Probability

- The most common way that people think about probability is in term of *outcomes*, or *set of outcomes*, of an experiment
- Suppose we have an experiment that has N possible outcomes and we conduct the experiment n_T times
- If the outcome w_i occurs n_i times,
  - we say that the *relative frequency of occurrence* of the outcome w_i is

$$\frac{n_i}{n_T}$$

  - we can then define the *probability of occurrence* of the outcome w_i as

$$P(w_i) = \lim_{n_T \to \infty} \frac{n_i}{n_T}$$

Probability

- In practice, we do not have the ability to conduct an experiment an infinite number of times and hence the probability of occurrence can not be calculated
- Instead of calculating the probability of occurrence as defined before, we often use the *relative frequency of occurrence* as *an approximation or an estimate* to the *probability of occurrence*

Probability

Example 1:

- Suppose that we turn on a television 1,000,000 times
  - 200,000 times the television was turned on during a commercial and
  - 800,000 times the television was turned on during a noncommercial
- Our experiment here is *turning on a television set*
- The outcomes are commercial or noncommercial
- We could say that the *relative frequency of occurrence*, or the *estimate of the probability of occurrence*, of
  - turning on a television set in the middle of a commercial is 0.2
  - turning on a television set in a noncommercial is 0.8

Topic 03: Codeword Encoding--Huffman Encoding

Probability

Example 2:

- Suppose that we turn on a television 10 times
  - 2 times the television was turned on during a commercial and
  - 8 times the television was turned on during a noncommercial
- We could say that the *relative frequency of occurrence*, or the *estimate of the probability of occurrence*, of
  - turning on a television set in the middle of a commercial is 0.2
  - turning on a television set in a noncommercial is 0.8
- What is the difference between the two examples?

Probability

- Each example gives an *estimate of the probability of occurrence*
- The *relatively larger* the number of experiment repetition, the better the probability of occurrence approximation, or estimation, is

Measuring Information

- Suppose that we have an event A
- If P_A is the probability that the event A will occur, then the *self-information* associated with A is given by

$$i(A) = \log_2(1/P_A) = -\log_2(P_A) \text{ bits}$$

- If the probability of an event is *low*, the amount of self-information associated with it is *high*
- If the probability of an event is *high*, the amount of self-information associated with it is *low*

Measuring Information

- Consider a process which produces either A or B output (only one of them occurs at a time)
  - The probability of A to be produced = P_A
  - The probability of B to be produced = P_B
- The amount of information you will get
  - When seeing A produced = $-\log_2(P_A)$ bits
  - When seeing B produced = $-\log_2(P_B)$ bits
- The average information you will get per event is:

$$-P_A \times \log_2(P_A) - P_B \times \log_2(P_B) \text{ bits per output}$$

Topic 03: Codeword Encoding--Huffman Encoding

Measuring Information

Example 1: If $P_A = 1/2$ and $P_B = 1/2$

- The amount of information you will get
  - When seeing A produced = $-\log_2(1/2) = 1$ bits
  - When seeing B produced = $-\log_2(1/2) = 1$ bits
- On average, you will get

$$1/2 \times 1 + 1/2 \times 1 = 1 \text{ bits per output}$$

Measuring Information

Example 2: If $P_A = 1/8$ and $P_B = 7/8$

- The amount of information you will get
  - When seeing A produced = $-\log_2(1/8) = 3$ bits
  - When seeing B produced = $-\log_2(7/8) = 0.19$ bits

- On average, you will get

$$1/8 \times 3 + 7/8 \times 0.19 = 0.54 \text{ bits per output}$$

Measuring Information

Example 3:

What will happen if P_A is very small and P_B is very large?

Measuring Information

- Consider a process which produces either A, B, C, or D output (only one of them occurs at a time)
  - The probability of A to be produced = P_A
  - The probability of B to be produced = P_B
  - The probability of C to be produced = P_C
  - The probability of D to be produced = P_D
- The amount of information you will get
  - When seeing A produced = $-\log_2(P_A)$ bits
  - When seeing B produced = $-\log_2(P_B)$ bits
  - When seeing C produced = $-\log_2(P_C)$ bits
  - When seeing D produced = $-\log_2(P_D)$ bits
- The average information you will get per event is:
  $-P_A \times \log_2(P_A) - P_B \times \log_2(P_B) - P_C \times \log_2(P_C) - P_D \times \log_2(P_D)$ bits per output

Topic 03: Codeword Encoding--Huffman Encoding

Measuring Information

Example 1: If $P_A = 1/4$, $P_B = 1/4$, $P_C = 1/4$, and $P_D = 1/4$

- The amount of information you will get
  - When seeing A produced = $-\log_2(1/4) = 2$ bits
  - When seeing B produced = $-\log_2(1/4) = 2$ bits
  - When seeing C produced = $-\log_2(1/4) = 2$ bits
  - When seeing D produced = $-\log_2(1/4) = 2$ bits
- On average, you will get

  $1/4 \times 2 + 1/4 \times 2 + 1/4 \times 2 + 1/4 \times 2 = 2$ bits per output

Measuring Information

Example 1: If $P_A = 1/8$, $P_B = 1/8$, $P_C = 1/4$, and $P_D = 1/2$

- The amount of information you will get
  - When seeing A produced = $-\log_2(1/8) = 3$ bits
  - When seeing B produced = $-\log_2(1/8) = 3$ bits
  - When seeing C produced = $-\log_2(1/4) = 2$ bits
  - When seeing D produced = $-\log_2(1/2) = 1$ bits

- On average, you will get

  $1/8\times3 + 1/8\times3 + 1/4\times2 + 1/2\times1 = 1.75$ bits per output

Measuring Information

- The quantity

$$-\sum_{i=1}^{L} P_i \times \log_2(P_i)$$

- is called the *entropy*, or the *uncertainty*, associated with the experiment, or the event
- ***Entropy* is** *the amount of information, on average, that a certain experiment, or event, can deliver*

- **It is also the *theoretical lower bound* for the *required bit rate for codeword encoding***

Measuring Information

- It is proved (see the information theory if you wish) that the maximum entropy (uncertainty) occurs when all the outputs has the same equal probability

- Applying this principle to Huffman encoder, you can see that the maximum bit rate to encode data occurs when all possible data elements have the same probability (equiprobable outputs)

Measuring Information

- When probability values are equal 2^{-n}, where n is a positive integer number, Huffman encoder produces an optimal codeword assignment, i.e, the average bits per symbol = entropy
- If probability values are not equal 2^{-n} and each output is encoded separately, it is not possible to design a code with average bit rate equals the entropy
- In this case, Huffman will generate a near optimal codeword assignment (i.e., close to the entropy)

- In the first Huffman example (slide # 32), the average number of bits per value was 1.813 bits, while the entropy was 1.7516 bits
- In the second and third Huffman examples (slides #55 and #59), the average number of bits per value is equal to the entropy

Another Huffman Example

- Consider a process which produces A, B, or C output, one at a time
  - The probability of A to be produced = $P_A = 0.95$
  - The probability of B to be produced = $P_B = 0.02$
  - The probability of C to be produced = $P_C = 0.03$
- The entropy = $-0.95\times\log_2(0.95) - 0.02\times\log_2(0.02) - 0.03\times\log_2(0.03) = 0.335$ bits/symbol
- A Huffman code would be:
  - A: 0
  - B: 10
  - C: 11
- The average length for this code is: $0.95 \times 1 + 0.02 \times 2 + 0.03 \times 2 = 1.05$ bits/symbol

Topic 03: Codeword Encoding--Huffman Encoding

Another Huffman Example

- The difference between the average code length and the entropy is called *code redundancy*
- In this example, the *code redundancy* is 0.715 bits/symbol, i.e., 213% of the entropy!!!
- Is there a way to improve this situation?

String Versus Individual Symbols Encoding

- Encoding each symbol individually needs, at least, 1 bit per symbol
- In the previous Huffman example, consider generating all strings of two symbols, i.e., AA, AB, AC, BA, BB, BC, CA, CB, and CC
- The associated probability for these symbols are:
  P_AP_A=0.9025, P_AP_B=0.0190, P_AP_C=0.0285,
  P_BP_A=0.0190, P_BP_B=0.0004, P_BP_C=0.0006,
  P_CP_A=0.0285, P_CP_B=0.0006, P_CP_C=0.0009
- Using Huffman scheme, the following codewords 0, 111, 100, 1101, 110011, 110001, 101, 110010, 110000, may be assigned to AA, AB, AC, BA, BB, BC, CA, CB, and CC, respectively
- In this case, the average length for each string of two symbols is 1.222 bits/symbol
- Hence, each symbol needs 0.611 bits/symbol, in average

Topic 03: Codeword Encoding--Huffman Encoding

String Versus Individual Symbols Encoding

- As more symbols are combined, codewords become longer, and the average bits/symbol gets better
- In brief, dealing with strings (combined symbols), rather than individual symbols, can, in principle, yield better bits/symbol results
- How about complexity?

String Versus Individual Symbols Encoding

- When combining more symbols together, the number of *all possible combinations* dramatically increases:
  - Each individual symbol by itself → 3 possible strings
  - All strings of 2 symbols together → 9 possible strings
  - All strings of 3 symbols together → 27 possible strings
  - All strings of 4 symbols together → 81 possible strings
  - All strings of 5 symbols together → 243 possible strings
  - All strings of 6 symbols together → 729 possible strings
  - All strings of 7 symbols together → 2187 possible strings
  - All strings of 8 symbols together → 6561 possible strings

String Versus Individual Symbols Encoding

- Even though combining more symbols yield smaller average bits/symbol, they may not be practical:
  - Storing a large code requires a huge amount of memory
  - Decoding a Huffman code of this large size would be a highly inefficient and time-consuming procedure
- To overcome this problem, we need a way of assigning codewords to particular strings without having to generate codes for ***all*** strings of similar length
- The *arithmetic* encoding scheme fulfills this requirement

Huffman Encoding

- So far, we assumed that the probability of each symbol is provided
- What if we do not have these probabilities?
  - For simplicity, can we assume that all symbols will have same probability?

Topic 03: Codeword Encoding--Huffman Encoding

Huffman Encoding

- Assuming that all symbols will have same probability will ***not*** lead us to ***any*** compression
- Using a prior knowledge about probability might achieve a small amount of compression, based on how good the prior knowledge is
- One solution to this issue is to make Huffman a two-pass procedure:
  - The first pass to collect statistics
  - The second pass to encode
- The other solution is to utilize an adaptive procedure to construct Huffman in one-pass