Hypothesis Testing With Finite Memory in Finite Time

RICHARD A. FLOWER AND MARTIN E. HELLMAN

Abstract—**The hypothesis-testing problem has recently been studied under a finite-memory constraint. However, most work has been concerned with the large-sample theory. Here we study the small-sample theory for binary-valued observations.**

Introduction

Let $X_1,X_2,\cdots$ be a sequence of independent observations each drawn from an observation space $\mathscr{X}$ and each distributed according to a probability density function $p(x)$. There are two possible states of nature H_0 and H_1 with *a priori* probabilities π_0 and π_1. Under H_0, $p(x) = p_0(x)$, a known density, while under H_1, $p(x) = p_1(x)$, also a known density. To determine the true state of nature we formulate a sequence of decision rules $d_1(X_1),d_2(X_1,X_2),\cdots$, where $d_n \in \{H_0,H_1\}$. The objective is to minimize the probability of error.

Since each decision may depend upon all previous observations, the amount of data to be stored increases without bound. To bring the problem further into the real world, Hellman and Cover [1] have applied the constraint that the data be summarized by an m-valued statistic $T \in \{1,2,\cdots,m\}$. The statistic is updated after each observation according to the rule

$$T_n = f(T_{n-1},X_n),$$

where T_n is the statistic value at time n and f is a time-invariant (but possibly randomized) updating function.

The decision at time n,d_n, is required to depend on the data only through the value of T_n; i.e.,

$$d_n = d(T_n),$$

where d: $\{1,2,\cdots,m\} \to \{H_0,H_1\}$ is a time-invariant function.

The algorithm pair (f,d) may be thought of as an automaton with inputs $X_n \in \mathscr{X}$, internal state $T_n \in \{1,2,\cdots,m\}$, and outputs $d(T_n) \in \{H_0,H_1\}$. We seek an algorithm that minimizes P_N, the probability of error at time N.

For the infinite-sample problem ($N = \infty$), Hellman and Cover [1] have found the greatest lower bound to the probability of error $P_\infty{}^*$. Define the likelihood ratio $l(x)$ of an observation x in the usual manner, $l(x) = p_0(x)/p_1(x)$. Denote the essential supremum on the likelihood ratio by l_u: i.e.,

$$l_u = \sup \frac{\int_A p_0(x)\,dx}{\int_A p_1(x)\,dx},$$

where the supremum is over all sets A such that $\Pr\{A\} > 0$. Similarly, denote the essential infimum on the likelihood ratio by l_l. Define $\gamma = l_u/l_l$. As shown in [1], the greatest lower bound on the infinite-time probability of error is

$$P_\infty{}^* = \inf_{f,d}\{P_\infty\} = \min\left\{\frac{2(\pi_0\pi_1\gamma^{m-1})^{1/2} - 1}{\gamma^{m-1} - 1}, \pi_0, \pi_1\right\}$$

for an m-state algorithm. Although in general no algorithm can achieve $P_\infty{}^*$, there exists a sequence of algorithms $\{\mathscr{A}_i\}_{i=1}^{\infty}$ such that the limit of the associated sequence of probabilities of error is $P_\infty{}^*$.

To test between H_0 and H_1, a reasonable automaton might be the following saturable counter. The automaton moves up one state on observations favoring H_0 (i.e., $l(x) > 1$) and moves down one state on observations favoring H_1, so long as these moves do not violate the requirement that $T_n \in \{1,2,\cdots,m\}$. If the move indicated would violate that requirement, then the automaton stays in the same state.

Several intuitively pleasing modifications improve the performance of this saturable counter for the infinite-sample problem. First, make transitions only on observations with likelihood ratios close to l_u or l_l. Thus transitions are made only when strong evidence for H_0 or H_1 is available. However, one must be willing to wait a long time, if necessary, for events with the extreme likelihood ratios.

Second, introduce randomization in transitions out of the end states. Leaving states 1 and m with small probabilities δ and $k\delta$, respectively, causes the automaton to be in the end states a large portion of the time. Decisions made in the end states have the lowest probabilities of error, so that as $\delta \to 0$, with k fixed at its optimal value, the asymptotic probability of error is $P_\infty{}^*$. It should be noted that as $\delta \to 0$, the time constant of the machine becomes large [2]. Thus a machine that is close to optimal for a large or infinite number of samples is far from optimal for a small number of observations.

Problem Statement

Consider the following symmetric coin-toss problem. The possible experimental outcomes are heads (H) and tails (T). Under H_0, $\Pr\{H\} = p$ and $\Pr\{T\} = 1 - p \triangleq q$, while under H_1, $\Pr\{H\} = q$ and $\Pr\{T\} = p$. Furthermore, $\pi_0 = \pi_1 = \frac{1}{2}$. Without loss of generality one may assume $p > \frac{1}{2}$. Though conceptually simple, the symmetric coin-toss experiment contains the important aspects of the more general problem. Any problem may be readily reduced to a possibly asymmetric coin-toss problem by designating certain outcomes as "pseudoheads" (e.g., $\{X \mid l_u - \varepsilon_1 < l(x)\}$) and others as "pseudotails" (e.g., $\{X \mid l_l + \varepsilon_2 > l(x)\}$). Any other observations induce no change in machine state and may be considered to correspond to the coin landing on its edge, although this event may occur with high probability. Of course this is not an information-preserving transformation, but as seen in [1] it entails no loss for the infinite-sample problem.

We wish to find the algorithm pair (f,d) that minimizes P_N for $N < \infty$. As a first step, note that the updating rule f may be represented by two $m \times m$ stochastic matrices $P(H)$ and $P(T)$, where

$$P_{ij}(H) = \Pr\{T_n = j \mid T_{n-1} = i, X_n = H\}$$

and $P_{ij}(T)$ is defined similarly.

The independence of the observations $\{X_i\}$ induces a Markov process on the state space $S = \{1,2,\cdots,m\}$ under each hypothesis. The state transition matrices under H_0 and H_1 are thus

$$P^0 = pP(H) + qP(T) \qquad P^1 = qP(H) + pP(T).$$

Letting $\mu_i{}^j(n) = \Pr\{T_n = i \mid H_j\}$, the probability distribution on the states at time n, under H_j, is

$$\boldsymbol{\mu}^j(n) = [\mu_1{}^j(n),\mu_2{}^j(n),\cdots,\mu_m{}^j(n)].$$

Clearly

$$\boldsymbol{\mu}^0(n) = \boldsymbol{\mu}(0)(P^0)^n$$

$$\boldsymbol{\mu}^1(n) = \boldsymbol{\mu}(0)(P^1)^n.$$

Thus, given $\boldsymbol{\mu}(0)$, $P(H)$, $P(T)$, and p, it is possible to calculate

$$P_N = \tfrac{1}{2}\left[\sum_{i \in S_1} \mu_i{}^0(N) + \sum_{i \in S_0} \mu_i{}^1(N)\right],$$

where $S_0 = \{i: d(i) = H_0\}$ and $S_1 = \{i: d(i) = H_1\}$. The initial distribution we shall use is $\mu_{(m+1)/2}(0) = 1$, if m is odd; $\mu_{m/2}(0) = \mu_{(m/2)+1}(0) = \frac{1}{2}$, if m is even.

On the basis of a study of optimal infinite-time algorithms [1], it was conjectured that for this symmetric coin-toss problem the optimal finite-time automaton would have the following form.

Manuscript received July 6, 1971; revised November 10, 1971. This work was supported in part by NSF Grant GK 5800 and in part by the NSF Graduate Fellowship Program.

R. A. Flower is with the Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, Mass. 02139.

M. E. Hellman is with the Department of Electrical Engineering and Stanford Electronics Laboratories, Stanford University, Stanford, Calif. 94305.

0) Right transitions occur only on H, i.e., $p_{ij}(T) = 0$, $i < j$. Left transitions occur only on T, i.e., $p_{ij}(H) = 0$, $i > j$.

1) Transitions occur only between adjacent states.

2) The optimal machine is symmetric: $p_{ij}(H) = p_{m-i+1,m-j+1}(T)$.

3) Randomization will be found only in transitions that lead to states with higher probability of error (i.e., states closer to the center of the machine).

4) The randomization factors for transitions from the end states ($p_{12}(H)$ and $p_{m,m-1}(T)$) will tend monotonically to 0 as $N \to \infty$.

Results

A search for optimal automata with up to 7 states of memory and up to 2^{22} observations was made using a high-speed digital computer. The results are as follows.

Conjecture 0): Though it was necessary to assume Conjecture 0) true to reduce the number of parameters to be searched, there are preliminary indications that it is true. Additionally, there are strong intuitive grounds that support it.

Conjecture 1): It was found to be true that transitions occur only between adjacent states in the optimal machine.

Conjecture 2): It was found to be false that the optimal machine is symmetric. For m odd, a symmetric machine occupying the middle state provides no information in favor of either hypothesis. However, by introducing asymmetries, the occupation of any state provides evidence in favor of one or the other hypothesis. Though the probability of error in the end states increases, the overall effect is a reduction in P_N. For example, consider testing $\Pr\{H\} = 0.75$ versus $\Pr\{H\} = 0.25$ (i.e., $p = 0.75$) with a three-state automaton. The nondeterministic transition factors are graphed in Fig. 1. Though $p_{12}(H) \neq p_{32}(T)$, as N increases the optimal machine approaches symmetry. For $m = 4$ the optimal machine is symmetric. It is now conjectured that the optimal machine is symmetric for m even and asymmetric for m odd. The infinite-sample theory guarantees that as $N \to \infty$ the optimal machine tends to a symmetric form.

Conjecture 3): Conjecture 3) was found to be true. Randomization is found only on transitions leading to states with a higher probability of error. For example, consider testing H_0: $\Pr\{H\} = 0.75$ versus H_1: $\Pr\{H\} = 0.25$ with a seven-state machine constrained to be symmetric. The nondeterministic transition factors are graphed in Fig. 2.

Conjecture 4): It is true that the randomization factors $p_{12}(H)$ and $p_{m,m-1}(T)$ tend monotonically to 0. It is interesting to note that this randomization factor appears to decrease as $1/N$ (see Figs. 1 and 2).

Much can be learned about this behavior by considering a symmetric machine with randomization out of the end states only and with $p_{12}(H) \triangleq p_{m,m-1}(T) = \delta$. This is the exact form of the optimal machine for $m = 4$ and provides an upper bound to P_N^* for $m \neq 4$. Furthermore, we feel that this upper bound is fairly tight and exhibits the correct asymptotic behavior.

Let $P_N(\delta)$ denote the probability of error for such a machine after N observations and let δ_N^* denote the value of δ that minimizes $P_N(\delta)$. Standard methods for finding state-occupation distributions yield the stationary distribution for $N = \infty$, $\mu^i(\infty)$, $i = 0,1$.

The probability of occupying an interior state is proportional to δ. It is easily seen that the probability of error for decisions made in the end states is P_∞^*. Thus, for small δ,

$$P_\infty(\delta) \doteq P_\infty^* + k_1\delta,$$

where the symbol $\doteq$ indicates asymptotic equality.

In [2] it is shown that $1/\delta$ has the significance of a kind of time constant for the machine. Thus we assume that

$$P_N(\delta) \doteq (P_\infty^* + k_1\delta) + k_2e^{-k_3\delta N}.$$

For $m = 3$ it was possible to verify that this is the correct form. Setting $\partial P_N(\delta)/\partial\delta = 0$ yields

$$\delta_N^* = \frac{\ln N}{k_3N} + \frac{\ln (k_2k_3/k_1)}{k_3N}.$$

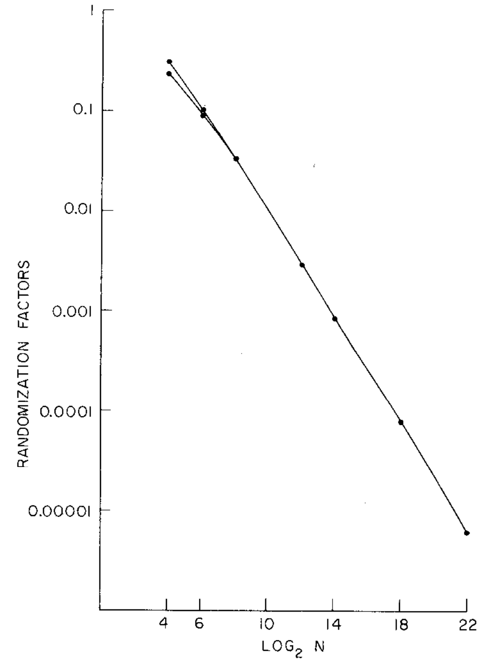

Fig. 1. Values of $p_{12}(H)$ and $p_{32}(T)$ versus $\log_2 N$.

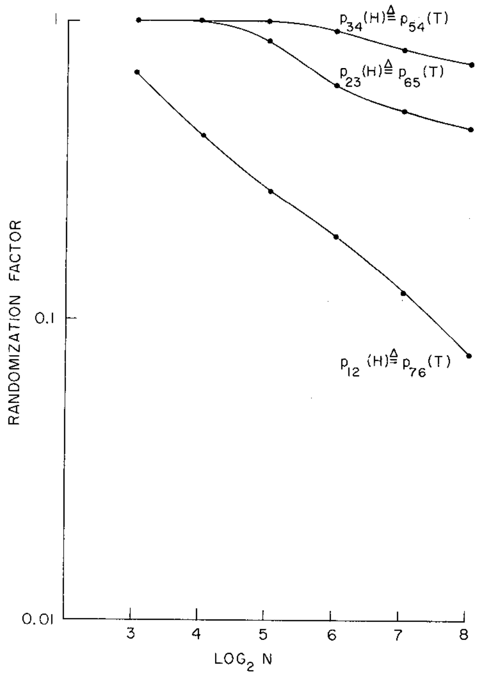

Fig. 2. Randomization factors versus $\log_2 N$.

Using this value of δ, the expression for $P_N(\delta)$ becomes

$$P_N^* = P_N(\delta_N^*) = P_\infty^* + \frac{k_1}{k_3N}\left[1 + \ln N + \ln\left(\frac{k_2k_3}{k_1}\right)\right].$$

Thus, asymptotically, δ_N^* and P_N^* both approach their limits (0 and P_∞^*) as $(\ln N)/N$.

It is of interest to consider how the finite-memory restriction affects the minimum probability of error attainable in finite time. Let $P_{N,\mathrm{sym}}^*(m)$ denote the minimum probability of error attainable with an m-state symmetric machine after N observations and let $P_N^*(\infty)$ denote the minimum probability of error attainable with infinite memory (i.e., the minimum attainable with knowledge of all prior

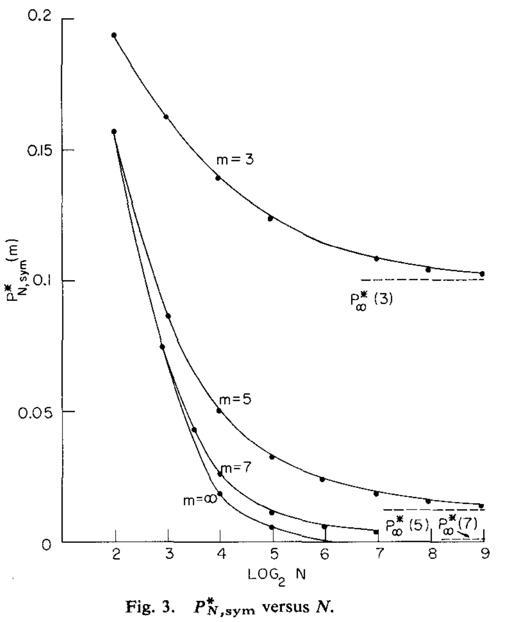

Fig. 3. $P^*_{N,\text{sym}}$ versus N.

observations). Direct calculation of $P_N^*(\infty)$ was possible for small values of N. For larger values of N it became necessary to upper-bound $P_N^*(\infty)$. A Chernoff bound on $P_N^*(\infty)$ is

$$P_N^*(\infty) \leq (2\sqrt{pq})^N/2.$$

Fig. 3 shows $P^*_{N,\text{sym}}(m)$ for various numbers of observations N and various sizes of memory m. The Chernoff bound on $P_N^*(\infty)$ was used for $N \geq 32$.

Quite naturally one does better with more memory. The $P^*_{N,\text{sym}}(m)$ curve for any given value of m follows the $P_N^*(\infty)$ line for low values of N, diverges from it for larger values of N, and approaches a nonzero limit $P_\infty^*(m)$ as $N \to \infty$. This behavior is easily explained. Any given machine can "remember" all of the observations for low values of N. Here infinite memory offers no advantages. For larger values of N, a finite-state machine necessarily loses some information and thus does not do so well as one with infinite memory. As $N \to \infty$, $P^*_{N,\text{sym}}(m)$ approaches $P_\infty^*(m)$, the infinite-time lower bound on the probability of error, since from [1] we know that for $N = \infty$ the optimal machine is symmetric.

Conclusions

Experimentally it has been found to be true that the optimal machine has the conjectured form except for the symmetry requirement. Transitions are only between adjacent states. Randomization is found only in transitions to states with a higher probability of error. As $N \to \infty$, the probability of a transition out of an end state tends monotonically to 0. It was found that P_N^* and $p_{12}(H)$ approach their respective limits (P_∞^* and 0) as $(\ln N)/N$. A more fundamental argument was given for this behavior.

It was felt that further experiments would only tend to confirm the findings presented. It would be more interesting at this point to attempt mathematical proofs of behavior, specifically of the conjectures. It is hoped that these experimental results will aid in this endeavor. Investigation of deterministic machines for continuous-probability distributions is also of interest and is being pursued [3].

References

[1] M. Hellman and T. M. Cover, "Learning with finite memory," *Ann. Math. Statist.*, vol. 41, pp. 765–782, 1970.
[2] M. E. Hellman, "Learning with finite memory," Ph.D. dissertation, Stanford Univ., Stanford, Calif., Mar. 1969.
[3] M. A. Freedman, "A finite memory, finite time, Gaussian hypothesis testing problem," M.S. thesis, Dep. Elec. Eng., Massachusetts Inst. Technol., Cambridge, Sept. 1971.

The Reduced Nearest Neighbor Rule

GEOFFREY W. GATES

Abstract—**A further modification to Cover and Hart's nearest neighbor decision rule, the reduced nearest neighbor rule, is introduced. Experimental results demonstrate its accuracy and efficiency.**

The nearest neighbor rule was originally proposed by Cover and Hart [1], [2] and is currently being used by several workers. One reason for the use of this rule is its conceptual simplicity, which leads to straightforward, if not necessarily the most efficient, programming. In a subsequent paper, Hart [5] suggested a means of decreasing memory and computation requirements. This paper introduces a technique, the reduced nearest neighbor rule, that can lead to even further savings. The results of this new rule are demonstrated by applying it to the "Iris" data [6].

The nearest neighbor rule (NN) is described in several places in the literature [1], [2]. For background a simple statement will be included here. First, some notation must be defined. Assume there are M pattern classes, numbered $1,2,\cdots,M$. Let each pattern be defined in an N-dimensional feature space and let there be K training patterns. Each training pattern is a pair (x^i,θ_i), $1 \leq i \leq K$, where $\theta_i \in \{1,2,\cdots,M\}$ denotes the correct pattern class and

$$x^i = (x_1{}^i,x_2{}^i,\cdots,x_N{}^i)$$

is the set of feature values for the pattern. Let $T_{\text{NN}} = \{(x^1,\theta_1), (x^2,\theta_2),\cdots,(x^K,\theta_K)\}$ be the nearest neighbor training set. Given an unknown pattern x, the decision rule is to decide x is in class θ_j if

$$d(x,x^j) \leq d(x,x^i), \qquad 1 \leq i \leq K,$$

where $d(\cdot,\cdot)$ is some N-dimensional distance metric.

Actually, the preceding rule is more properly called the 1-NN rule, since it uses only one nearest neighbor. An obvious generalization of this is the k-NN rule, which takes the k nearest patterns $i_1,i_2,\cdots,i_k$ and decides upon the pattern class that appears most frequently in the set $\theta_{i_1},\theta_{i_2},\cdots,\theta_{i_k}$.

Hart [5] describes a revised rule called the condensed nearest neighbor rule (CNN). Actually, this is not a new decision rule since it still chooses the class of the nearest neighbor. Rather, the word condensed refers to a procedure for choosing a subset of T_{NN}, which we call T_{CNN}, that should perform almost as well as T_{NN} in classifying unknown patterns. In this case, a possible drop in performance is being traded for greater efficiency, both in the amount of memory required to store the training set and in the computation time required to reach a decision. Simulation can be used to decide whether the increased efficiency is worth the degradation in performance.

In the formation of T_{CNN}, the notion of a consistent subset of T_{NN} is important. It is simply a subset of T_{NN} that will classify all the patterns in T_{NN} correctly. Then the minimal consistent subset is the smallest and therefore the most efficient subset of T_{NN} that will properly classify every pattern in T_{NN}. The minimal consistent subset of T_{NN} is important since, in some sense, it has generalized all the important information out of T_{NN}. It will turn out that T_{CNN} must be consistent, but cannot be guaranteed to be minimal.

The algorithm for constructing T_{CNN} proceeds as follows.

1) The first sample pattern is copied from T_{NN} to T_{CNN}.

2) T_{CNN} is used as the training set to classify each pattern of T_{NN}, starting with the first. This is done until one of the following two cases arises:

Manuscript received June 18, 1971; revised September 28, 1971.
The author is with the Department of Computer Science, Michigan State University, East Lansing, Mich. 48823.