A small(-ish) case

Introduction

A few weeks ago we faced a problem that can be (re)formulated in the following way. A sequence of N integers are drawn from a uniform integer distribution with interval bounds $[1, b]$ where the upper bound b is unknown. You see one realization from this distribution at a time and the task is to estimate the value b (with some confidence level) so that you get "close enough" to the true value b.

In many cases N might be very large (up in the millions), but we don't want to go through every single element to find the value b if we can get a good estimate simply by going through $n \ll N$ and get a good estimate from that.

A small example

Let's set $N = 15$ and let us assume that we know that the upper bound b is given by $b = 10$. Hence, the uniform integer distribution is given by:

$$P(i|a = 1, b = 10) = \frac{1}{b - a + 1} = \frac{1}{10} \tag{1}$$

for $i = 1, \dots, 10$.

Suppose we let $n = 15$ before we make our guess. One particular realization might be given by:

$$6, 9, 5, 1, 6, 7, 6, 1, 8, 9, 5, 2, 5, 3$$

Let b' be the estimate of the true value b. One approach might be to take $b' = \max\{6, 9, 5, 1, 6, 7, 6, 1, 8, 9, 5, 2, 5, 3\}$ of the sequence, which gives $b' = 9$. The deviation here is $b - b' = 1$, which is a fairly good approximation.

Another approach might be to set $n = 5$, and multiply that number by, say, 30%. Then, our estimate of b would be $\max\{6, 9, 5, 1, 6, 7\} * 1.30 = 9 * 1.3 = 11.7$. Because we know that b must be an integer we can do better by rounding it to the closest integer, which in this case is 12. The error here is $b - b' = 12 - 10 = 2$, which is slightly worse. On the other hand, this approach looked at far less samples ($n = 5$ instead of $n = 15$).

Problem formulation

The following parameters are given:

I The length of the entire sequence, N_t on problem instance t

II Allowed estimation error, m_t on problem instance t

III Number of problem instances T

Next, define the *hit* function as follows, where m_t, b_t and b'_t denotes the allowed deviation, correct upper bound and estimated upper bound on problem t, respectively:

$$h(b'_t) = \begin{cases} 1, & \text{if } b_t - m_t \le b'_t \le b_t + m_t \\ 0, & \text{otherwise} \end{cases}$$

And for a given trial t we define the penalty as:

$$\frac{n_t}{N_t} + (1 - h(b'_t)) \tag{2}$$

In the second example, we have $N_t = 15$ and $n_t = 5$. Furthermore, the true b_t and predicted b'_t is 10 and 12, respectively. If $m_t = 1$, then no hit occurs, and the penalty is $\frac{n_t}{N} + (1 - h(b'_t)) = \frac{5}{10} + 1 = 1.5$. If $m_t = 3$ (for example), then a hit occurs and the penalty is only $\frac{n_t}{N} = 0.5$.

Now we seek to minimize the following penalty function:

$$L = \min \sum_{t=1}^{T} \frac{n_t}{N_t} + (1 - h(b'_t)) \tag{3}$$

Or, equivalently, maximizing the score function:

$$\text{mean score} = 1 - L/T \tag{4}$$

Some interpretation of equation 2: The first term (n_t/N_t) is the ratio of the numbers you see against the total numbers in the sequence before making a prediction. The fewer, the better. The second term introduce an additional penalty of 1 every time there is a miss (i.e. no hit). The penalty is pretty hard, so there is a trade-off between looking at as few numbers as possible before making a guess and looking at enough numbers so that the guess is ”good enough”.

Equation 4 is simply a ”score function” that is initially 1 and decreases with the average penalty.

Code submission

Code is written in numpy and we would like the submission also be written in python. Feel free to use any library you would like. If you are more comfortable in another language, feel free to use that language as well (but I guess that it might be some work associated with setting the required functions up).

This can be solved by just trying a bunch of different strategies that seem to work fine. Also, using statistical properties might be a good approach (for example, there's a lot of information about the continuous uniform distribution on Wikipedia).

We will mainly look at *how* the problem is solved and the achieved mean score. We don't care if you use external libraries or the algorithm is running very slowly, as long as the code is well written and it achieves a reasonably good mean score. Good luck!