Image Processing and Computer Vision

Prof.(Dr.) Deepali S. Jadhav
Dept. of Information Technology, VIT, Pune

RUN LENGTH CODING

- Run-length encoding (RLE) is a very simple form of data compression encoding.
- RLE is a lossless type of compression
- It is based on simple principle of encoding data. This principle is to every stream which is formed of the same data values (repeating values is called a *run*) i.e sequence of repeated data values is replaced with count number and a single value.
- This intuitive principle works best on certain data types in which sequences of repeated data values can be noticed;
- RLE is usually applied to the files that a contain large number of consecutive occurrences of the same byte pattern.

RUN LENGTH CODING

- RLE may be used on any kind of data regardless of its content, but data which is being compressed by RLE determines how good compression ratio will be achieved.
- RLE is used on text files which contains multiple spaces for indention and formatting paragraphs, tables and charts.
- Digitized signals also consist of unchanged streams so such signals can also be compressed by RLE.
- A good example of such signal are monochrome images, and questionable compression would be probably achieved if such compression was used on continous-tone (photographic) images.

RUN LENGTH CODING

- Fair compression ratio may be achieved if RLE is applied on computer generated color images.
- RLE is a lossless type of compression and cannot achieve great compression ratios,
- but a good point of that compression is that it can be easily implemented and quickly executed

Example1

- WWWWWWWWWWWWBWWWWWWWWWWWWB
BB
  WWWWWWWWWWWWWWWWWWWWWWWW
  BWWWWWWWWWWWWWW

- **If we apply a simple run-length code to the above hypothetical scan line, we get the following:**

- 12WB12W3B24WB14W

- Shift code:
- A shift code is generated by
- Arranging the source symbols so that their probabilities are monotonically decreasing
- Dividing the total number of symbols into symbol blocks of equal size.
- Coding the individual elements within all blocks identically, and
- Adding special shift-up or shift-down symbols to identify each block. Each time a shift-up or shift-down symbol is recognized at the decoder, it moves one block up or down with respect to a pre-defined reference block.

Arithmetic coding

- Unlike the variable-length codes described previously,
- *arithmetic coding*, generates non-block codes.

- In arithmetic coding, a one-to-one correspondence between source symbols and code words does not exist.

- Instead, an entire sequence of source symbols (or message) is assigned a single arithmetic code word.

- Arithmetic coding, is entropy coder widely used, the only problem is it's speed, but compression tends to be better than can achieve

Arithmetic coding

- The code word itself defines an interval of real numbers between 0 and 1
- As the number of symbols in the message increases, the interval used to represent it becomes smaller and the number of information units (say, bits) required to represent the interval becomes larger
- Each symbol of the message reduces the size of the interval in accordance with the probability of occurrence.
- It is suppose to approach the limit set by entropy.

Arithmetic coding

- The idea behind arithmetic coding is to have a probability line, 0-1
- assign to every symbol a range in this line based on its probability
- higher the probability, the higher range which assigns to it.
- Once we have defined the ranges and the probability line, start to encode symbols
- every symbol defines where the output floating point number lands

Example

Symbol	Probability	Range
a	2	[0.0 , 0.5)
b	1	[0.5 , 0.75)
c	1	[0.7.5 , 1.0)

Algorithm to compute the output number

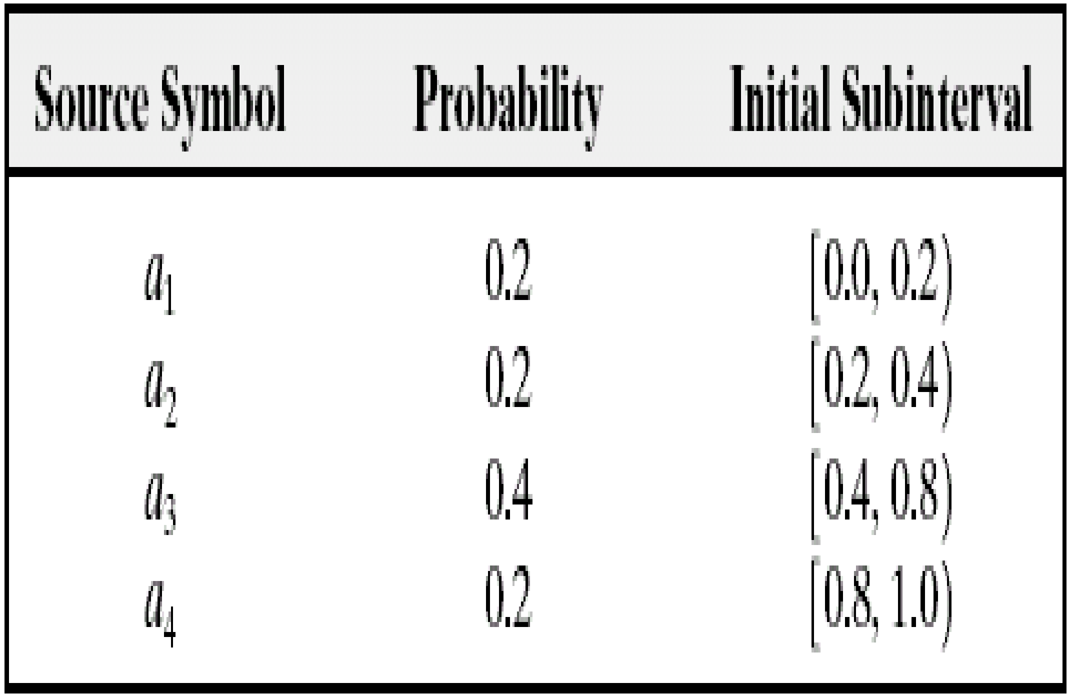
- Low = 0
- High = 1
- Loop. For all the symbols.

```
  Range = high - low
  High = low + range *  high_range of
                          the symbol being coded
  Low = low + range * low_range of the symbol
                          being coded
```

Symbol	Range	Low value	High value
		0	1
b	1	0.5	0.75
a	0.25	0.5	0.625
c	0.125	0.59375	0.625
a	0.03125	0.59375	0.609375

The output number will be 0.59375

Arithmetic coding



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	$[0.8, 1.0)$

TABLE 8.6
Arithmetic coding example.

Let the message to be encoded be $a_1a_2a_3a_3a_4$

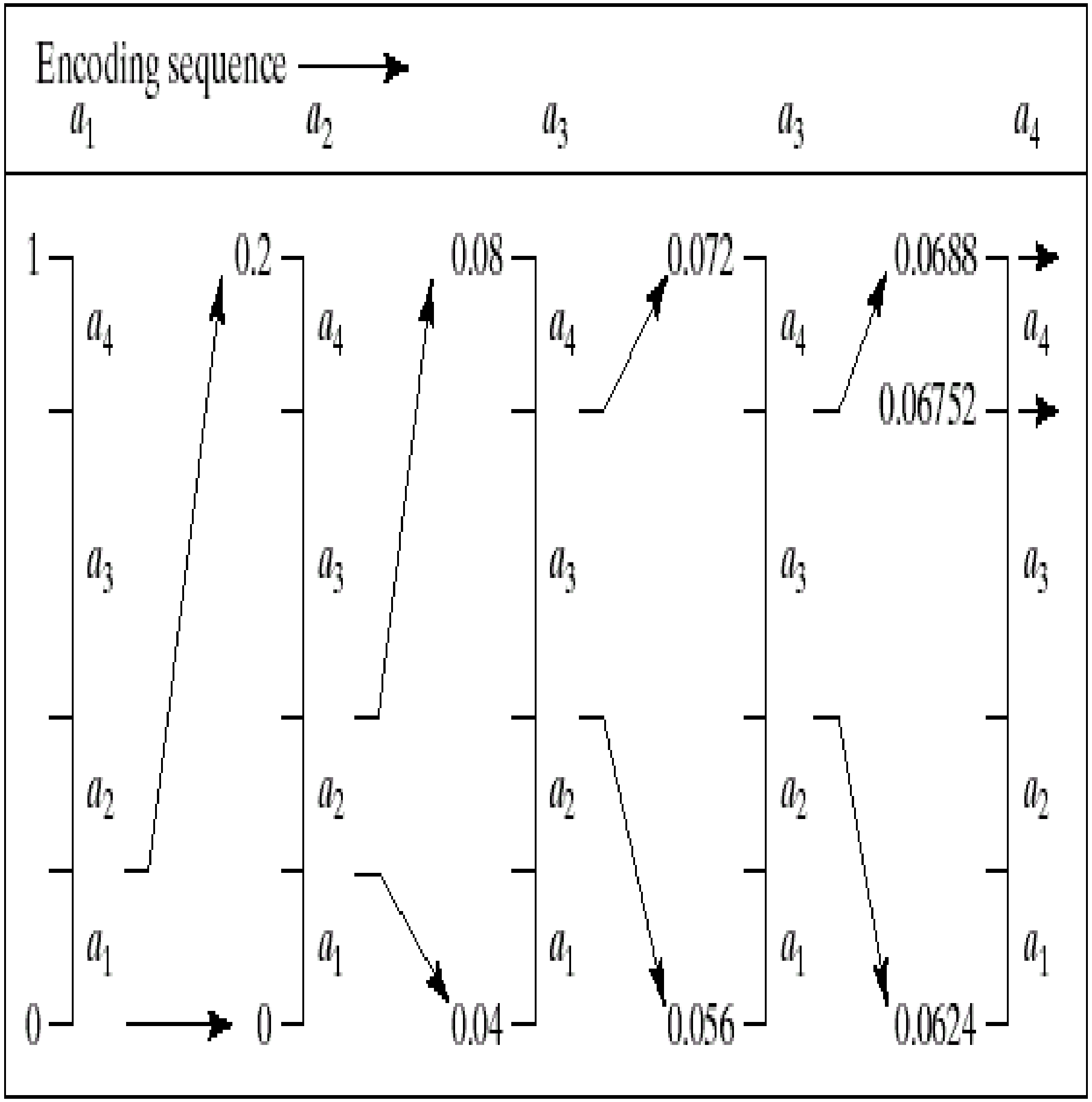
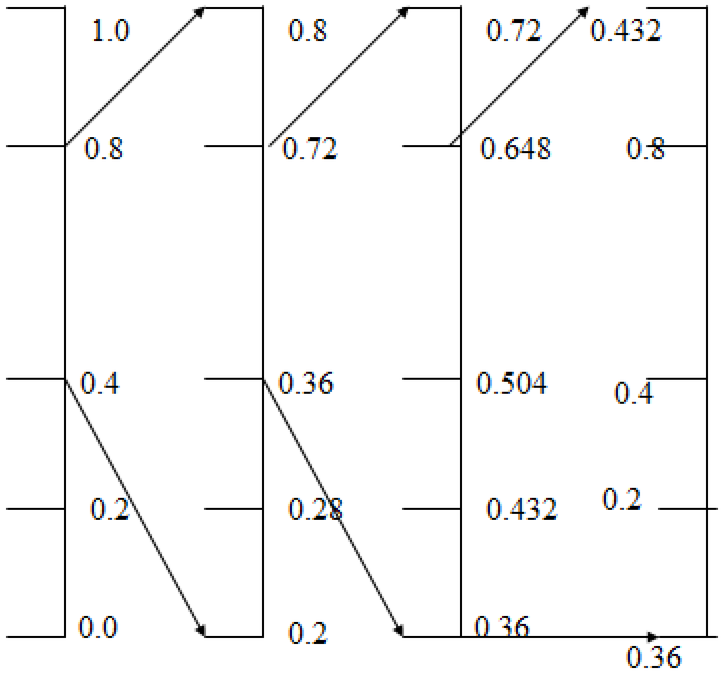


FIGURE 8.13
Arithmetic coding procedure.

So, any number in the interval [0.06752,0.0688) , for example 0.068 can be used to represent the message.

Decode 0.39.

Since 0.8>code word > 0.4, the first symbol should be a_3.


1.0
0.8
0.4
0.2
0.0
0.8
0.72
0.36
0.28
0.2
0.72
0.648
0.504
0.432
0.36
0.432
0.8
0.4
0.2
0.36