Information Theory
Lecture 6

A. Huffman Coding Algorithm
B. Optimality of Huffman Coding
C. Shannon-Fano-Elias Coding
D. Arithmetic Coding
E. Lempel-Ziv Algorithms

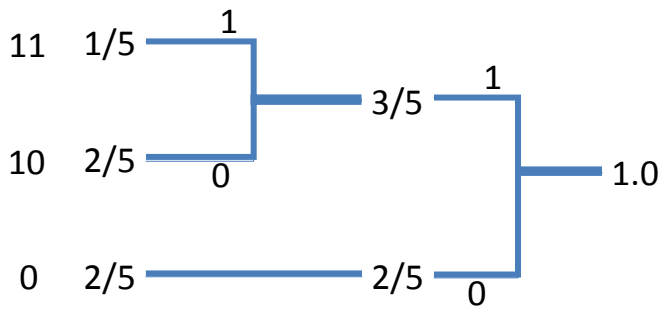
Part 6A:
Huffman Coding Algorithm

Huffman Codes

- An algorithm for constructing the most efficient instantaneous codes.
- Always achieves $L \leq H(X)+1$.
- Sometimes achieves $L = H(X)$.

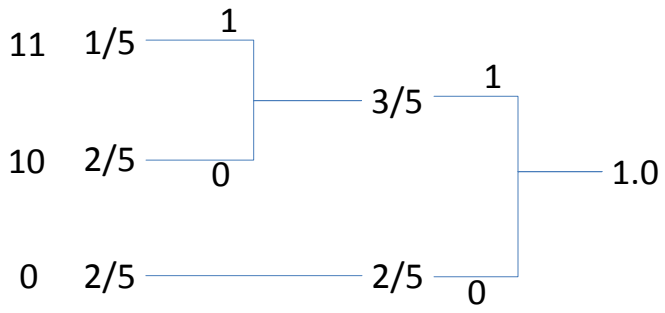
Huffman Coding Example

x	a	b	c
p(x)	1/5	2/5	2/5



Huffman Coding Example

x	a	b	c
p(x)	1/5	2/5	2/5
C(x)	11	10	0



Huffman algorithm

1. Pick the two smallest p(x)'s and draw braches merging them. Label branches with 0 and 1. Extend the other probabilities.

2. Repeat until all probability has merged into one node.

3. Read codeword symbols right to left on the Huffman tree.

Part 6B:
Optimality of Huffman Coding

Lemma 5.8.1

For any distribution, there is an instantaneous code with minimum length that satisfies the following properties:

1. $p_j > p_k \Rightarrow l_j \leq l_k$ (otherwise switch).

2. The two longest words have the same length (otherwise truncate).

3. Two of the longest codewords differ only in the last bit (Otherwise, the last bit is not necessary.) and correspond to the two least likely symbols (by 1).

Optimality of Huffman Coding

- Huffman codes are the most efficient codes that obey rules 1, 2 and 3.
- Restricting attention to codes that obey the properties, we reduce the design of an optimal m-word code to the design of an optimal m-1 word code.

Suppose we have an optimal m-word code.

$p_j < p_{j-1}$

C_m

p_1	l_1
$\vdots$	$\vdots$
p_{m-1}	l_{m-1}
p_m	l_m

$l_{m-1} = l_m$ differ only in last bit

$C_{m\text{-}1}$

p_1	l_1
p_2	l_2
$\vdots$	$\vdots$
$p'_{m-1} = p_m + p_{m-1}$	$l'_{m-1} = l_m - 1$

$$L(C_m) = L(C_{m-1}) + p_{m-1} + p_m$$

Since the average lengths differ only by a constant, minimizing $L(C_m)$ is the same as minimize $L(C_{m\text{-}1})$.

Continue combining probabilities.

- Continue merging until there are only two elements left. Then we are done since the optimal code for a two-letter alphabet is a simple one-bit code.
- Then we backtrack to construct the original C_m.
- This optimal procedure is exactly the Huffman algorithm.

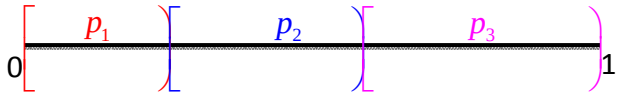
Part 6C:
Shannon-Fano-Elias Coding

Shannon-Fano-Elias Coding

- Huffman coding gets within one bit of entropy, but we may want to do better.
- Consider S-F-E which uses $l(x) = \left\lceil \log\left(\frac{1}{p(x)}\right) \right\rceil + 1.$
- This is not be as good as Huffman for a single letter code, but it leads to arithmetic coding, which can beat Huffman by not being single letter.

Assigning real estate in the unit interval.

- Assign each p_i a length equal to p_i on the (0,1) interval.



- Send a point in p_i's interval to communicate x_i.

Example

$$l(x) = \lceil -\log p(x) \rceil + 1$$

x	$p(x)$	interval	mid	binary	$l(x)$	C
a	0.6	[0, 0.6)	0.3	$0.01\overline{0011}$	2	01
b	0.3	[0.6, 0.9)	0.75	0.11	3	110
c	0.1	[0.9, 1.0)	0.95	0.11110…	5	11110

$$L(C) \le 2 \times 0.6 + 3 \times 0.3 + 5 \times 0.1 = 2.6$$

$$H(X) = 1.3$$

Truncating center points to uniquely identify each interval

- Truncate the center point of p_i's interval to $\lceil -\log p_i \rceil + 1$ bits. $L(C) \le H(X) + 2$
- The transmitted point is to the left of the center point.
- Can it be too far to the left?

Truncation doesn't make point too small.

$$\frac{1}{4}+\frac{1}{8}+\cdots=\frac{1}{2} \qquad \sum_{i=k+1}^{\infty} 2^{-i} = 2^{-k}$$

- Truncation subtracts at most

$$2^{-(\lceil -\log p_i \rceil+1)} \le 2^{-(-\log p_i+1)}$$
$$= \frac{1}{2} p_i$$

- The truncation never moves the point into another codeword's region.

Sequel bits can't make the point to large.

- Interval is p_i wide.
- The transmitted point is either the center exactly or left of center.
- The transmitted point is smallest value that could be interpreted (assumes all zeros after).
- If all the following bits are ones, the value would be the transmitted point plus

$$2^{-(\lceil -\log p_i \rceil+1)} \le 2^{-(-\log p_i+1)}$$
$$= \frac{1}{2} p_i$$

S-F-E is prefix free

- This codeword cannot be the prefix of any other codeword because all of the extensions of this codeword still lie within the region of p_i.

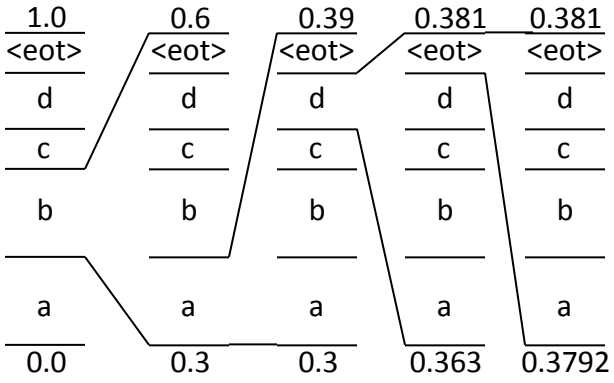
Part 6D:
Arithmetic Coding

Arithmetic coding

- Like S-F-E, we transmit a point in an interval.
- Now, one point, sent in progressively higher and higher precision, encodes the entire string.
- The scheme require an <eot> to tell the receiver when the sequence ends.

A bad<eot> example.

x	p(x)	interval	LL(x)	UL(x)
a	0.3	[0, 0.3)	0	0.3
b	0.3	[0.3, 0.6)	0.3	0.6
c	0.1	[0.6, 0.7)	0.6	0.7
d	0.2	[0.7, 0.9)	0.7	0.9
<eot>	0.1	[0.9, 1.0)	0.9	1.0



Matlab function for arithmetic encoder

```
[new_low, new_high] = encode_symbol(symbol,low,high)
range = high-low
new_low = low + range*LL(symbol)
new_high = low + range*UL(symbol)
```

Sending bits

- But, ultimately, which point in the interval do we send?
- Can we use finite precision? (Yes.)
- Whenever you know a bit, send it and rescale.
  - if (high<0.5) bit=0
    - low = low*2; high=high*2;
  - else if (low>0.5) bit =1
    - low = (low-0.5)*2; high = (high-0.5)*2;
  - else don't send a bit

Binary encoding of bad<eot>

initial interval	[0, 1)	can't send a bit
encode "b"	[0.3, 0.6)	can't send a bit
encode "a"	[0.3, 0.39)	can send a 0
send 0	[0.6, 0.78)	can send a 1
send 1	[0.2, 0.56)	can't send a bit

- After sending <eot>, when you can't send another bit (i.e. when interval contains 0.5), send a final 1 to guarantee <eot> will be received.
- bad<eot> → .011000011 (0.380859375)

Arithmetic decoding

- We decode one symbol at a time.
- [symbol,new_value]=decode_symbol[value]
- Find the symbol from the interval containing the value.
- Set the new value based on its position in the interval.

Decoding example

- Set low and high equal to that intervals low and high.
- new_value = (value - low)/(high - low)
- value = 0.38085795
- symbol = b
- new_value = (0.38085795-0.3)/(0.6-0.3)
  = 0.26953125

Part 6E: Lempel-Ziv Algorithms

Lempel-Ziv Algorithm

- Encoder does not have to explicitly know the source statistics to compress close to the entropy.
- Ziv & Lempel produced two algorithms, one in 1977 and one in 1978 that compress to the entropy asymptotically.

LZ-77

- Dictionary consists of previous text.
- Compress to triples <pointer, length, new_character>
- a b d c a d d c a d d c d a b c
  previous text new text

  <4, 5, d>
- a b d c a d d c a d d c d a b c

  <1, 2, c>

LZ-78

- Dictionary is derived from previous text.
- Initial dictionary is empty except for a null pointer.
- Compress to pairs <pointer, new_character>
- Each new string is added to the dictionary.

LZ-78 Example

aaabbabaabaaabab

a aa b ba baa baaa bab

Compressed file

1. <0, a>
2. <1, a>
3. <0, b>
4. <3, a>
5. <4, a>
6. <5, a>
7. <4, b>

Dictionary

0. nothing
1. a
2. aa
3. b
4. ba
5. baa
6. baa
   a
7. bab

Lempel-Ziv-Welch modification of LZ-78

- Initial dictionary contains all the single letters.
- Groups of letters are compressed as a pointer to a matching string already in the dictionary.
- New dictionary entries are formed from each received string plus first letter of next received string.

LZW Encoding Example

aababababaaa

Compressed file	Dictionary
1. 1	1. a 2. b 3. a…

LZW Encoding Example

a,abababaaa

Compressed file

1. 1
2. 1

Dictionary

1. a
2. b
3. aa
4. a…

LZW Encoding Example

a,a,bababaaa

Compressed file

1. 1
2. 1
3. 2

Dictionary

1. a
2. b
3. aa
4. ab
5. b…

LZW Encoding Example

a,a,b,ababaaa

Compressed file

1. 1
2. 1
3. 2
4. 4

Dictionary

1. a
2. b
3. aa
4. ab
5. ba
6. ab…

LZW Encoding Example

a,a,b,ab,abaaa

Compressed file

1. 1
2. 1
3. 2
4. 4
5. 6

Dictionary

1. a
2. b
3. aa
4. ab
5. ba
6. aba
7. aba…

LZW Encoding Example

a,a,b,ab,aba,aa

Compressed file

1. 1
2. 1
3. 2
4. 4
5. 6
6. 3

Dictionary

1. a
2. b
3. aa
4. ab
5. ba
6. aba
7. abaa

LZW Decoding Example

a,

Compressed file

1. 1
2. 1

Dictionary

1. a
2. b
3. a…

LZW Decoding Example

a,a,

Compressed file

1. 1
2. 1
3. 2

Dictionary

1. a
2. b
3. aa
4. a…

LZW Decoding Example

a,a,b,

Compressed file

1. 1
2. 1
3. 2
4. 4

Dictionary

1. a
2. b
3. aa
4. ab
5. b…

LZW Decoding Example

a,a,b,ab,

Compressed file

1. 1
2. 1
3. 2
4. 4
5. 6

Dictionary

1. a
2. b
3. aa
4. ab
5. ba
6. ab…

LZW Decoding Example

a,a,b,ab,aba

Compressed file

1. 1
2. 1
3. 2
4. 4
5. 6
6. 3

Dictionary

1. a
2. b
3. aa
4. ab
5. ba
6. aba
7. aba…

LZW Decoding Example

a,a,b,ab,aba,aa

Compressed file

1. 1
2. 1
3. 2
4. 4
5. 6
6. 3

Dictionary

1. a
2. b
3. aa
4. ab
5. ba
6. aba
7. abaa