for the relationship of the problem discussed in Aizerman et al.[7] and Braverman and Pyatnitskii [8] with the pattern classification problem discussed here).

T. J. Wagner
J. M. Pitt
B. F. Womack
Dept. of Elec. Engrg.
University of Texas
Austin, Tex.

References

[1] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis*. New York: Wiley, 1958.
[2] L. A. Zadeh and C. A. Desoer, *Linear System Theory*. New York: McGraw-Hill, 1963.
[3] J. D. Patterson and B. F. Womack, "An adaptive pattern classification system," Labs. for Electronics and Related Science Research, Dept. of Elec. Engrg., University of Texas, Austin, Tech. Rept. 9, August 1965.
[4] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer, "The probability problem of pattern recognition learning and the method of potential functions," *Automat. remote Control*, vol. 25, pp. 1175–1190, March 1965.
[5] C. Blaydon, "On a pattern classification result of Aizerman, Braverman, and Rozonoer," *IEEE Trans. Information Theory (Correspondence)*, vol. IT-12, pp. 82–83, January 1966.
[6] Y. C. Ho and C. Blaydon, "On the abstraction problem in pattern classification," Cruft Lab., Harvard University, Cambridge, Mass., Tech. Rept. 476, August 1965.
[7] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer, "The method of potential functions for the problem of restoring the characteristic of a function converter from randomly observed points," *Automat. remote Control*, vol. 25, pp. 1546–1556, May 1965.
[8] E. Braverman and E. Pyatnitskii, "Estimation of the rate of convergence of algorithms based on the potential functions method," *Automation and Control*, vol. 27, pp. 80–100, January 1966.
[9] C. Blaydon and Y. C. Ho, "On the abstraction problem in pattern classification," *Proc. NEC*, pp. 857–862, 1966.
[10] C. Blaydon and R. Kashyap, "Recovery of functions from noisy measurements taken at randomly selected points and its application to pattern classification," *Proc. IEEE (Letters)*, vol. 54, pp. 1127–1129, August 1966.

The Number of Different Possible Compact Codes

***Abstract*—For a source with a given number q of messages and an unspecified set of probabilities, the number $X(q)$ of nontrivially different compact codes that are possible increases in a predictable fashion as q increases. Distinct binary compact codes of q messages correspond to distinct oriented binary trees with q terminal nodes. The theorem of this correspondence shows that, by using a recursion relation, and given that there is one compact code tree for $q = 2$, all compact code trees for any $q > 2$ can be automatically constructed. This is done by splitting, for all integers $s \geq 1$, s bottom level nodes of all compact code trees which have $q - s$ terminal nodes and which end in s or more bottom level nodes.**

I. Introduction

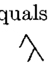
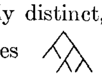
For a source with a given number q of messages or code words, a compact code is a uniquely decodable code (uniquely decodable for all finite sequences of code words) with the minimum possible average word length, where average word length L is the sum of the products of each code word's length l times its probability P.[1]

$$L = \sum_{i=1}^{q} P_i l_i.$$

If the possibility of error is negligible or disregarded, these are the most efficient codes. Notice, however, that since the average word length depends on the set of probabilities of the code words, there may be more than one possible compact code for a given source if the probabilities are allowed to vary. For instance, for the source with four equiprobable words the compact code is the uniquely decodable code with four words of length 2. But for the source with four words with probabilities $P_1 \geq P_2 \geq P_3 \geq P_4$ and $P_1 > (1 - P_2)/2$ a compact code must have $l_1 = 1$, $l_2 = 2$, $l_3 = 3$, $l_4 = 3$; when $P_1 < (1 - P_2)/2$ a compact code must have $l_1 = l_2 = l_3 = l_4$. These are the only possible compact codes for a source with four messages. Note that the codes 0, 10, 110, 111, and 0, 11, 100, 101, and all other uniquely decodable codes with these word lengths, are only trivially different with respect to compactness, since it is the set of word lengths, not the words themselves, that is of interest. Since they are uniquely decodable and their average word length is minimized, binary compact codes always have an even number of words of length $l_{\max}$. (From here on, this correspondence is restricted for brevity's sake to binary compact codes, to which subclass "compact codes" is now always meant to refer.)

It would be of interest to the coder to know exactly how many different compact codes are possible for a given source with q messages. (As will be shown, this is the same problem as that of counting distinct binary trees.) But heretofore the algorithm behind the sequence of numbers of compact codes for increasing q has been elusive. For $q = 2$ there is only one compact code. For $q = 3$ there is still only one. There are two for $q = 4$, three for $q = 5$, and five for $q = 6$. So far the numbers appear to be in a Fibonacci sequence. However, for $q = 7$ there are nine compact codes, instead of the expected eight. But the sequence is quite regular, being based on the nature of the codes. It is the thesis of this correspondence that for a source with a given number q of messages and an unspecified set of probabilities, the number $X(q)$ of compact codes that are possible increases in a predictable fashion as q increases.

II. Theorem

Let us use Fano's oriented binary tree representation of codes,[2] in which a terminal node at level R down from the top represents a code word of length $1 = R$, and the height of the tree equals the length of the longest code word. For example, the tree  represents a code with one word of length 1 and two words of length 2. Codes corresponding to trees having the same number of terminal nodes, at each level, as each other, are only trivally distinct, and will not be counted as different (e.g., the trees  and  are equivalent). This is not the usual notion of isomorphic trees. Let us also use Neumann's numeral representation of codes[3] (.04 represents a code with zero words of length 1 and four words of length 2; this is equivalent to the tree ). A node whose level from the top is equal to the height of the tree will be called a bottom level node (*bln*). Since every compact code corresponds to a tree with strictly binary branching, every tree will end in an even number of *bln*'s, and every numeral representation will end in an even number.

A table $\overline{T(p, q)}$ can be constructed by computing, for each q and each even number p, the number of trees with q terminal nodes and p *bln*'s. (Note that p is always $\leq q$.) The construction of this table is based on the following theorem.

Let $T(p, q)$ = the set of all the nontrivially different trees with q terminal nodes and p *bln*'s.

Theorem

$|T(2s, q + s)| = |\cup_{p \geq s} T(p, q)|$ (read "The cardinality of $T(2s, q + s)$ equals the cardinality of $\cup_{p \geq s} T(p, q)$"), for $q \geq 2$, $s \geq 1$.

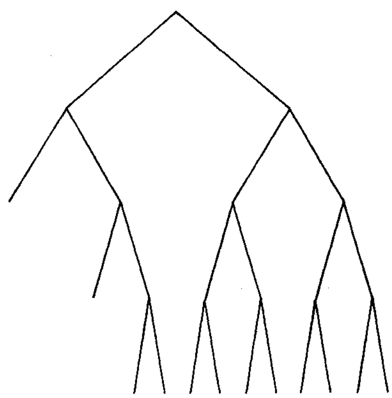
Proof: Let $S(s, q) = \cup_{p \geq s} T(p, q)$. We want to set up a one-to-one correspondence ϕ from $S(s, q)$ to $T(2s, q + s)$. For a given tree t in $S(s, q)$, let $\phi(t)$ be the tree formed from t by splitting s of the *bln*'s of t (splitting a node of a tree refers to the process of making the node nonterminal by adding two branches leading from it to two new nodes at the next level).

Manuscript received August 15, 1966; revised March 7, 1967. The research in this work was supported in part by National Science Foundation Grant GN-329.

TABLE I

$q \rightarrow$ / $p \downarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
2	1	1	1	2	3	5	9	16	28	50	89	159	285	510	914	1639	2938	5269	9451	16952	30410	54555
4			1	1	2	3	5	9	16	28	50	89	159	285	510	914	1639	2938	5269	9451	16952	30410
6						1	1	2	4	7	12	22	39	70	126	225	404	725	1299	2331	4182	7501
8							1	1	2	4	7	12	22	39	70	126	225	404	725	1299	2331	4182
10											1	2	3	6	11	20	37	66	119	215	385	692
12												1	2	3	6	11	20	37	66	119	215	385
14														1	1	2	4	8	15	27	49	89
16															1	1	2	4	8	15	27	49
18																				1	3	5
20																					1	3

TABLE II



.011(10)

TABLE III

p	Decimal Representation of First Tree Ending in p	Row p Begins at $q =$
18	.0111(18)	$18 + 1 + 1 + 1 = 21$
20	.0110(20)	$20 + 1 + 1 = 22$
22	.0101(22)	$22 + 1 + 1 = 24$
24	.0100(24)	$24 + 1 = 25$
26	.0011(26)	$26 + 1 + 1 = 28$
28	.0010(28)	$28 + 1 = 29$
30	.0001(30)	$30 + 1 = 31$
32	.0000(32)	$32 = 32$

ϕ is a single-valued function since all of the different ways of splitting s of the *bln*'s of a given tree yield equivalent trees. Also; ϕ is one-to-one since if t_1 and t_2 are nonequivalent, then splitting s of the *bln*'s of each cannot make them equivalent. For a given $t \in S(s, q)$, $\phi(t)$ will be in $T(2s, q + s)$, since splitting s *bln*'s results in $2s$ *bln*'s in the new tree with a net increase of s terminal nodes. ϕ is onto since if t' is a tree in $T(2s, q + s)$, then t' is the image of the tree t obtained by erasing the $2s$ *bln*'s from t'.

Hence, ϕ is a one-to-one correspondence between $S(s, q)$ and $T(2s, q + s)$ and therefore $|T(2s, q + s)| = |S(s, q)|$. Q.E.D.

Corollary

$|T(2s, q + s)| = \sum_{p \geq s} |T(p, q)|$, since $|\bigcup_{p \geq s} T(p, q)| = \sum_{p \geq s} |T(p, q)|$ (because the sets $T(p, q)$ are clearly disjoint).

In Table I, then, any column q summed from row p (p is always even) on down determines $T(p', q')$, where $p' = 2p - 2$ and $q' = q + p - 1$, and it determines $T(p'', q'')$, where $p'' = 2p$ and $q'' = q + p$. (E.g., the sum of any column q from $p = 2$ on down yields the number of trees ending in 2 *bln*'s for $q + 1$, and it also yields the number of trees ending in 4 *bln*'s for $q + 2$. But for $p = 6$ and $p = 8$, where 3 and 4 *bln*'s of previous trees must be split, trees ending in only 2 *bln*'s are unproductive, and the column must be summed from $p = 4$ on down.) Every entry in Table I except $T(2, 2)$ is determined by a summation of previous entries. In this way the whole table can be automatically constructed, given only the initial 1 in $T(2, 2)$. The sum of any column q also equals $X(q)$ for that q.

III. Beginnings of Rows

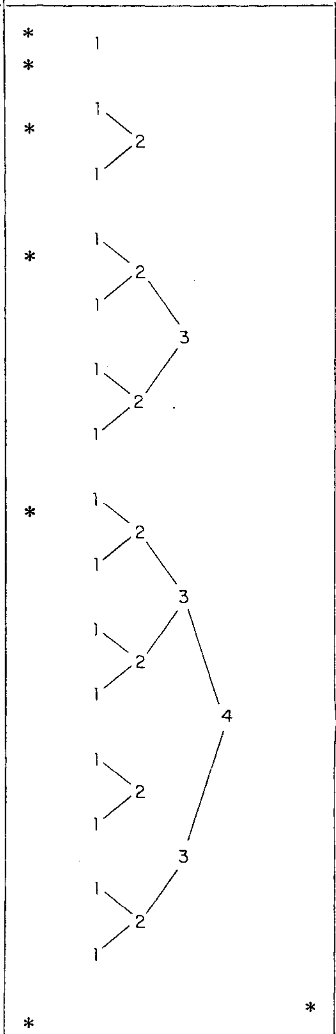
In checking this result to see if each row begins in the right place, some interesting properties of the sequence of intervals between beginnings of rows emerges. There are three ways of determining where the rows begin, each justifying the next.

The first method is based directly on the nature of the code trees. Since all branchings are binary, the number of nodes (terminal and nonterminal) on each level must be even. When $p = 10$, then, the next higher level must have a minimum of not just 5, but 6 nodes, of which 1 will be terminal. Above these 6, in turn, there must be not just 3, but 4 nodes, and again 1 is terminal. Above 4 there are 2, and 2 is, of course, the beginning of every tree. Now, since it has been shown that there must be at least 2 terminal nodes above the bottom level with 10 *bln*'s, we see that the row for $p = 10$ will begin at $q = 12$ (see Table II). In general, row p begins at q equal to p plus the number of ones that must be added in a process of dividing p by 2, adding 1 to the quotient if it is odd, then dividing this number by 2, adding 1 if odd, and so on until the quotient is 2.

TABLE IV

	row for $p =$	begins at $q =$	intervals between q's		row for $p =$	begins at $q =$	intervals between q's
*	2	2	2		66	71	1
*	4	4	3		68	72	2
	6	7	1		70	74	1
	8	8	4		72	75	3
	10	12	1		74	78	1
	12	13	2		76	79	2
	14	15	1		78	81	1
*	16	16	5		80	82	4
	18	21	1		82	86	1
	20	22	2		84	87	2
	22	24	1		86	89	1
	24	25	3		88	90	3
	26	28	1		90	93	1
	28	29	2		92	94	2
	30	31	1		94	96	1
*	32	32	6		96	97	5
	34	38	1		98	102	1
	36	39	2		100	103	2
	38	41	1		102	105	1
	40	42	3		104	106	3
	42	45	1		106	109	1
	44	46	2		108	110	2
	46	48	1		110	112	1
	48	49	4		112	113	4
	50	53	1		114	117	1
	52	54	2		116	118	2
	54	56	1		118	120	1
	56	57	3		120	121	3
	58	60	1		122	124	1
	60	61	2		124	125	2
	62	63	1		126	127	1
*	64	64	7	*	128	128	8

Nesting Structure Exhibited



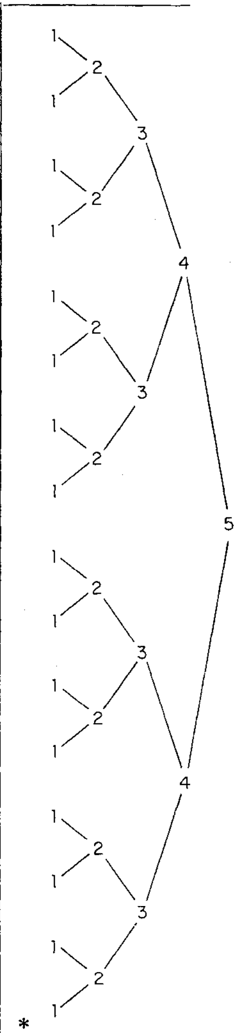
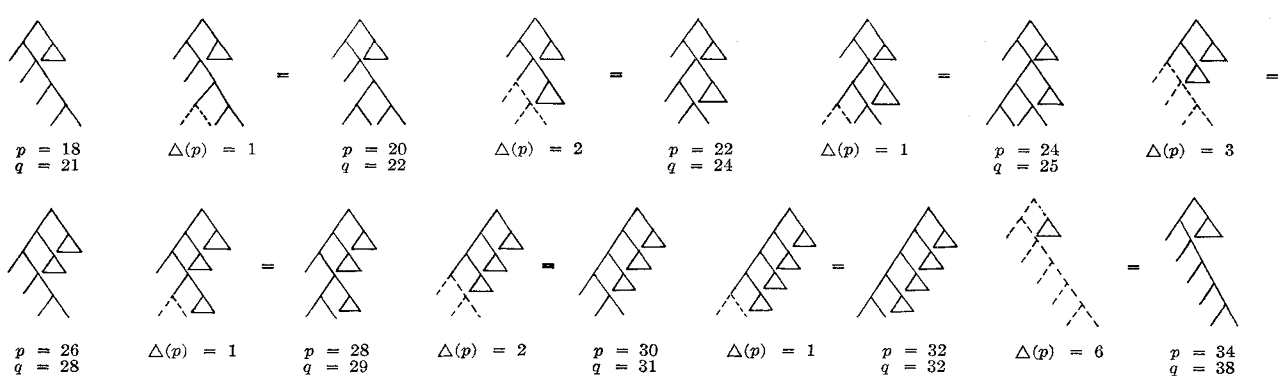
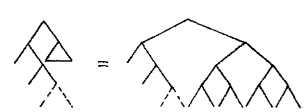


TABLE V



Note:  denotes a closed path, i.e., one which has no terminal nodes above the bottom level as indicated by the longest visible path.



However, this can be an arduous process for large p or for long strings of p's, and a second approach is possible. Consider the sequence of the numeral representations of the tree structures of the first tree in each row p for all values of p between 2^n and 2^{n+1}. The first number in the sequence, that for the first tree ending in $p = 2^n + 2$ *bln*'s, is .0 followed by $n - 1$ 1's followed by p. The last number in the sequence, that for the first tree ending in $p = 2^{n+1}$ *bln*'s, is n 0's followed by p. Now, if the last number, p, and the first number, 0, of each numeral representation are disregarded, the sequence is a countdown of binary numbers, each of length $n - 1$. This countdown generalization allows one to quickly generate the sequence of numeral representations of row-beginning trees for all values of p between 2^n and 2^{n+1} for any n. Each row p in the table begins at q equal to the sum of all the digits in the numeral representations ending in p (see Table III).

But a third method is possible. The intervals between beginnings of rows are predictable. Let $Q(p)$ be the value of q at which the row for the given p begins. Let $\Delta(p) = Q(p+2) - Q(p)$. Then $\Delta(2^n) = n + 1$, and the sequence of values of $\Delta(p)$ for p, $2^n < p < 2^{n+1}$, exhibits a symmetric nesting property (see Table IV).

The reason for this can be illustrated with either trees or numeral representations. In terms of the numeral representations of method 2, the countdown is performed by a right to left algorithm which begins with the right-most digit (except p) of numeral representation r_i in the sequence and changes all 0's to 1's until the first 1 is reached, which is then changed to 0. (If no 1 is found, a 0 is concatenated to the left of the representation and the decimal point is moved to the left of it.) Then p is stepped by 2 and the next representation r_{i+1} has been obtained. If the digit to the left of p in r_i was a 1 (as it will be for every odd i between the numbers for $p = 2^n$ and $p = 2^{n+1}$), $\Delta(p)$ is only 1. For the even i's between the numbers for $2^n < p < 2^{n+1}$, $\Delta(p)$ must be ≥ 2. For those divisible by 4, $\Delta(p)$ must be ≥ 3. In general, $\Delta(p) = k + 1$ where k is an integer such that p is divisible by 2^k but not by 2^{k+1}. k is limited by $n - 2$. (The k-generalization holds good only for $2^n < p < 2^{n+1}$.)

The reason for the nested sequence of intervals can also be illustrated in terms of trees. Consider the first tree in the row $p = 2^n + 2$. This tree will have $n - 1$ terminal nodes above the bottom level, with one on each level except the first (and the last). From this tree each tree in the sequence of row-beginning trees, up to that for $p = 2^{n+1}$, can be obtained by splitting, successively, the minimum number of non-*bln*'s of the previous tree necessary to produce two new *bln*'s. The number that must be split determines $\Delta(p)$. Table V illustrates why this will give a nested sequence. An algorithm for generating this sequence can be formulated, using a binary search technique.

It is possible to reduce the table $\overline{T(p, q)}$ to an equation depending only on q and $X(q)$, but so far all attempts to do so have had to take into account the nesting sequence of intervals obtained by method 3 above, so that the table remains the simpler and clearer method of arriving at the sequence of $X(q)$'s.

Acknowledgment

The author wishes to express her gratitude and indebtedness to W. A. Woods, Jr. for his criticism and help in the formulation of the theorem; and would also like to thank Prof. N. Abramson for bringing attention to the problem and for his comments and encouragement during its solution.

Emily Norwood
Computation Laboratory
Harvard University
Cambridge, Mass. 02138

References

[1] N. Abramson, *Information Theory and Coding*. New York: McGraw-Hill, 1963.
[2] R. Fano, *Transmission of Information*. New York: Wiley, 1961.
[3] P. G. Neumann, "Efficient error-limiting variable-length codes," Ph.D. dissertation, Harvard University, Cambridge, Mass., 1961.

Generating Random Numbers Having Probability Distributions Occurring in Signal Detection Problems

In many problems related to signal detection it is either impossible or impractical to obtain solutions analytically. One alternative is to resort to Monte Carlo simulation, and usually such simulation depends on having available sequences of numbers which are random samples from some specified probability distribution. Some methods are known for generating genuinely random numbers.[1],[2] More often we are satisfied with sequences of pseudo-random numbers generated in some deterministic manner by a digital computer.

Much of the literature on random numbers (from here on the term "random" will include pseudo-random) has been restricted to the generation and testing for randomness of sequences of numbers uniformly distributed in the unit interval. A random variable uniformly distributed in the unit interval will be referred to as $U(0, 1)$. The random variable Y is $U(0, 1)$ if

$$P(Y \leq y) = \begin{cases} 0 & y < 0 \\ y & 0 \leq y \leq 1. \\ 1 & y > 1 \end{cases} \tag{1}$$

Several authors have treated the problem of generating normally distributed random numbers from $U(0, 1)$ random numbers.[1] Some of the suggested methods are discussed and appraised by Muller.[5] His "direct" method is the following. Let

$$T = \sqrt{-2 \ln U} \cos(2\pi V) \tag{2}$$

$$Y = \sqrt{-2 \ln U} \sin(2\pi V) \tag{3}$$

where U and V are independent $U(0, 1)$ random variables. Then T and Y are independent normally distributed random variables with zero mean and unit variance.[6] This can be demonstrated as follows. From (2) and (3) it follows that

$$U = \exp[-(T^2 + Y^2)/2] \tag{4}$$

$$V = (1/2\pi) \operatorname{Tan}^{-1}(Y/T). \tag{5}$$

Since U and V are independent $U(0, 1)$ random variables, the joint density $f(t, y)$ of T and Y is given by $|J|$, where J is the Jacobian of the transformation. Using (4) and (5)

$$\begin{aligned} f(t, y) &= |J| = (1/2\pi) \exp[-(t^2 + y^2)/2] \\ &= (1/\sqrt{2\pi}) \exp[-t^2/2] \cdot (1/\sqrt{2\pi}) \exp[-y^2/2] \\ &= f(t) f(y). \end{aligned} \tag{6}$$

In radar applications it is often assumed that the total noise voltage (from combinations of thermal, cosmic, and shot-noise sources) is a normally distributed random variable. After the receiver voltage is passed through linear and/or nonlinear devices (e.g., a square-law detector) the resulting voltage has a probability distribution which can be obtained by performing the appropriate transformations on the normal probability distribution. Hence, we may simulate observations of the resulting voltage by first obtaining normally distributed random numbers from $U(0, 1)$ random numbers and then performing the proper transformations on the normally distributed random numbers. If we use the direct method mentioned above, these can be looked upon as transformations immediately from $U(0, 1)$ random numbers to the desired numbers, leaving out the intermediate step of obtaining normally distributed random

Manuscript received March 24, 1967; revised May 23, 1967.

[1] See the reference notes at the end of this correspondence.