inequality

- This is a simple inequality characterizing the code word length distribution for a uniquely decodable tree code
- Every code word is located on one node of the tree
- The nodes reachable traveling the tree from this node are said to be controlled
- A node at depth ν_i controls $2^{\nu_{\max} - \nu_i}$ nodes at depth ν_{\max}
- The sub-tree stemming from it up to depth ν_{\max} does not contain other codewords to satisfy the prefix-free condition
- We have the inequality

$$\sum_{i=1}^M 2^{\nu_{\max} - \nu_i} \leq 2^{\nu_{\max}}$$







Kraft inequality

- Inequality

$$\sum_{i=1}^M 2^{\nu_{\max} - \nu_i} \leq 2^{\nu_{\max}}$$

- The total number of controlled nodes at depth ν_{\max} cannot exceed the number of nodes at depth ν_{\max} , i.e., $2^{\nu_{\max}}$
- Dividing both sides, we obtain the Kraft inequality:

$$\sum_{i=1}^M 2^{-\nu_i} \leq 1$$

Lower bound to \bar{v}

- We want to choose the code word lengths v_i to minimize the average number of bits per symbol

$$\bar{v} = \sum_{i=1}^M p_i v_i$$

- Our constraint is the Kraft inequality:

$$\sum_{i=1}^M 2^{-v_i} \leq 1$$

- Applying the method of Lagrange multipliers, we build the Lagrangian function for this problem:

$$\mathcal{L}(v_1, \dots, v_M) = \sum_{i=1}^M p_i v_i + \lambda \left(\sum_{i=1}^M 2^{-v_i} - 1 \right)$$

constraint
objective fct.

Lower bound to \bar{v}

$$p_i = 2^{-v_i} \rightarrow v_i = -\log_2 p_i$$

- This is a convex optimization problem which can be solved by the following equations:

$$\frac{\partial \mathcal{L}}{\partial v_i} = p_i - \lambda \cdot \ln 2 \cdot 2^{-v_i} = 0$$
$$\lambda \left(\sum_{i=1}^M 2^{-v_i} - 1 \right) = 0 \quad (*)$$
$$\sum_i p_i = 1 = \lambda \ln 2 \cdot \sum_i 2^{-v_i}$$
$$= \lambda \ln 2$$
$$\lambda = \log_2 e$$

- The solution requires $\lambda \cdot \ln 2 = 1$ since $\sum_{i=1}^M p_i = 1$ so that $(*)$ is satisfied
- We get $v_i = -\log_2 p_i$ and

$$\bar{v} = \sum_{i=1}^M p_i (-\log_2 p_i) = H(X)$$

Lower bound to \bar{v}

$$p_1 = \frac{1}{2} \quad p_2 = \frac{1}{4} \quad p_3 = \frac{1}{4}$$

- The solution $v_i = -\log_2 p_i$ is acceptable only if the lengths are all integer numbers
- In that case, $\bar{v} = H(X)$
- Otherwise, the source entropy is a strict lower bound:
$$\bar{v} > H(X)$$

- Summarizing, Kraft inequality implies that

$$\bar{v} \geq H(X)$$

Upper bound to \bar{v}

$$\lceil 3.1 \rceil = 4 < 3.1 + 1$$

- We can replace the lengths $v_i = -\log_2 p_i$ by their integer upper bounds:

$$v_i = \lceil -\log_2 p_i \rceil$$

- For every real number x , $\lceil x \rceil < x + 1$

- Then,

$$v_i = \lceil -\log_2 p_i \rceil < -\log_2 p_i + 1$$

- Thus,

$$\bar{v} = \sum_{i=1}^M p_i v_i < \sum_{i=1}^M p_i (-\log_2 p_i + 1) = H(X) + 1$$

Shannon Theorem

- There always exists a uniquely decodable source code such that the average number of bits per symbol \bar{v} , required to encode a stationary independent source with entropy $H(X)$, satisfies the inequalities

$$H(X) \leq \bar{v} < H(X) + 1$$

- The lower bound is necessary for the existence of the code
- The upper bound is sufficient

Information Theory and Applications

Huffman codes

Prof. Giorgio Taricco

Politecnico di Torino – DET

Examples

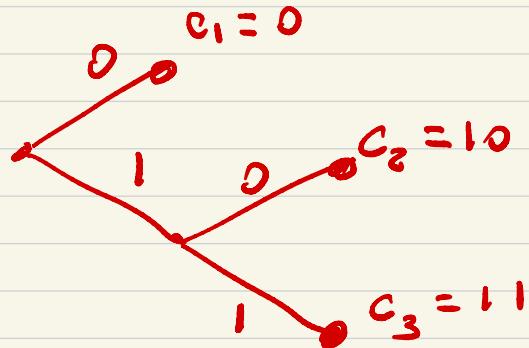
- Example 1: $p = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{2}\right)$ [Check Kraft inequality and build tree code]
- Example 2: $p = \left(\frac{1}{8}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}\right)$ [Check Kraft inequality and build tree code]
- Example 3: $p = \left(\frac{1}{8}, \frac{3}{8}, \frac{1}{2}\right)$ [Check Kraft inequality and build tree code]

$$\underline{p} = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4} \right)$$

$$v_1 = 1 \quad v_2 = 2 = v_3$$

$$\bar{v} = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 = 1.5$$

$$H(\underline{p}) = 1.5$$



Cross entropy

- The precise knowledge of the source symbol distribution is not always possible
- Let p_i be the true probability of generation of the symbol ξ_i and q_i an estimated probability which may be different from p_i
- The optimum source code based on the estimated probabilities q_i has lengths $v_i = -\log_2 q_i$
- The average length based on the true probability distribution is the cross entropy:

$$\bar{v} = H(p, q) = - \sum_i p_i \log_2 q_i \geq - \sum_i p_i \log_2 p_i$$

Cross entropy inequalities

- We notice that $H(\mathbf{p}) = H(\mathbf{p}, \mathbf{p})$
- We expect that the mismatch between the true distribution (\mathbf{p}) and the estimated distribution (\mathbf{q}) increases \bar{v} :

$$H(\mathbf{p}, \mathbf{q}) \geq H(\mathbf{p}, \mathbf{p})$$

- This can be proved by applying the inequality

$$\ln x \leq x - 1, \quad (x > 0)$$

- In fact,

$$H(\mathbf{p}, \mathbf{p}) - H(\mathbf{p}, \mathbf{q}) = \sum_i p_i \log_2 \frac{q_i}{p_i} = \frac{\sum_i p_i \ln \frac{q_i}{p_i}}{\ln 2} \leq \frac{\sum_i p_i \left(\frac{q_i}{p_i} - 1 \right)}{\ln 2} = 0$$

$= \sum_i (q_i - p_i) = \sum_i q_i - \sum_i p_i = 1 - 1 = 0$

Kullback-Leibler divergence

- The difference between the mismatched and optimum \bar{v} is the Kullback-Leibler divergence:

$$D(\mathbf{p} \parallel \mathbf{q}) \stackrel{\text{def}}{=} H(\mathbf{p}, \mathbf{q}) - H(\mathbf{p}, \mathbf{p}) \geq 0$$

- We also can write

$$D(\mathbf{p} \parallel \mathbf{q}) \stackrel{\text{def}}{=} \sum_i p_i \log_2 \frac{p_i}{q_i}$$

- There is no ordering, instead, between $H(\mathbf{p}, \mathbf{q})$ and $H(\mathbf{p}, \mathbf{q})$, i.e., the difference $H(\mathbf{p}, \mathbf{q}) - H(\mathbf{q}, \mathbf{q}) = \sum_i (q_i - p_i) \log_2 q_i$ can be negative, zero, or positive

Codes for stationary Markovian sources

- A Markovian source is characterized by a conditional probability distribution $p(x|\sigma)$
 - $x \in \mathcal{X}$ is the source symbol
 - $\sigma \in \mathcal{S}$ is the source state
- Choosing a specific source state σ , we can repeat the previous analysis and select a specific source code depending on σ
 - The optimum lengths are $-\log_2 p(x|\sigma)$
 - The average number of bits per symbol, conditional on σ , satisfies

$$-\sum_{x \in \mathcal{X}} p(x|\sigma) \log_2 p(x|\sigma) \leq \bar{v}_\sigma$$

Codes for stationary Markovian sources

- There exists a source code such that the average number of bits per symbol, conditional on σ , satisfies

$$\bar{v}_\sigma < - \sum_{x \in \mathcal{X}} p(x|\sigma) \log_2 p(x|\sigma) + 1$$

- Averaging with respect to the stationary state distribution $p(\sigma)$ and defining the average number of bits per symbol as v , we get

$$\underbrace{\left\{ \sum_{\sigma \in \mathcal{S}} p(\sigma) \left[- \sum_{x \in \mathcal{X}} p(x|\sigma) \log_2 p(x|\sigma) \right] \right\}}_{\bar{H} = H(X|\Sigma)} \leq \sum_{\sigma \in \mathcal{S}} p(\sigma) \bar{v}_\sigma = \bar{v}$$

- In a similar way:

$$\bar{v} < \bar{H} + 1$$

Huffman codes

- Huffman codes are prefix-free tree codes that minimize the average number of bits per symbol
- In this sense they are optimal source codes
- They are derived by constructing the code tree starting from the leaves
- The detailed construction algorithm is described in the following slide for the usual stationary independent source with symbols from

$$\mathcal{X} = \{\xi_1, \dots, \xi_M\}$$

- The symbol probabilities are $p_i = P(X_n = \xi_i)$ for $i = 1, \dots, M$

Huffman code construction algorithm

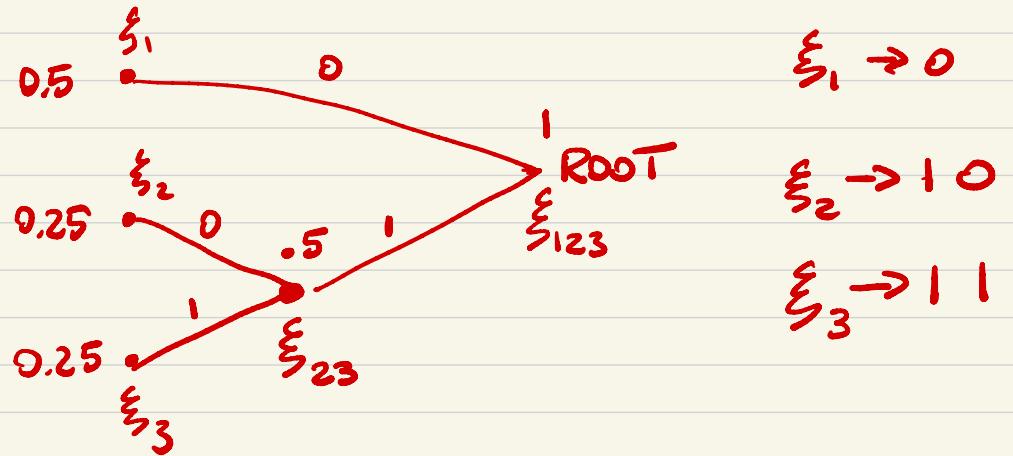
- The starting point is the set of M leave nodes corresponding to the symbols $\xi_i, i = 1, \dots, M$
- The nodes are processed sequentially according to the following rules:
 - Select two nodes with minimum probabilities
 - Build a sub-tree stemming from a new node with arcs labelled 0 and 1 reaching the two original nodes
 - Replace the two nodes with the new node and assign it a probability equal to the sum of the probabilities of the two original nodes
 - Terminate the construction when only one node (the tree root) remains

Examples

- Example 1: $p = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{2}\right)$ [Build the Huffman code]
- Example 2: $p = \left(\frac{1}{8}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}\right)$ [Build the Huffman code]
- Example 3: $p = \left(\frac{1}{8}, \frac{3}{8}, \frac{1}{2}\right)$ [Build the Huffman code]

Example 1

$$\underline{p} = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4} \right)$$

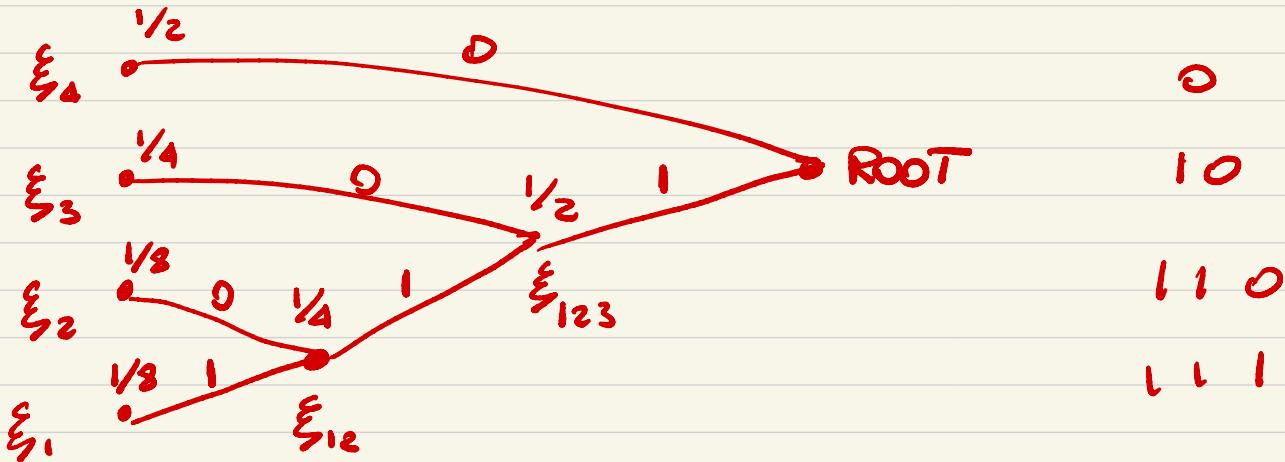


$$\xi_1 \rightarrow 0$$

$$\xi_2 \rightarrow 1 0$$

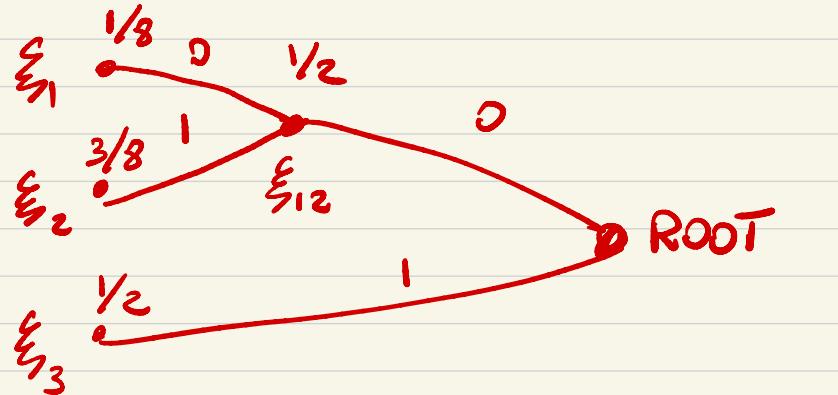
$$\xi_3 \rightarrow 1 1$$

Ex. 2 $\underline{P} = \left(\frac{1}{8}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2} \right)$



$$\bar{v} = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + 2 \cdot \frac{1}{8} \cdot 3 = 1.75 = H(x)$$

Ex. 3 $\underline{p} = \left(\frac{1}{8}, \frac{3}{8}, \frac{1}{2} \right)$

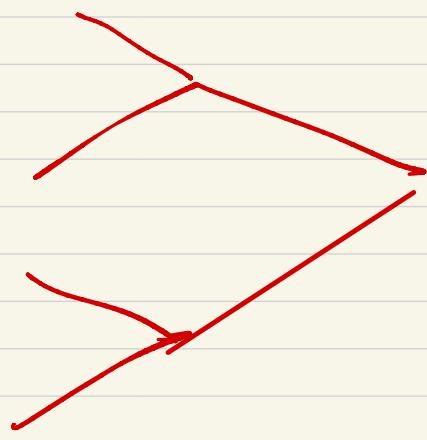
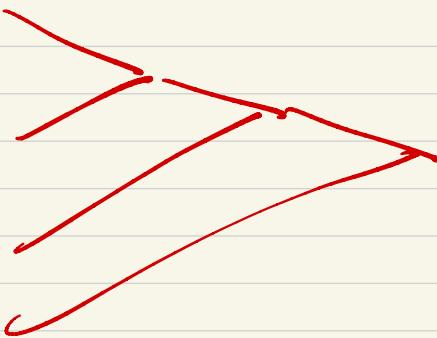


00
01
1

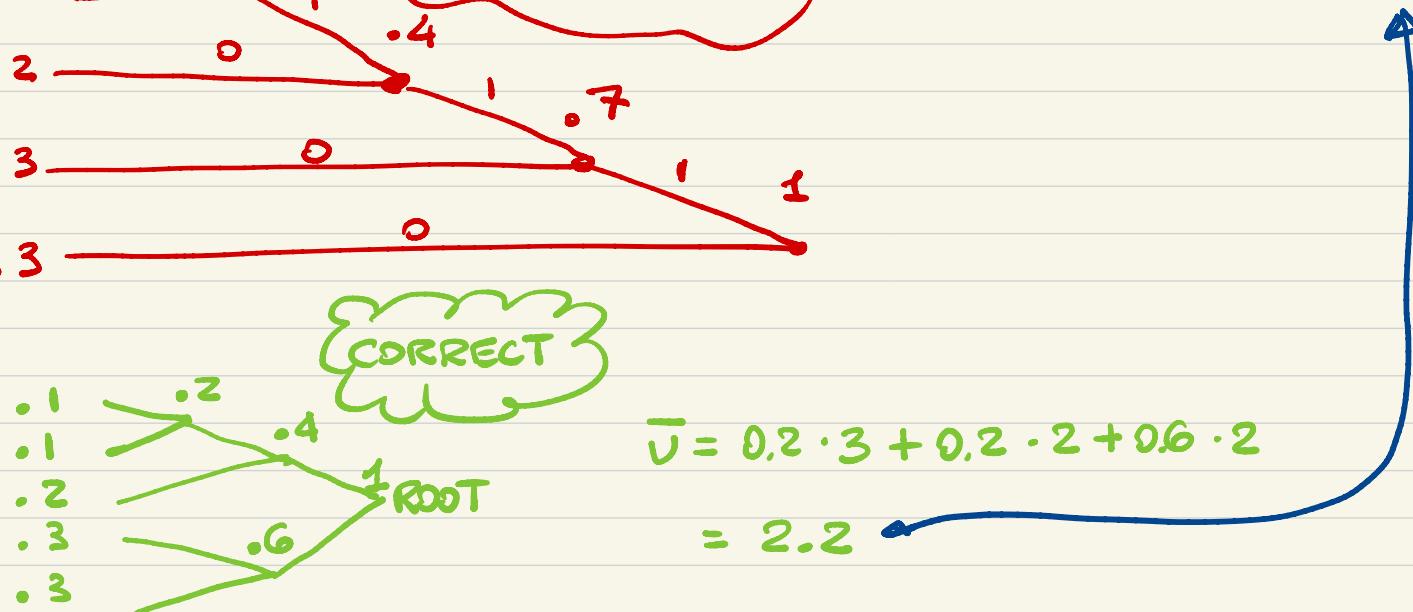
$$\bar{v} = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2 = 1.5$$

$$\eta = \frac{1.4056}{1.5}$$

$$H(x) = \frac{1}{8} \cdot 3 + \frac{3}{8} \log \frac{8}{3} + \frac{1}{2} \cdot 1 = 1.4056$$



Example of Huffman code which may be confusing



Storage of the compressed data

- The application of a source code to a sequence of source symbols of given length requires two types of data storages:
 - The code tree containing the symbols and their bit encodings
 - The sequence of encoded bits
- Consider for example a ternary source with symbols A,B,C with probabilities 0.25,0.25,0.5
- The tree (Huffman) code is
 - A → 00
 - B → 01
 - C → 1

$$\bar{v} = 1.5 \text{ bit/symbol}$$

Storage of the compressed data

- Consider the source sequence “ABCCCCBABA BACCCC” (16 symbols)
- Its encoding is “000111110100010001001111” (24 bits)
- The stored data are:
 - The tree: 00,A,01,B,1,C
 - The encoded bits: 000111110100010001001111
- Usually, the storage of the tree code is negligible with respect to the storage of the encoded bits
- In this example, the actual number of bits per symbol required is
$$\bar{v} = \frac{24}{16} = 1.5$$
- The source entropy is $\bar{H} = H(0.25, 0.25, 0.5) = 1.5 = \bar{v}$

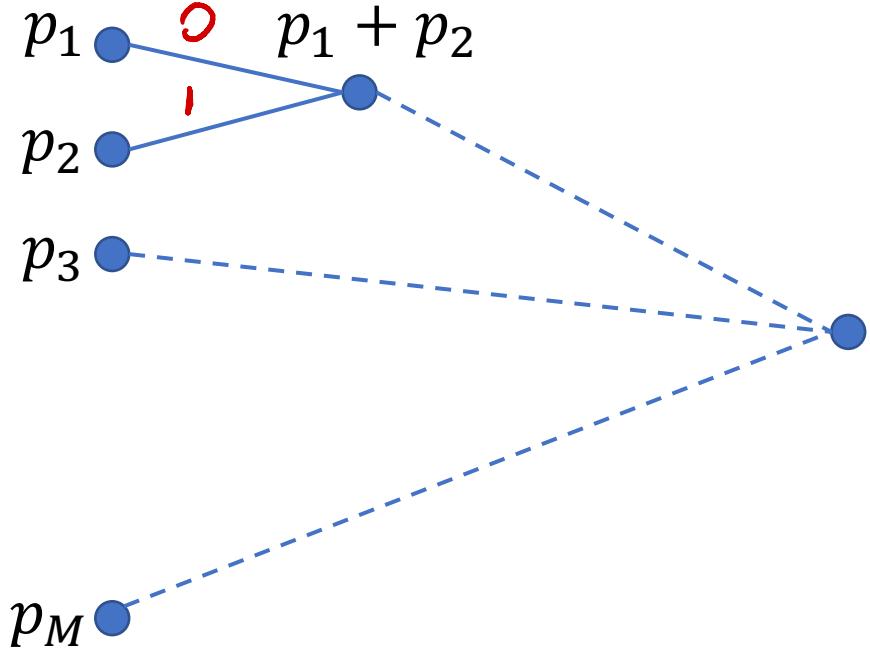
$$00, 01, 1*$$

$$\frac{24+6}{16} = 1.875$$

Calculating \bar{v} for Huffman codes

- The average number of bits per symbol \bar{v} can be calculated recursively
- Clearly, \bar{v} is a function of the probability vector $\mathbf{p} = (p_1, \dots, p_M)$:
$$\bar{v} = \phi(\mathbf{p})$$
- This function is invariant to permutations of the probabilities so that we can assume them sorted in increasing (at least nondecreasing) order: $p_1 \leq p_2 \leq \dots \leq p_M$
- Now we consider one step in the Huffman code construction algorithm

Calculating \bar{v} for Huffman codes



- The full code is obtained by adding two nodes to the dashed code
- The contribution to \bar{v} in the full code of the two nodes p_1 and p_2 is
$$p_1 v_1 + p_2 v_2$$
- If the path to the node $p_1 + p_2$ has length v_{12} , we have
$$v_1 = v_2 = v_{12} + 1$$
- Then,
$$\begin{aligned} p_1 v_1 + p_2 v_2 &= (p_1 + p_2)(v_{12} + 1) \\ &= (p_1 + p_2)v_{12} + p_1 + p_2 \end{aligned}$$
- The contribution to \bar{v} in the dashed code of the node $p_1 + p_2$ is $(p_1 + p_2)v_{12}$
- The other nodes p_3 to p_M give the same contribution to \bar{v} in the full code and in the dashed code: $p_3 v_3 + \dots + p_M v_M$
- Then,
$$\phi(p_1, p_2, p_3, \dots, p_M) = p_1 + p_2 + \phi(p_1 + p_2, p_3, \dots, p_M)$$

Examples

- Apply the previous recursive rule to calculate \bar{v} with the previous recursive rule and check the inequalities $H(X) \leq \bar{v} < H(X) + 1$

1. $p = (0.1, 0.3, 0.6)$

2. $p = (0.1, 0.1, 0.2, 0.2, 0.4)$

3. $p = (0.5, 0.25, 0.125, 0.125)$

1) $\bar{v} = \phi(0.1, 0.3, 0.6) = 0.4 + \phi(0.4, 0.6) = 1.4 \quad H(X) = 1.2955$

$\begin{array}{r} \cancel{0.1} \\ \cancel{0.3} \\ \cancel{0.6} \end{array} \rightarrow \bar{v} = 0.4 \cdot 2 + 0.6 \cdot 1 = 1.4$

2) $\bar{v} = 0.2 + \phi(0.2, 0.2, 0.2, 0.4) = 0.2 + 0.4 + \phi(0.4, 0.2, 0.4) = 0.2 + 0.4 + 0.6 + 1 = 2.2 \quad H(X) = 2.12$

3) $\bar{v} = \frac{1}{4} + \phi\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right) = \frac{1}{4} + \frac{1}{2} + 1 = 1.75 = H(X)$

Information Theory and Applications

Dictionary methods

Prof. Giorgio Taricco

Politecnico di Torino – DET

LIMITATIONS OF HUFFMAN-TYPE CODES (FIXED - TO - VARIABLE)

- estimation of the source prob. distribution
- storage of the code
- non universality

⇒ DICTIONARY METHODS

VAR - TO - FIXED

Variable-to-fixed encoding

- Complementary approach wrt fixed-to-variable
- Variable length source strings are encoded into fixed length bit vectors
- The set of possible variable length strings is called dictionary so that these are called **dictionary methods**
- The dictionary changes dynamically, as the source symbols are read off, according to an encoding algorithm
- Dictionary methods were introduced by Jacob Ziv, Abraham Lempel, and Terry Welch

Dictionary methods

- For an extended study of dictionary methods see:
 - W.A. Pearlman and A. Said, *Digital Signal Compression*, Cambridge University Press 2011
 - D. Salomon and G. Motta, *Handbook of Data Compression*, 5th ed., Springer, 2010
- The fixed-to-variable methods we considered before are **statistical methods**
- These methods use a statistical model of the data, and the quality of compression they achieve depends on how good that model is
- **Dictionary methods are not based on a statistical model**, they select strings of symbols and encode each string as a token using a dictionary
- The dictionary holds strings of symbols, and it may be **static** or **dynamic**
- Static is permanent, sometimes permitting the addition of strings but no deletions
- Dynamic holds strings previously found in the input stream, allowing for additions and deletions as new input is read

English dictionary method

- The simplest example of a **static** dictionary is a dictionary of the English language used to compress English text
- Imagine a dictionary with half a million words (without definitions)
- A word is read from the input stream and the dictionary is searched
- If a match is found, an index to the dictionary is written into the output stream
- Otherwise, the uncompressed word is written
- The compressed data contains indexes and raw words
- We distinguish them by an extra bit at the beginning

English dictionary method

- With 19 bits we can select each word from an English dictionary containing $2^{19} = 524,288$ words
- If the word to encode is in the dictionary we need 20 bits (including the extra bit)
- If it is not in the dictionary, we may encode it as follows:
 - Extra bit
 - 8 bits denoting the number of characters (1-256)
 - Character encoding according to some alphabet, eg, ASCII
- If the dictionary is comprehensive, most source words require 20 bit

English dictionary method

- A simple improvement can be obtained by using 2 extra bits
 - 00, if the word is in the first 1024 most likely words in the dictionary 10 bits
 - 01, if it is in the next 16384 most likely words 14 bits
 - 10, if it is in the remaining words 19 bits
 - 11, if it is not in the dictionary
- The number of bits required for the encoding is
 - Case 00: $2 + 10 = 12$ bits
 - Case 01: $2 + 14 = 16$ bits
 - Case 10: $2 + 19 = 21$ bits
 - Case 11: variable

Adaptive methods

- The efficiency of a static method depends on the goodness of the statistical model (word probability distribution)
- With text files, it depends on the language, of course
- With other types of files (*eg*, binary files), a statistical model is difficult to predict
- An **adaptive** method is usually much better
 - It starts with a small default dictionary
 - It adds words as they are found from the source
 - It deletes old unused words to keep the size small and the search time short

String compression

- Source codes based on strings of symbols are more efficient than those based on individual symbols
- Example: stationary source generating binary symbols with probability $P(X_n = 0) = 0.7$
- Apply Huffman coding to strings of m symbols
- The probability distribution of a string of two symbols is:

String	Probability
00	0.49
01	0.21
10	0.21
11	0.09

String compression

- By applying Huffman encoding to progressively longer strings we report the values of the average number of bits per symbol

m	$\frac{\bar{v}}{m}$
1	1
2	0.9050
3	0.9087
4	0.8918
5	0.8890
6	0.8882

- The sequence converges very slowly to $H_b(0.7) = 0.8813$

String compression

- The slow convergence is due to the fact that the encoded string length is fixed
- The number of possible strings grows exponentially with their length (it is 2^m)
- Strings to be encoded must be selected more carefully

Lempel-Ziv codes

- Jacob Ziv and Abraham Lempel were the developers of the first dictionary methods for data compression in the late 1970's
- Their first algorithm was LZ77
- The principle of LZ77 is to fill the dictionary with parts of the input stream as it is read off
- The method is based on a sliding window divided in two parts:
 - Search buffer
 - Look-ahead buffer

LZ77

- The search buffer contains already encoded symbols
- The look-ahead buffer contains symbols to be encoded
- In practical implementations, the search buffer is long a few thousands symbols and the look-ahead buffer a few tens
- The encoded output consists of tokens represented by three components:
 - Offset = distance of the first encoded symbol to the end of the search buffer
 - Length = length of the encoded string
 - Next symbol = first symbol after the encoded string
- The two buffers are implemented as sliding windows

LZ77 algorithm

- Initially,
 - the search buffer is empty
 - the symbols to be encoded fill the look-ahead buffer
- Repeat until the look-ahead buffer is empty:
 - Scan the search buffer to find the longest string matching the first symbols in the look-ahead buffer
 - If nothing is found
 - Output token <0,0,first symbol in look-ahead buffer>
 - else
 - Output token <offset, length, next symbol after encoded string in look-ahead buffer>
 - Move the buffer boundary to the right up to the next symbol to encode

LZ77 example: “cat cat catering”

- Assume buffer length is 8
- The \$ sign is used to terminate the string to encode

Search	Look-ahead	Token
"	" "cat cat "	<0,0,"c">
" c"	"at cat c"	<0,0,"a">
" ca"	"t cat ca"	<0,0,"t">
" cat"	" cat cat"	<0,0," ">
" cat "	" <u>cat cate</u> "	<4,4,"c">
"at cat c"	" <u>atering\$</u> "	<4,2,"e">
" <u>cat cate</u> "	"ring\$ "	<0,0,"r">
"at cater"	"ing\$ "	<0,0,"i">
"t cateri"	"ng\$ "	<0,0,"n">
" caterin"	"g\$ "	<0,0,"g">
"catering"	"\$ "	<0,0,"\$">

Another example: “sir sid eastman easily teases sea sick seals”

```
"      " --- "sir sid " --- <0,0,"s">      "an easil" --- "y teases" --- <0,0,"y">
"      s" --- "ir sid e" --- <0,0,"i">      "n easily" --- " teases " --- <7,1,"t">
"      si" --- "r sid ea" --- <0,0,"r">      "easily t" --- "eases se" --- <8,3,"e">
"      sir" --- " sid eas" --- <0,0," " >      "ly tease" --- "s sea si" --- <2,1," " >
"      sir " --- "sid east" --- <4,2,"d">      " teases " --- "sea sick" --- <4,2,"a">
" sir sid" --- " eastman" --- <4,1,"e">      "ases sea" --- " sick se" --- <4,2,"i">
"ir sid e" --- "astman e" --- <0,0,"a">      "s sea si" --- "ck seals" --- <0,0,"c">
"r sid ea" --- "stman ea" --- <6,1,"t">      " sea sic" --- "k seals$" --- <0,0,"k">
"sid east" --- "man easi" --- <0,0,"m">      "sea sick" --- " seals$" --- <5,2,"e">
"id eastm" --- "an easil" --- <4,1,"n">      " sick se" --- "als$"      " --- <0,0,"a">
" eastman" --- " easily " --- <8,4,"i">      "sick sea" --- "ls$"      " --- <0,0,"l">
"man easi" --- "ly tease" --- <0,0,"l">      "ick seal" --- "s$"      " --- <4,1,"$">
```

ZIP methods

- Several compression methods named *ZIP have been proposed in later years
- The first was PKZIP developed by P.W. Katz in 1987
- The core of PKZIP is the method called DEFLATE
- DEFLATE is a variation of LZ77 combined with Huffman codes

It has been shown by Lempel & Ziv that their encoding algorithm achieves asymptotically the Shannon limit, ie, the entropy rate

DEFLATE

- DEFLATE does not output tokens as LZ77
- DEFLATE still scans the search buffer for the longest string matching the initial part of the look-ahead buffer but it outputs
 - An unmatched character when the search fails
 - A pair (offset, length) when it was successful
- Unmatched characters and offsets have byte dimension
- Lengths have 15 bits
- DEFLATE output is stored by using a special Huffman code

Book references

- T.M. Cover and J.A. Thomas, *Elements of Information Theory*. Wiley, 2006
- R.M. Gray, *Entropy and Information Theory*, Springer, 2011
- G. Grimmett and D. Stirzaker, *Probability and Random Processes* 3rd ed. Oxford, 2001
- G. Grimmett and D. Stirzaker, *One Thousand Exercises in Probability*. Oxford, 2001
- M. Lefebvre, *Basic Probability Theory with Applications*. Springer Undergraduate Texts in Mathematics and Technology, 2009
- A. El Gamal and Y.-H. Kim, *Network Information Theory*. Cambridge University Press, 2011
- M.R.D. Rodriguez and Y. Eldar, *Information-Theoretic Methods in Data Science*, Cambridge University Press, 2021