# A high-speed algorithm for computing conditional probabilities of substrings of sequentially observed data*

LLOYD M. NIRENBERG†
*Space Biology Laboratory, Brain Research Institute*

JOCHEN HABER
*Health Sciences Computing Facility*

and

SAMUEL L. MOISE, JR.
*Space Biology Laboratory, Brain Research Institute*
*University of California, Los Angeles, California 90024*

**An algorithm is described that computes relative frequencies of occurrence of all arbitrarily long substrings of sequential data, such as are obtained from experiments in learning/memory and verbal interaction. The algorithm offers high speed and provides systematization for the computation of empirical conditional probabilities. Use of this algorithm allows application of probabilistic and information theoretic disciplines to reveal dependencies between events separated arbitrarily in time.**

Behavioral experiments often produce data sequentially in time. For example, experiments designed to examine learning and memory processes are often presented trial by trial and produce time sequences of parameters which describe stimulus-response events at each trial. Another sequential data sequence may be produced by experiments in natural interaction, in which utterances and signs are sequentially coded in time. In these cases, since the observed data is characteristically sequential, the E may wish to account for the time properties of the data in conjunction with other forms of data analysis.

The purpose of this paper is to specify an algorithm that allows the complete tabulation of empirical probabilities of occurrence of any particular substring of the data, given that some substring of arbitrary length has just preceded it. This probabilistic approach to the analysis of sequential data reveals dependencies of the present observation on past observations, allows direct application of information theory concepts, and generally systematizes the analysis of sequence information in the data.

The algorithm processes one arbitrarily long sequence of data under the assumption of stationarity. The stationarity assumption is reasonable from a computational point of view, since to assume otherwise would require such tight controls on data acquisition that data could not be obtained practically. This procedure provides a probabilistic state description of the data stream, whether the data is stationary or not. If knowledge of positional information about the data sequences is desirable, the E should examine substrings over intervals short enough to intuitively ensure stationarity but long enough for statistical validity.

The algorithm does not describe a Markov chain (MC) analysis of the data. The user of the present probabilistic analysis reveals Markov properties if they exist, but to assume Markov properties is to assume a more tightly constrained form of dependence than may be required. The difference between MC analysis and the present method is that the former assumes that sequences occurring far enough in the past have no effect on the present. The existence of this past-influence "truncation" property may be estimated by probabilistic analysis.

## NOTATION AND DEFINITIONS

Suppose the experiment consists of N trials (or observations) $(D_k)_{k=1}^{N}$ occurring consecutively at times $1, 2, \cdots, N$. Here, each $D_k$ stands for the values of a describing set of variables for each trial. Let

$$D_k = (d_{k1}, \cdots, d_{kn}) \tag{1}$$

be the descriptor for Trial k. Since quantization of a continuous variable can be made fine enough to ensure no loss of accuracy, it may be assumed that each $d_{kj}$ takes only discrete values.

Assume each $D_k$ can take up to $r_0$ different values. On any one trial k, $r_0$ can be an enormous number of combinations of variables $d_{kj}$, and the assessment of dependences between trials can become extremely complex. In order to systematize the search for multiple-trial influences, let

$Pr(D_k \mid D_{k-M}, \cdots, D_{k-1})$ = the probability that on trial k descriptor $D_k$ equals some particular allowable n-tuple, given the occurrence of M preceding specific consecutive n-tuple descriptors. (2)

*This research was supported in part by USPHS Grant GM 16058. Computing assistance was obtained from the UCLA Health Sciences Computing Facility, sponsored by NIH Special Research Resources Grant RR-3.

†Now with Aerospace Corporation, El Segundo, California.

| Time Index k | Observed Data $D_k = (d_{k1}$ | $d_{k2}$ | $d_{k3})$ | Indexed Data $x_k$ |
|---|---|---|---|---|
| 1 | 0 | 0 | 2 | 0 |
| 2 | 0 | 1 | 2 | 1 |
| 3 | 1 | 0 | 0 | 2 |
| 4 | 2 | 0 | 1 | 4 |
| 5 | 1 | 1 | 2 | 3 |
| 6 | 1 | 1 | 2 | 3 |
| 7 | 1 | 1 | 2 | 3 |
| 8 | 2 | 0 | 1 | 4 |
| 9 | 0 | 0 | 2 | 0 |
| 10 | 0 | 1 | 2 | 1 |
| 11 | 1 | 0 | 0 | 2 |

**Fig. 1. Example of recoding the observed data.**

Knowledge of all $r_0^{M+1}$ possible probabilities for each k, $1 \leq k \leq N$, defines the sense in which is understood the effect at Time k of the previous M events.

The defined conditional probability indicates a relationship among whole descriptors, $D_k$. But the effect due to an individual variable $d_{kj}$ may also be desired. This may be obtained by suitable applications of the rules of marginal probability and Bayes's theorem.

This systematic approach is independent of the number of previous trials M which may be considered. In fact, the effects of the "memory" M may be elicited by the same technique. However, these computations require the measurement of the necessary probabilities.

Even though each $D_k$ can theoretically take up to $r_0$ different values, in any given data set $(D_k)_{k=1}^N$, the usual case is that there are only $r < r_0$ different values that the descriptors actually take on. Therefore, we index each $D_k$ so that it may be referred to by its index in the set J = $(0, 1, \cdots, r-1)$. Figure 1 shows an example of an observed data set and its indexed data set. Each variable may take the values 0, 1, or 2. Notice that five different observed 3-tuples occur, although 27 possibilities exist. Therefore, each observed $D_k$ gets a label from 0 to 4. This is an obvious, but immediate and simplifying, recoding of the data for computational purposes.

## ALGORITHM FOR EMPIRICAL COMPUTATION OF CONDITIONAL PROBABILITIES

Suppose we have the sequence of numbers $X = (x_1, x_2, x_3, \cdots, x_N)$, where each $x_i \epsilon J$ for some integer $r \geq 2$. The symbols $(x_i)$ will sometimes be referred to as r-digits. If the position in the sequence, k, has no effect upon the probabilities in Eq. 2, then we say X is a stationary sequence. Since we assume for computational simplicity that X is stationary, no account of k is made. Finite sequence X is the only data observed. Let $\eta(x_{k+1}, \cdots, x_{k+K})$ be the number of times sequence $x_{k+1}, \cdots, x_{k+K}$ occurs in the observed data vector X independently of starting position k + 1. The sequence $x_{k+1}, \cdots, x_{k+K}$ is simply an arbitrary substring of Length K, of the observed data, X. An intuitive estimator for

$$P_K(x) = \Pr(x_k = x \mid x_{k-K}, \cdots, x_{k-1})$$

is

$$\hat{P}_K(x) = \frac{\eta(Z_K x)}{\eta(Z_K)},$$

where for any $k \geq 2$

$$Z_K = x_{k-K}, \cdots, x_{k-1}$$

stands for any string of K consecutive symbols. In the algorithm, K varies from 0 [yielding Pr(x)] to M. Length M is the largest depth backward in time (maximum "memory"). Furthermore, it can be shown (Billingley, 1961; Anderson & Goodman, 1957) that $\hat{P}_K(x)$ is a maximum likelihood estimator for $P_K(x)$, if, in fact, X is a Markov chain. Since we observe only a finite sequence X, it is computationally simpler to ignore the endpoint condition wherein, for example, $Z_K$ occurs only as the last K observed symbols. Here, $\eta(Z_K) = 1$ and $\eta(Z_K x)$ is undefined; no symbol is observed to follow $Z_K$. Rather than checking for this case, it is simpler to let

$$\hat{P}_K(x) = \frac{\eta(Z_K x)}{\sum_{j=0}^{r-1} \eta(Z_K j)}. \tag{3}$$

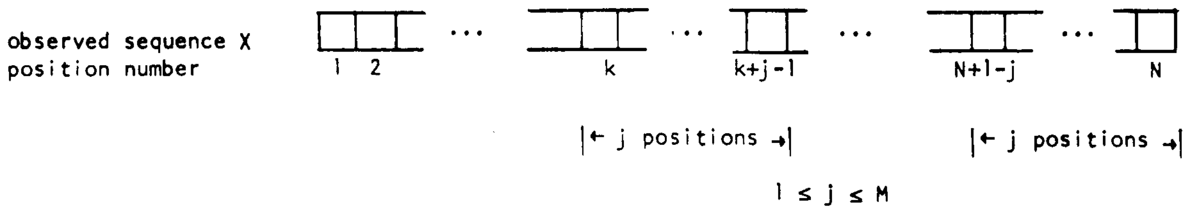
If $Z_K j_0$ does not occur, $\eta(Z_K j_0)$ is defined to be 0. This deviation from the maximum likelihood estimator will not cause significant differences in the result. The direct approach to compute Eq. 3 requires that $\eta(Z_K x)$ be evaluated by counting the occurrences of $Z_K x$ for each of the $r^{K+1}$ possible values of $Z_K x$. This amount of computation is intolerable for values of r and K of reasonable interest. Instead, the method used here simply counts the occurrences of each sequence of Length K that appears at least once in X, yet does not look for particular sequences which may not appear.

## METHOD

Let A(j,k) equal the decimal representation for the sequence of j r-digits of X which begins in Position k of X, $1 \leq k \leq N-j+1$, $1 \leq j \leq M$. These r-digits are assumed right-adjusted. (Figure 2 shows the construction of an A array.) The jth row of A contains the decimal values of the sequences of Length j formed by sequentially scanning X. Observe that

$$\Pr[x \mid Z_K] = \frac{\eta(Z_K x)}{\sum_{m=0}^{r-1} \eta(Z_K m)} = \frac{\eta(Z_K x)}{\sum_{m \epsilon J_K} \eta(Z_K m)},$$

where $J_K = [m : \eta(Z_K m) \neq 0]$.



"window" of length j moves from $x_1$ to $x_{N+1-j}$. The window contains a sequence of j r-digits, right adjusted. The last position for a j-window to start is N+1-j.

Fig. 2. Construction of A Array.

If $V(Z_K m)$ equals the decimal representation for the right-adjusted r-digits $Z_K m$, then if $|Z_K m| = K + 1 = j$, $\eta(Z_K n)$ equals the number of times $V(Z_K m)$ appears in Row j of A. The position information contained in the column index of A does not affect this counting (see below). From A, compute the following sequence of pairs for each Row j, $1 \leqslant j \leqslant M$:

$$(v_{j1},n_{j1}), (v_{j2},n_{j2}), \cdots, (v_{jL_j},n_{jL_j}),$$

where $L_j$ different numbers appear in Row j of A, and $v_{ji}$ is one of them, $1 \leqslant i \leqslant L_j$. The number of times $v_{ji}$ appears in Row j of A is $n_{ji}$. With $|Z_K| = j - 1$, $PR(x \mid Z_K) \neq 0$ implies that in Base r, the right r-digit of $v_{jq}$ is x for some q, $1 \leqslant q \leqslant L_j$ and the left $j - 1$ r-digits of $v_{jq}$ (Base r) form $Z_K$. Then

$$Pr[x \mid Z_K] = \frac{n_{jq}}{\sum\limits_{m \epsilon \tilde{J}_K} n_{jm}},$$

where $\tilde{J}_K$ = (m: left $j - 1$ r-digits of $v_{jm}$ form $Z_K$). In particular, $q\epsilon\tilde{J}$.

In other words,

$$Pr\left[x = v_{jq}MOD(r) \mid Z_K = \left\lfloor \frac{v_{jq}}{r} \right\rfloor\right] = \frac{n_{jq}}{\sum\limits_{m \epsilon J_K} n_{jm}},$$

where $\lfloor a \rfloor$ = integer part of a = largest integer $\leqslant$ a, for any number a. The computation for $Pr(x \mid Z_K)$ can be considerably speeded up by capitalizing on the fact that column information in A is irrelevant. Hence, individually sort each row of A keyed on its columns to bring all values together in a row in sections. Figure 3 shows a typical sorted row of A.

## COMPUTER TIME AND MEMORY REQUIREMENTS

The computer time and memory requirements are, to a large degree, dependent on the maximum sequence memory length, M, and the length of the input string, N. In fact, for fixed M and N, the A array is the only varying item among the program variables. Its size is:

$$\sum_{j=1}^{M} (N + 1 - j) = M\left[N - \frac{(M - 1)}{2}\right] \leqslant MN \text{ words.}$$

The number of operations required to fill up the A array is also proportioned to this quantity. In addition, the A array must be sorted, requiring a number of operations proportional to $MN \log_2 MN$. One pass through the A array counts the number of equal entries that require a number of operations proportional to MN.

Since, in practice, the number of different subsequences of length $\leqslant M$ is much less than MN, one might be tempted to count the equal subsequences as they are found. Here one could use a dynamic list to store all different subsequences as they are encountered in the search along with the number of occurrences of each subsequence. Using this method, the total time

$$\underbrace{v_{j1}\ v_{j1} \cdots v_{j1}}_{n_{j1}}\quad \underbrace{v_{j2}\ v_{j2} \cdots v_{j2}}_{n_{j2}}\quad \underbrace{v_{jm} v_{jm} \cdots v_{jm}}_{n_{jm}}\quad \underbrace{v_{jL_j}\ v_{jL_j} \cdots v_{jL_j}}_{n_{jL_j}}$$

$$A\ (j,k) = v_{ji} \quad \text{for} \quad \sum_{m=1}^{i-1} n_{jm} + 1 \leq k \leq \sum_{m=1}^{i} n_{jm}\ , \quad 1 \leq i \leq L_j$$

Fig. 3. Typical row of the sorted A Array.

required to search the list is proportional to:

$$MN\mu^2/4$$

where $\mu$ is the number of unique strings found. Since this time could become intolerably large, we would have to resort to some kind of direct-access table whose size must be $L_1 = 2\mu$ words. If a hash table is used for purposes of computational efficiency, we must allocate at least $L_2 = 1.33\ L_1$ words (see Morris, 1968). If we assign $\mu$ at its upper bound, we obtain

$$L_2 > MN \text{ words,}$$

and no memory saving is achieved.

The use of a hash table, although more expensive in terms of memory, will save some computation time. Here, the number of operations required will be proportional to:

$$(k_1 + 2k_2)MN + k_3L_2/2,$$

where $k_1$, $k_2$, $k_3$ are constants of proportionality, for generation of subsequences, insertion into hash table, and gathering the elements from the table, respectively. This procedure will not output the subsequences in sorted order, as will the method presented in this paper.

Given these considerations, our method of computation and memory management appears appropriate. These algorithms have been implemented as a series of FORTRAN programs for a 360-91 computer installation, and have been applied to behavioral data from monkeys performing in a delayed matching from sample task.

## REFERENCES

Anderson, T. W., & Goodman, L. A. Statistical inference about Markov chains. Annals of Mathematical Statistics, 1957, 28, 89-110.

Billingsley, P. Statistical methods in Markov chains. Annals of Mathematical Statistics, 1961, 32, 12-40.

Morris, R. Scatter storage techniques. Communications of the Association for Computing Machinery, 1968, 11, 38-44.

(Received for publication December 6, 1972; revision received March 1, 1973.)