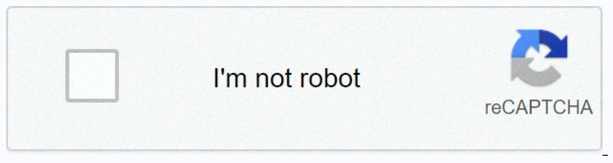

Continue

Shannon code emblem

In the field of data compression, Shannon-Fano encoding, named after Claude Shannon and Robert Fano, is a name given to two different but related techniques for building a prefix code based on a set of symbols and their probabilities (estimated or measured). Shannon's method selects a prefix code where a source symbol i {\displaystyle i} ⌈ the code word-length l i = ⌈ --log 2 p i ⌉ {\displaystyle l_{i}=\lceil -\log _{2}p_{i}\rceil } . One common way to select the code words uses the binary extension of the cumulative probabilities. This method was proposed in Shannon's A Mathematical Theory of Communication (1948), his article that introduced the field of information theory. Fano's method divides the source symbols into two sets (0 and 1) with probabilities as close to 1/2 as possible. Then those sets are divided themselves into two, and so on, until each set contains only one symbol. The code word for that symbol is the string 0s and 1s that make records which half of the separators it fell. This method was suggested in a later technical report by Fano (1949). Shannon-Fano codes are suboptimal in that they don't always achieve the lowest possible expected code word length, as Huffman coding does. [1] However, Shannon-Fano codes have an expected code word length within 1 bit. Fano's method usually produces encoding with shorter expected lengths than Shannon's method. However, Shannon's method is easier to theoretically analyze. Shannon–Fano coding should not be confused with Shannon-Fano-Elias coding (also known as Elias coding), the precursor to computational coding. Naming the confusion in the two different codes referred to by the same name, Writes Krajči et al[2] : Around 1948, both Claude E. Shannon (1948) and Robert M. Fano (1949) independently proposed two different source coding algorithms for an effective description of a discrete memoryless source. Unfortunately, despite being different, both schemes became known under the same name Shannon-Fano coding. There are several reasons for this mixture. For one thing, in discussing its coding scheme, Shannon calls Fano's scheme and calls it significantly the same (Shannon, 1948, p. 17). For another, both Shannon's and Fano's coding schemes are similar in the sense that they are both efficient, but suboptimal prefix-free coding schemes with a similar performance Shannon(1948) method, using predefined word lengths, are called Shannon-Fano encoding by Cover and Thomas[3], Goldie and Pinch[4], Jones and Jones[5], and Han and Kobayashi. [6] It is called Shannon coding by Yeung. [7] Fano's (1949) method, using binary division of probabilities, is called Shannon-Fano coding by Salomon[8] and Gupta. [9] It Fano encryption called by Krajči et al[2]. Shannon's Code: Predefined Word Lengths Main Article: Shannon Shannon Shannon's algorithm Shannon's method begins by deciding on the lengths of all the code words, then selecting a prefix code with those word lengths. Given a source with probabilities p 1 , p 2 , ... , p n {\displaystyle p_{1},p_{2},\dots, p_{n}} the desired code word lengths are l i = ⌈ - log 2 p i ⌉ {\displaystyle l_{i}=\lceil -\log _{2}p_{i}\rceil } Here⌈ x ⌉ {\displaystyle\lceil x\rceil } the ceiling function, meaning the smallest integer greater than or equal to x {\display style x} . Once the code word lengths are determined, we must select the code words ourselves. One method is to choose code words to choose from most likely to least likely symbols, choosing each code word to be the lexicologically first word of the correct length that maintains the prefix-free property. A second method uses cumulative probabilities. First, the probabilities are written in descending order p 1 ≥ p 2 ≥ ⋯ ≥ p n {\displaystyle p_{1}\geq p_{2}\geq\cdots\geq p_{n}} . Then, the cumulative probabilities are defined as c 1 = 0 , c i = ∑ j = 1 i - 1 p i for i ≥ 2 , {\displaystyle c_{1}=0,\qquad c_{i}=\sum _{j=1}^{i-1}p_{i}{\text{ for }}i\geq 2,} so c 1 = 0, c 2 = p 1 , c 3 = p 1 + p 2 {\display style c_{1}=0,c_{2}=p_{1} ,c_{3}=p_{1}+p_{2}} and so on. The code word for symbol i {\displaystyle i} is selected to be the first l i {\displaystyle l_{i}} binary digits in the binary extension of c i {\displaystyle c_{i}}. Example This example shows the construction of a Shannon-Fano code for a small alphabet. There are 5 different source symbols. Scathing 39 total symbols have been observed with the following frequencies, from which we can estimate the symbol probabilities. Symbol A B C D E Count 15 7 6 6 5 Probabilities 0.385 0.179 0.154 0.154 0.128 This source has entropy H (X) = 2.186 {\display style H(X)=2.186} bits. For the Shannon–Fano code, we must calculate the desired word lengths l i = ⌈ - log 2 p i ⌉ {\displaystyle l_{i}=\lceil -\log _{2}p_{i}\rceil } . Symbol A B C D E Probabilities 0.385 0.179 0.154 0.154 0.128 - log 2 p i {\display style -\log _{2}p_{i}} 1.379 2.480 2.700 2.7 2.963 Word lengths ⌈ - log 2 p i ⌉ {\screenstyle\lceil -\log _{2}p_{i}\rceil } 2 3 3 3 3 We can select code words to maintain the lexicographic first word of the correct length that the prefix-free property holds , select. Clearly A gets the code word 00. To maintain the prefix-free property, B's code word may not start 00, so the lexicographically first available word of length is 3,010. Continuing so, we get the following code: Symbol A B C D E Probabilities 0.385 0.179 0.154 0.154 0.128 Word Lengths ⌈ - log 2 <2> <5> p i ⌉ {\displaystyle\lceil -\log _{2}p_{i}\rceil } 2 3 3 3 3 Codewords 00 010 011 100 101 Alternatively, we can use the cumulative probability method. Symbol A B C And Waarskynlikhede Waarskynlikhede 0.179 0.154 0.154 0.128 Cumulative probabilities 0.000 0.385 0.564 0.718 0.872 ... in binary 0.00000 0.01100 0.10010 0.10110 0.11011 Word Lengths ⌈ - log 2 p i ⌉ {\display style \l -\log _{2}p_{i}\rceil } 2 3 3 3 3 Codewords 00 011 100 101 110 Note that although the code words under the two methods are different, the word lengths are the same. We have lengths of 2 bits for A, and 3 bits for B, C, D and E, giving an average length of 2 bits of · (15) + 3 bits · (7+ 6 + 6+ 5) 39 symbols ≈ 2.62 bits per symbol, {\displaystyle {\frac {2\,{\text{bits}}\cdot(15)+3\,{\text{bits}\cdot(7+6+6+5)}{39\,{\text{symbols}}}\approx.2.62\,{\text{bits per symbol .}}} which is within one bit of the entropy. Expected word length For Shannon's method saturates the word lengths l i = ⌈ - log 2 p i ⌉ ≤ - log 2 p i +1. {\displaystyle l_{i}=\lceil -\log _{2}p_{i}\rceil\leq -\log _{2}p_{i}+1.} Hence the expected word length satisfy E L = ∑ i = 1 n p i ≤ ∑ i = 1 n p i (- log 2 p i + 1) = - ∑ i = 1 n p i log 2 p i + ∑ i = 1 n p i = H (X) + 1. {\displaystyle\mathbb {E} L=\sum _{i=1}^{n}p_{i}l_{i}\leq \sum _{i=1}^{n}p_{i}(-\log _{2}p_{i}+1)=-\sum _{i=1}^{n}p_{i}\log _{2}p_{i}+\sum _{i=1}^{n}p_{i}=H(X)+1.} Here, H(X) = - ∑ i = 1 n p i log 2 p i {\displaystyle H(X)=-\textstyle\sum _{i=1}^{n}p_{i}\log _{2}p_{i}} is the entropy, and Shannon's source coding theorem says that any code should have an average length of at least H (X) {\displaystyle H(X)}. That's why we see that the Shannon-Fano code is always within one bit of the optimal expected word length. Fano's code: binary splitting Breakdown of Fano's code In Fano's method, the symbols are arranged in order from most likely to the least likely, and then split into two sets whose total probabilities are as close as possible to equal. All symbols then assigned the first digits of their codes; symbols in the first set receive 0 and symbols in the second set receive 1. As long as any sets remain with more than one member, the same process is repeated on those sets to determine sequential digits of their codes. When a set is reduced to one symbol, it means that 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 partition are in fact of equal likelihood, the one bit of information used to distinguish them is most effectively used. Unfortunately, Shannon-Fano encryption doesn't 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 encryption. Fano's version of Shannon-Fano coding is used in IMPLODE compression method, which is part of the ZIP file format. [10] The tree A Shannon-Fano tree is built according to a specification designed to define an effective code table. The actual algorithm is simple: For a given list of symbols, develop a corresponding list of probabilities or frequency counts so that each symbol's relative frequency of appearance is known. Sort the lists of symbols by frequency, with the most common symbols on the left and the least common on the right. Divide the list into two parts, with the total frequency counts of the left part being as close to the total from the right as possible. 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. Apply the steps 3 and 4 repeatedly to each of the two halves, subdivision groups and add bits to the codes until each symbol has become a corresponding code leaf on the tree. Example Shannon-Fano Algorithm We continue with the previous example. Symbol A B C D E Score 15 7 6 6 5 Probabilities 0.385 0.179 0.154 0.154 0.128 All symbols are sorted by frequency, from left to right (in Figure a). Putting the line of division 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 reduces the difference in totals between the two groups. With this section, A and B will each have a code starting with a 0-bit, and the C, D and E codes will all start with a 1, as shown in Figure b. After that, the left half of the tree gets a new split between A and B, which putS A on a leaf with code 00 and B on a leaf with code 01. After four section procedures, a tree of codes results. In the final tree, the three symbols with the highest frequencies were all assigned 2-bit codes, and two symbols with lower scores have 3-bit codes as table indicated below: Symbol A B C D E Probabilities 0.385 0.179 0.154 0.154 0 .128 First section 0 1 Second section 0 1 0 1 Third section 0 1 Code words 00 01 10 110 111 This results in lengths of 2 bits for A , B and C and per 3 bits for D and E, return an average length of 2 bits · (15 + 7 + 6) + 3 bits · (6 + 5) 39 symbols ≈ 2.28 bits per symbol. {\displaystyle {\frac {2\,{\text{bits}}\cdot(15+7+6)+3\text{bits}}\cdot(6+5)}{39\,{\text{symbols}}}\approx.28\,{\text{bits per symbol.}}}} We see that Fano's method, with an average length of 2.28, surpassed Shannon's method, with an average length of 2.62. Expected word length This is shown by Krajči et al[2] that the expected length of Fano's method length above is bound by E L ≤ H (X) + 1 - p min {\displaystyle\mathbb {E} L\leq H(X)+1-p_{\text{min}}} , where p min = min i {\displaystyle p_{\text{min}}=\text style _{i}p_{i}} is the probability of least common Comparison with other coding methods Neither Shannon-Fano algorithm is guaranteed to generate an optimal code. For this reason, Shannon-Fano codes are almost never used; Huffman coding is almost as calculatedly simple and produces prefix codes that always reach 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 limitation that is often unnecessary, since the codes will be packaged end-to-end in long lines. If we consider groups of codes at a time, symbol-by-symbol Huffman encoding is optimal only if the probabilities of the symbols are independent and are some force of a half, i.e. 1/2 k {\display style\text style 1/2^{k}} . In most situations, arithmetic coding can produce greater overall compression than either Huffman or Shannon-Fano, since it can encode into fractional numbers of bits that approach the actual information content of the symbol more closely. However, arithmetic coding has not replaced Huffman as Huffman replaces Shannon-Fano, both because computational coding is more calculative and because it is covered by multiple patents. [citation needed] Huffman's coding article: Huffman coding a few years later, David A. Huffman (1949) gave another algorithm that always produces an optimal tree for any given symbol probabilities. While Fano's Shannon-Fano tree is created by dividing the root to the leaves, the Huffman algorithm works in the opposite direction, merging the leaves to the root. Create a leafcdode for each symbol and add it to a priority queue, using its frequency of appearance as the priority. While there is more than one node in the string: Remove the two nodes of lowest probability or frequency of the queue Prepend 0 and 1 respectively to any code assigned to these nodes Create a new internal node with these two nodes as children and with probability equivalent to the sum of the two nodes' probabilities. Add the new node to the string. The remaining node is the root node and the tree is complete. Example with Huffman coding Huffman Algorithm We use the same frequencies as for the Shannon-Fano example above, viz: Symbol A B C D E Score 15 7 6 6 5 Probabilities 0.385 0.179 0.154 0.154 0.128 In this case, D&E has the lowest frequencies and so is awarded 0 and 1 and grouped together with a combined probability of 0.228 respectively. The lowest pair are now B and C so they are awarded 0 and 1 and grouped along with a combined probability of 0.333. This leaves BC and DE now with the lowest probabilities as 0 and 1 are pre-equiped on their codes and they are combined. That then leaves only A and BCDE, which are 0 and 1, respectively and then be combined. This leaves us with a single node and our algorithm is complete. The code lengths for the characters this time are 1 bit for A and 3 bits for all other characters. Symbol A B C D E Code words 0 100 101 110 111 This results in the lengths of 1 bit for A and per 3 bits for B, C, D and E, give an average length of 1 bit · 15+ 3 bits · (7+6+6+5) 39 symbols ≈ 2.23 bits per symbol. {\displaystyle {\frac {1\,{\text{bit}}\cdot 15+3\\, {\text{bits}}\cdot(7+6+6+5)}{39\,{\text{symbols}}}\approx 2.23\,{\text{bits per symbol.}}} We see that the Huffman code surpassed both types of Shannon-Fano code, which expected lengths of 2.62 and 2.28. Notes ^ Kaur, Sandeep; Singh, Sukhjeet (May 2016). Entropy coding and various coding techniques (PDF). Journal of Network Communication and Emerging Technologies. 6 (5): 5. Retrieved 3 December 2019. ^a b c Stanislav Krajči, Chin-Fu Liu, Ladislav Mikeš and Stefan M. Moser (2015), Performance Analysis of Fano Coding, 2015 IEEE International Symposium on Information Theory (ISIT). ^ Thomas M. Cover and Joy A. Thomas (2006), Elements of Information Theory (2nd ed.), Wiley-Interscience. Historical Notes to Chapter 5. ^ Charles M. Goldie and Richard G. E. Pinch (1991), Communication Theory, Cambridge University Press. Section 1.6. ^ Gareth A. Jones and J. Mary Jones (2012), Information and Coding Theory (Springer). Section 3.4. ^ Te Sun Han and Kingo Kobayashi (2007), Mathematics of Information and Coding, American Mathematical Society. Subsection 3.7.1. ^ Raymond W Yeung (2002), a first course in information theory, Springer. Subsection 3.2.2. ^ David Salomon (2013), Data Compression: The Complete Reference, Springer. Section 2.6. ^ Prakash C. Gupta (2006), Data Communication and Computer Networks, Phi Publishing. Subsection 1.11.5. ^ APPNOTE. TXT - . ZIP file format specification. PKWARE Inc. 2007-09-28. Retrieved 2008-01-06. The Imploding algorithm is actually a combination of two separate algorithms. The first algorithm compresses repeated byte rows using a sliding dictionary. The second algorithm is used to complement the encoding of the sliding dictionary output, using several Shannon-Fano trees. ^Huffman, D. (1952). A method for the construction of minimum redundancy codes (PDF). Proceedings of the IRE. 40 (9): 1098–1101. Doi:10.1109/JRPROC.1952.273898. References Fano, R.M. (1949). The transfer of information. Technical Report No. 65. Cambridge (Mass.), USA: Research Laboratory of Electronics at MIT. CS1 main: ref=harv (link) Shannon, C.E. (July 1948). A mathematical Theory of Communication. Bell System Technical Journal. 27: 379–423. Retrieved