Mathematical Background on Lossless Data Compression

National Chiao Tung University
Chun-Jen Tsai
9/22/2014

Measuring Information Amount

- Shannon defines a quantity, self-information, of an event A with probability $P(A)$ as follows:

$$i(A) = \log_b \frac{1}{P(A)} = -\log_b P(A).$$

- Note that:
  - $i(A) = 0$ for $P(A) = 1$ (this event is predictable)
  - $i(A) \geq 0$ for $0 \leq P(A) \leq 1$ (well, this is debatable)
  - $i(A) > i(B)$ for $P(A) < P(B)$
  - $i(AB) = i(A) + i(B)$ if A and B are independent events
- Counter-example of Shannon's idea:
  - A random string of letters versus a meaningful statement

† If the base $b = 2$, the unit of self-information is bits.

Flipping of a Fair Coin

- Let H and T be the outcomes of flipping a coin, if

$$P(H) = P(T) = 0.5$$

then,

$$i(H) = i(T) = 1 \text{ bit.}$$

$\rightarrow$ Shannon used this case (an uniform distribution with binary outcome) as a basis of defining the unit of self-information.

Entropy of Random Events

- If we have a set of independent events A_i, S is the sample space of all events, then the average self-information is given by

$$H = \sum P(A_i) i(A_i) = -\sum P(A_i) \log_b P(A_i).$$

  This quantity is called the entropy **associated with the experiment**.

- Given a data source S, its entropy is the *minimal* average number of bits one must use to describe an output symbol of the source

Entropy of a Data Source

- For a general source S with alphabet $A = \{1, 2, \ldots, m\}$ that generates a sequence $\{X_1, X_2, \ldots\}$, the entropy is given by

$$H(\mathrm{S}) = \lim_{n\to\infty} \frac{1}{n} G_n,$$

where

$$G_n = -\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m} P(X_1 = i_1, \cdots, X_n = i_n) \log P(X_1 = i_1, \cdots, X_n = i_n).$$

- Question: what happens if the output is i.i.d.?

Estimation of Source Entropy (1/2)

- It is in general not possible to know the entropy of a physical source, for example

$$S \to 1\ \ 2\ \ 3\ \ 2\ \ 3\ \ 4\ \ 5\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10 \ldots$$

But, can we estimate it?

† Hint: a "reasonable" estimate of the entropy of S is 3.25 bits.

Estimation of Source Entropy (2/2)

- Given the same source S, if we compute the difference between neighboring samples, we have the **residual** sequence:

$$R \rightarrow 1 \;\; 1 \;\; 1 \;\; -1 \;\; 1 \;\; 1 \;\; 1 \;\; -1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \; . \; . \; .$$

  The sequence has only two symbols, estimated entropy is only 0.70 bits

  Question: Are the two sources S and R equivalent?

Is the Entropy of a Source Constant?

- Given a source S, can you perform some invertible manipulations to the output of S so that the entropy of S is reduced?
  - Information theory tells you no!
  - But what just happened in previous example?

- Let's try it again! What is the entropy of the following source?

$$S \rightarrow 1\ 2\ 1\ 2\ 3\ 3\ 3\ 3\ 1\ 2\ 3\ 3\ 3\ 3\ 1\ 2\ 3\ 3\ 1\ 2$$

Why Do We Use Log Function?

- Given a set of independent events $A_1, A_2, \ldots, A_n$ with probability $p_i = P(A_i)$, we want the definition of the information measure $H()$ to satisfy:

  1. Small change in p_i cause small change in the measure (i.e., $H()$ is a continuous function of p_i)

  2. If $p_i = 1/n$, for all i, then the measure $H()$ should be a monotonically increasing function of n

  3. Partition the outcome of a source into several groups shall not change its total entropy

Example: Partition a Source

- Assume that $S = \{A_1, A_2, A_3\}$ and $P(A_i) = p_i$. If we partition S into two different sources S_1 and S_2, where $S_1 = \{A_1\}$ and $S_2 = \{A_2, A_3\}$. Then, we have:

$$H(p_1, p_2, p_3) =$$

$$H(P(S_1), P(S_2)) + P(S_1)H\left(\frac{p_1}{P(S_1)}\right) + P(S_2)H\left(\frac{p_2}{P(S_2)}, \frac{p_3}{P(S_2)}\right).$$

Note that $P(S_1) = p_1$ and $P(S_2) = p_2 + p_3$.

Shannon's Great Contribution

- Shannon showed that the only way all these conditions could be satisfied was if

$$H = -K \sum p_i \log p_i,$$

where K is an arbitrary positive constant.

- Conclusion
  - if you agree that the probability of an event is the exclusive factor of the amount of information it carries, then the $\log$ function is the only way to define the information measure!

Models for Coding

- Physical Models
  - Vocal cord model for speech coding
  - Head and shoulder model for video coding
- I.I.D. Probability Models
  - For source alphabet $A = \{a_1, a_2, \ldots, a_M\}$, we can have a probability model $\mathbf{P} = \{P(a_1), P(a_2), \ldots, P(a_M)\}$ if we can assume that the symbols coming from the source are independent to each others.
- Markov Models
  - The past always changes the future;
    But, how can we mathematically describe such influences?

Markov Models

- A sequence $\{x_n\}$ fits a kth-order Markov model if

$$P(x_n \mid x_{n-1}, \ldots, x_{n-k}) = P(x_n \mid x_{n-1}, \ldots, x_{n-k}, \ldots).$$

  $\rightarrow$ Probability of the next symbols can be determined completely by knowing the past k symbols.

- Each sequence of x_{n-1}, …, x_{n-k} is called a state; if the alphabet set has size m, the number of states is m^k.

- First-order Markov model is most commonly used.

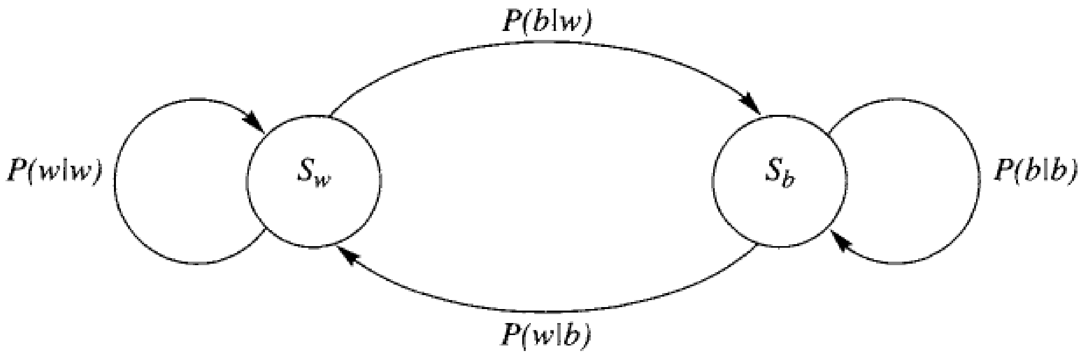
Markov Source Model

- The sequence generated by a linear 1st-order Markov model can be expressed by

$$x_n = \rho x_{n-1} + \varepsilon_n,$$

where ε_n is a white noise.

- Markov model can also be described using a state transition diagram, e.g. a two-state Markov model:



Entropy of Finite State Process

- The entropy of a finite state process with state S_i can be computed by:

$$H = \sum_{i=1}^{M} P(S_i) H(S_i),$$

where $H(S_i)$ is the entropy of a state S_i.

- For example, for the two-state Markov model:

$$H(S_w) = -P(b \mid w) \log P(b \mid w) - P(w \mid w) \log P(w \mid w),$$

where $P(w \mid w) = 1 - P(b \mid w)$.

S_w is treated as a data source that outputs a black or a white pixel in next time instance. The entropy of S_w, $H(S_w)$, is computed using probabilities leaving state S_w (for the generation of the next pixel).

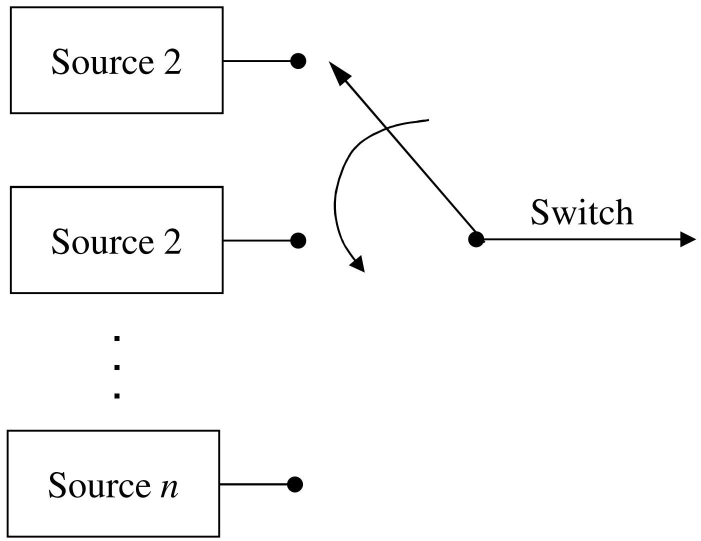
Example: I.I.D. vs. Markov Model

- For the two-state model, assume that
  $P(S_w) = 30/31$, $P(S_b) = 1/31$,
  $P(w \mid w) = 0.99$, $P(b \mid w) = 0.01$,
  $P(b \mid b) = 0.7$, $P(w \mid b) = 0.3$.

  What is the entropy of the source given i.i.d. model?
  How about Markov model?

Composite Source Model

- A composite source model can be described by n different sources and the probability P_i to select ith source:



Definitions of Coding

- Coding is the process of assigning binary sequences to an alphabet.

  For example:

$$\underbrace{1000011}_{\text{codeword}} \rightarrow \underbrace{a}_{\text{alphabet, symbol}}$$

- The set of all codewords is called a code. The average number of bits per symbol is called the rate of the code.

Uniquely Decodable Codes

- Given any messages represented using a code, if there is only one way to reconstruct the messages in the original alphabet, the code is uniquely decodable

- Example: $A = \{a_1, a_2, a_3, a_4\}$.

Letters	Probability	Code 1	Code 2	Code 3	Code 4
a_1	0.5	0	0	0	0
a_2	0.25	0	1	10	01
a_3	0.125	1	00	110	011
a_4	0.125	10	11	111	0111
Average length		1.125	1.25	1.75	1.875

Instantaneous Code

- If a codeword can be identified before we see the next codeword, the code is called instantaneous code.

- Example: can we decode the following message?

01111111111111111

Letter	Codeword
a_1	0
a_2	01
a_3	11

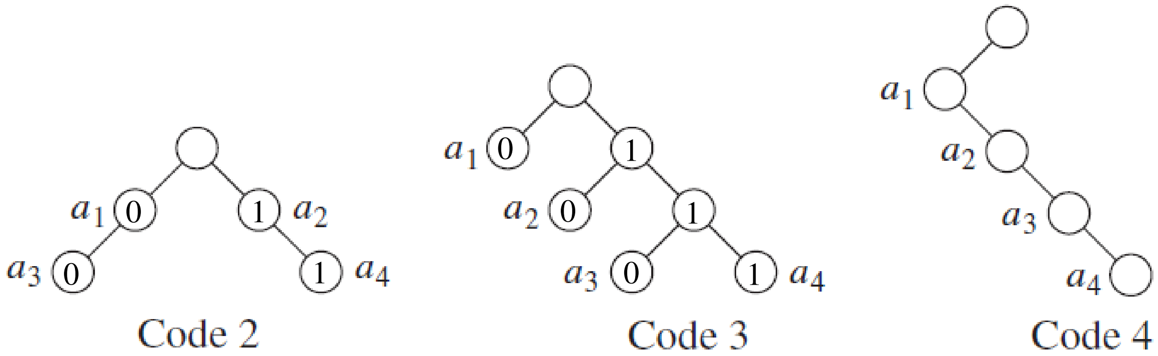
A Test for Unique Decodability

- *Definitions*
  - Prefix: if the beginning subsequence of codeword b is equal to codeword a, then a is a prefix
  - Dangling suffix: if a is a prefix of b, then the subsequence of b excluding the prefix a is called a dangling suffix
- *Procedure:*
  - Check if any codeword is a prefix of another codeword, if so, add the dangling suffix to the codeword list unless it has already been added in a previous iteration.
  - Repeat the procedure until:
    (1) you get a dangling suffix that is a codeword,
    (2) there are no more unique dangling suffixes.

    If you get (1), the code is not uniquely decodable.

Prefix Code

- If no codeword is a prefix of the others, the code is definitely uniquely decodable. In this case, we call the code a **prefix code**.
- A code tree is a binary tree where each codeword of a code corresponds to a node in the binary tree.



A prefix code has all codewords on leaf nodes.

Kraft-McMillan Inequality

- Let C be a code with N codewords with lengths l_1, l_2, …, l_N. If C is uniquely decodable, then

$$K(C) = \sum_{i=1}^{N} 2^{-l_i} \leq 1.$$

Key points of the proof:

1) If $[K(C)]^n$ does not grow exponentially, $K(C) \leq 1$.
2) There can be at most 2^k different *decodable messages of* length k (i.e., $l_{i1} + l_{i2} + \ldots + l_{in} = k$). If A_k is the number of possible messages of length k of this code, $A_k \leq 2^k$.
3) $[K(C)]^n = \Sigma_{k=n..nl} A_k 2^{-k} \leq n(l-1)+1$, where l is the max codeword length and Thus, $[K(C)]^n$ does not grow exponentially.

Efficiency of Prefix Code

- Given a set of integers $l_1, l_2, \ldots, l_N$ that satisfy the inequality,

$$\sum_{i=1}^{N} 2^{-l_i} \leq 1.$$

we can always find a prefix code with codeword lengths $l_1, l_2, \ldots, l_N$.

Key point of the proof:
Use binary representation of $\Sigma_{i=1..j-1} 2^{-l_i}$ as the codeword prefix for l_j, $j > 1$, and 0 as the codeword prefix for l_1. The trailing bits of codeword are all zeros.

Algorithmic Information Theory

- Kolmogorov complexity $K(x)$ of a sequence x is the size of the program, including all the required input data, needed to generate x.
  - Bad news: no systematic way of computing or approximating $K(x)$.

- Worse case scenario of $K(x)$: the size of the data sequence plus the size of a small program that output a sequence.

  But, *what is the lower bound?*

Minimum Description Length Principle

- J. Risannen in 1978 argued that:

  Let M_j be a model from a set of models M that attempt to characterize the structure in a sequence x. Let D_{M_j} be the number of bits required to describe the model M_j, $R_{\mathsf{M}_j}(x)$ be the number of bits required to represent x w.r.t. the model M_j. The minimum description length would be given by

$$\min_j (D_{\mathsf{M}_j} + R_{\mathsf{M}_j}(x)).$$

Example: MDL Principle

- ❑ If we use kth-order polynomial to model a set of data,
  - High-order model: accurate, hard to describe
  - Low-order model: less accurate, easy to describe
- ❑ Question: which one do you prefer?