Run-Length Encodings

I. A CONTEXT FOR THE PROBLEM

Secret Agent 00111 is back at the Casino again, playing a game of chance, while the fate of mankind hangs in the balance. Each game consists of a sequence of favorable events (probability p), terminated by the first occurrence of an unfavorable event (probability $q = 1 - p$). More specifically, the game is roulette, and the unfavorable event is the occurrence of 0, which has a probability of $q = 1/37$. No one seriously doubts that 00111 will come through again, but the Secret Service is quite concerned about communicating the blow-by-blow description back to Whitehall.

The bartender, who is a free-lance agent, has a binary channel available, but he charges a stiff fee for each bit sent. The problem perplexing the Service is how to encode the vicissitudes of the wheel so as to place the least strain on the Royal Exchequer. It is easily seen that, for the case $p = q = 1/2$, the best that can be done is to use 0 and 1 to represent the two possible outcomes. However, the case at hand involves $p \gg q$, for which the "direct coding" method is shockingly inefficient.

Finally, a junior code clerk who has been reading up on Information Theory, suggests encoding the *run lengths* between successive

Manuscript received May 6, 1965; revised November 9, 1965. This work was supported in part by the Jet Propulsion Laboratory under Contract 951076.

TABLE I
RUN-LENGTH DICTIONARIES FOR SMALL m

$m = 1$			$m = 2$			$m = 3$			$m = 4$		
n	$G(n)$	Codeword	n	$G(n)$	Codeword	n	$G(n)$	Codeword	n	$G(n)$	Codeword
0	1/2	0	0	0.293	00	0	0.206	00	0	0.151	000
1	1/4	10	1	0.207	01	1	0.164	010	1	0.128	001
2	1/8	110	2	0.116	100	2	0.130	011	2	0.109	010
3	1/16	1110	3	0.104	101	3	0.103	100	3	0.092	011
4	1/32	11110	4	0.073	1100	4	0.081	1010	4	0.078	1000
5	1/64	111110	5	0.051	1101	5	0.064	1011	5	0.066	1001
6	1/128	1111110	6	0.036	11100	6	0.051	1100	6	0.056	1010
7	1/256	11111110	7	0.025	11101	7	0.041	11010	7	0.048	1011
8	1/512	111111110	8	0.018	111100	8	0.032	11011	8	0.040	11000
9	1/1024	1111111110	9	0.013	111101	9	0.026	11100	9	0.034	11001
10	1/2048	11111111110	10	0.009	1111100	10	0.021	111010	10	0.029	11010

unfavorable events. In general, the probability of a run length of n is $p^n q$, for $n = 0, 1, 2, 3,...$, which is the familiar *geometric distribution*. (See Feller,[1] page 174.)

II. The Encoding Procedure

If the list of possible outcomes were finite, we could list them with their probabilities, and apply Huffman coding[2] (as done by Abramson,[3] page 77 et seq.). However, with an infinite list, it is clear that we cannot start at the bottom and work our way up. Fortunately, the fact that the probabilities follow a distribution law furnishes a short cut, as follows.

Let $m = -\log 2/\log p$. (That is, $p^m = 1/2$.) The results will be most readily applicable for those p such that m is an integer (viz., $p = 0.5, p = 0.707..., p = 0.794..., p = 0.849..., p = 0.873...$, etc.). The resulting coding scheme is especially simple when m is a power of 2, but *any* integer m is a favorable case.

If $p^m = 1/2$, then a run of length $n + m$ is only half as likely as a run of length n. (The respective probabilities are $p^{m+n} q = \frac{1}{2} p^n q$ and $p^n q$.) Thus, we would expect the codeword for run-length $n + m$ to be one bit longer than the codeword for run-length n. This argument, although nonrigorous, leads to the correct conclusion that there should be m codewords of each possible wordlength, except for the shortest wordlengths, which are not used at all if $m > 1$, and possibly one transitional wordlength which is used fewer than m times. Knowing this answer, there is a rigorous proof by mathematical induction. The dictionaries for the first several values of m are as shown in Table I, where $G(n)$ is used to designate $p^n q$.

In general, let k be the smallest positive integer such that $2^k \geqq 2m$. Then the corresponding code dictionary contains exactly m words of every word length $\geqq k$, as well as $2^{k-1} - m$ words of length $k - 1$. (The simplification which occurs for m a power of 2 is that the collection of words of length $k - 1$ is empty.) This result is obtained by seeing how much "signal space" is used up by having m words of every length $\geqq k$. This consumes

$$\frac{m}{2^k} + \frac{m}{2^{k+1}} + \frac{m}{2^{k+2}} + \cdots = \frac{m}{2^{k-1}},$$

leaving $1 - m/2^{k-1} = (2^{k-1} - m)/2^{k-1}$ unused, which means that $2^{k-1} - m$ words of length $k - 1$ may be adjoined.

III. Further Examples

We will consider the cases $m = 14$ and $m = 16$, to illustrate what happens when m is not a power of 2 and when m is a power of 2, respectively. The dictionaries in these two cases are shown in Table II. In the case $m = 14$, we find $k = 5$, and $2^{k-1} - m = 2$, so that there are two codewords of length 4, followed by fourteen codewords of lengths 5, 6, 7, etc. On the other hand, since $m = 16$ is a power of 2, the corresponding dictionary contains exactly 16 words of every wordlength starting with length 5.

In a practical situation, if $m = -\log 2/\log p$ is not an integer, then the best dictionary will oscillate between $[m]$ words of a given length and $[m] + 1$ words of another length. (Here $[m]$ denotes the greatest integer $\leqq m$.) For large m, however, there is very little penalty for picking the *nearest* integer when designing the code. Very often, the underlying probabilities are not known accurately enough to justify picking a non-integral value of m. (For example, saying $p = 0.95$ on the basis of statistical evidence may involve as large a round-off error as saying $m = 14$.) For Agent 00111, the approximation $m = 25$ corresponds closely to $q = 1/37$.

TABLE II
RUN-LENGTH DICTIONARIES FOR $m = 14$ AND $m = 16$

$m = 14$				$m = 16$			
n	Codeword	n	Codeword	n	Codeword	n	Codeword
0	0000	24	101100	0	00000	24	101000
1	0001	25	101101	1	00001	25	101001
		26	101110	2	00010	26	101010
2	00100	27	101111	3	00011	27	101011
3	00101	28	110000	4	00100	28	101100
4	00110	29	110001	5	00101	29	101101
5	00111			6	00110	30	101110
6	01000	30	1100100	7	00111	31	101111
7	01001	31	1100101	8	01000		
8	01010	32	1100110	9	01001	32	1100000
9	01011	33	1100111	10	01010	33	1100001
10	01100	34	1101000	11	01011	34	1100010
11	01101	35	1101001	12	01100	35	1100011
12	01110	36	1101010	13	01101	36	1100100
13	01111	37	1101011	14	01110	37	1100101
14	10000	38	1101100	15	01111	38	1100110
15	10001	39	1101101			39	1100111
		40	1101110	16	100000	40	1101000
16	100100	41	1101111	17	100001	41	1101001
17	100101	42	1110000	18	100010	42	1101010
18	100110	43	1110001	19	100011	43	1101011
19	100111			20	100100	44	1101100
20	101000	44	11100100	21	100101	45	1101101
21	101001	45	11100101	22	100110	46	1101110
22	101010	46	11100110	23	100111	47	1101111
23	101011	47	11100111				

IV. Decoding

The dictionaries in Table II exhibit striking patterns which suggest that a rather simple decoding procedure might be employed. For the case $m = 16$, the following rule for decoding is adequate.

Start at the beginning (left end) of the word, and count the number of 1's preceding the first 0. Let this number be $A \geqq 0$. Then the word consists of $A + 5$ bits. Let the last 5 bits be regarded as the ordinary binary representation of the integer R, $0 \leqq R \leqq 15$. Then the correct decoding of the word is $16A + R$. This simple decoding reveals an equally simple method of *encoding*. To encode the number N, we divide N by 16 to get $N = 16A + R$, and write A 1's followed by the 5-bit binary representation of R.

The case $m = 14$ is only slightly more complicated. Suppose a word starts in A 1's, and the next three bits are *not all 0's*. Then we consider the word to consist of $A + 5$ bits altogether. Let the last 5 bits be the binary representation of the integer R. Then the correct decoding of the codeword is $14A + R - 2$. On the other hand, if the initial A 1's are followed by three or more 0's, we regard the codeword as consisting of a total of $A + 4$ bits. Letting the last 4 bits be the binary representation of an integer R', the correct decoding in this case is $14A + R'$. This procedure also can be inverted to describe direct encoding from ordinary numbers to codewords.

[1] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. 1. New York: Wiley, 1950.
[2] D. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE*, vol. 40, pp. 1098–1101, September 1952.
[3] N. Abramson, *Information Theory and Coding*. New York: McGraw-Hill, 1963.

V. Implementation

The Senior Cryptographer observes that although run length coding is a big improvement over no coding at all, it is less than 100 percent efficient for the mission at hand. He has heard that a method invented at M.I.T. is 100 percent efficient. However, a hasty briefing on this method convinces Operations that it is unimplementable, because it requires infinite computing capability. The run-length system is employed after all. As it turns out, however, Agent 00111 has bribed the croupier, and the "Unfavorable Case" occurs only half as often as expected. Fortunately, the coding procedure is such that the cost of communicating has also decreased as a result!

It is appropriate to mention that there really is a method, invented by Elias and Shannon (see Abramson,[3] page 61), which is 100 percent efficient for communicating events from a p: q distribution. Moreover, the assertion that "infinite computing capability" is required is a gross overstatement. Nevertheless, British Intelligence quite possibly made the correct practical decision. We shall leave it to the reader to judge.

VI. Perspective

The literature in statistical communication theory generally contains a significant shift in viewpoint between the discrete and the continuous case. In the latter context, a particular distribution is assumed almost from the outset, and most of the theorems refer to such things as the "white Gaussian noisy channel," or other equally specific assumptions. For the discrete case, on the other hand, the results are rarely evaluated in terms of specific distributions. The present remarks are intended as a step in this direction, viz., the explicit form which Huffman coding assumes when applied to the geometric distribution. It would also be appropriate to have explicit answers for the binomial distribution, the Poisson distribution, etc.

Solomon W. Golomb
Dept. of Elec. Engrg.
University of Southern California
Los Angeles, Calif.