Information Theory for Data Science

Information sources

Prof. Giorgio Taricco

Politecnico di Torino – DET

Program of this part of the course

- Time allocation:
    - 30 hours = 20 slots (of 1.5-hours)
- Information sources, source codes (lossless), and lossy source coding
    - THEORY (5 slots)
    - FIRST PART OF THE ASSIGNMENT (5 slots)
    - THEORY (5 slots)
    - SECOND PART OF THE ASSIGNMENT (5 slots)

Information sources

- Information sources generate a data stream representing any kind of information
- They are modeled by a sequence of random variables X_n, $n \in \mathbb{Z}$, the set of integer numbers
- The complete description requires all the possible probabilities

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

  - $\mathcal{S}$ is an arbitrary subset of $\mathbb{Z}$
  - The event $X_{\mathcal{S}} = x_{\mathcal{S}}$ corresponds to $\overset{\text{random variable}}{\boxed{X_n}} = \underset{\text{deterministic variable}}{\boxed{x_n}}\ \forall\, n \in \mathcal{S}$
  - X_n is random, x_n is deterministic
  - $x_n \in \mathcal{X} = \{\xi_1, \dots, \xi_M\}$, a source alphabet

    $M = |\mathcal{X}|$ = cardinality

Example 1

- Binary source, $\mathcal{X} = \{0,1\}$

- Sample subset: $\mathcal{S} = \{0,2,4,10\}$

- Probability:

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

Example 2

- Ternary information source, $\mathcal{X} = \{0,1,2\}$

- Sample subset: $\mathcal{S} = \{-2,0,2\}$

- Probability:

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

Stationarity of an information source

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

$$P(X_{\mathcal{S}} = x) = P(X_{\mathcal{S}+\Delta} = x) \qquad (*)$$

- $\Delta \in \mathbb{Z}$

- $\mathcal{S} + \Delta \overset{\text{def}}{=} \{n + \Delta, n \in \mathcal{S}\}$

- Stationarity holds if eq. $(*)$ holds for every $\mathcal{S}$ and every Δ

Example 2 (cont)

- Sample subset: $\mathcal{S} = \{-2,0,2\}$, shift $\Delta = 4$:

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

- $P(X_{-2} = 2, X_0 = 1, X_2 = 0) = P(X_2 = 2, X_4 = 1, X_6 = 0)$

mapping of the indexes

so this case is there
is stationarity
property

Stationarity of an information source

- If $|\mathcal{S}| = 1$, stationarity implies that

$$P(X_n = x) = P(X_0 = x) \overset{\text{def}}{=} \boxed{p_X(x)} \rightarrow \text{no dependence on time}$$

- So, the probability distribution is independent of n
- The function $p_X(x)$ is the probability distribution function of the stationary informatin sourche

- If $|\mathcal{S}| = 2$, stationarity implies that

$$P(X_n = x_1, X_{n+\Delta} = x_2) = P(X_0 = x_1, X_\Delta = x_2) \overset{\text{def}}{=} \boxed{p_\Delta(x_1, x_2)}$$

dependence on the different of time beetween the two variables

- The two-point probability distribution depends only on the time difference Δ

Markov sources

- Markov sources are an important class of information sources *with memory*
- The information source $X_1, X_2, X_3, \ldots$ is called a Markov source with memory L if the following property holds for every $n > L$:

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

  - 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

- 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 $\Sigma_n \overset{\text{def}}{=} X_{n-L:n-1}$ represent the state of the source at time n $(> L)$
- If the symbol alphabet is $\mathcal{X}$, the state alphabet is $\mathcal{X}^L$

- We characterize time-invariant Markov sources by the distribution of the initial state $\Sigma_{L+1} = X_{1:L}$ and by the conditional probabilities

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

*Time-invariance is different from stationarity. It allows the beginning of the source at a finite index

P_{ij} := probability of going from state i to state j

Markov sources

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

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

- Usually, the states are represented by integer numbers:

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

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

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

Evolution equation

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

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

- This is a consequence of the total probability law:

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

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

$$= \sum_{i \in \mathbb{S}} (\boldsymbol{p}_n)_i \, (\boldsymbol{P})_{i,j}$$

Existence of a stationary state

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

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

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

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

- When a Markov chain is **irreducible and aperiodic**, $\boldsymbol{P}^m$ converges, as $m \to \infty$, to a matrix whose rows are all equal
- In this case, the limit

$$\boldsymbol{p}_\infty \overset{\text{def}}{=} \lim_{m\to\infty} \{\boldsymbol{p}_m = \boldsymbol{p}_0 \boldsymbol{P}^m\}$$

exists, it is unique, and it is independent of $\boldsymbol{p}_0$

- This limit is called *stationary state probability distribution*

PROVE

$$\underline{p}_\infty = \underline{p}_0 P^\infty = \underline{p}_0 \cdot \begin{pmatrix} \underline{\tilde{p}} \\ \vdots \\ \underline{\tilde{p}} \end{pmatrix} = p_{01}\underline{\tilde{p}} + p_{02}\underline{\tilde{p}} + \dots + p_{0n}\underline{\tilde{p}} = (p_{01} + p_{02} + \dots + p_{0n})\underline{\tilde{p}} = 1 \cdot \underline{\tilde{p}}$$

Irreducibility

- A Markov chain is **irreducible** if all states are connected to each other, i.e., if $\boldsymbol{P}^m$ has all nonzero probabilities from $m \geq m_0$:

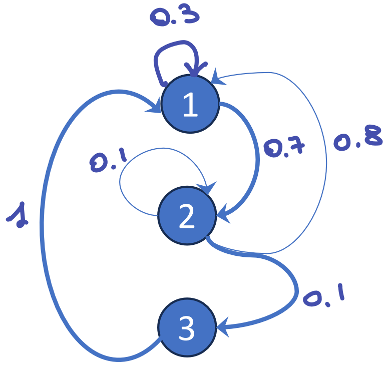
for this porpouse we consider a particoular matrix

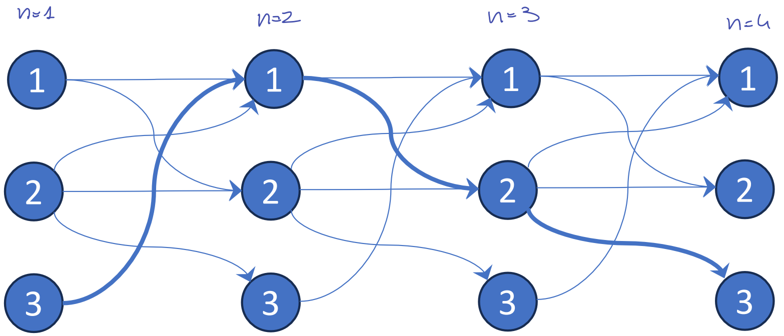
$$\boldsymbol{P} = \begin{pmatrix} 0.3 & 0.7 & 0 \\ 0.8 & 0.1 & 0.1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{P}^2 = \begin{pmatrix} 0.65 & 0.28 & 0.07 \\ 0.42 & 0.57 & 0.01 \\ 0.3 & 0.7 & 0 \end{pmatrix}, \qquad \boldsymbol{P}^3 = \begin{pmatrix} 0.489 & 0.483 & 0.028 \\ 0.592 & 0.351 & 0.057 \\ 0.65 & 0.28 & 0.07 \end{pmatrix}$$

Irreducibility

- The graph of this **irreducible** Markov chain is





A path from 3 to 3 in three steps

this follow the evolution of the matrix P

(1 → 1) probability to go from states 1 to state 1

$$\boldsymbol{P} = \begin{pmatrix} \overset{1\to 1}{0.3} & \overset{1\to 2}{0.7} & \overset{1\to 3}{0} \\ \overset{2\to 1}{0.8} & \overset{2\to 2}{0.1} & \overset{2\to 3}{0.1} \\ \overset{3\to 1}{1} & \overset{3\to 2}{0} & \overset{3\to 1}{0} \end{pmatrix}, \quad \boldsymbol{P}^2 = \begin{pmatrix} 0.65 & 0.28 & 0.07 \\ 0.42 & 0.57 & 0.01 \\ 0.3 & 0.7 & 0 \end{pmatrix}, \quad \boldsymbol{P}^3 = \begin{pmatrix} 0.489 & 0.483 & 0.028 \\ 0.592 & 0.351 & 0.057 \\ 0.65 & 0.28 & 0.07 \end{pmatrix}$$

Aperiodicity

- A Markov chain is **aperiodic** if all states are aperiodic
- A state i is aperiodic if its period d_i is equal to 1
- The period of a state i is d_i if the chain can return to the state i only at multiples of d_i steps
- Formally, defining the set of integers $\mathcal{S} = \{m: m \geq 1, (\boldsymbol{P}^m)_{i,i} > 0\}$,

$$d_i = GCD(\mathcal{S})$$

Aperiodicity

- Example: $\boldsymbol{P} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ has two states with period 2, so that the Markov chain is periodic
- In fact,

$$\boldsymbol{P}^1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad \boldsymbol{P}^2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \boldsymbol{P}^3 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \dots$$

- A sufficient condition for aperiodicity is that $(\boldsymbol{P})_{i,i} > 0$ for all i

Information Theory for Data Science

Information sources: metrics computation

Prof. Giorgio Taricco

Politecnico di Torino – DET

Calculation of $\boldsymbol{p}_\infty$

- One way is calculating the successive powers of $\boldsymbol{P}$ until it converges to a matrix with equal rows (equal to $\boldsymbol{p}_\infty$)
- Another way is solving the equation

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

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

all-1 row-vector

Calculation of $\boldsymbol{p}_\infty$

- To solve the homogeneous linear equation system, we proceed as follows:
  - Transpose the linear equations: $(\boldsymbol{I} - \boldsymbol{P}^T)\boldsymbol{p}_\infty^T = \boldsymbol{0}$
  - Discard one row of $(\boldsymbol{I} - \boldsymbol{P}^T)$
  - Replace it with an all-one row
  - Use $(0, \dots, 0,1)^T$ as the rhs $\boldsymbol{1}^T = 1$
- The Perron–Frobenius Theorem guarantees that the solution components are nonnegative, so that $\boldsymbol{p}_\infty$ is probability vector

Example 1

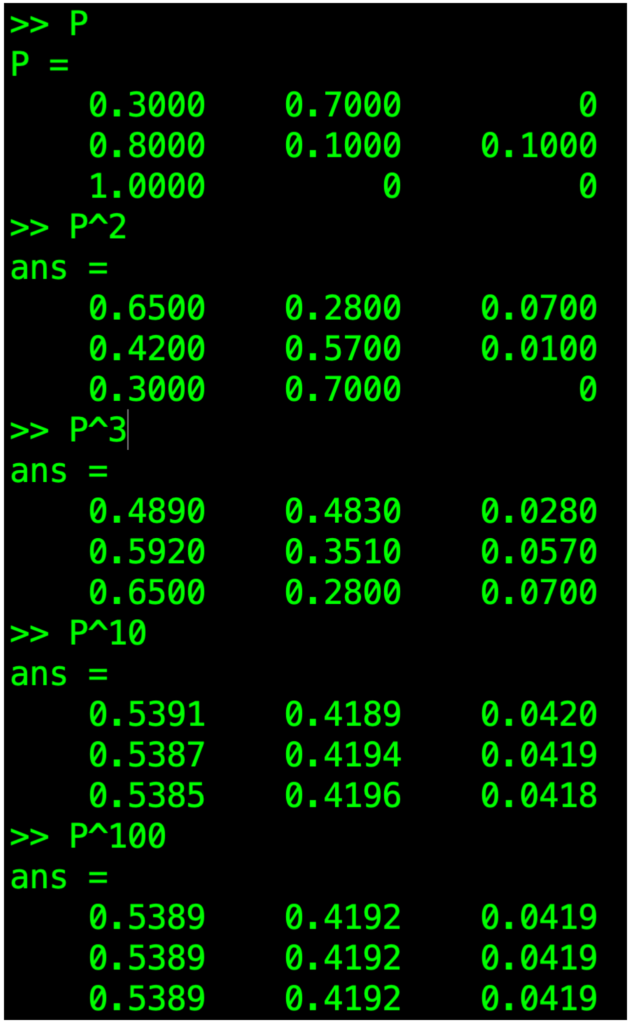
- Let $\boldsymbol{P} = \begin{pmatrix} 0.3 & 0.7 & 0 \\ 0.8 & 0.1 & 0.1 \\ 1 & 0 & 0 \end{pmatrix}$
- Then, $\boldsymbol{I} - \boldsymbol{P}^T = \begin{pmatrix} 0.7 & -0.8 & -1 \\ -0.7 & 0.9 & 0 \\ 0 & -0.1 & 1 \end{pmatrix}$
- Replace the first row with $(1,1,1)$ and solve $\begin{pmatrix} 1 & 1 & 1 \\ -0.7 & 0.9 & 0 \\ 0 & -0.1 & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$
- We get:
  - $-0.7p_1 + 0.9p_2 = 0 \rightarrow p_1 = \frac{9}{7}p_2$
  - $-0.1p_2 + p_3 = 0 \rightarrow p_3 = \frac{1}{10}p_2$
  - $p_1 + p_2 + p_3 = \left(\frac{9}{7} + 1 + \frac{1}{10}\right)p_2 = \frac{167}{70}p_2 = 1 \rightarrow p_2 = \frac{70}{167} \rightarrow p_1 = \frac{90}{167}, p_3 = \frac{7}{167}$

Example 1

- Sequence of matrices $\boldsymbol{P}^n, n = 1,2,3, \ldots$

- Stationary distribution:

$$\boldsymbol{p}_\infty = \left(\frac{70}{167},\frac{90}{167},\frac{7}{167}\right)$$
$$= (0.4192 \quad 0.5389 \quad 0.0419)$$



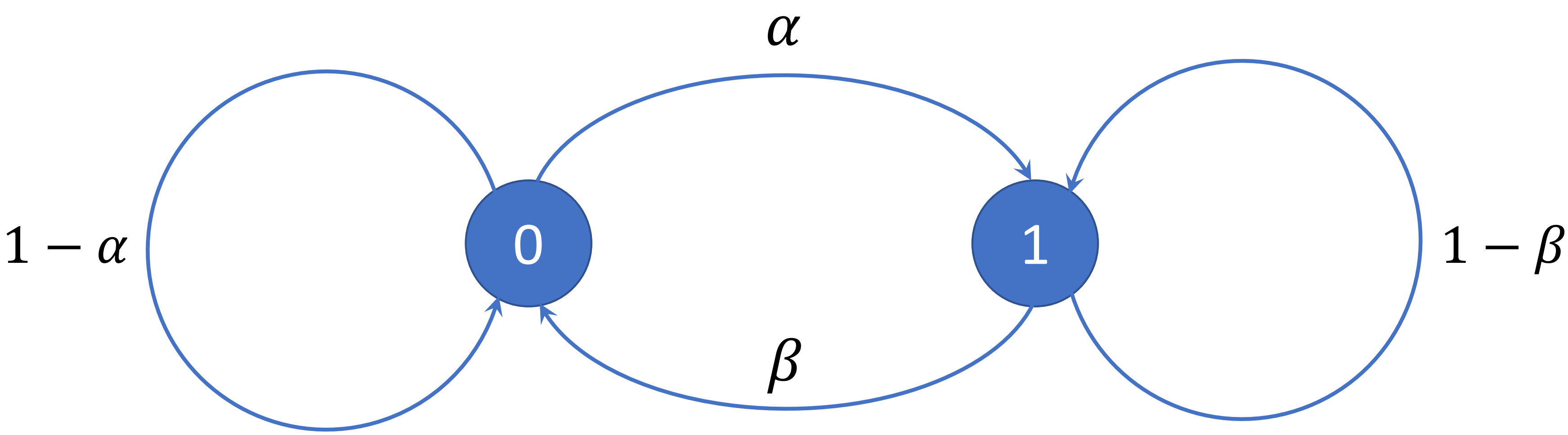
```
>> P
P =
    0.3000    0.7000         0
    0.8000    0.1000    0.1000
    1.0000         0         0
>> P^2
ans =
    0.6500    0.2800    0.0700
    0.4200    0.5700    0.0100
    0.3000    0.7000         0
>> P^3
ans =
    0.4890    0.4830    0.0280
    0.5920    0.3510    0.0570
    0.6500    0.2800    0.0700
>> P^10
ans =
    0.5391    0.4189    0.0420
    0.5387    0.4194    0.0419
    0.5385    0.4196    0.0418
>> P^100
ans =
    0.5389    0.4192    0.0419
    0.5389    0.4192    0.0419
    0.5389    0.4192    0.0419
```

Example 2: Two-state Markov chain

- Probability matrix

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

- Graph:

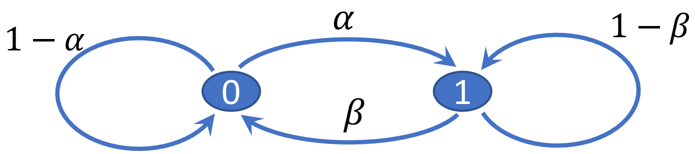


© prof. Giorgio Taricco

Example 2

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

$$\begin{gathered}(1-\alpha)p_0 + \beta p_1 = p_0 \;\Rightarrow -\alpha p_0 + \beta p_1 = 0 \\ \alpha\, p_0 + (1-\beta)p_1 = p_1 \Rightarrow \alpha p_0 - \beta p_1 = 0 \\ p_0 + p_1 = 1\end{gathered}$$



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

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

Entropy rate of a Markov source

- General definition:

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

- Stationary independent source:

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

Time index dropped for stationarity

- For a stationary (not independent) source,

$$\bar{H} = \lim_{n\to\infty} H(X_n|X_{1:n-1})$$

- 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

Markov source based on Example 2

- The Markov chain of the source is the two-state Markov chain
- Additionally,
  - In state 0, the source emits symbols with probability distribution vector $\boldsymbol{p}_0$
  - In state 1, the source emits symbols with probability distribution vector $\boldsymbol{p}_1$
- The entropy rate is

$$\overline{H} = p_\infty(0) H(\boldsymbol{p}_0) + p_\infty(1) H(\boldsymbol{p}_1)$$

$$= \frac{\beta}{\alpha+\beta} H(\boldsymbol{p}_0) + \frac{\alpha}{\alpha+\beta} H(\boldsymbol{p}_1)$$

Example 3

- Consider a binary source
  - with memory $L = 2$;
  - defined by the conditional symbol probabilities

$$P(X_n = 0|X_{n-1} + X_{n-2} = 0) = p_0 \quad \Rightarrow P(X_n = 1|X_{n-1} + X_{n-2} = 0) = 1 - p_0$$
$$P(X_n = 0|X_{n-1} + X_{n-2} = 1) = p_1$$
$$P(X_n = 0|X_{n-1} + X_{n-2} = 2) = p_2$$

- To find the stationary state distribution we must find the matrix $\boldsymbol{P}$
- We define
  - Starting (or departure) state: X_{n-2}, X_{n-1}
  - Ending (or arrival) state: X_{n-1}, X_n

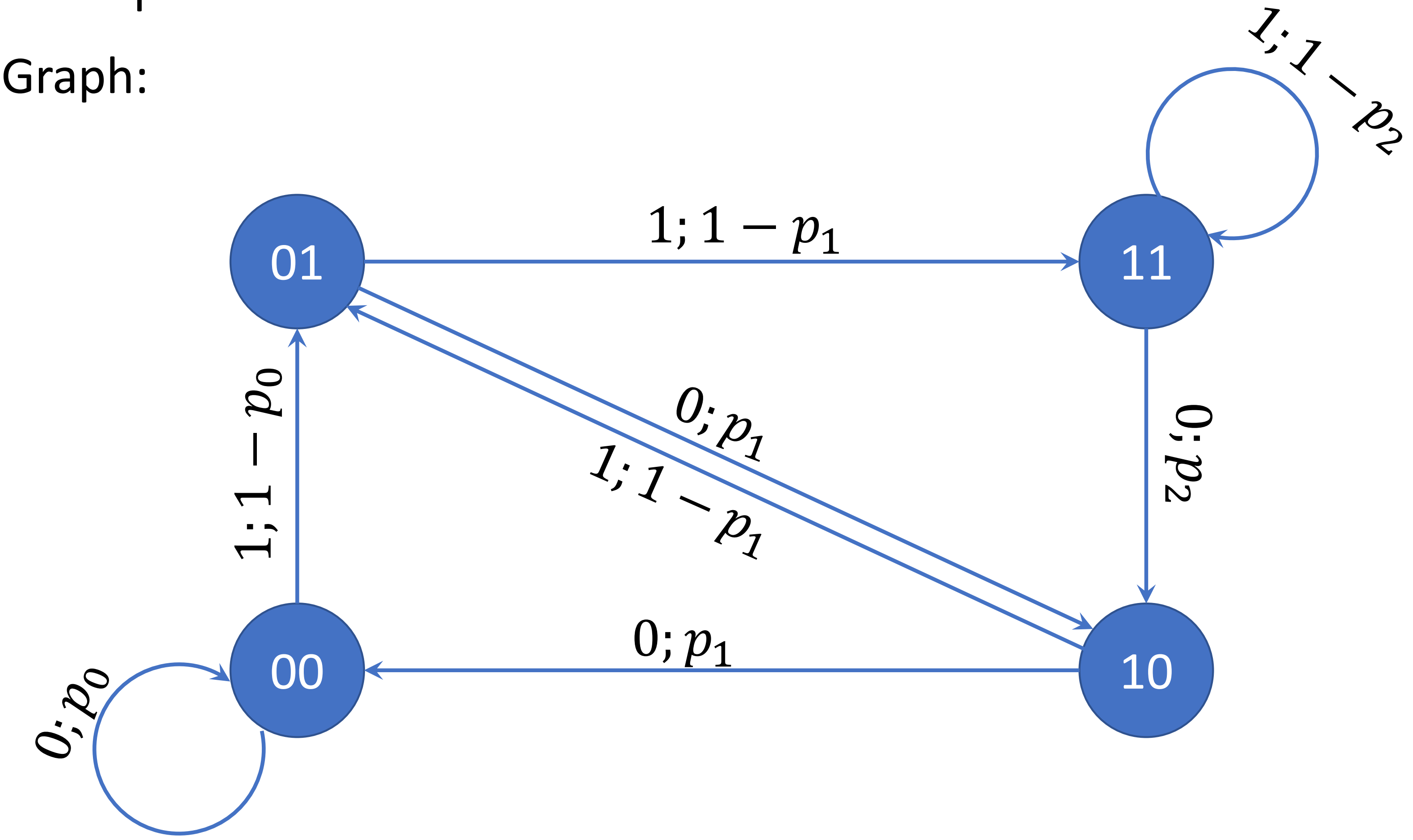
Example 3

- 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	00	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	$1 - p_2$

Example 3

- Graph:



© prof. Giorgio Taricco

Example 3

- Matrix:

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

- The states are listed as
  1. 00
  2. 01
  3. 10
  4. 11

Example 3

- Finally, we solve the stationary state probability equations:

$$\begin{aligned} p_{00} &= p_0 p_{00} + p_1 p_{10} \\ p_{01} &= (1 - p_0) p_{00} + (1 - p_1) p_{10} \\ p_{10} &= p_1 p_{01} + p_2 p_{11} \\ 1 &= p_{00} + p_{01} + p_{10} + p_{11} \end{aligned}$$

p_{ij} stands for $p_\infty(ij)$

- The solution is

$$p_{00} = \frac{\alpha p_1 p_2}{p_1 p_2 + 2(1 - p_0) p_2 + (1 - p_0)(1 - p_1)}$$

$$p_{01} = p_{10} = \frac{\alpha (1 - p_0) p_2}{p_1 p_2 + 2(1 - p_0) p_2 + (1 - p_0)(1 - p_1)}$$

$$p_{11} = \frac{\alpha (1 - p_0)(1 - p_1)}{p_1 p_2 + 2(1 - p_0) p_2 + (1 - p_0)(1 - p_1)}$$

Example 3

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

$$\overline{H}=\frac{p_1p_2H_b(p_0)+2(1-p_0)p_2H_b(p_1)+(1-p_0)(1-p_1)H_b(p_2)}{p_1p_2+2(1-p_0)p_2+(1-p_0)(1-p_1)}$$

$$H_b(p)\overset{\text{def}}{\equiv}-p\log_2 p-(1-p)\log_2(1-p)$$

Information Theory and Applications

Source codes

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 1
  - Fixed-to-variable 2
  - Variable-to-fixed 3

1

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

the requirement is at least that $2^N-1 \geq M^N-1$

- Since n bits are sufficient to represent all integers between 0 and $2^n - 1$, we must have $2^n \geq M^N$ or $\boxed{n \geq N \log_2 M}$

n is the minimum number of bits required to rapresent all the possible combination

- Since n must be an integer, its minimum value is given by

$$\nu = \lceil N \log_2 M \rceil$$

- The corresponding number of bits per encoded symbol is

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

when $N \to \infty$ we can say that $\bar{n} = \log_2 M$

Example

- Consider the English alphabet with some punctuation characters $\mathcal{X}$ with a total of $M = |\mathcal{X}| = 32$ characters

$\bar{n} \approx \log_2 M = \log_2 32 = 5$

- $\mathcal{X}$ is encoded as follows:

5 is the minimum number of bits necessary to rapresent all the possible combination

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

- Then, let us encode the sentence “GOOD NIGHT!”

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						

we obtain a string of bits

00110	01110	01110	00011	11100	01101	01000	00110	00111	10011	11111

- Decoding is easy because every symbol corresponds to 5 bits (no boundary problem)
- Simplicity is traded off with performance: the number of bits per symbol is fixed and corresponds to a benchmark which must be improved (reduced)

however decodin is not particoular difficoult in this case because each symbol is equal to five bits, this is the case where we have fix-to fix coding

2 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 if the symbols have different probabilities
  - More likely source symbols are assigned shorter codewords
  - Less likely are assigned longer codewords, but their impact is limited on the average codeword length
  - For example, let a source contain 128 symbols
    - The total probability of the most likely 8 symbols is 0.9
    - That of the less likely 120 symbols is 0.1
    - Assign 3 bits to the more likely and 7 to the less likely
    - Average codeword length: $0.9 \times 3 + 0.1 \times 7 = 3.4$ bits/symbol
    - Fixed-to-fixed source coding requires 7 bits/symbol

decoding ambiguity is a thing that must be avoiding becouse we need to recover the original set of symbols example :

$M=3$

$N=5$

$\xi_1\ \xi_2\ \xi_1\ \xi_1\ \xi_3$

$v_1\ v_2\ v_1\ v_1\ v_3$

$N_1 = 3$

$N_2 = 1$

$N_3 = 1$

$3v_1 + v_2 + v_3 =$

Fixed-to-variable source coding

- A stationary source generates symbols from the alphabet $\mathcal{X}$ with probability distribution

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

- Let the codeword lengths be ν_i (bit/symbol)
- The average number of bit per symbol for an encoded string is

$$\boxed{\langle\nu\rangle = \frac{1}{N}\sum_{i=1}^{M} N_i \nu_i}$$

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

Fixed-to-variable source coding

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

$$\frac{N_i}{N} \approx p_i, \qquad i = 1, \ldots, M$$

→ = it's the frequence of ξ_i in the string

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

$$\bar{\nu} = \sum_{i=1}^{N} p_i \nu_i$$

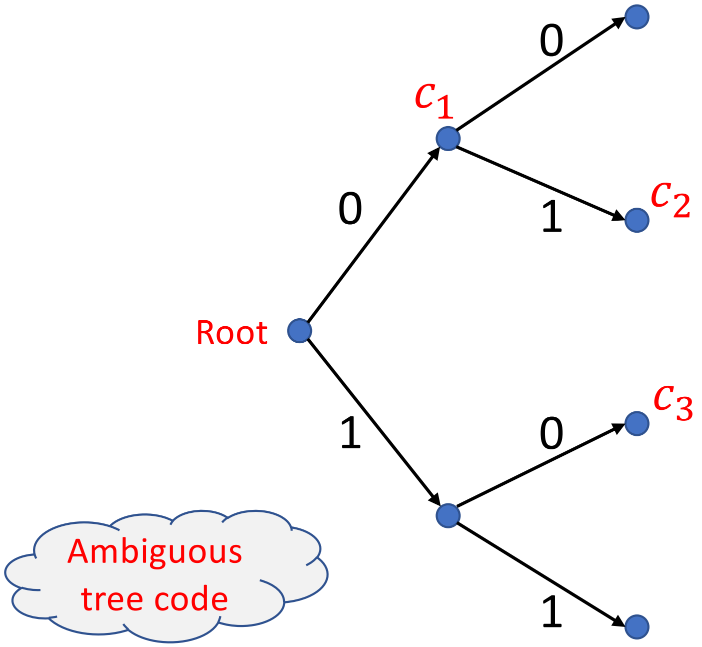
- $\bar{\nu}$ is the expected average number of bit per symbol of the source code and represents a quality measure for the source code
- The lower $\bar{\nu}$, the better the source code   we try to minimize this number

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
$$c(\xi_1) = 0, \qquad c(\xi_2) = 01, \qquad c(\xi_3) = 10$$
- 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



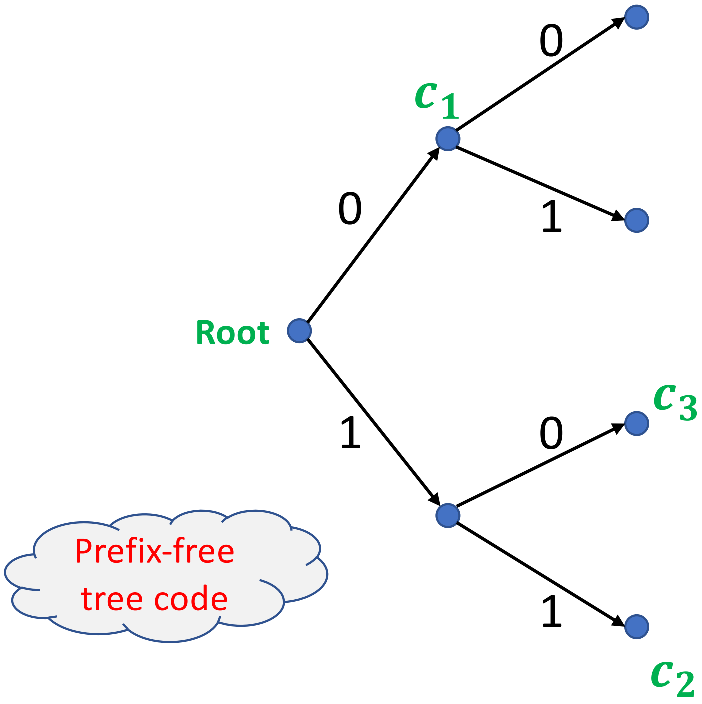
- The code words (bit sequences representing each symbol) are located at tree nodes
- The bits of each code word are read off traveling the tree from the root to the node
- For example, c_2 is read passing through a path labelled 0 and another labelled 1
- If one code word is at a node inside the path 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

the prefix-free condition is the condition for the absense of the decoding ambiguity and holds the absense of symbols coding by smaller string of bits (like c_2 in this case)

↓ solution to eliminate the problem of decoding ambiguity

Decoding fixed-to-variable source codes

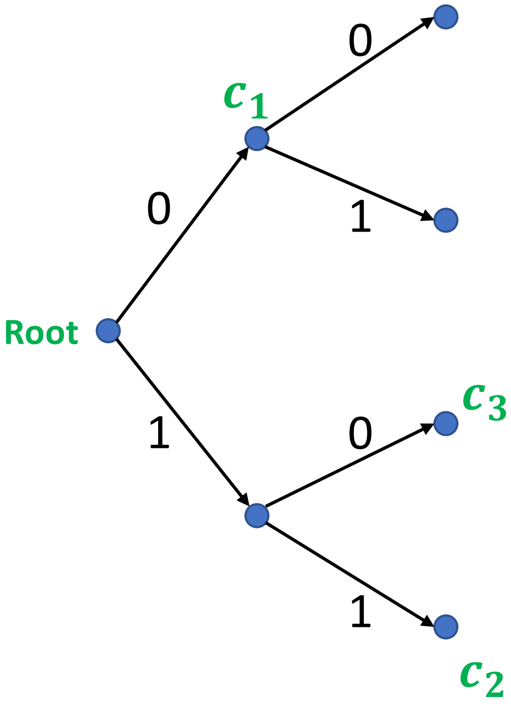
- The following code satisfies the no-prefix condition:



- 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

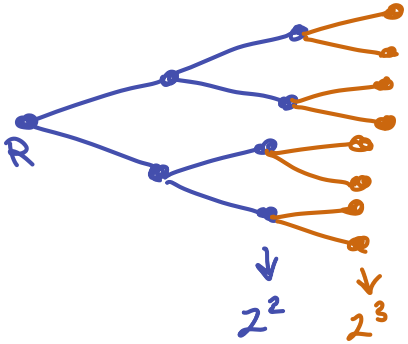
Decoding fixed-to-variable source codes

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



- Exercise:
  Decode the bit sequence 0111001110

- The result is $\xi_1, \xi_2, \xi_3, \xi_1, \xi_2, \xi_3$



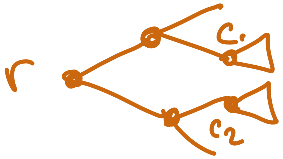
if I consider a root R and a tree we deeps 2 the number of node at the 2^ level is equal to 2^2
if I add a level I'll have 2^3 nodes at the third level

Kraft inequality

- This is a simple inequality characterizing the code word length distribution for a 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 $\boxed{2^{\nu_{\max}-\nu_i}}$ nodes at depth $\nu_{\max}$ ⇒ V_{max} represent the maximum level possible
- The sub-tree stemming from it up to depth $\nu_{\max}$ does not contain other codewords to satisfy the prefix-free condition
- We have the inequality

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

in else we know that if we take a root and two different nodes C_1 and C_2 that respect the fact that C_1 can't be in the path beetween root and C_2 and in else C_2 can't be in the path beetween root and C1, in this way the subtree after C1 is form by different nodes from the subtree generates by C2



Kraft inequality

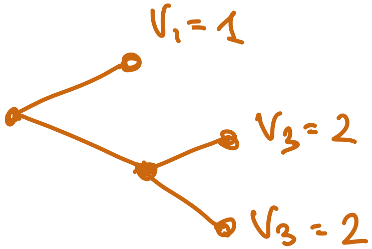
but if the PF is violated the set of nodes generate isn't distint

- Inequality

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

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

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



$$2^{-1}+2^{-2}+2^{-2}$$
$$=\frac{1}{2}+\frac{1}{4}+\frac{1}{4}=1\leq 1$$

Just an application

Information Theory and Applications

Shannon Theorem for source coding

Prof. Giorgio Taricco

Politecnico di Torino – DET

Lower bound on $\bar{\nu}$

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

$$\bar{\nu} = \sum_{i=1}^{M} p_i \nu_i$$

- Our constraint is the Kraft inequality:

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

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

$$\mathcal{L}(\nu_1, \dots, \nu_M) = \sum_{i=1}^{M} p_i \nu_i + \lambda \left(\sum_{i=1}^{M} 2^{-\nu_i} - 1 \right)$$

Lower bound to $\bar{\nu}$

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

$$\frac{\partial \mathcal{L}}{\partial \nu_i} = p_i - \lambda \cdot \ln 2 \cdot 2^{-\nu_i} = 0$$

$$\lambda \left(\sum_{i=1}^{M} 2^{-\nu_i} - 1 \right) = 0 \qquad (*)$$

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

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

$\left(p_i = 2^{-\nu_i}\right)$

$\Longleftrightarrow \nu_i = -\log_2 p_i$

© prof. Giorgio Taricco

Lower bound to $\bar{\nu}$ $\longrightarrow$ Entropy

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

$$\bar{\nu} > H(X)$$

- Summarizing, Kraft inequality implies that

$$\Rightarrow \bar{\nu} \geq H(X)$$

with equality only if all the probabilities are negative powers of 2

Upper bound to $\bar{\nu}$

- We can replace the lengths $\nu_i = -\log_2 p_i$ by their integer ceilings*:

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

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

so apply in our case:

- Then,

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

- Thus,

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

*The integer ceiling of x is the minimum integer number $\geq x$, e.g., $\lceil 0.5 \rceil = 1, \lceil 1 \rceil = 1, \lceil -0.2 \rceil = 0, \lceil 0.2 \rceil = 1,$ *etc.*

Shannon Theorem

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

So in final we can say ⇒

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

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

Cross entropy

= for calculate the correlation beetween two symbols

is sometimes use in statistical learning

- 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 $\nu_i = -\log_2 q_i$
- The average length based on the true probability distribution is the cross entropy:

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

the expectation is that $H(p,q) \geq H(p,p)$

$= H(p)$

l'entropia incrociata (o cross-entropia) fra due distribuzioni di probabilità
p e q, relative allo stesso insieme di eventi, misura il numero medio di bit necessari per identificare un evento estratto dall'insieme
nel caso sia utilizzato uno schema ottimizzato per una distribuzione di probabilità
q piuttosto che per la distribuzione vera
p.

Cross entropy inequalities

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

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

- This can be proved by applying the inequality

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

- In fact,

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

because the sum

$$\sum_i p_i\left(\frac{q_i}{p_i} - 1\right) = \sum_i (q_i - p_i)$$

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

misura non simmetrica della differenza tra due distribuzioni di probabilità P e Q. Specificamente, la divergenza di Kullback–Leibler di Q da P, indicata con DKL(P||Q), è la misura dell'informazione persa quando Q è usata per approssimare P

Kullback-Leibler divergence

- The difference between the mismatched and optimum $\bar{\nu}$ is the Kullback-Leibler divergence:
$$D(\boldsymbol{p}||\boldsymbol{q}) \overset{\text{def}}{=} H(\boldsymbol{p},\boldsymbol{q}) - H(\boldsymbol{p},\boldsymbol{p}) \geq 0$$
- We also can write
$$D(\boldsymbol{p}||\boldsymbol{q}) \overset{\text{def}}{=} \sum_i p_i \log_2 \frac{p_i}{q_i}$$
- There is no ordering, instead, between $H(\boldsymbol{p},\boldsymbol{q})$ and $H(\boldsymbol{p},\boldsymbol{q})$, i.e., the difference $H(\boldsymbol{p},\boldsymbol{q}) - H(\boldsymbol{q},\boldsymbol{q}) = \sum_i (q_i - p_i) \log_2 q_i$ can be negative, zero, or positive

Codes for stationary Markov 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

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

Markov information source, or simply, a Markov source, is an information source whose underlying dynamics are given by a stationary finite Markov chain.

Codes for stationary Markov sources

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

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

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

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

- In a similar way:

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

Information Theory and Applications

Huffman codes & dictionary methods

Prof. Giorgio Taricco

Politecnico di Torino – DET

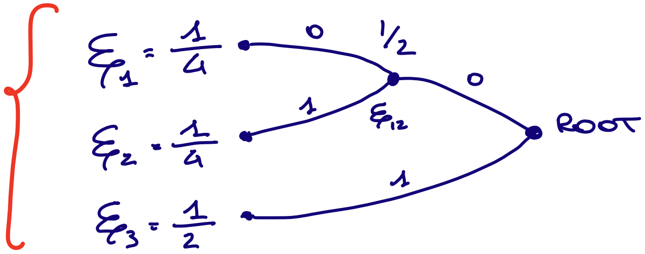
Huffman codes

- 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

1^ EXAMPLE :



$c(\xi_1) = 00 \quad r_1 = 2$

$c(\xi_2) = 01 \quad r_2 = 2$

$c(\xi_3) = 1 \quad r_3 = 1$

$\bar{r} = \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 + \frac{1}{2} \cdot 1 = 1.5$ bits/symbol

Examples

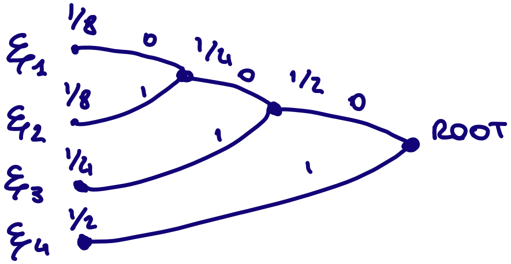
- Example 1: $\boldsymbol{p} = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{2}\right)$ [Build the Huffman code]

$\bar{H} = H(1/4 \; 1/4 \; 1/2) = 2\,1/4 \log_2 4 + \frac{1}{2} \log_2 2 = 1.5$

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

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

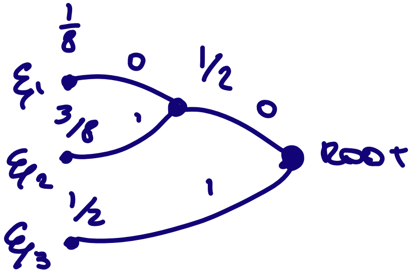
2^ EXAMPLE:



$\bar{H} = H\left(\frac{1}{8}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}\right) = 1.75$

$\bar{r} = 3 \cdot \frac{1}{8} + 3 \cdot \frac{1}{8} + 2 \cdot \frac{1}{4} + \frac{1}{2} = \frac{3}{4} + 1 = \frac{7}{4} = 1.75$

3^ EXAMPLE



$\bar{r} = 2 \cdot \frac{1}{8} + 2 \cdot \frac{3}{8} + \frac{1}{2} = \frac{1}{4} + \frac{3}{4} + \frac{1}{2} = 1.5$

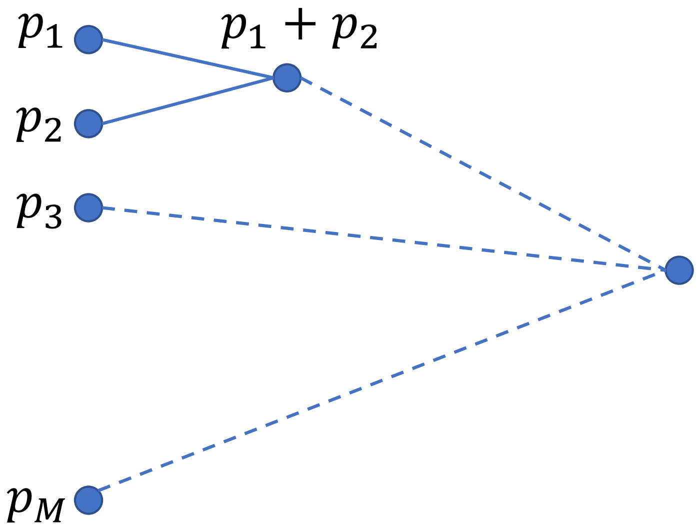
$\bar{H} = 1.4056 \quad \bar{r} \simeq \bar{H}$

wich means that we can obtain a better result but 1 symbol for bit is too short

Calculating $\bar{\nu}$ for Huffman codes

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

Calculating $\bar{\nu}$ for Huffman codes



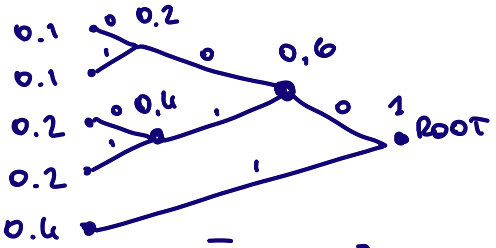
- The full code is obtained by adding two nodes to the dashed code
- The contribution to $\bar{\nu}$ in the full code of the two nodes p_1 and p_2 is
$$p_1 \nu_1 + p_2 \nu_2$$
- If the path to the node $p_1 + p_2$ has length ν_{12}, we have $\nu_1 = \nu_2 = \nu_{12} + 1$
- Then,
$$p_1 \nu_1 + p_2 \nu_2 = (p_1 + p_2)(\nu_{12} + 1)$$
- The contribution to $\bar{\nu}$ in the dashed code of the node $p_1 + p_2$ is $(p_1 + p_2)\nu_{12}$
- The other nodes p_3 to p_M give the same contribution to $\bar{\nu}$ in the full code and in the dashed code: $p_3 \nu_3 + \cdots + p_M \nu_M$
- Then,
$$\phi(p_1, p_2, p_3, \ldots, p_M) = p_1 + p_2 + \phi(p_1 + p_2, p_3, \ldots, p_M)$$

Examples

- Apply the previous recursive rule to calculate $\bar{\nu}$ with the previous recursive rule and check the inequalities $H(X) \leq \bar{\nu} < H(X) + 1$
  1. $\boldsymbol{p} = (0.1, 0.3, 0.6)$
  2. $\boldsymbol{p} = (0.1, 0.1, 0.2, 0.2, 0.4)$
  3. $\boldsymbol{p} = (0.5, 0.25, 0.125, 0.125)$

2^ EXAMPLE

$$\phi(0.1, 0.1, 0.2, 0.2, 0.4) = 0.2 + \phi(0.2, 0.2, 0.2, 0.4)$$
$$= 0.6 + \phi(0.4, 0.2, 0.4)$$
$$= 1.2 + \phi(0,6, 0,4) = 1,6 + 0,6$$
$$= 2,2$$



$\bar{r} = 0.6 \cdot 3 + 0,4 = 2,2$

Note that :

encoding one symbol at a time: $H(x) \le \bar{r} < H(X) + 1$

encoding n-symbols at a time: $H(x^n) \le \bar{r}_n < H(x^n) + 1$

in else if the source is indipendent :

$$H(x^n) = H(x_1, x_2, \dots x_n) = H(x_1) + H(x_2) + \dots H(x_n) = n H(x)$$

So: $n H(x) \le \bar{r}_n < n H(x) + 1 \Rightarrow H(x) \le \bar{r} < H(x) + \frac{1}{n}$

The Shamon Theorem lower bound can be approached arbitrarily close by letting $n \to \infty$

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

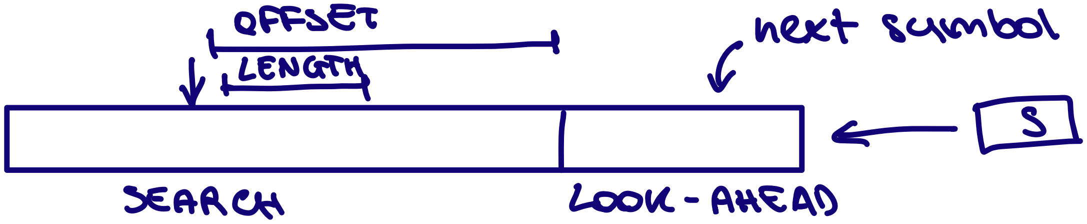
- A simple improvement can be obtained by using 2 extra bits
  - 00, if the word is in the first 1024 most likely words in the dictionary
  - 01, if it is in the next 16384 most likely words
  - 10, if it is in the remaining words
  - 11, if it is not in the dictionary
- The number of bits required for 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

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 because they publish it in 1977
- 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 " "cat cate" <4,4,"c">
"at cat c" "atering$" <4,2,"e">
"cat cate" "ring$   " <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

The main contribution by Lempel and Ziv was to show that their algor. provide an average num. of bits per symbol converging to the entropy of the source asymptotically

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
- 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