Huffman Coding

Presentation by:
Joe Louis Paul I

A Simple Example

- Suppose we have a message consisting of 5 symbols, e.g. [►♣♣♠☻►♣☼►☻]
- How can we code this message using 0/1 so the coded message will have minimum length (for transmission or saving!)
- 5 symbols → at least 3 bits
- For a simple encoding,
  length of code is 10*3=30 bits

►	000
♣	001
☻	010
♠	011
☼	100

A Simple Example – Cont…

- **Intuition:** Those symbols that are more frequent should have smaller codes, yet since their length is not the same, there must be a way of distinguishing each code
- For Huffman code, length of encoded message will be ▶♣♣♠☻▶♣☼▶☻ =3*2+3*2+2*2+3+3 =24bits

Symbol	Freq.	Code
▶	3	00
♣	3	01
☻	2	10
♠	1	110
☼	1	111

A simple example – cont.

- Intuition: Those symbols that are more frequent should have sma ler codes ,yet since their length is not the same, there must be a way of distinguishing each code

- For Huffman code,

length of encoded message will be ▶♣♣♠☻▶♣☼▶☻

=3*2 +3*2+2*2+3+3=2

Symbol	Freq.	Code
▶	3	00
♣	3	01
☻	2	10
♠	1	110
☼	1	111

Huffman Coding

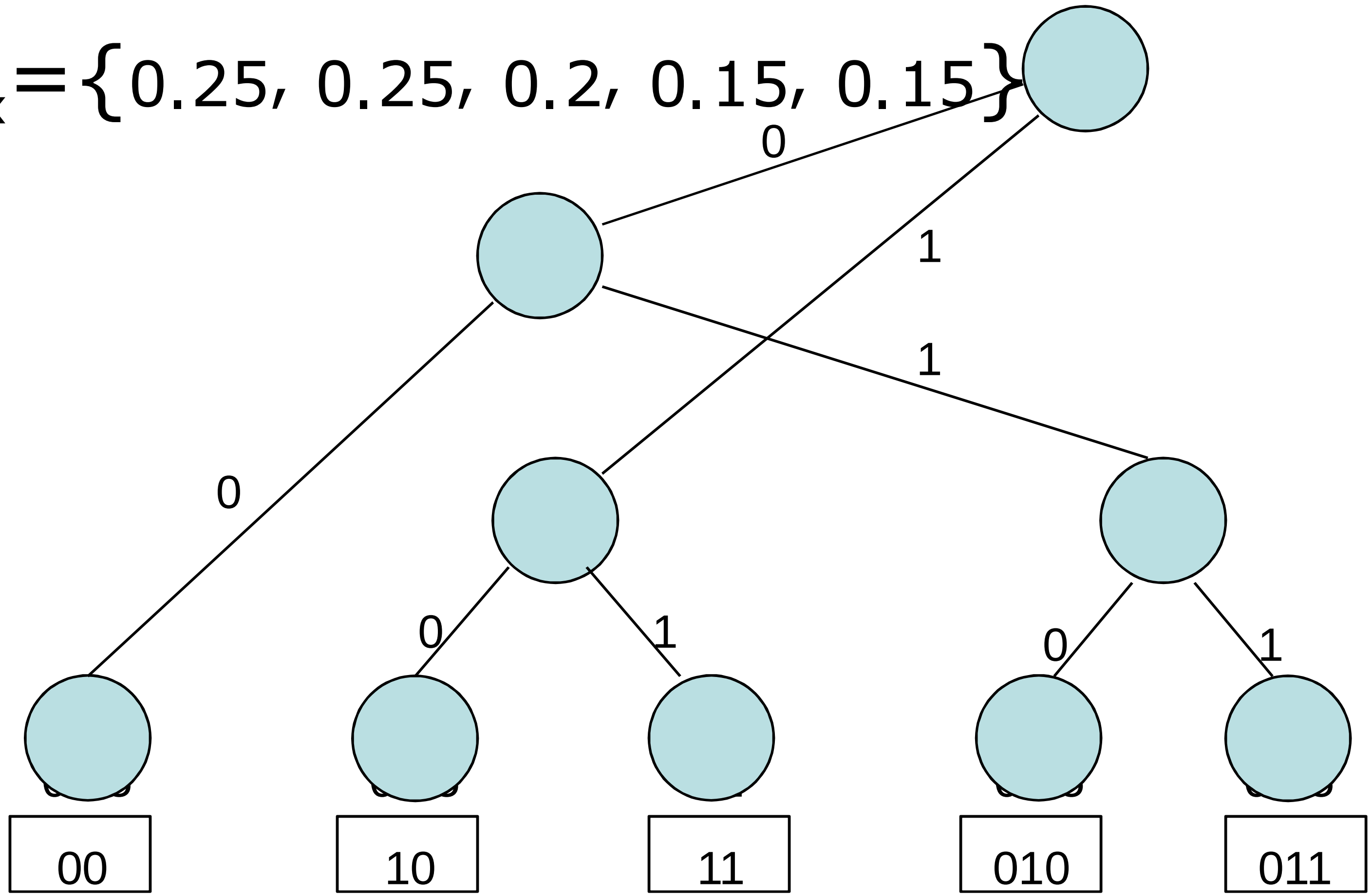
1. Take the two least probable symbols in the alphabet

   (longest codewords, equal length, differing in last digit)

2. Combine these two symbols into a single symbol, and repeat.

Example

- $A_x = \{ a, b, c, d, e \}$
- $P_x = \{0.25, 0.25, 0.2, 0.15, 0.15\}$



Codeword Table

a_i	p_i	$h(p_i)$	l_i	$c(a_i)$
a	0.25	2.0	2	00
b	0.25	2.0	2	10
c	0.2	2.3	2	11
d	0.15	2.7	3	010
e	0.15	2.7	3	011

The Purpose of Huffman Coding

- Proposed by Dr. David A. Huffman in 1952
  - *“A Method for the Construction of Minimum Redundancy Codes”*
- Applicable to many forms of data transmission.

The Huffman Coding Algorithm- History

- In 1951, David Huffman and his MIT information theory classmates given the choice of a term paper or a final exam
- Huffman hit upon the idea of using a frequency-sorted binary tree and quickly proved this method the most efficient.
- In doing so, the student outdid his professor, who had worked with information theory inventor Claude Shannon to develop a similar code.
- Huffman built the tree from the bottom up instead of from the top down.

Huffman Algorithm (1/3)

***Step 1**: Arrange the source symbols in descending order with respect to their probabilities.*

***Step 2**: Find two source symbols with least probabilities in the list and assign 0 and 1 for these symbols respectively.*

*This stage is known as **splitting stage**.*

Huffman Algorithm (2/3)

Step 3: *Combine the source symbols with least probabilities into a new symbol, where the probability for the new symbol is the sum of the probabilities of the original symbol.*

This stage is known as ***combining stage*** *or* ***combination stage***.

Step 4: *The new probability should be placed in the list, so that the new symbol always having the highest priority.*

Huffman Algorithm (3/3)

Step 5:*The above steps should be repeated recursively until the number of symbols become only 2.*

Note: *Huffman algorithm uses* ***Bottom-Up Approach***.

Huffman Tree

- The codeword for every source symbol is determined by backward tracing of 0's and 1's.

Disadvantages of the Huffman Code

- **Changing ensemble**
  - If the ensemble changes→ the frequencies and probabilities change → the optimal coding changes
  - e.g. in text compression symbol frequencies vary with context
  - Re-computing the Huffman code by running through the entire file in advance?!
  - Saving/ transmitting the code too?!

- **Does not consider ‘blocks of symbols’**
  - ‘strings_of_ch’ →the next nine symbols are predictable ‘aracters_’ , but bits are used without conveying any new information