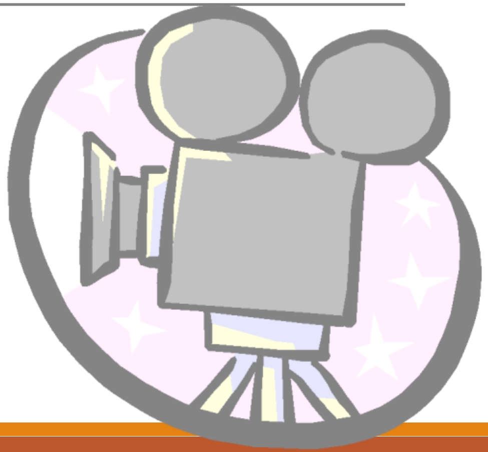
Image Processing

CS-317/CS-341



Outline

- Lossless Compression
  - Huffman Coding
  - Arithmetic Coding

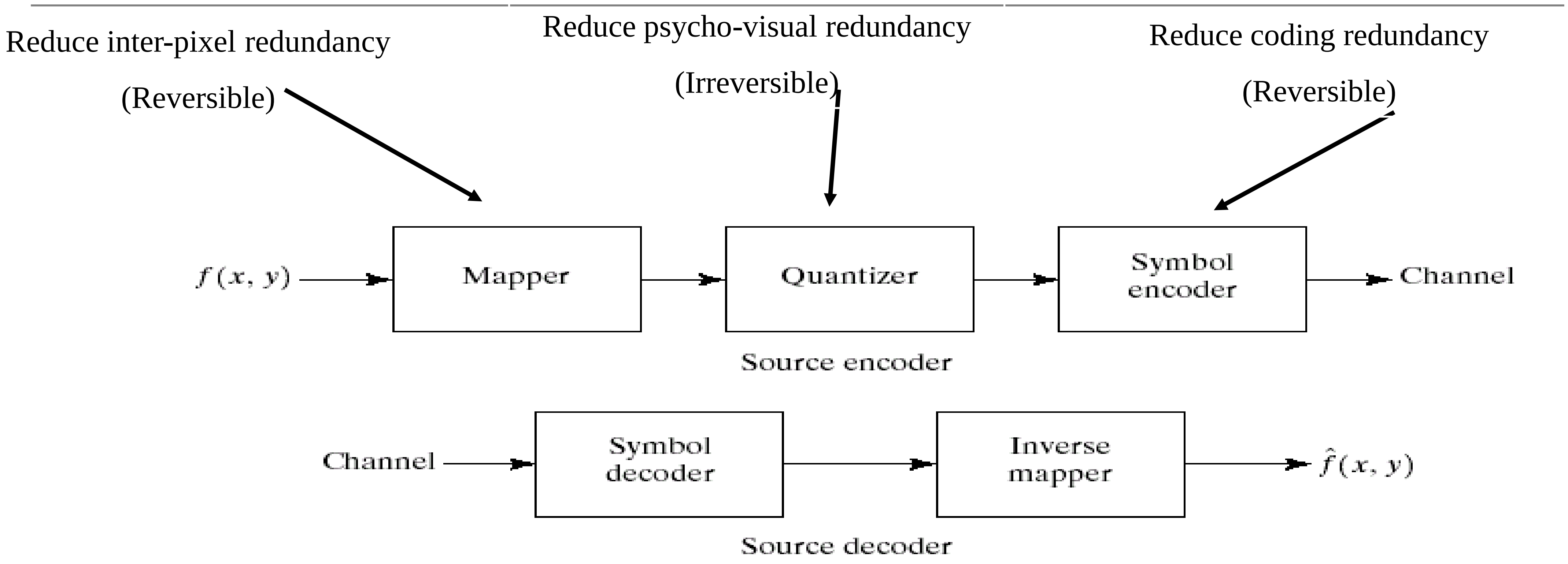
Techniques of Lossless Compression

Huffman Coding

Arithmetic Coding

Run Length Coding

Image Compression model



(a) Source encoder and (b) source decoder model.

Lossless or Error-Free Compression

Variable-length Coding

Huffman coding (optimal code)

Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				

FIGURE 8.11 Huffman source reductions.

Lossless or Error-Free Compression

Variable-length Coding

Huffman coding

FIGURE 8.12
Huffman code assignment procedure.

Original source			Source reduction							
Sym.	Prob.	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

$$L_{avg} = (0.4)(1) + (0.3)(2) + (0.1)(3) + (0.1)(4) + (0.06)(5) + (0.04)(5)$$
$$= 2.2\, bits / symbol$$
$$entropy = 2.14\, bits / symbol$$

Example:

a3 a1 a2 a2 a6 =010100111100

Lossless or Error-Free Compression

Variable-length Coding

Huffman decoding

FIGURE 8.12
Huffman code assignment procedure.

Original source			Source reduction							
Sym.	Prob.	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

Example:

010100111100 = a3 a1 a2 a2 a6

Lossless or Error-Free Compression

Variable-length Coding

Huffman coding

- Variable length code whose length is inversely proportional to that character's frequency
- must satisfy non-prefix property to be uniquely decodable
- two pass algorithm
  - first pass accumulates the character frequency and generate codebook
  - second pass does compression with the codebook

Lossless or Error-Free Compression

Variable-length Coding

Huffman coding

- create codes by constructing a binary tree
  - 1. consider all characters as free nodes
  - 2. assign two free nodes with lowest frequency to a parent nodes with weights equal to sum of their frequencies
  - 3. remove the two free nodes and add the newly created parent node to the list of free nodes
  - 4. repeat step2 and 3 until there is one free node left. It becomes the root of tree

Arithmetic (or Range) Coding

- The main weakness of Huffman coding is that it encodes source symbols one at a time.
- Arithmetic coding encodes sequences of source symbols together.
  - There is no one-to-one correspondence between source symbols and code words.
- Slower than Huffman coding but can achieve better compression.

Arithmetic Coding (cont'd)

A sequence of source symbols is assigned to a sub-interval in [0,1) which can be represented by an arithmetic code, e.g.:

	sub-interval	arithmetic code
$a_1\ a_2\ a_3\ a_3\ a_4$	→ [0.06752, 0.0688)	→ 0.06816

Start with the interval [0, 1) ; a sub-interval is chosen to represent the message which becomes smaller and smaller as the number of symbols in the message increases.

Arithmetic Coding (cont'd)

Encode message: $a_1\ a_2\ a_3\ a_3\ a_4$

Source Symbol	Probability
a_1	0.2
a_2	0.2
a_3	0.4
a_4	0.2

1) Start with interval [0, 1)

0 1

2) Subdivide [0, 1) based on the probabilities of α_i

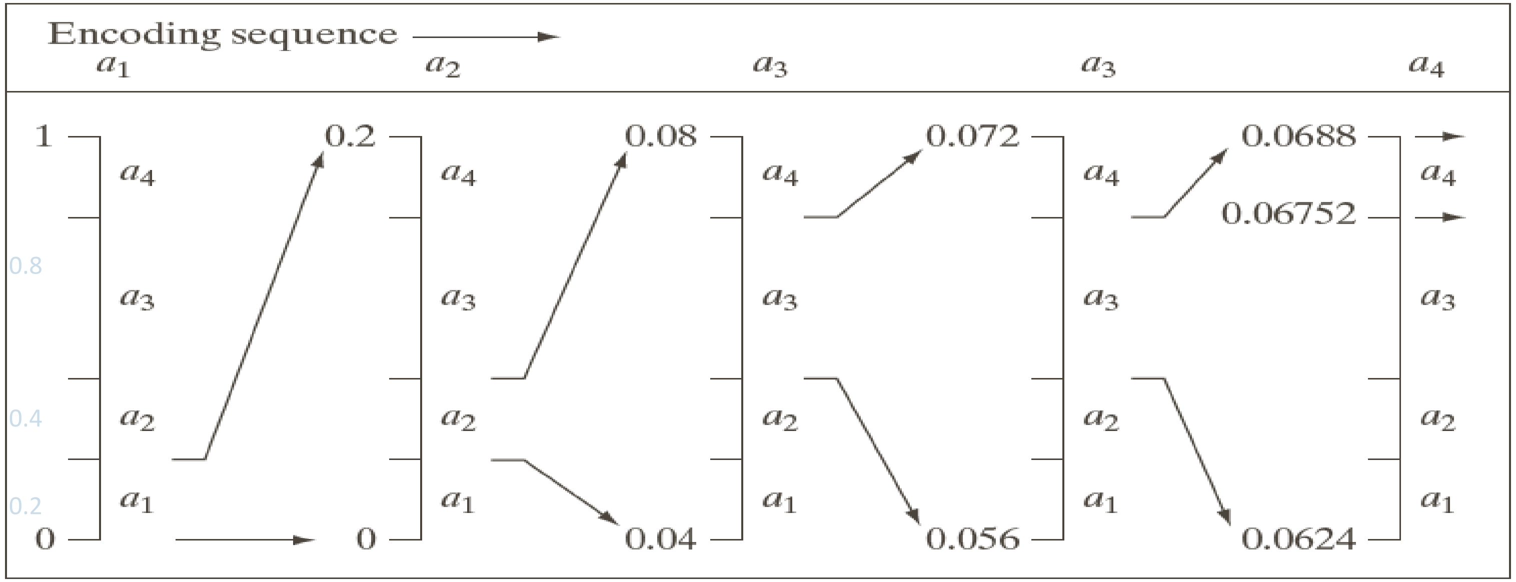
Initial Subinterval
[0.0, 0.2)
[0.2, 0.4)
[0.4, 0.8)
[0.8, 1.0)

3) Update interval by processing source symbols

Arithmetic Coding

Example: Encode the message $a_1\ a_2\ a_3\ a_3\ a_4$

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)



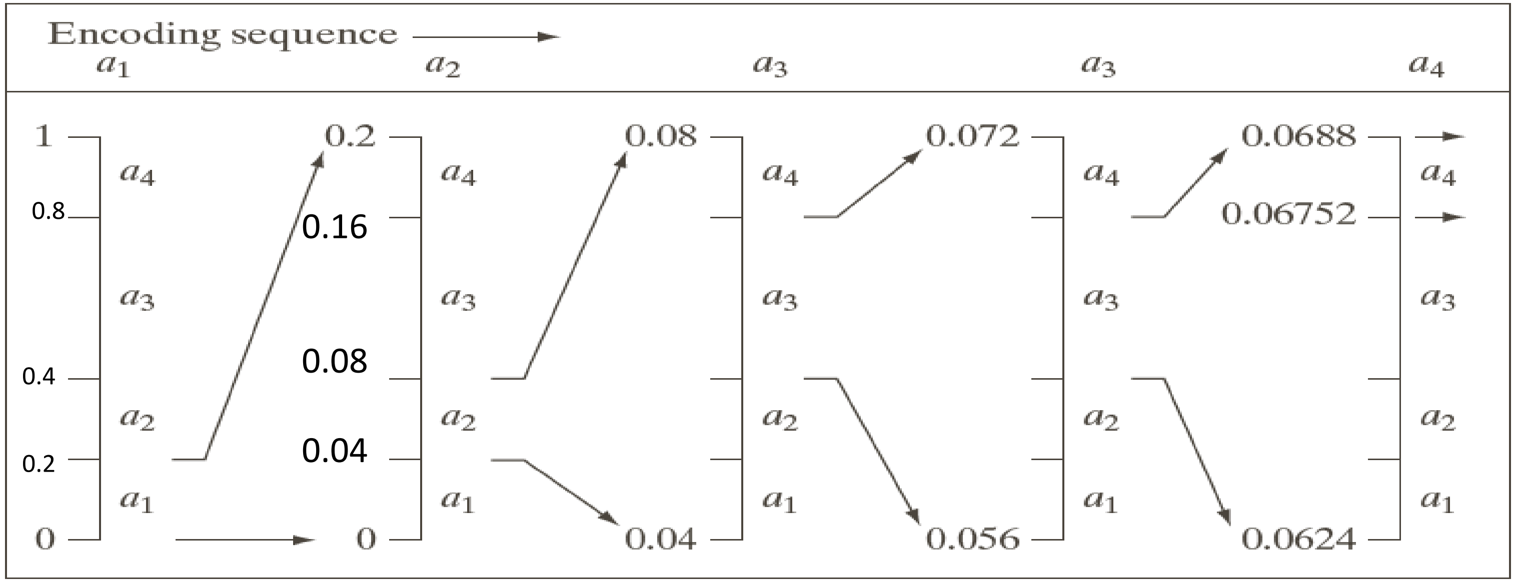
Example: Encode the message a_1 a_2 a_3 a_3 a_4

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

d=upper bound-lower bound

Range of symbol=lower limit+ d *(probability of Symbol)

d=0.2-0=0.20
Range of ‘a1’=0+0.2*0.2=0.04



Range of ‘a2’=0.04+0.2*0.2=0.08

Range of ‘a3’=0.08+0.2*0.4=0.16

Range of ‘a4’=0.16+0.2*0.2=0.2

Example: Encode the message a_1 a_2 a_3 a_3 a_4

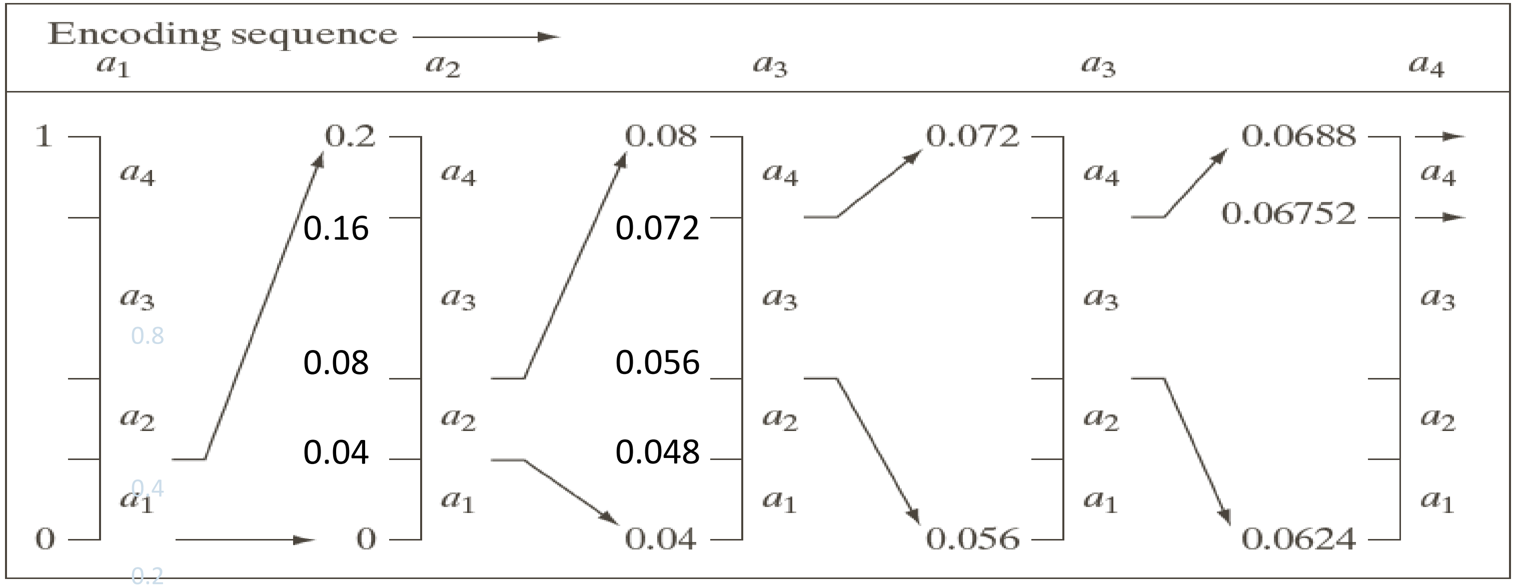
d=upper bound-lower bound

Range of symbol=lower limit+ d *(probability of Symbol)

d=0.08-0.04=0.04

Range of ‘a1’=0.04+0.04*0.2=0.048

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)



Range of ‘a2’=0.048+0.04*0.2=0.056

Range of ‘a3’=0.056+0.04*0.4=0.072

Range of ‘a4’=0.072+0.04*0.2=0.08

Example: Encode the message $a_1\ a_2\ a_3\ a_3\ a_4$

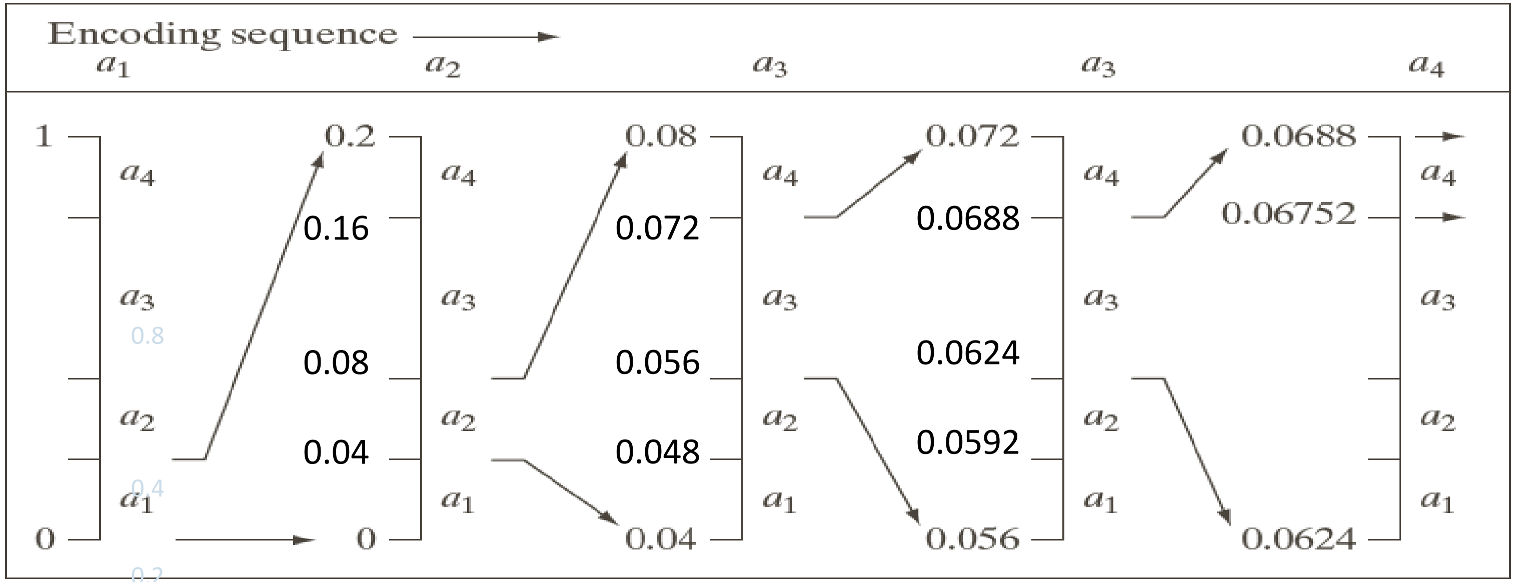
d=upper bound-lower bound

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

Range of symbol=lower limit+ d *(probability of Symbol)

d=0.072-0.056=0.016

Range of ‘a1’=0.056+0.016*0.2=0.0592



Range of ‘a2’=0.0592+0.016*0.2=0.0624

Range of ‘a3’=0.0624+0.016*0.4=0.0688

Range of ‘a4’=0.0688+0.016*0.2=0.072

Example: Encode the message $a_1\ a_2\ a_3\ a_3\ a_4$

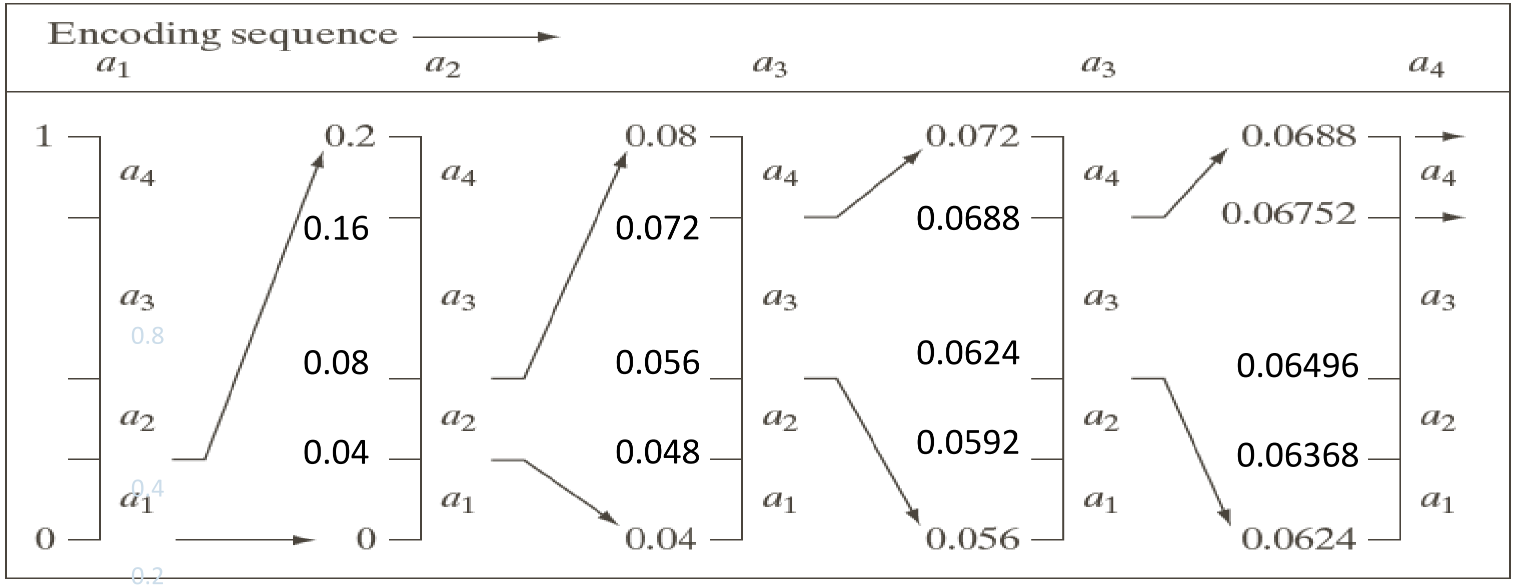
Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

d=upper bound-lower bound

Range of symbol=lower limit+ d *(probability of Symbol)

d=0.0688-0.0624=0.0064

Range of 'a1'=0.0624+0.0064*0.2=0.06368



Range of 'a2'=0.06368+0.0064*0.2=0.06496

Range of 'a3'=0.06496+0.0064*0.4=0.06752

Range of 'a4'=0.06752+0.0064*0.2=0.0688

Example (cont..)

Encode $a_1\ a_2\ a_3\ a_3\ a_4$

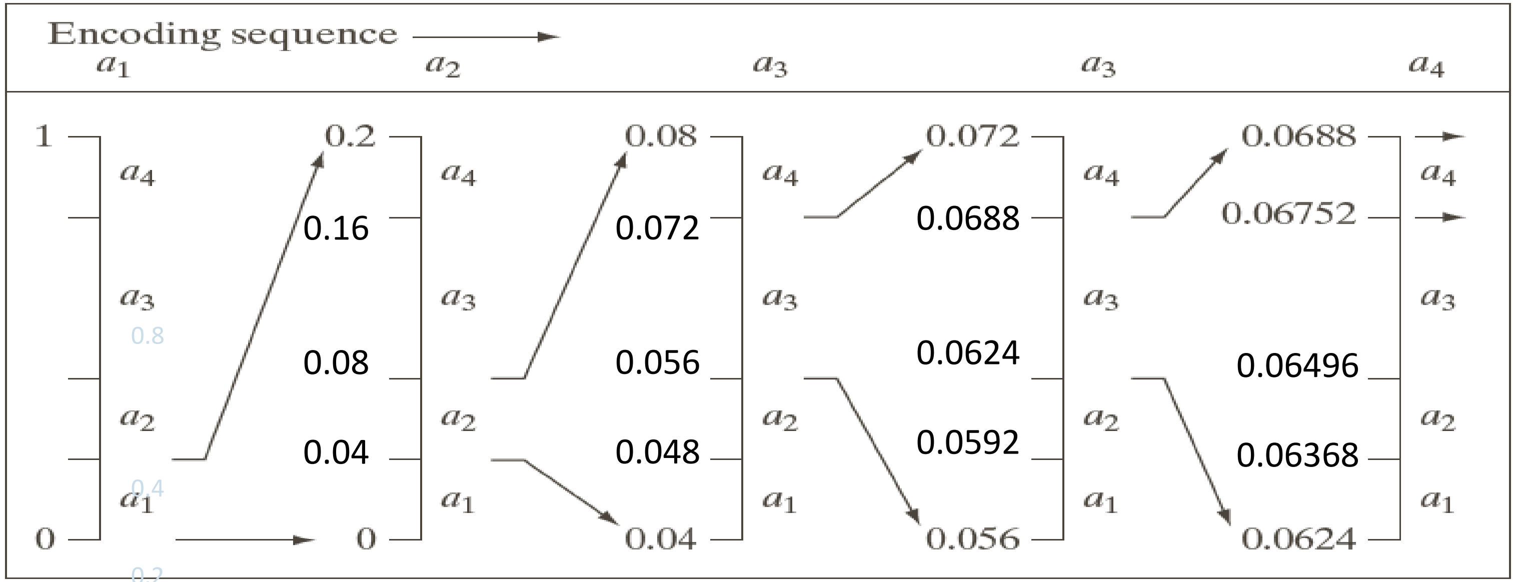
Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

Code lies in this range [0.06752, 0.0688)

or

Tag=(upper limit+lower linit)/2 = 0.06816

(must be inside sub-interval)

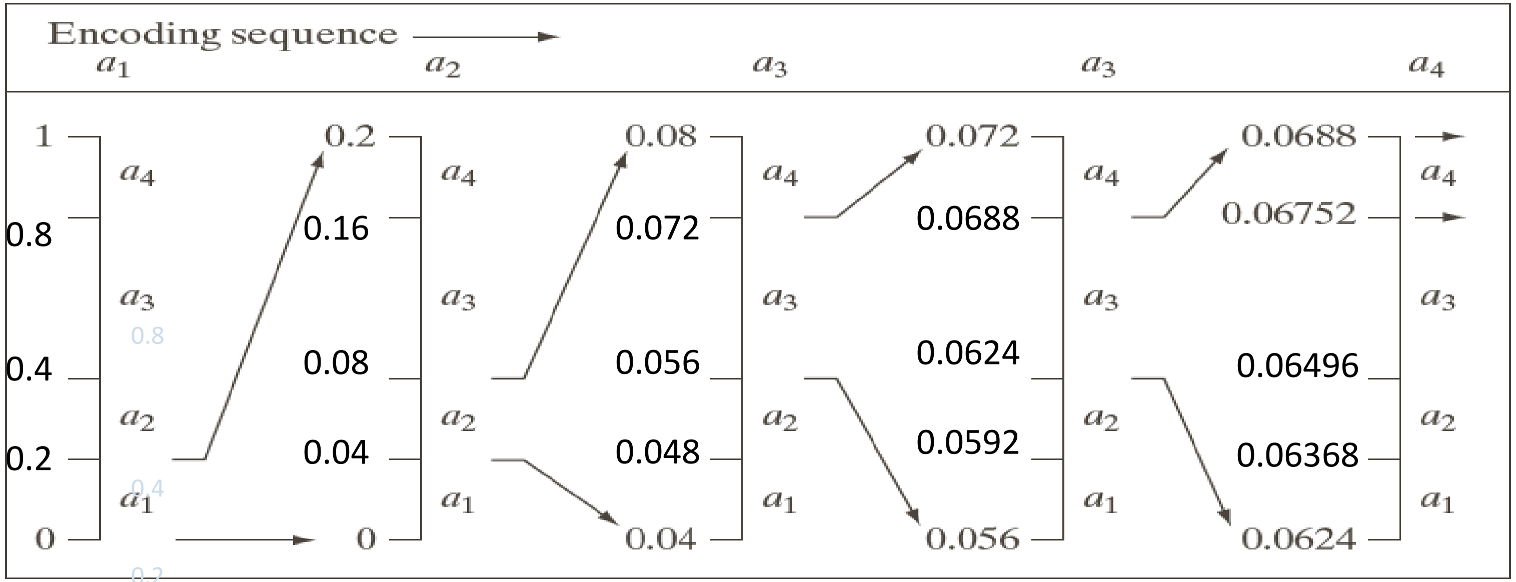


Arithmetic Decoding

Example

Decode the message 0.06816

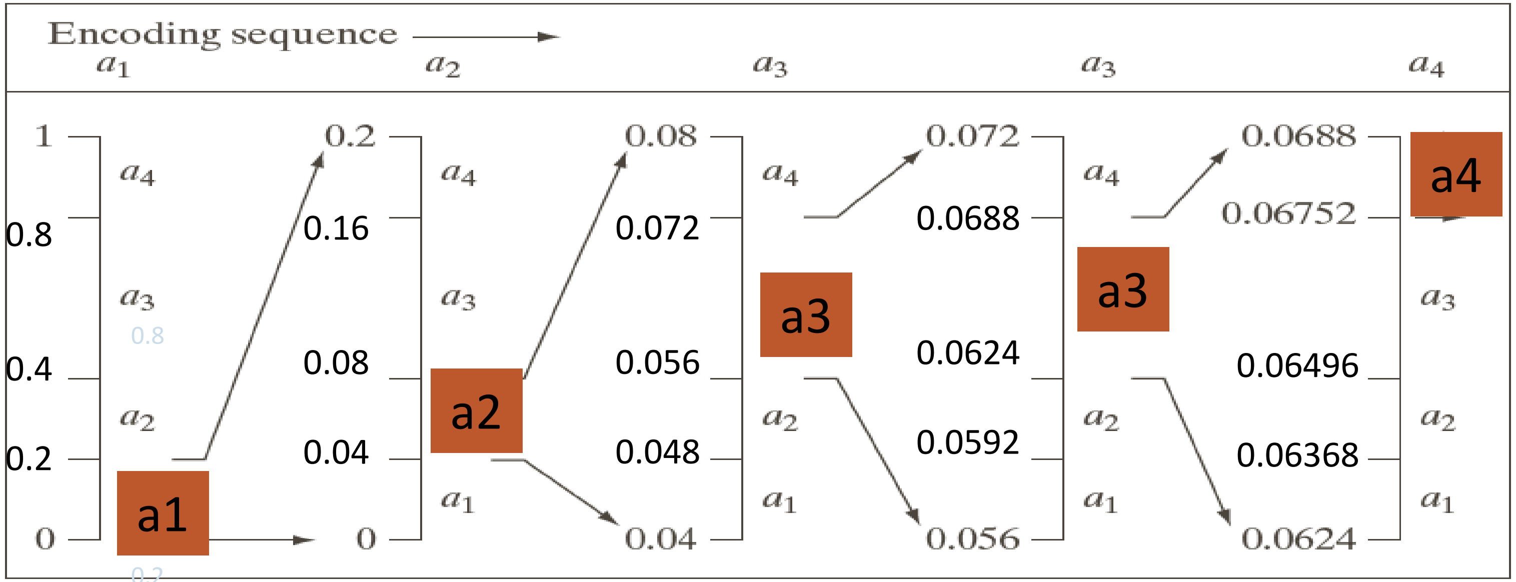
Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)



Example

Decode the message 0.06816

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)



Decoded message

$a_1a_2a_3a_3a_4$

Suggested Readings

- **Digital Image Processing by Rafel Gonzalez, Richard Woods, Pearson Education India, 2017.**

- **Fundamental of Digital image processing by A. K Jain, Pearson Education India, 2015.**

Thank you