Aalto University School of Electrical Engineering
Department of Communications and Networking

EXAM
12.12.2013

S-72.2410 Information Theory

1. (6p.) Entropy. Consider the board in Figure 1 with squares numbered $0, 1, \ldots$. You start in square 0 and flip a coin to decide whether to advance or not. With heads you advance one square to the right and with tails you don't move. Let Y_1, Y_2, and Y_3 be your position after one, two, and three flips, respectively.

   (a) Determine $H(Y_1, Y_2)$.

   (b) Determine $H(Y_3|Y_2)$.

   (c) Determine $I(Y_1; Y_3)$.

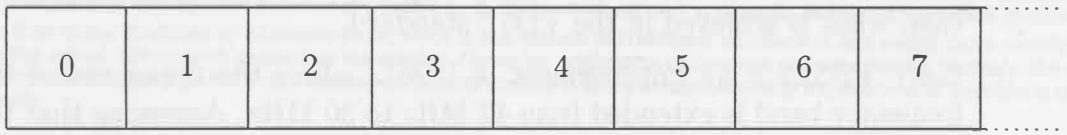


Figure 1: The board

2. (12p.) Source coding. In addition to the source coding methods considered in the course, many others have been proposed. One such method is Shannon–Fano coding, described in an attached Appendix. In the sequel, assume a source with alphabet $\{A, B, C, D, E, F, G\}$ and probabilities $p(A) = 0.09$, $p(B) = 0.12$, $p(C) = 0.13$, $p(D) = 0.14$, $p(E) = 0.15$, $p(F) = 0.16$, and $p(G) = 0.21$.

   (a) (2p.) What is the entropy of the source?

   (b) (4p.) Use Shannon–Fano coding to obtain a binary source code. What is the expected codeword length?

   (c) (3p.) Use one of the methods discussed in the course—Shannon coding *or* Huffman coding—to obtain a binary source code. What is the expected codeword length?

   (d) (1p.) If one outcome of the source is encoded at a time, what is the smallest possible value for the expected codeword length of a binary code?

   (e) (2p.) What is the smallest possible value for the length of an encoded binary sequence per outcome of the source, when encoding a long sequence of outcomes of the source with any method? In other words, if n outcomes of the source are encoded into a binary sequence of length αn, how small can α be? Moreover, how can this be achieved?

CONTINUES ON THE NEXT PAGE

3. (6p.) Channel capacity. Very-high-bit-rate digital subscriber line (VDSL) is a technology providing data transmission faster than ADSL over copper wires and coaxial cables. VDSL uses the frequency band covering all frequencies from 25 kHz to 12 MHz. VDSL supports rates of up to 52 Mbit/s downstream and 16 Mbit/s upstream for copper wires and rates of up to 85 Mbit/s downstream as well as upstream for coaxial cables. (downstream = direction from Internet to user; upstream = direction from home user to Internet) We assume short distances so that issues such as attenuation can be ignored.

   (a) (2p.) What signal-to-noise ratio (in dB) is *at least* required to make the rates of VDSL possible? Answer the question **both for copper wires and coaxial cables**.

   (b) (2p.) In practice, a signal-to-noise ratio of at least 30 dB can probably be assumed for copper wires. Calculate the capacity **for copper wires** based on this assumption and give **one** reason why this value is higher than what is achieved in the VDSL standard.

   (c) (2p.) VDSL2 is an improvement of VDSL, where the upper end of the frequency band is extended from 12 MHz to 30 MHz. Assuming that the ratio of the transmission rates equal the ratio of the theoretical bounds **and** that VDSL2 divides the capacity between downstream and upstream so that both get an equal share, estimate the rates that can be achieved with VDSL2. Answer the question **both for copper wires and coaxial cables**.

Shannon–Fano coding

From Wikipedia, the free encyclopedia

In the field of data compression, **Shannon–Fano coding**, named after Claude Shannon and Robert Fano, is a technique for constructing a prefix code based on a set of symbols and their probabilities (estimated or measured). It is suboptimal in the sense that it does not achieve the lowest possible expected code word length like Huffman coding; however unlike Huffman coding, it does guarantee that all code word lengths are within one bit of their theoretical ideal $-\log P(x)$. The technique was proposed in Shannon's "A Mathematical Theory of Communication", his 1948 article introducing the field of information theory. The method was attributed to Fano, who later published it as a technical report.[1] Shannon–Fano coding should not be confused with Shannon coding, the coding method used to prove Shannon's noiseless coding theorem, or with Shannon–Fano–Elias coding (also known as Elias coding), the precursor to arithmetic coding.

In Shannon–Fano coding, the symbols are arranged in order from most probable to least probable, and then divided into two sets whose total probabilities are as close as possible to being equal. All symbols then have the first digits of their codes assigned; symbols in the first set receive "0" and symbols in the second set receive "1". As long as any sets with more than one member remain, the same process is repeated on those sets, to determine successive digits of their codes. When a set has been reduced to one symbol, of course, this means the symbol's code is complete and will not form the prefix of any other symbol's code.

The algorithm produces fairly efficient variable-length encodings; when the two smaller sets produced by a partitioning are in fact of equal probability, the one bit of information used to distinguish them is used most efficiently. Unfortunately, Shannon–Fano does not always produce optimal prefix codes; the set of probabilities {0.35, 0.17, 0.17, 0.16, 0.15} is an example of one that will be assigned non-optimal codes by Shannon–Fano coding.

For this reason, Shannon–Fano is almost never used; Huffman coding is almost as computationally simple and produces prefix codes that always achieve the lowest expected code word length, under the constraints that each symbol is represented by a code formed of an integral number of bits. This is a constraint that is often unneeded, since the codes will be packed end-to-end in long sequences. If we consider groups of codes at a time, symbol-by-symbol Huffman coding is only optimal if the probabilities of the symbols are independent and are some power of a half, i.e., $\frac{1}{2^n}$. In most situations, arithmetic coding can produce greater overall compression than either Huffman or Shannon–Fano, since it can encode in fractional numbers of bits which more closely approximate the actual information content of the symbol. However, arithmetic coding has not superseded Huffman the way that Huffman supersedes Shannon–Fano, both because arithmetic coding is more computationally expensive and because it is covered by multiple patents.

Shannon–Fano coding is used in the `IMPLODE` compression method, which is part of the `ZIP` file format.[2]

Contents

Shannon–Fano Algorithm

A Shannon–Fano tree is built according to a specification designed to define an effective code table. The actual algorithm is simple:

1. For a given list of symbols, develop a corresponding list of probabilities or frequency counts so that each symbol's relative frequency of occurrence is known.
2. Sort the lists of symbols according to frequency, with the most frequently occurring symbols at the left and the least common at the right.
3. Divide the list into two parts, with the total frequency counts of the left part being as close to the total of the right as possible.
4. The left part of the list is assigned the binary digit 0, and the right part is assigned the digit 1. This means that the codes for the symbols in the first part will all start with 0, and the codes in the second part will all start with 1.
5. Recursively apply the steps 3 and 4 to each of the two halves, subdividing groups and adding bits to the codes until each symbol has become a corresponding code leaf on the tree.

Example

The example shows the construction of the Shannon code for a small alphabet. The five symbols which can be coded have the following frequency:

Symbol	A	B	C	D	E
Count	15	7	6	6	5
Probabilities	0.38461538	0.17948718	0.15384615	0.15384615	0.12820513

All symbols are sorted by frequency, from left to right (shown in Figure a). Putting the dividing line between symbols B and C results in a total of 22 in the left group and a total of 17 in the right group. This minimizes the difference in totals between the two groups.

With this division, A and B will each have a code that starts with a 0 bit, and the C, D, and E codes will all start with a 1, as shown in Figure b. Subsequently, the left half of the tree gets a new division between A and B, which puts A on a leaf with code 00 and B on a leaf with code 01.

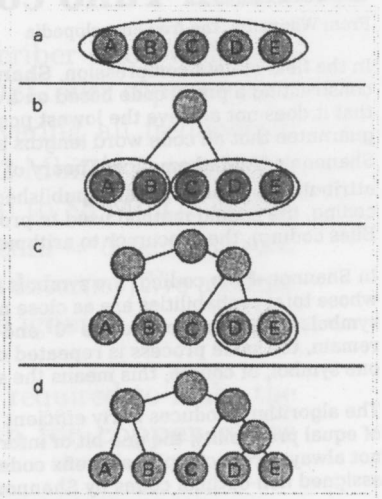
After four division procedures, a tree of codes results. In the final tree, the three symbols with the highest frequencies have all been

assigned 2-bit codes, and two symbols with lower counts have 3-bit codes as shown table below:

Symbol	A	B	C	D	E
Code	00	01	10	110	111

Results in 2 bits for A, B and C and per 3 bits for D and E an average bit number of

$$\frac{2\,\text{bits} \cdot (15+7+6) + 3\,\text{bits} \cdot (6+5)}{39\,\text{symbols}} \approx 2.28\,\text{bits per symbol.}$$



Shannon-Fano Algorithm