Institut
Mines-Telecom

Lossless coding principles

Marco Cagnazzo,
cagnazzo@telecom-paristech.fr

MN910 – Advanced Compression

Outline

Outline

Introduction

Lossless compression is based on data statistical properties. The basic idea is very simple: instead of using fixed-length coding, we use variable-length coding (VLC) with

- **Short** codewords for **probable** symbols
- **Long** codewords for **not probable** symbols

Definitions:

Alphabet : $\mathcal{X} = \{x_1, x_2, \ldots, x_M\}$ set of symbols to encode

- $\{0, 1, \ldots, 255\}$ for luminance values
- Latin alphabet for a text in French, English, ...

Code : application between $\mathcal{X}$ and $\{0, 1\}^*$, the set of finite-length bit strings

- Fixed-length coding
- Variable-length coding

Choosing a code

Code: $\mathcal{C} : x_i \in \mathcal{X} \rightarrow c_i \in \{0,1\}^*$
Fixed-length coding (FLC)

- All codewords have the same length
- E.g., if $M = 256$ symbols, we need $\lceil \log M \rceil = 8$ bits to encode each symbol
- For a text, $M = 26$, we need $\lceil \log M \rceil = 5$ bpS (bit per symbol)

Variable-length coding (VLC)

- ℓ_i: length of codeword c_i
- We achieve *lossless coding* if:
  - Decodability condition: prefix condition
  - Non-equiprobable symbols

Example: French text compression

Method	FLC
Number of symbols M	26
Coding rate ($\mathcal{L}$)	5 bpS
Compression ratio	1

- Each letter is represented on 5 bits
- No compression (this is our reference)

Examples of codes

Symbol	Code 1	Code 2	Code 3	Code 4
A	0	0	0	0
B	11	10	01	01
C	111	110	011	11

- Code 1 is not uniquely decodable : 111111 can be decoded as BBB or CC
- Code 2 is instaneously decodable (prefix condition)
- Code 3 is decodable with a maximum delay (1 bit)
- Code 4 is decodable but with undefined delay: 0111111...11 can be decoded as ACCC...CC or BCC...C depending on the number of 1's is even or odd

VLC: decodability condition

- We do not have “separators” between codewords
- Instantaneous and decodable codes
- Kraft’s inequality: there exists an instantaneous code with lengths $\{\ell_1, \ldots, \ell_M\}$ if and only if:

$$\sum_i 2^{-\ell_i} \leq 1$$

- Decodable codes do not improve performance with respect to instantaneous codes

Kraft's inequality

Prefix condition $\Rightarrow \sum_i 2^{-\ell_i} \leq 1$

- First we build a complete binary tree of depth $\ell_{\max}$
- Each codeword is associated with a node
- Fore each leaf, we go back to the root: how many codeword do we find along this path?

Kraft's inequality

Prefix condition $\Rightarrow \sum_i 2^{-\ell_i} \leq 1$

- First we build a complete binary tree of depth $\ell_{\max}$
- Each codeword is associated with a node
- Fore each leaf, we go back to the root: how many codeword do we find along this path?
  - Zero or one (prefix condition)
- Number of leaves = $A \geq B$ = Number of leaves with exactly one codeword on the path to the root
- $A = 2^{\ell_{\max}}$
- $B = \sum_{i=1}^{M}$ Leaves descending from the i-th codeword $= \sum_{i=1}^{M} 2^{\ell_{\max} - \ell_i}$

Kraft's inequality

$\sum_i 2^{-\ell_i} \leq 1 \Rightarrow$ prefix condition

- Build a complete binary tree of depth $\ell_{\max}$
- First codeword c_1: start from a leaf and go toward the root up to level ℓ_1
- This is the node associated to c_1
- Cut the sub-tree rooted in c_1: this assures the prefix condition for this codeword
- For each new codeword, we cut the subtree rooted in the associated node
- The only question is: are there enough leaves to generate the entire codebook, or we have cut too many leaves?

Kraft's inequality

$\sum_i 2^{-\ell_i} \leq 1 \Rightarrow$ Prefix condition

- Recurrence proof
- We know that we can find c_1
- Recurrence: if we have $\{c_i\}_{i=1}^{n-1}$, we can find c_n, with $n \leq M$
- How many leaves have been deleted up to step $n-1$?
- For codeword c_i we cut $2^{\ell_{\max}-\ell_i}$ leaves; in total:

$$\sum_{i=1}^{n-1} 2^{\ell_{\max}-\ell_i} = 2^{\ell_{\max}} \sum_{i=1}^{n-1} 2^{-\ell_i} \tag{1}$$

$$< 2^{\ell_{\max}} \sum_{i=1}^{M} 2^{-\ell_i} \leq 2^{\ell_{\max}} \tag{2}$$

- Therefore there is at least one remaining leaf: we go to the root and take the node at level ℓ_n: it is the node associated to c_n

Kraft's inequality

- In summary, an instantaneous code is defined by the set of lengths: $\{\ell_1, \ldots, \ell_M\}$ provided that it satisfies the Kraft's inequality
- From the set of lengths we can build the tree
- From the tree we build the code

Information and entropy

- Symbol x_i has probability p_i
- Average length of the code: $\mathcal{L} = \sum p_i \ell_i$
- Information associated to x_i is $I(x_i) = -\log p_i$
  - $I(x_i) \geq 0$
  - If $p_i = 1$, $I = 0$
  - If two symbols are independent, $I(x_i, x_j) = I(x_i) + I(x_j)$
- Source entropy: average symbol information

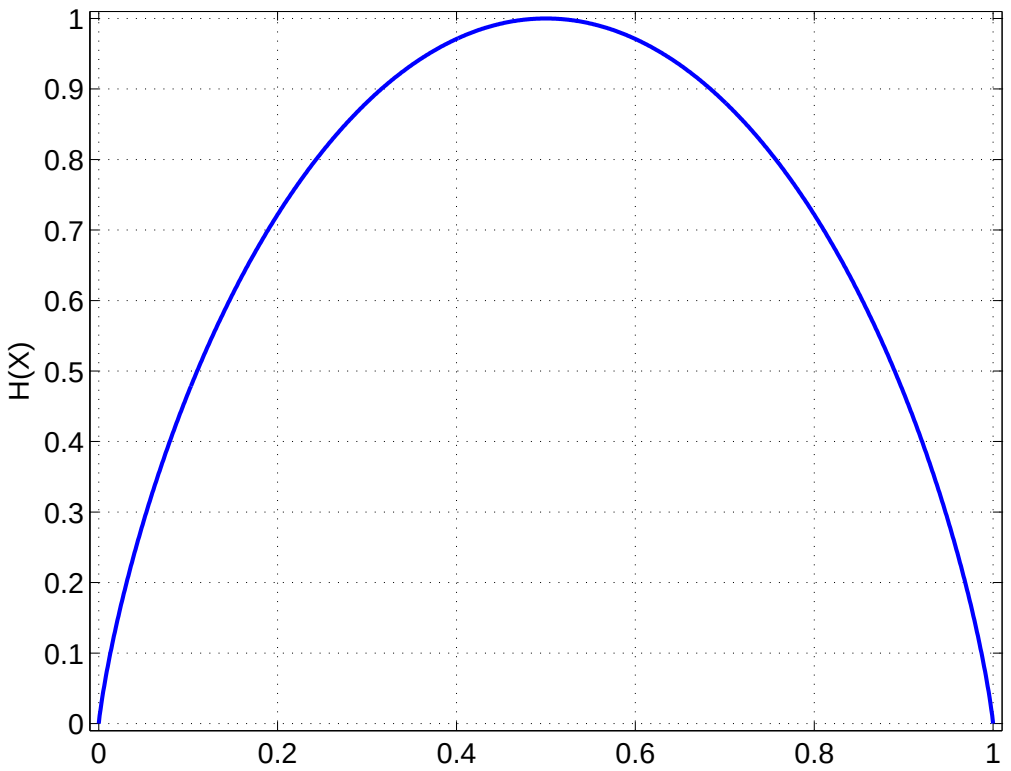
$$H(X) = -\sum_i p_i \log p_i$$

Binary variable entropy

Example

$$p = P\{X = 0\} \qquad q = P\{X = 1\} = 1 - p$$

$$H(X) = -p\log(p) - (1-p)\log(1-p)$$



Distribution with maximum entropy

Constrained maximization problem:

$$\mathbf{p}^* = \arg\max_{\mathbf{p}} \sum_{i=1}^{M} p_i \log \frac{1}{p_i} \qquad \sum_{i=1}^{M} p_i = 1$$

The distribution maximizing the entropy of a M-ary discrete r.v. is

Distribution with maximum entropy

Constrained maximization problem:

$$\mathbf{p}^* = \arg\max_{\mathbf{p}} \sum_{i=1}^{M} p_i \log \frac{1}{p_i} \qquad \sum_{i=1}^{M} p_i = 1$$

The distribution maximizing the entropy of a M-ary discrete r.v. is $\mathbf{p}^* = [p_1^* \, p_2^* \, \ldots \, p_M^*]$ such that $p_i^* = \frac{1}{M} \qquad \forall i \in \{1, 2, \ldots, M\}$

$$J(\mathbf{p}) = -\sum_{i=1}^{M} p_i \log p_i + \lambda \left(\sum_{i=1}^{M} p_i - 1 \right) \qquad \frac{\partial J}{\partial p_i}(\mathbf{p}^*) = 0$$

$$\frac{\partial J}{\partial p_i} = -\left(\frac{1}{\ln 2} + \log p_i \right) + \lambda \qquad p_i^* = \lambda - \log e = \text{constant}$$

Joint entropy

- For a couple of r.v. X and Y
- Joint probability distribution $p_{i,j} = \mathrm{P}\{X = x_i, Y = y_j\}$
- Joint entropy: average information of couples

$$H(X,Y) = -\sum_{i,j} p_{i,j} \log p_{i,j}$$

- Formally, there is no difference between the joint probability of a couple X, Y and the entropy of a single r.v. having the same set of probabilities
- The entropy does not depends on the values, only on the probabilities

Conditional entropy

- We consider a couple of r.v. X and Y
- Let $p_j = \mathrm{P}\{Y = y_j\}$
- Conditional entropy:

$$H(X|Y) = \sum_j p_j H(X|Y = y_j)$$

- It is easy to show that:

$$\begin{aligned} H(X,Y) =& H(Y) + H(X|Y) \\ & H(X) + H(Y|X) \end{aligned}$$

Properties

$$
\begin{aligned}
H(X) &\geq 0 \\
H(X,Y) &= H(Y) + H(X|Y) \\
&= H(X) + H(Y|X) \\
H(X,Y) &\leq H(X) + H(Y) && \text{with equality} \Leftrightarrow \text{independence} \\
H(X|Y) &\leq H(X) && \text{with equality} \Leftrightarrow \text{independence} \\
H(X) &\leq \log_2 M && \text{with equality} \Leftrightarrow X \sim \mathcal{U}
\end{aligned}
$$

Optimal code

- We drop condition $\ell_i \in \mathbb{N}$
- Constrained minimization:

$$\underline{\ell}^* = \arg\min_{\underline{\ell}} \sum_i p_i \ell_i \qquad \text{subject to} \quad \sum_i 2^{-\ell_i} = 1$$

Optimal code

- We drop condition $\ell_i \in \mathbb{N}$
- Constrained minimization:

$$\underline{\ell}^* = \arg\min_{\underline{\ell}} \sum_i p_i \ell_i \qquad \text{subject to} \quad \sum_i 2^{-\ell_i} = 1$$

$$J(\underline{\ell}) = \sum_i p_i \ell_i + \lambda \left(\sum_i 2^{-\ell_i} - 1 \right) \qquad \frac{\partial J}{\partial \ell_i} = p_i - (\lambda \ln 2) 2^{-\ell_i^*} = 0$$

$$\sum_i p_i = (\lambda \ln 2) \sum_i 2^{-\ell_i^*} \qquad 1 = \lambda \ln 2$$

$$2^{-\ell_i^*} = p_i \qquad \ell_i^* = -\log_2 p_i$$

$$\mathcal{L}^* = \sum_i -p_i \log_2 p_i = H(X)$$

Shannon's source coding theorem

Reintroducing condition $\ell_i \in \mathbb{N}$, it can be shown that

- **Shannon's theorem**

$$\mathcal{L}^* \geq H(X)$$

with equality if and only if:

$$\forall i \in \{1, 2, \ldots M\},\ \exists k \in \mathbb{N} \,|\, p_i = 2^{-k}$$

Entropy coding

Shannon's theorem:

Optimal coding rate $\geq$ source entropy

therefore optimal coding is called *Entropy coding*.

- It is a strict identity if the probability distribution is *dyadic*, i.e., $\forall i, \exists k \in \mathbb{N} : p_i = 2^{-k}$
- However, when M increases, the optimal coding rate is practically very close to H

Entropy coding

- It easy to find an instantaneous code such that $\mathcal{L} < H(X) + 1$, we just take $\ell_i = \lceil -\log_2 p_i \rceil$
- Kraft's inequality is satisfied:

$$\ell_i = -\log p_i + \delta_i \qquad 0 \leq \delta_i < 1$$

$$2^{-\ell_i} = p_i \times 2^{-\delta_i} = \epsilon_i p_i \qquad \frac{1}{2} < \epsilon_i \leq 1$$

$$\sum_i 2^{-\ell_i} = \sum_i \epsilon_i p_i \leq \sum_i p_i = 1$$

- The average length is:

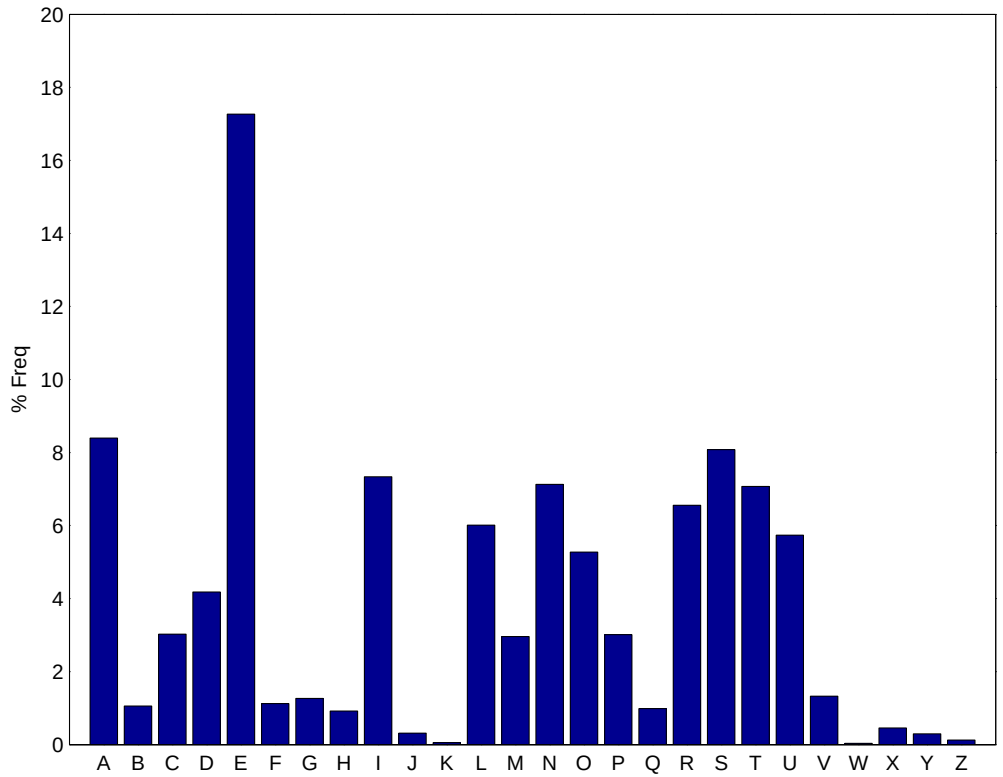
$$\ell_i = \lceil -\log_2 p_i \rceil < -\log_2 p_i + 1 \qquad p_i \ell_i < -p_i \log_2 p_i + p_i$$

$$\sum_i p_i \ell_i < \sum_i (-p_i \log_2 p_i + p_i) \qquad \mathcal{L} < H(X) + 1$$

- Since $\mathcal{L}^* \leq \mathcal{L}$, $\boxed{H(X) \leq \mathcal{L}^* < H(X) + 1}$

Example: French text compression

Source entropy	3.999 bpS
Method	Any VLC
Coding rate ($\mathcal{L}$)	$\geq$ 3.999 bpS
Compression ratio	$\leq$ 1.25



Letter distribution in a French text

Outline

Huffman coding

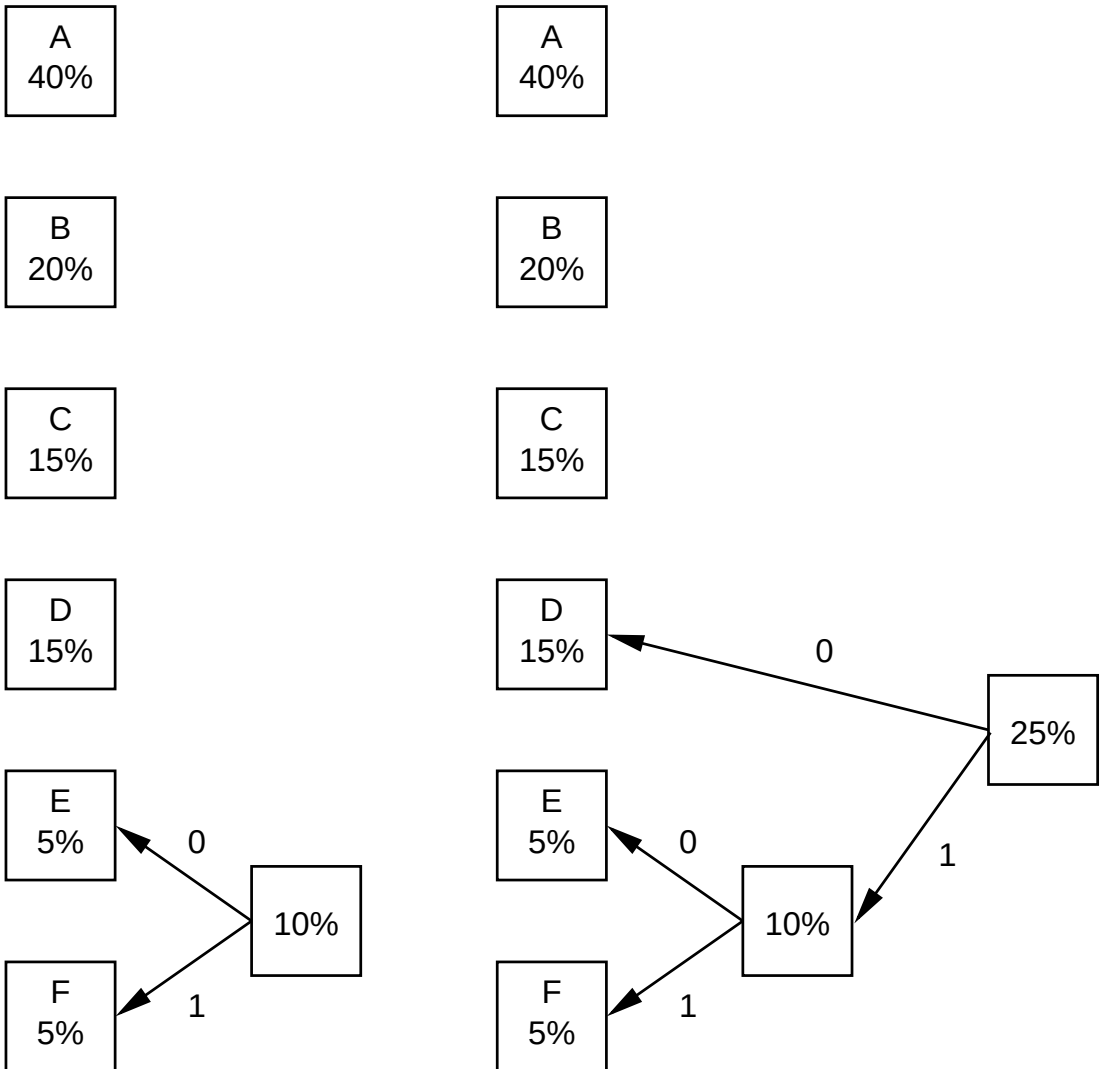
Huffman discovered how to build the optimal lossless coder for any source with known probabilities.

Example:

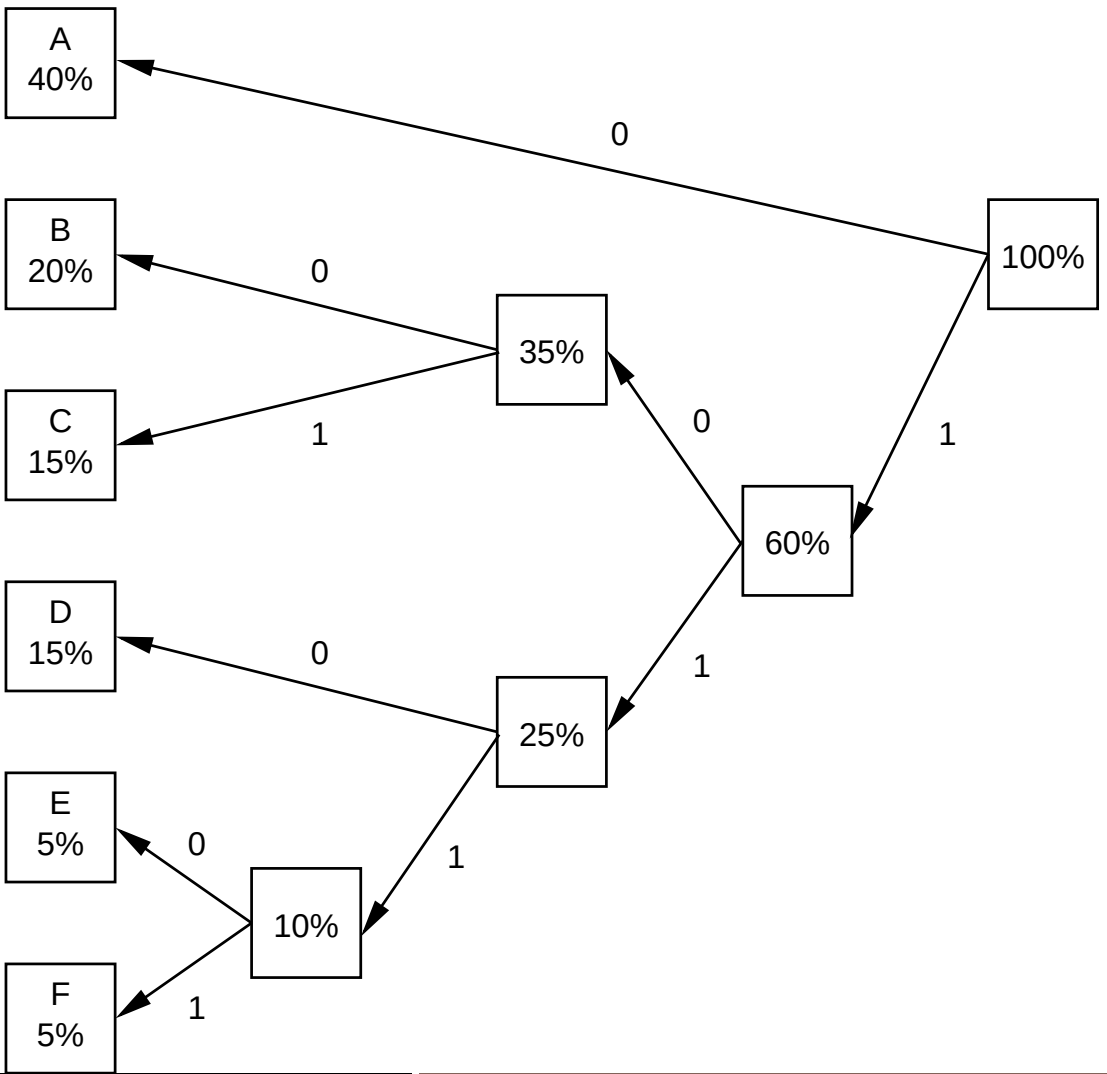
Symbol	Probability
A	0.4
B	0.2
C	0.15
D	0.15
E	0.05
F	0.05

6 symbols, 3 bits par symbol (without coding)

Huffman coding



Huffman coding



Huffman coding

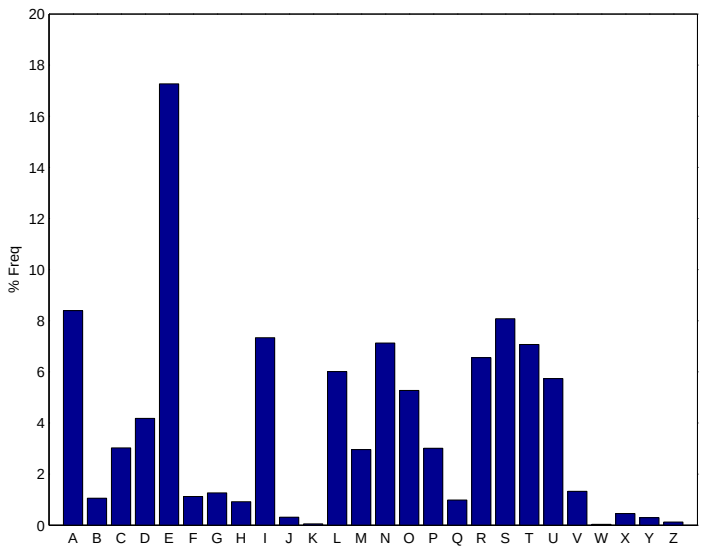
Symbol	Probability	Codeword
A	0.4	0
B	0.2	100
C	0.15	101
D	0.15	110
E	0.05	1110
F	0.05	1111

$$\mathcal{L} = 0.4 \cdot 1 + 0.2 \cdot 3 + 0.15 \cdot 3 + 0.15 \cdot 3 + 0.05 \cdot 4 + 0.05 \cdot 4$$
$$= 2.3 \text{ bits/Symbol}$$

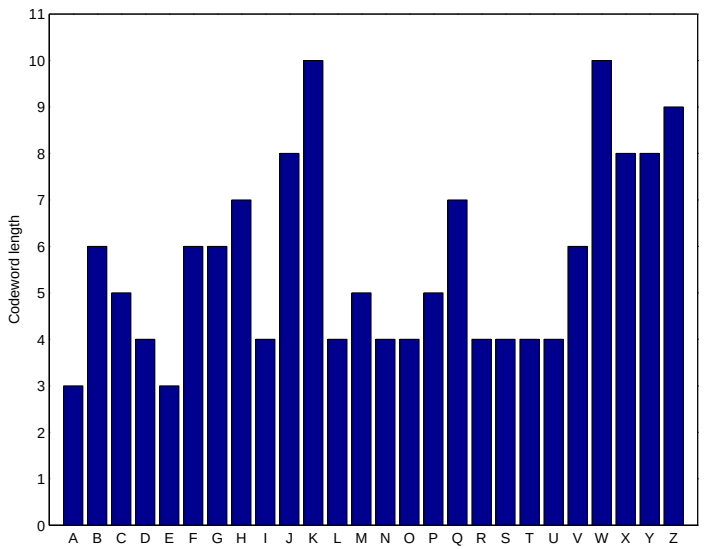
$$H = 0.4 \cdot \log_2 \frac{1}{0.4} + 0.2 \cdot \log_2 \frac{1}{0.2} + 2 \cdot 0.15 \cdot \log_2 \frac{1}{0.15}$$
$$+ 2 \cdot 0.05 \cdot \log_2 \frac{1}{0.05}$$
$$\cong 2.2464 \text{ bits/Symbol}$$

Example: Compression of French text

Method	Huffman
Source entropy	3.999 bpS
Taux de codage ($\mathcal{L}$)	4.041 bpS
Compression ratio	1.238



Letter probabilities in a French text



Huffman (i.e., optimal) codeword lengths

Huffman coding

- Example: codage image binaire
- $\Pr(\{X = B\}) = p \ll 1$
- $\Pr(\{X = W\}) = 1 - p$
- $H(X)$

Huffman coding

- Example: codage image binaire
- $\Pr(\{X = B\}) = p \ll 1$
- $\Pr(\{X = W\}) = 1 - p$
- $H(X) \approx p(\log \frac{1}{p} + \frac{1}{\ln 2}) \ll 1$
- Huffman coding: $B \to 0 \quad W \to 1$
- $\mathcal{L} = 1 \gg H(X)$

Huffman coding

How to improve performance?

- Let us refer to the block of the first K symbols of X_i as X^K
- $H(X^K) \leq \sum_i H(X_i)$ with equality if and only if components of X^K are independent from each other
- Block coding: we reduce the number of bits per symbol

Huffman coding

Block coding

- Example: coding a B/W image with independent pixels

Huffman coding

Block coding

- Example: coding a B/W image with independent pixels
- $\Pr(BB) = p^2$
- $\Pr(WB) = p(1-p) \approx p$
- $\Pr(BW) = p(1-p) \approx p$
- $\Pr(WB) = (1-p)^2 \approx 1-2p$
- $H(X_1, X_2) = 2H(X) \ll 2$
- Code de Huffman :

Huffman coding

Block coding

- Example: coding a B/W image with independent pixels
- $\Pr(BB) = p^2$
- $\Pr(WB) = p(1-p) \approx p$
- $\Pr(BW) = p(1-p) \approx p$
- $\Pr(WB) = (1-p)^2 \approx 1-2p$
- $H(X_1, X_2) = 2H(X) \ll 2$
- Code de Huffman : $WB \rightarrow 0 \quad WB \rightarrow 10$ $BW \rightarrow 110$ $BB \rightarrow 111$
- $\mathcal{L} \approx 1$
- $\mathcal{L}_{\mathcal{S}} \approx \frac{1}{2}$

Huffman coding

Block coding

- Hypothesis: $\frac{H(X^K)}{K}$ is converging
- This is true for example in the case of stationary process
- Average lenght of optimal code:

$$H(X^K) \leq \mathcal{L}^* < H(X^K) + 1$$

$$\frac{H(X^K)}{K} \leq \frac{\mathcal{L}^*}{K} = \mathcal{L}_S^* < \frac{H(X^K)}{K} + \frac{1}{K}$$

$$\lim_K \mathcal{L}_S^* = \lim_K \frac{H(X^K)}{K} = \mathcal{H}(X)$$

$$\mathcal{L}_S^* \to \mathcal{H}(X) \leq H(X)$$

- $\mathcal{H}(X)$ is called *entropic rate*

Huffman coding

Block coding

- Block coding:

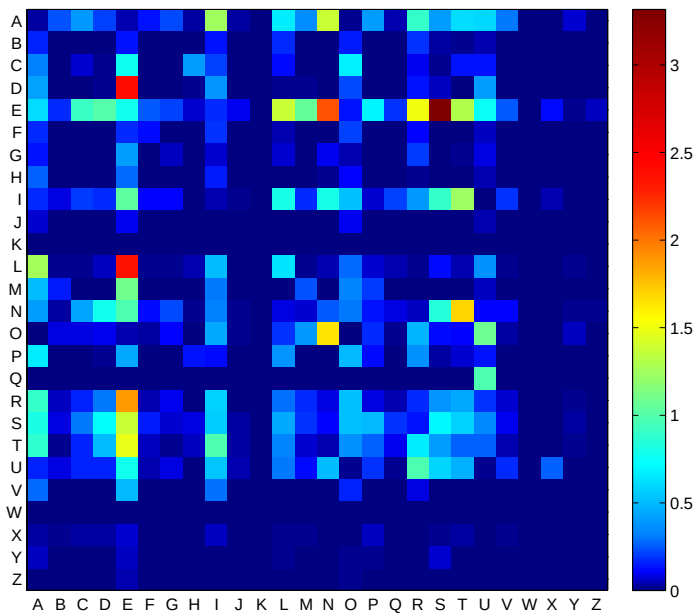
$$\mathcal{L}_S^* \to \mathcal{H}(X) \leq H(X)$$

- The best performance is attained when the whole message of K symbols is encoded as a single block
- This means an alphabet of N^K symbols
- Block coding improves performance even for iid variables, since it asymptotically removes the 1-bit penalty for non-dyadic distributions

Example: Compression of French text

K=1	Letter Entropy	3.999 bpB	3.999 bpS
K=2	Digram Entropy	7.440 bpB	3.720 bpS
K=3	Trigrams Entropy	9.452 bpB	3.151 bpS

bpB: bits per block; bpS: bits per symbol, bits per letter



Digram distribution in a French text

Most common trigrams:

ait	ent	les
1.59%	1.25%	0.94%
lle	des	ant
0.78%	0.72%	0.70%
que	our	ien
0.67%	0.63%	0.60%

Limits of Huffman coding

We would like to use Huffman with larger and large K
This is practically impossible

- Complexity is exponential with K
- The joint probability estimation for large blocks is costly and unreliable

Limits of Huffman coding

We would like to use Huffman with larger and large K
This is practically impossible

- Complexity is exponential with K
- The joint probability estimation for large blocks is costly and unreliable

Arithmetic coding solves (at least) the first problem

Arithmetic coding

- *Arithmetic coding* allows to perform block coding or context-based coding with linear complexity
- Instead of producing the entire dictionary and then selecting the codeword, only the codeword associated to the block of n input symbols is built
- The arithmetic is suboptimal, but *asymptotically* optimal

$$\mathcal{L} \leq H(X^K) + 2$$

$$\mathcal{L}_S = \mathcal{L}/K$$

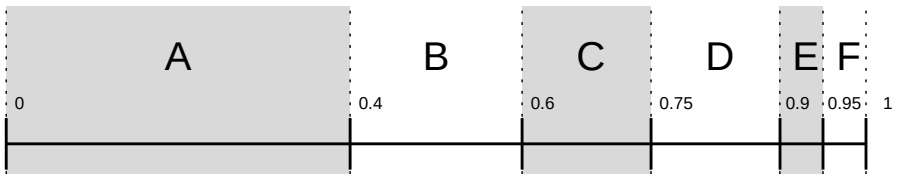
$$\lim_{K\to\infty} \mathcal{L}_S = \lim_{K\to\infty} \frac{H^K}{K} = \mathcal{H}(X)$$

- Low-complexity encoding and decoding: two sums and two multiplications per input symbols (arithmetic operations)

Arithmetic coding: example

Symbole	A	B	C	D	E	F
Probability	0.4	0.2	0.15	0.15	0.05	0.05

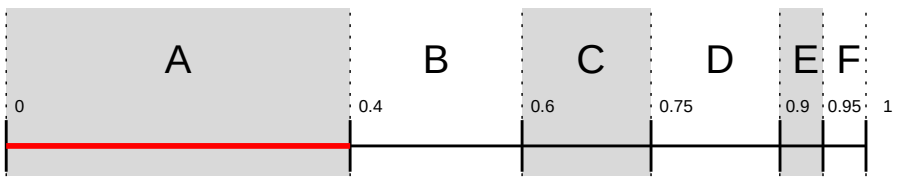
Input sequence: ACFD



Arithmetic coding: example

Symbole	A	B	C	D	E	F
Probability	0.4	0.2	0.15	0.15	0.05	0.05

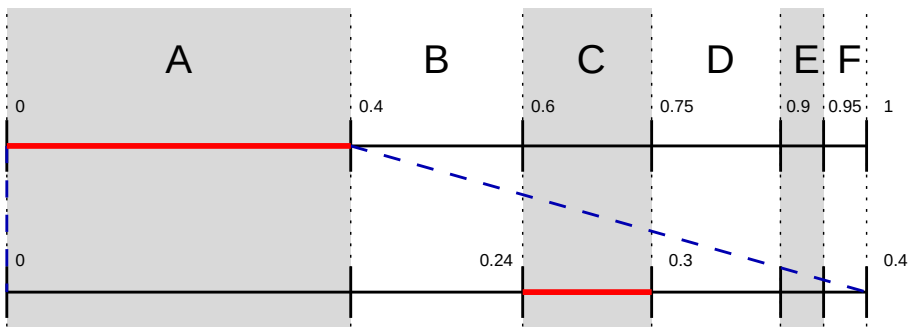
Input sequence: ACFD



Arithmetic coding: example

Symbole	A	B	C	D	E	F
Probability	0.4	0.2	0.15	0.15	0.05	0.05

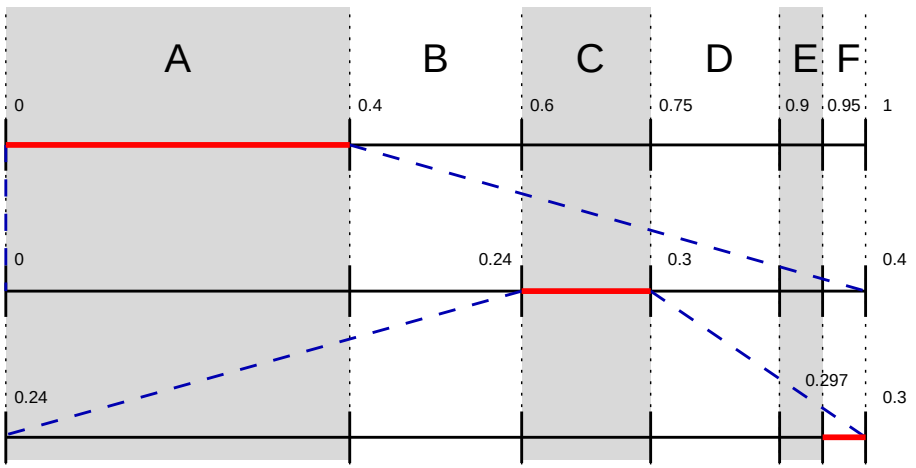
Input sequence: ACFD



Arithmetic coding: example

Symbole	A	B	C	D	E	F
Probability	0.4	0.2	0.15	0.15	0.05	0.05

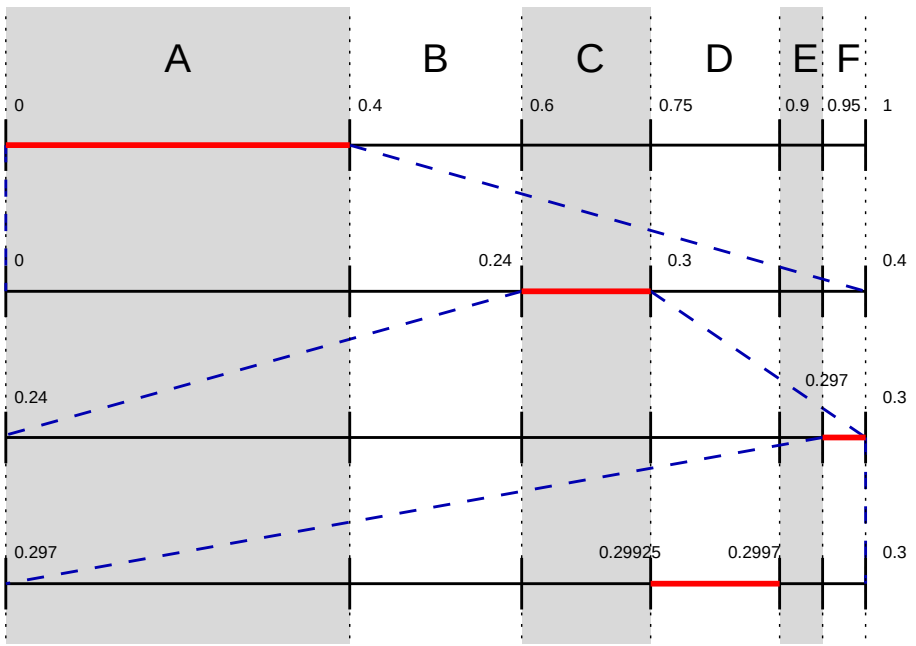
Input sequence: ACFD



Arithmetic coding: example

Symbole	A	B	C	D	E	F
Probability	0.4	0.2	0.15	0.15	0.05	0.05

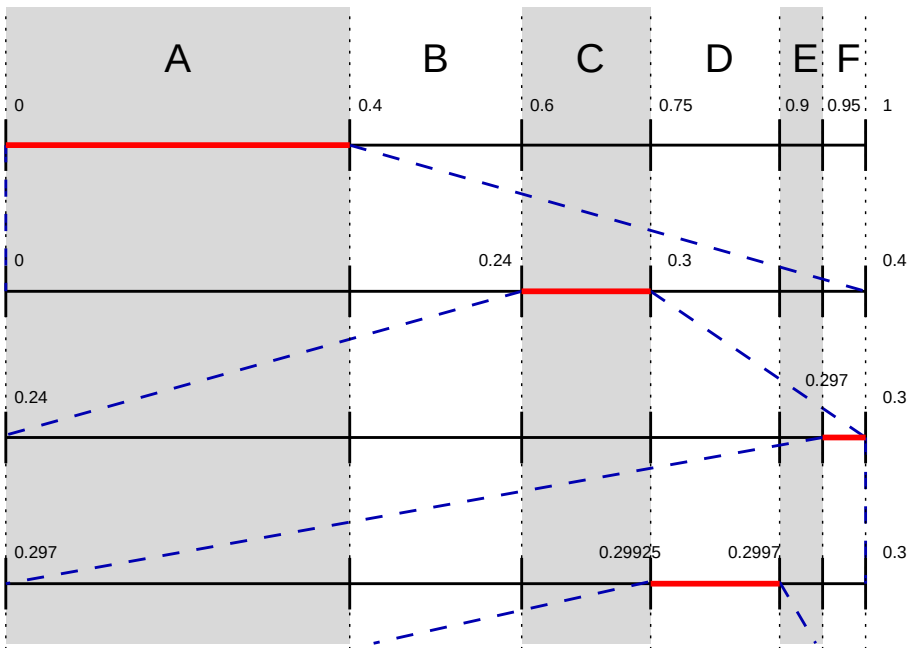
Input sequence: ACFD



Arithmetic coding: example

Symbole	A	B	C	D	E	F
Probability	0.4	0.2	0.15	0.15	0.05	0.05

Input sequence: ACFD



Arithmetic coding

- Each sequence deetrmines an interval in $[0, 1]$
- For each new symbol, 2 multiplications and 2 sums are needed to update the interval
- A sequence is encoded as the center interval, represented with *the suitable precision*, i.e. half the size of the interval

$$\ell_i = \lceil \log_2 p_i \rceil + 1$$
$$\log_2 p_i + 1 \leq \ell_i < \log_2 p_i + 2$$
$$H(X) + 1 \leq \mathcal{L}_{AC} < H(X) + 2 \leq \mathcal{L}^* + 2$$

Arithmetic coding

- Problem: estimation of $P(X^K)$, where K can be as large as the size of the message
- In the previous exemple, symbols wehere supposed independent, $P(X^K) = \prod_{i=1}^{K} p(x_i)$
- Symbols statistics can be learned during the encoding process (adaptivity)

Context-based coding

- Estimation of joint probability $P(X^K)$ may be difficult
- On the other hand, often a symbol only depends on a few neighbors
- We define as *context* a set of a few, let us say N_S previous symbols that mainly impacts on the current one
- The number of possible contexts is $N_C M^{N_S}$ at most
  - Contexts may by clustered in order to reduce complexity and improve probability estimation
- Using N_C contexts is equivalent to having N_C different arithmethic encoders and switching from the one to the other

Context-based coding : B/W image

Let X and Y be two neighboring pixels in a B/W image.

Y \ X	B	W
B	0.15	0.05
W	0.05	0.75

Joint probability of *X* and *Y*

Y \ X	B	W
B	0.75	0.25
W	$\frac{1}{16}$	$\frac{15}{16}$

Conditional probability of *X* given *Y*

H(X)	H(X,Y)/2	H(X|Y)	HE	AE	CB AE
0.722	0.577	0.432	1	0.722	0.432

HE: Huffman Encoder, One Symbol
AE: Arithmetic Encoder
CB-AE: Context-Based AE

Arithmeric coding: conclusions

Advantages

- It allows encoding long sequences of symbols with linear complexity
- Block coding: it benefits from high order dependencies and removes the penalty due to non-dyadic distributions
- Context-based coding: simple model for high order statistics
- Adaptivity: non-stationary sources

Arithmeric coding: conclusions

Disadvantages and difficulties

- Implementation may be tricky
- Need of initialization
- How to choose contexts
- Adaptivity: robust estimation demands for a potentially large training set

Outline

Dictionary-based coding

- A dictionary of most frequent sequences of symbols is built during the encoding process
- The dictionary automatically adapts to non-stationary signals
- No initialisation needed (universal coding)
- Used in popular lossless coding softwares (zip, gzip, bzip, *etc.*)

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Table

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1

Output

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Present 0

Table *Output*

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

New 0 0

Table *Output*

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Add 0 0

Table *Output*

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1 0

0

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Output 0 ~~0~~

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0												
		0												

Output

0

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Present 0

Table

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1 0
0

Output

0

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Present 0 0

Table

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1 0
0

Output

0

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

New 0 0 1

Table

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1 0

0

Output

0

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Add 0 0 1

Table

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1 0 0
0 0
1

Output

0

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Output 0 0 1

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0											
		0	0											
			1											

Output

0 2

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Present 1

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0											
		0	0											
			1											

Output

0 2

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

New 1 0

Table

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1 0 0
0 0
1

Output

0 2

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Add 1 0

Table

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1 0 0 1
0 0 0
1

Output

0 2

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Output 1 ~~0~~

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1										
		0	0	0										
			1											

Output

0 2 1

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Present 0

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1										
		0	0	0										
			1											

Output

0 2 1

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Present 0 0

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1										
		0	0	0										
			1											

Output

0 2 1

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

New 0 0 0

Table

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1 0 0 1
0 0 0
1

Output

0 2 1

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Add 0 0 0

Table

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1 0 0 1 0
0 0 0 0
1 0

Output

0 2 1

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Output 0 0 ~~0~~

Table

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1 0 0 1 0
0 0 0 0
1 0

Output

0 2 1 2

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Prefix

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1	0									
		0	0	0	0									
			1		0									

Output

0 2 1 2

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Prefix

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1										
		0	0	0										
		0	1											

Output

0 2 1 2

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Present 0

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1										
		0	0	0										
		0	1											

Output

0 2 1 2

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Present 0 0

Table

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
0	1	0	0	1										
		0	0	0										
		0	1											

Output

0 2 1 2

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Present 0 0 0

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1										
		0	0	0										
		0	1											

Output

0 2 1 2

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

New 0 0 0 0

Table

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1 0 0 1
0 0 0
0 1

Output

0 2 1 2

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Add 0 0 0 0

Table

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1 0 0 1 0
0 0 0 0
0 1 0
0

Output

0 2 1 2

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Output 0 0 0 0

Table

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
0	1	0	0	1	0									
		0	0	0	0									
		0	1		0									
					0									

Output

0	2	1	2	2

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Present 0

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1	0									
		0	0	0	0									
		0	1		0									
					0									

Output

0 2 1 2 2

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

New 0 1

Table

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
0	1	0	0	1	0									
		0	0	0	0									
		0	1		0									
					0									

Output

0 2 1 2 2

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Add 0 1

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1	0	0								
		0	0	0	0	1								
		0	1		0									
					0									

Output

0 2 1 2 2

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Output 0 1

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1	0	0								
		0	0	0	0	1								
		0	1		0									
					0									

Output

0 2 1 2 2 0

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Prefix 0 1

Table

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
0	1	0	0	1	0	0								
		0	0	0	0	1								
		0	1		0									
					0									

Output

0 2 1 2 2 0

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Prefix 0 1

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1	0									
1		0	0	0	0									
		0	1		0									
					0									

Output

0 2 1 2 2 0

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Present 1

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1	0									
1		0	0	0	0									
		0	1		0									
					0									

Output

0 2 1 2 2 0

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Present 1 0

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1	0									
1		0	0	0	0									
		0	1		0									
					0									

Output

0 2 1 2 2 0

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

New 1 0 1

Table

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
0	1	0	0	1	0									
1		0	0	0	0									
		0	1		0									
					0									

Output

0 2 1 2 2 0

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Add 1 0 1

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1	0	1								
1		0	0	0	0	0								
		0	1		0	1								
					0									

Output

0 2 1 2 2 0

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Output 1 0 X

Table

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
0	1	0	0	1	0	1								
1		0	0	0	0	0								
		0	1		0	1								
					0									

Output

0 2 1 2 2 0 4

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

New 1 0 0

Table

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
0	1	0	0	1	0	1								
1		0	0	0	0	0								
		0	1		0	1								
					0									

Output

0 2 1 2 2 0 4

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Output 1 0 0

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1	0	1	1							
1		0	0	0	0	0	0							
		0	1		0	1	0							
					0									

Output

0 2 1 2 2 0 4 4

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Prefix 1 0 0

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1	0	1	1							
1		0	0	0	0	0	0							
		0	1		0	1	0							
					0									

Output

0 2 1 2 2 0 4 4

Dictionary-based coding: example

Input

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0

Prefix 1 0 0

Table

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
0	1	0	0	1	0	1								
1		0	0	0	0	0								
		0	1	0	0	1								
					0									

Output

0 2 1 2 2 0 4 4

Decoding: example

Table

`0 1 2 3 4 5 6 7 8 9 A B C D E`

0 1

Input

`0 2 1 2 2 0`

New

Output

Decoding: example

Table

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1

Input

0 2 1 2 2 0

New

2 0x

Output

0x

Decoding: example

Table

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1 0

x

Input

0 2 1 2 2 0

New

2 0x

Output

0x

Decoding: example

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0												
		x												

Input

0 2 1 2 2 0

New

3 0xy

Output

0x

00xy

Decoding: example

Table

0 1 2 3 4 5 6 7 8 9 A B C D E

0 1 0 0
0 0
y

Input

0 2 1 2 2 0

New

3 00y

Output

000y

Decoding: example

Table

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
0	1		0	0										
			0	0										
				y										

Input

0	2	1	2	2	0

New

4	1z

Output

000y
0001z

Decoding: example

Table

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**
0	1	0	0	1										
		0	0	z										
			1											

Input

0 2 1 2 2 0

New

4 1z

Output

0001z

Decoding: example

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1										
		0	0	z										
			1											

Input

0 2 1 2 2 0

New

5	00w

Output

0001z

000100w

Decoding: example

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1	0									
		0	0	0	0									
			1		w									

Input

0 2 1 2 2 0

New

5 00w

Output

000100w

Decoding: example

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1	0									
		0	0	0	0									
			1		w									

Input

0 2 1 2 2 0

New

5 00w

Output

000100w

Decoding: example

Table

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***A***	***B***	***C***	***D***	***E***
0	1	0	0	1										
		0	0	0										
		0	1											

Input

0 2 1 2 2 0

New

2 000

Output

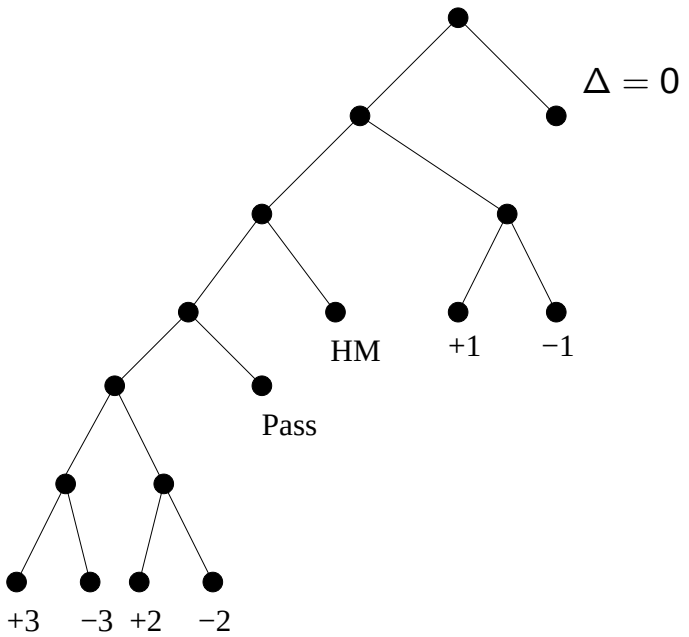
0001000

Run-Length Encoding (RLE)

- B/W image coding
- Long runs of zeros or ones
- Idea: to encode the length of the run instead of the individual pixel value

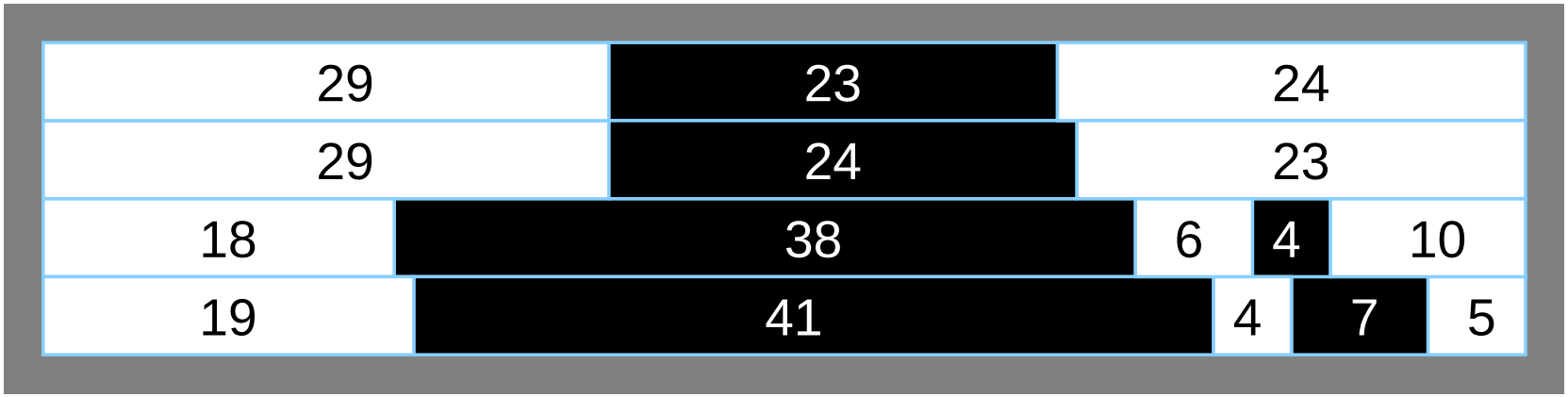
Run-Length Encoding (RLE)

- Huffman code: too large alphabet
- Huffman only on powers of two
- Each length is represented as sum of powers of two (binary representation)
- Horizontal mode: absolute length
- Vertical mode: difference with upper row (if it is ≤ 3)
- Pass: new block



Vertical mode: symbols and Huffman code

Run-Length Encoding (RLE) : Example

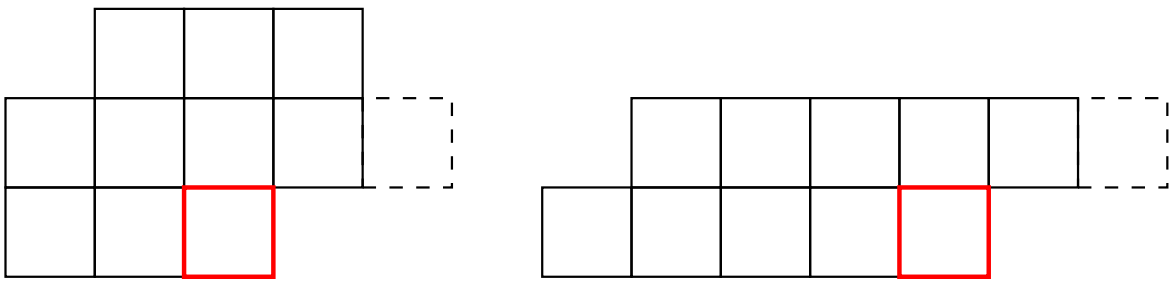


HM 16+8+4+1	16+4+2+1	16+8		
VM 0	**+1**	**-1**		
HM 16+2	32+4+2	4+2	PASS 4	PASS 8+2
VM +1	**+3**	**-2**	**+3**	HM 4+1

JBIG-1

Joint Bi-level Image Experts Group

- Standard ISO/IEC 11544 and recommandation ITU-T T.82
- Context-based arithmetic coding
- Progressive coding (resolution scalability)
- *Template* of 10 pixels, 2 shape with a single variable pixel position



Two rows template: faster but $\approx$ 5% rate increase

JBIG-2

Joint Bi-level Image Experts Group

- Standard ISO/IEC 14492 and recommandation ITU-T T.88
- Image segmented into **text**, **halftones**, and **other**
- **Text**: A dictionnary of symbols is created and used for coding
- **Halftone**: the gray-scale image is encoded using a disctionary of *halftone patterns*
- **Other**: Context-based arithmetic coding
- PDF files (version 1.4 and behind) can contain JBIG2 data

Outline

Quantization and coding

- Typically, quantization is followed by entropy coding (EC)
- Should we modify the desing of a quantizer knowing that it is followed by EC?
- Is it possible to estimate the performance of Q+EC?

Quantization and coding

- A generic quantizer can be modeled as a uniform quantizer (UQ) preceeded by a non-linear function $f(x)$
- We want to minimize the quantization error energy:

$$\sigma_Q^2 = \frac{\delta}{12} \int \frac{p_X(x)}{f'^2(x)} dx$$

- subject to an entropy constraint: $H(\hat{X}) \leq b$
- In the high resolution (HR) hypothesis, it can be shown that the optimum is obtained when f' is constant

Quantization and coding

In HR, it can be shown that:

- For a given MSE, the minimum the minimum quantizer entropy is obtained with UQ
- For a given quantizer entropy, the minimum MSE is obtained with UQ
- A UQ followed by an EC has a gain of 2.81 dB over a Lloyd-Max quantizer without EC in the case of Gaussian i.i.d. source
- The RD curve has still the behavior: $D \propto 2^{-2R}$