Picking the Best Expert from a Sequence

Ruth Bergman*
Laboratory for Artificial Intelligence
Massachusetts Institute of Technology
Cambridge, MA 02139

Ronald L. Rivest†
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

March 17, 1995

Abstract

We examine the problem of finding a good expert from a sequence of experts. Each expert has an "error rate"; we wish to find an expert with a low error rate. However, each expert's error rate is unknown and can only be estimated by a sequence of experimental trials. Moreover, the distribution of error rates is also unknown. Given a bound on the total number of trials, there is thus a tradeoff between the number of experts examined and the accuracy of estimating their error rates.

We present a new expert-finding algorithm and prove an upper bound on the expected error rate of the expert found. A second approach, based on the sequential ratio test, gives another expert-finding algorithm that is not provably better but which performs better in our empirical studies.

1 Introduction

Suppose you are looking for an expert, such as a stock broker. You have limited resources and would like to efficiently find an expert who has a low error rate. There are two issues to face. First, when you meet a candidate expert you are not told his error rate, but can only find this out experimentally. Second, you do not know a priori how low an error rate to aim for. We give here an algorithm to find a good expert given limited resources, and show that the algorithm is efficient in the sense that it finds an expert that is almost as good as the expert you could find if each expert's error rate was stamped on his forehead (given the same resources).

If each expert's error rate were stamped on his forehead then finding a good expert would be easy. Simply examine the experts one at a time and keep the one with the lowest error rate. If you may examine at most n experts you will find the best of these n experts, whose expected error rate we denote by b_n. You cannot do any better than this without examining more experts.

Since experts do not typically come marked with their error rates, you must test each expert to estimate their error rates. We assume that we can generate or access a sequence of independent experimental trials for each expert.

If the number of available experts is finite, you may retain all of them while you test them. In this case the interesting issues are determining which expert to test next (if you cannot test

*Supported by ARO grant N00014-89-J-1988, NSF grant 9217041-ASC (funded in part by the DARPA HPCC program), NSF grant CCR-9310888, and the Siemens Corporation. email address: ruth@ai.mit.edu

†Supported by ARO grant N00014-89-J-1988, NSF grant 9217041-ASC (funded in part by the DARPA HPCC program), NSF grant CCR-9310888, and the Siemens Corporation. email address: rivest@theory.lcs.mit.edu

all the experts simultaneously), and determining the best expert given their test results. These issues have been studied in reinforcement learning literature and several interesting algorithms have been developed (see Watkins (1989), Sutton (1990), Sutton (1991), and Kaelbling (1990) for some examples).

Here we are interested in the case where we may test only one expert at a time. The problems in this case are: (1) what is the error rate of a "good" expert, and (2) how long do we need to test an expert until we are convinced that he is good or bad?

First consider the case that we have a predetermined threshold such that an error rate below this threshold makes the expert "good" (acceptable). This is a well-studied statistical problem. There are numerous statistical tests available to determine if an expert is good; we use the ratio test which is the most powerful among them. The ratio test is presented in section 3.1.

However, in our problem formulation we have no prior knowledge of the error rate distribution. We thus do not have an error-rate threshold to define a good expert, and so cannot use the ratio test. The algorithm in section 3.2 overcomes this limitation by setting lower and lower thresholds as it encounters better experts. Section 3 contains the main result of this paper: our algorithm finds an expert whose error rate is close to the error rate of the best expert you can expect to find given the same resources.

Section 4 presents a similar expert-finding algorithm that uses the sequential ratio test (Wald 1947) rather than the ratio test. Wald (1947) shows empirically that the sequential ratio test is twice as efficient as the ratio test when the test objects are normally distributed. While the theoretical bound we give for the sequential-ratio expert-finding algorithm is weaker than the bound for the ratio-test expert-finding algorithm, empirical results with specific distributions in section 5 indicate that the former algorithm performs better in practice.

2 An AI Application: Learning World Models

Consider the problem of learning a world model where rules describe causal relationships of the environment. A rule has the form

$$\text{precondition} \rightarrow \text{action} \rightarrow \text{postcondition}$$

with the meaning that if the preconditions are true in the current state and the action is taken, then the postcondition will be true in the next state. These are predictive rules as in (Drescher 1989), as opposed to the prescriptive rules in reinforcement learning (Watkins 1989, Holland 1985) or operators in Soar (Laird, Newell & Rosenbloom 1978).

An algorithm to learn rules uses triples of previous state, S, action, A, and current state to learn. It may isolate a postcondition, P, in the current state, and generate preconditions that explain the postcondition from the previous state and action. For any precondition PC that is true in state S, the rule $\text{PC} \rightarrow \text{A} \rightarrow \text{P}$ has some probability p of predicting incorrectly. To learn a world model, the algorithm must find the rules with low probability of prediction error, and discard rules with high probability of prediction error.

The problem of finding a good rule to describe the environment is thus an expert-finding problem. It fits into the model discussed here since (1) each rule has an unknown error rate, (2) the distribution of rules' error rates is unknown and depends both on the environment and the learning algorithm, and (3) the learning algorithm can generate arbitrarily many rules.

3 Finding Good Experts from an Unknown Distribution

First, let us reformulate the expert-finding problem as a problem of finding low error-rate coins from an infinite sequence $c_1, c_2, \ldots$ of coins, where coin c_i has probability r_i of "failure" (tails) and probability $1 - r_i$ of "success" (heads). The r_i's are determined by independent draws from the interval $[0, 1]$, according to some unknown distribution. We want to find a "good" coin, i.e. a coin with small probability r_i of failure (error). We are not given the r_i's, but must estimate them using coin flips (trials).

The main result of this section is:

Theorem 1 *There is an algorithm (algorithm* **FindExpert***) such that when the error rates of drawn coins are unknown quantities drawn from an unknown distribution, after t trials, with probability at least $1 - 1/t$, we expect to find a coin whose probability of error is at most $b_{t/\ln^2 t} + O(\frac{1}{\sqrt{\ln t}})$.*

This theorem states that after t trials, we expect the algorithm to find an expert that is almost as good as the best expert in a set of $t/\ln^2 t$ randomly drawn experts (who would have error rate $b_{t/\ln^2 t}$). We note that our result depends in a natural manner on the unknown distribution.

Recall that in t trials if the experts' error rates are known we can find the best of t experts' error rates (b_t). Compared to this, our algorithm must examine fewer experts because it must spend time estimating their error rates. For some distributions (such as for fair coins) $b_{t/\ln^2 t}$ and b_t are equal, while for other distribution they can be quite far apart.

The rest of this section gives the ratio test and our algorithm for finding a good expert.

3.1 The Ratio Test

Since we do not know the error rates of the coins when we draw them, we must estimate them by flipping the coins. If we knew that "good" coins have error rate at most p_1, we could use standard statistical tests to determine if a coin's error rate is above or below this threshold. Because it is difficult to test coins that are very close to a threshold, we instead use the ratio test, which tests one hypothesis against another. In this case the hypotheses are that the coin has error rate at most p_0, versus that the coin has error rate at least p_1, where p_0 is a fixed value less than p_1.

The Problem Given a coin with unknown rate of failure p.
Test if $p \leq p_0$ vs. $p \geq p_1$. Accept if $p \leq p_0$. Reject if $p \geq p_1$.

Requirements The probability of rejecting a coin does not exceed α if $p \leq p_0$, and the probability of accepting a coin does not exceed β if $p \geq p_1$. [1]

The Test Let m be the number of samples, and f_m be the number of failures in m samples. The ratio test is

$$\begin{array}{ll} reject & \text{if } f_m \geq (p_0 + \sqrt{\frac{\ln 1/\alpha}{2m}})m \\ accept & otherwise \end{array}$$

[1] We choose the ratio test since it has the most power, i.e., for a given α, i.e. it gives the least β (probability of accepting when the hypothesis H_0 is wrong (see (Rice 1988).)

3.2 An Algorithm for Finding a Good Expert

We know how to test if a coin is good given a threshold defining a good error rate, but when we do not know the error-rate distribution we can not estimate the lowest error rate b_t that we can expect to achieve in t trials. The following algorithm overcomes this handicap by finding better and better coins and successively lowering the threshold for later coins.

The algorithm for finding a good coin is the following.

Algorithm 1 FindExpert
Input: *t, an upper bound on the number of trials (coin flips) allowed.*

Let BestCoin = Draw a coin.
Flip BestCoin $\ln^3 t$ *times to find* $\widehat{p}$.
Set $p_1 = \widehat{p}$.
Repeat until all t *trials are used*
  Let $p_0 = p_1 - \epsilon(p_1)$, *where* $\epsilon(p_1) = \sqrt{4/\ln(t)}$.
  Let Coin = Draw a coin.
  Test Coin using the ratio test:
    Flip Coin $m = \ln^2 t$ *times.*
    Accept if $f_m < (p_1 - \epsilon(p_1)/2)m$.
  If the ratio test accepted then
    Set BestCoin = Coin.
    Flip BestCoin an additional $\ln^3 t$ *times to find an improved* $\widehat{p}$.
    Set $p_1 = \widehat{p}$.
Output BestCoin.

The proof that **FindExpert** satisfies the statement of Theorem 1 is too lengthy for this paper. The following is a high level summary of the proof.

Description of the proof: Since the error-rate distribution is unknown, we do not have any estimate of b_t, so the algorithm uses better and better estimates. It starts with a random coin and a good estimate of its error rate. It prepares a test to determine if a new coin is better than the current coin (with high probability). Upon finding such a coin it prepares a stricter test to find a better coin, and so on. We show that *the time to test each coin is short*, and thus *we see many coins*. Since *we almost always keep the better coin* we can *find a coin whose error rate is at most the expected best error rate of the coins that the algorithm saw (plus a small correction).*

4 A Faster (?) Test for Experts

A disadvantage of the ratio test in the previous section is that the length of each test is fixed. This length is chosen so as to guarantee (with high probability) a good determination as to whether the tested coin has error rate at least ϵ better than the current best coin. For coins that are much better or much worse, it may be possible to make this determination with many fewer trials.

The sequential ratio test given by Wald (1947) solves precisely this problem. After each coin toss it assesses whether it is sufficiently sure that the tested coin is better or worse than the current best coin. If not, the test continues. The sequential ratio test thus uses a *variable* number of flips to test a coin. One can hope that for same probability of erroneous acceptances and rejections, the sequential ratio test will use fewer coin flips than the ratio test. Although the worst case sample size is larger for the sequential ratio test, Wald (1947) shows that in experiments with normally

distributed error rates the sequential test is on average twice as efficient as the ratio test. Section 5 gives our experimental results comparing expert-finding algorithms based on the ratio test and on the sequential ratio test.

The rest of this section gives the sequential ratio test and the corresponding expert-finding algorithm.

4.1 The Sequential Ratio Test

This section describes the sequential ratio test due to Wald (1947).

The Problem Given a coin with unknown failure rate p.
Test if $p \leq p_0$ vs. $p \geq p_1$. Accept if $p \leq p_0$. Reject if $p \geq p_1$.

Requirements The probability of rejecting a coin does not exceed α if $p \leq p_0$, and the probability of accepting a coin does not exceed β if $p \geq p_1$.

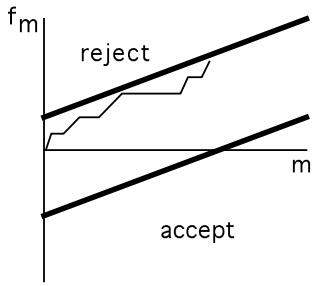
The Test Let m be the number of samples, and f_m be the number of failures in m samples.
Reject if

$$f_m \geq \frac{\log \frac{1-\beta}{\alpha}}{\log \frac{p_1}{p_0} - \log \frac{1-p_1}{1-p_0}} + m \frac{\log \frac{1-p_0}{1-p_1}}{\log \frac{p_1}{p_0} - \log \frac{1-p_1}{1-p_0}}.$$

Accept if

$$f_m \leq \frac{\log \frac{\beta}{1-\alpha}}{\log \frac{p_1}{p_0} - \log \frac{1-p_1}{1-p_0}} + m \frac{\log \frac{1-p_0}{1-p_1}}{\log \frac{p_1}{p_0} - \log \frac{1-p_1}{1-p_0}}.$$

Otherwise, draw another sample.



The sequential ratio test defines two lines with different intercepts and the same slope. Above the upper line is a reject region. Below the lower line is the accept region. The test generates a random walk starting at the origin which terminates when it reaches one of the two lines.

4.2 Finding a Good Expert Using the Sequential Ratio Test

The algorithm for finding a good coin using the sequential ratio test is as follows.

Algorithm 2 SeqFindExpert:
Input: *t, an upper bound on the number of trials allowed.*

Let BestCoin = Draw a coin.
Flip BestCoin $\ln^3 t$ *times to find* $\widehat{p}$.
Set $p_1 = \widehat{p}$.
Repeat until all t trials are used:
  Let $p_0 = p_1 - \epsilon(p_1)$, *where* $\epsilon(p_1) = \sqrt{\frac{4p_1(1-p_1)}{\log t}}$.
  Let Coin = Draw a coin.
  Test Coin using the sequential ratio test
    with parameters p_0, p_1, *and* $\alpha = \beta = 1/t^2$.
  If the sequential test accepts then
    Set BestCoin = Coin.
    Flip BestCoin $\log^3 t$ *more times to find an improved* $\widehat{p}$.
    Set $p_1 = \widehat{p}$.
Output BestCoin.

Because the worst case number of coin flips for the sequential ratio test is larger than the (fixed) number of coin flips for the ratio test, the bound we now prove for **SeqFindExpert** ratio test is not as strong as the bound shown above for **FindExpert**.

Theorem 2 *There is an algorithm (***SeqFindExpert***) such that when the coins are drawn according to an unknown error-rate distribution, after t trials, with probability at least* $1 - 1/t$, *we expect to find a coin whose probability of error is at most* $b_{t/\log^3 t} + O(\frac{1}{\sqrt{\log t}})$.

Theorem 2 shows that algorithm **SeqFindExpert**, which uses the sequential ratio test to find a low error-rate coin from coins drawn according to an unknown distribution, does almost as well as we can do if coins were labeled with their error rates, but see only $t/\log^3 t$ coins. The proof of Theorem 2 is similar to the proof of Theorem 1. The bound in Theorem 2 is not as tight as the bound for the **FindExpert**. In practice, however, **SeqFindExpert** often performs better because the test lengths are much shorter than the worst case test length used to prove Theorem 2.

For some distributions, such as the uniform distribution, the coins tested are typically *much* worse than the current best. (After seeing a few coins the algorithm already has a fairly good coin and most coins are much worse.) Thus, the sequential ratio tests will be short. When the error rates are uniformly distributed we expect that the algorithm **SeqFindExpert** will see more coins and find a better coin than **FindExert**. This argument is confirmed by our empirical results below. Our results also show the superiority of **SeqFindExpert** when the error rates are drawn from a (truncated) normal distribution.

5 Empirical Comparison of FindExpert and SeqFindExpert

To compare the performance of **FindExpert** and **SeqFindExpert** we ran experiments for uniform and normally distributed error rates. (The normal distribution was truncated to lie within the interval $[0, 1]$.) Table 1 gives results for both algorithms on the uniform distribution. All results reported are an average over 1000 repeated executions of the algorithm. Table 1(a) contains the average of 1000 runs each with trial limit $t = 1000$. Table 1(a) shows that the **SeqFindExpert** algorithm had shorter average test lengths and therefore tested more experts. **SeqFindExpert**

	Coins Tested	Test Length	Best Estimated Error Rate	Best Actual Error Rate
FindExpert	7.6	49	.1085	.1088
SeqFindExpert	21	44	.0986	.0979

(a) Uniform distribution; limit of $t = 1000$ trials.

	Coins Tested	Test Length Test Length	Best Estimated Error Rate	Best Actual Error Rate
FindExpert	66	100	.0185	.0187
SeqFindExpert	230	33	.01	.0101

(b) Uniform distribution; limit of $t = 10000$ trials.

Table 1: Empirical Comparison of **FindExpert** and **SeqFindExpert** with the uniform distribution. The numbers in the tables are averaged over 1000 runs.

was able to find experts with lower actual error rate (.0979 on the average compared with .1088 for **FindExpert**). The table contains both the average actual error rate of the best experts that the algorithm found and the average error rate from experiments for the same experts. Table 1(b) shows that given more time ($t = 10000$ trials) to find a good expert **SeqFindExpert** performs significantly better than **FindExpert**. The average test length is much shorter and the resulting best error rate is .0101 compared with .0187.

	Coins Tested	Test Length	Best Estimated Error Rate	Best Actual Error Rate
FindExpert	13	49	.4361	.4395
SeqFindExpert	29	61	.4144	.4204

(a) Normal distribution; limit of $t = 1000$ trials.

	Coins Tested	Test Length	Best Estimated Error Rate	Best Actual Error Rate
FindExpert	85	100	.3292	.3352
SeqFindExpert	470	31	.2670	.2741

(b) Normal distribution; limit of $t = 10000$ trials.

Table 2: Empirical Comparison of **FindExpert** and **SeqFindExpert** with the Normal Distribution (mean 0.5, variance 0.3) truncated at 0 and 1. The numbers in the tables are averaged over 1000 runs.

Experiments with the normal distribution used a normal with mean 0.5 and variance 0.3. These results are reported in table 2. Note that for this distribution most coins have error rate close to .5. Table 2(a) reports the average of 1000 executions with trial limit 1000. It is interesting that the **SeqFindExpert** both tested more experts and had a longer average test length. The long average test is due to a few very long tests (to compare close experts), but most tests are very short. As expected, the average error probabilities of the best coin is lower for the **SeqFindExpert** algorithm.

Table 2(b) shows that with a longer limit of 10000 trials the **SeqFindExpert** algorithm performs much better than **FindExpert**, giving an average best error rate of .2741 compared with .3352.

The experimental results in this section show that **SeqFindExpert** performs better than **FindExpert** for two distributions with different characteristics. The experimental results agree with the theoretical analysis in that some sequential tests are quite long (longer than the ratio tests), but the experiments also show that on the average the sequential test lengths are short especially when the trial limit is large. The average test length is short when the time limit is large because the best expert is already much better than the average population.

6 Conclusions

This paper presents two algorithms to find a low error expert from a sequence of experts with unknown error-rate distribution, a problem that arises in many areas, such as the given example of learning a world model consisting of good rules. The two algorithms **FindExpert** and **SeqFindExpert** are nearly identical, but use the ratio test and sequential ratio test respectively to determine if an expert is good.

Theorem 1 shows that **FindExpert** finds an expert which is the best expert of $t/\ln^2 t$, given trial limit t. This result is strong in the sense that it shows only a factor of $\ln^2 t$ loss from testing over the best expert we could find in t trials if we knew the exact error rate of each expert. Theorem 2 gives a weaker bound for **SeqFindExpert**. Empirical results in section 5, on the other hand, indicate that **SeqFindExpert** performs better than **FindExpert** in practice (at least for the uniform and normal distributions).

The obvious open question from this work is to prove that **SeqFindExpert** expects to find a lower error-rate expert for general or specific distributions than **FindExpert**.

References

Drescher, G. L. (1989), Made-Up Minds: A Constructivist Approach to Artificial Intelligence, PhD thesis, MIT.

Holland, J. H. (1985), Properties of the bucket brigade algorithm, *in* 'First International Conference on Genetic Algorithms and Their Applications', Pittsburg, PA, pp. 1–7.

Kaelbling, L. P. (1990), Learning in Embedded Systems, Technical Report TR-90-04, Teleos Research.

Laird, J. E., Newell, A. & Rosenbloom, P. S. (1978), 'SOAR: An Architecture for General Intelligence', *Artificial Intelligence* **33**, 1–64.

Rice, J. A. (1988), *Mathematical Statistics and Data Analysis*, Wadsworth & Brooks/Cole, Pacific Grove, CA.

Sutton, R. S. (1990), First Results with DYNA, an Integrated Architecture for Learning, Planning, and Reacting, *in* 'Proceedings, AAAI-90', Cambridge, Massachusetts.

Sutton, R. S. (1991), Reinforcement Learning Architectures for Animats, *in* 'First International Conference on Simulation of Adaptive Behavior', The MIT Press, Cambridge, MA.

Wald, A. (1947), *Sequential Analysis*, John Wiley & Sons, Inc., Chapman & Hall, LTD., London.

Watkins, C. (1989), Learning from Delayed Rewards, PhD thesis, King's College.