Indian Statistical Institute, Kolkata

Project for 'Statistical Methods -II'

Bachelor of Statistics (Hons.), 2023-26

Sequential Testing

Deep Jayesh Hariya (BS2324)
Abhirup Mistry (BS2302)
Aditya Aryan (BS2305)

Drishti Singla (BS2325)
Mrittika Giri (BS2332)
Nandita Sen (BS2333)

Advisor/Instructor

DR. ARNAB CHAKRABORTY

Associate Professor, Applied Statistics Unit

Wednesday 27th March, 2024

Contents

1 Abstract

This project delves into the exploration of sequential testing.

The desired test procedure should split the sample space into three parts: accept, reject and continue. This project will explore a simple toy version of this: there are 20 closed boxes, exactly 5 of which contain a Rs 5 coin. We have to pay Re 1 to open a box. We intend to find out the best strategy to maximise the expected gain.

2 The Optimal Approach

Building upon insights gained from our previous unsuccessful attempts, we opted for a fresh perspective to tackle the challenge. Our exploration led us to explore recursive solutions to our optimisation problem.

We examine expected gain at every state of our process and decide whether to stop the process or continue opening. Thus obtaining expected gain at every state of the process, assigning it zero in cases of negative expected gain i.e. deciding not to open any further.

2.1 Notations

Assume total number of boxes to be N, where we have a total of C reward boxes. After opening i ($\leq N$) arbitrary boxes, let x be the the number of reward boxes obtained having monetary value V, and y be the number of empty boxes obtained. Each box can be opened by expenditure of 1 unit currency. Thus we denote a game state as $S(x, y)$.

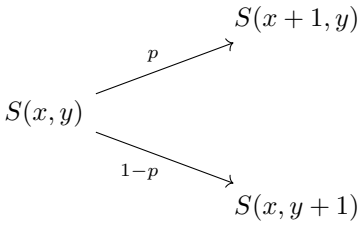
Let $\sigma_{x,y}$ denote any strategy for the game state $S(x, y)$ and $\Sigma_{x,y}$ denote set of all possible strategies for the game state $S(x, y)$ i.e. $\sigma_{x,y}$ defines a sequence of actions for subsequent game states, and $\Gamma_{\sigma_{x,y}}$ denote the gain of strategy $\sigma_{x,y}$ for game states after $S(x, y)$ on any random permutation of coin boxes.
Hence, $\mathbb{E}\left(\Gamma_{\sigma_{x,y}}\right)$ would be the expected gain of the strategy $\sigma_{x,y}$ over all permutations of coin boxes.

2.2 Backtracking Algorithm

At any game state $S(x, y)$ the game can either proceed further to game state $S(x + 1, y)$ or $S(x, y + 1)$. Let p be the probability of getting a reward box next i.e. advancing to game state $S(x + 1, y)$, hence $1 - p$ will be the probability of getting an empty box next i.e. advancing to game state $S(x, y + 1)$. p can be calculated as:

$$p = \frac{C - x}{N - x - y} \qquad x \leq C,\ y \leq N - C$$

obtained by dividing the number of remaining reward boxes $(C - x)$ by the total number of unopened boxes $(N - x - y)$.



Thus for any strategy $\sigma_{x,y}$ at game state $S(x, y)$, it contains information about all possible actions in the subsequent game states, $\sigma_{x+1,y}$ and $\sigma_{x,y+1}$ would contain the tail information about actions after the state $S(x, y)$. Thus it can be defined as sum of indicators as follows:
Similarly, $\Gamma_{\sigma_{x,y}}$ can be written as following accounting for the expenditure and gains when moving to the respective game states:
Definition: Now, we define a function $\textsc{gain}(x, y, C, V, N)$

$$\textsc{gain}(x, y, C, V, N) := \begin{cases} \max(\lambda, 0) & x < C \\ 0 & x = C \end{cases}$$

where

$$\lambda = p \cdot (V - 1 + \textsc{gain}(x + 1, y, C, V, N)) + (1 - p) \cdot (-1 + \textsc{gain}(x, y + 1, C, V, N))$$

With base case returning 0 if all reward boxes are opened i.e. $x \geq C$

Proposition 1. *Function* $\textsc{gain}(x, y, C, V, N)$*, defined recursively calculates the maximum achievable expected gain at any game state* $S(x, y)$ *i.e.*

$$\textit{GAIN}(x, y, C, V, N)) = \max_{\sigma_{x,y}} \mathbb{E}\left(\Gamma_{\sigma_{x,y}}\right)$$

Proof. We prove the proposition by backward induction.
Base case: If we open all coin boxes i.e. $x = C$ then there are no more possible gains. Hence, $\max_{\sigma_{x,y}} \mathbb{E}\left(\Gamma_{\sigma_{x,y}}\right) = 0$.

$$x = C \implies \textsc{gain}(x, y, C, V, N)) = 0$$

Hence our condition

$$\textsc{gain}(x, y, C, V, N)) = \max_{\sigma_{x,y}} \mathbb{E}\left(\Gamma_{\sigma_{x,y}}\right)$$

is true under base case.

Induction Hypothesis: Assume that at any game state $S(x, y)$ such that $x + y \geq k \ \ \forall\ k \leq N$

$$\textsc{gain}(x, y, C, V, N)) = \max_{\sigma_{x,y}} \mathbb{E}\left(\Gamma_{\sigma_{x,y}}\right)$$

Inductive Step: To prove: $\textsc{gain}(x, y, C, V, N)) = \max_{\sigma_{x,y}} \mathbb{E}\left(\Gamma_{\sigma_{x,y}}\right)$ such that $x + y = k - 1$

Since we always move to states $S(x + 1, y)$ and $S(x, y + 1)$ from $S(x, y)$, $\sigma_{x,y}$ can be derived as

$$\sigma_{x,y} = \mathbb{I}_{S(x+1,y)}\, \sigma_{x+1,y} + \mathbb{I}_{S(x,y+1)}\, \sigma_{x,y+1}$$

Similarly, $\Gamma_{\sigma_{x,y}}$ can be derived as:

$$\Gamma_{\sigma_{x,y}} = \mathbb{I}_{S(x+1,y)}\left(V - 1 + \Gamma_{\sigma_{x+1,y}}\right) + \mathbb{I}_{S(x,y+1)}\left(\Gamma_{\sigma_{x,y+1}} - 1\right)$$

Thus the expected gain is:

$$\mathbb{E}[\Gamma_{\sigma_{x,y}}] = \mathbb{E}[\mathbb{I}_{S(x+1,y)}\left(V - 1 + \Gamma_{\sigma_{x+1,y}}\right) + \mathbb{I}_{S(x,y+1)}\left(\Gamma_{\sigma_{x,y+1}} - 1\right)]$$

Now since $\mathbb{I}_{S(x+1,y)}$ and $\Gamma_{\sigma_{x+1,y}}$ are independent, also $\mathbb{I}_{S(x,y+1)}$ and $\Gamma_{\sigma_{x,y+1}}$ are independent.

$$\begin{aligned}
&= \mathbb{E}[\mathbb{I}_{S(x+1,y)}]\,\mathbb{E}[\left(V - 1 + \Gamma_{\sigma_{x+1,y}}\right)] + \mathbb{E}[\mathbb{I}_{S(x,y+1)}]\,\mathbb{E}[\left(\Gamma_{\sigma_{x,y+1}} - 1\right)] \\
&= p\left(V - 1 + \mathbb{E}[\Gamma_{\sigma_{x+1,y}}]\right) + (1 - p)\left(\mathbb{E}[\Gamma_{\sigma_{x,y+1}}] - 1\right)
\end{aligned}$$

Therefore, for maximum expectation over all $\sigma_{x,y}$ we have

$$\begin{aligned}
\max\left(\mathbb{E}[\Gamma_{\sigma_{x,y}}]\right) &= \max\left(p\left(V + \mathbb{E}[\Gamma_{\sigma_{x+1,y}}] - 1\right) + (1 - p)\left(\mathbb{E}[\Gamma_{\sigma_{x,y+1}}] - 1\right)\right) \\
&= p\left(V + \max\left(\mathbb{E}[\Gamma_{\sigma_{x+1,y}}]\right) - 1\right) + (1 - p)\left(\max\left(\mathbb{E}[\Gamma_{\sigma_{x,y+1}}]\right) - 1\right) \\
&= p\left(V + \textsc{gain}(x + 1, y, C, V, N)\right) - 1) + (1 - p)\left(\textsc{gain}(x, y + 1, C, V, N)\right) - 1) \\
&= \textsc{gain}(x, y, C, V, N))
\end{aligned}$$

This proves that the gain function gives the maximum return for $x + y = k - 1$ as well. Thus, by backward induction, we have proved that the function $\textsc{gain}(x, y, C, V, N)$, defined recursively, calculates the maximum achievable expected gain at any game state $S(x, y)$.

□

Algorithm 1 Recursion Algorithm

```
function GAIN(x, y, C, V, N)
    if x ≥ C or y > N − C then
        return 0
    end if
    p ← (C − x)/(N − x − y)
    λ ← p × (V − 1 + GAIN(x + 1, y, C, V, N)) + (1 − p) × (GAIN(x, y + 1, C, V, N) − 1)
    return max(λ, 0)
end function
```

2.3 Analysing the Problem

The problem deals with a specific case of our generalised version of the problem setup. For the specific case of C = 5, V = ₹5, and N = 20, the gain function is called, with initial parameters:
$x = 0$ (No reward boxes obtained)
$y = 0$ (No empty boxes obtained)
The given backtracking algorithm yields the maximum achievable expected gain for the game state $S(0, 0)$ i.e.

$$\boxed{\textsc{gain}(0, 0, 5, 5, 20) = \mathbf{7.6875644994840036}}$$

2.3.1 Constructing the Optimal strategy

To construct the optimal strategy, we establish cutoffs by determining the maximum number of boxes that can be opened while ensuring a positive expected gain. We employed our backtracking strategy to compute a cutoff value, utilizing the function $\textsc{gain}(x, y, C, V, N)$.

The process involves iterating over the total number of boxes, denoted as n, for a fixed number of reward boxes, denoted as c. At each iteration, we calculate the gain and return the maximal positive n for a positive gain. This calculation is performed for varying values of c, where $c \in \{1, 2, 3, 4, 5\}$.

Algorithm 2 Cutoff Algorithm

```
n ← 1
for c = 1, ..., 5 do
    G ← 1
    while G > 0 do
        n ← n + 1
        G ← GAIN(0, 0, c, 5, n)
    end while
    return [c, n − 1]
end for
```

xCalculating cutoffs from the above algorithm we get:

Reward Boxes(c)	1	2	3	4	5
Cutoff(n_c)	8	15	21	27	33

Now we construct our strategy for the problem statement using these cutoff values. Since the cutoff for three reward boxes exceeds the total number of boxes, we aim to open boxes until we obtain three reward boxes. Subsequently, we initiate another opening spree if the number of remaining boxes is less than or equal to the cutoff for two boxes, denoted as n_2 (which is 15 in this case), with the objective of obtaining one more reward. Finally, if the remaining number of boxes is less than or equal to the cutoff for one box, denoted as n_1 (which is 8) after the second opening spree, we continue opening boxes until completion. This sequential approach maximizes our expected gain while efficiently utilizing the determined cutoff values.

Algorithm 3 Derived strategy

```
α ← 5
for i = 1, ..., 20 do
    Open the box
    if A coin is found then
        α = α − 1
    end if
    if n_α < 20 − i then
        break
    end if
end for
```

After applying this strategy we saw that the expected gain of this strategy coincides with the value of $\textsc{gain}(0, 0, 5, 5, 20)$ (refer to A.3). Hence we conclude that this is our optimal strategy which maximises expected gain over oll possible permutations of coin boxes.

2.3.2 Analysing Cutoffs

We investigated the convergence of cutoff values with respect to the number of reward boxes. Employing our algorithm, cutoff values were computed for varying quantities of reward boxes. Subsequently, we generated a scatter plot to visually represent the relationship between the number of reward boxes and their corresponding cutoff values:

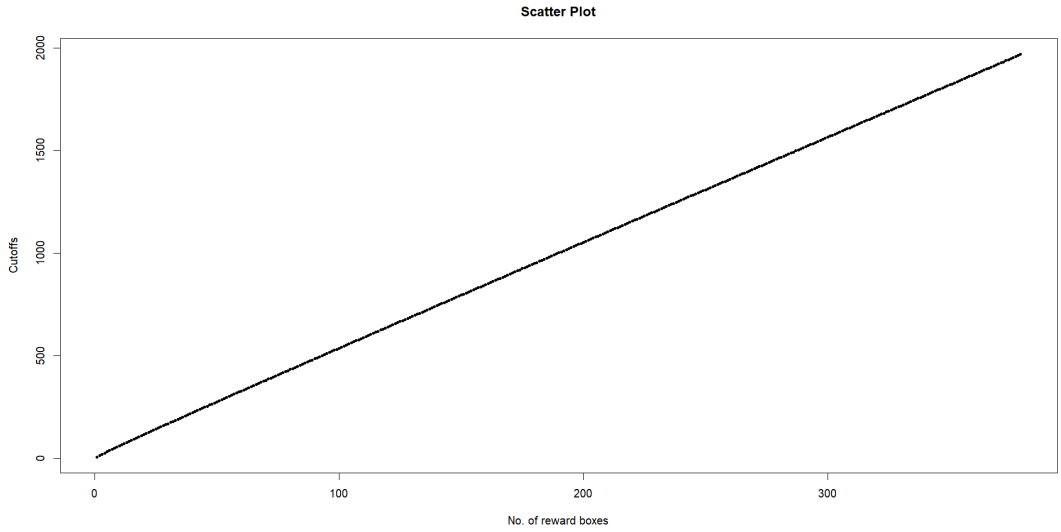


Figure 2.1: Scatter plot

Our analysis reveals a consistent linear increase in cutoff values as the number of reward boxes escalates.The observed linear trend in cutoff values indicates a lack of convergence in the sequence concerning higher numbers of reward boxes.

3 Conclusion

In conclusion, our project marked a significant success as we found an optimal strategy for opening boxes. This accomplishment showcases the potential of our approach in maximising the expected gain over all possible permutations of the arrangement of the boxes.

While our current approach excelled in finding the maximum optimal solution possible, the backtracking algorithm is still computationally heavy, as it iteratively calculates solution at each point. Although we used memoization to store values which exponentially decreased the calculations at each recursion, we were still unable to compute solutions for higher values. Furthermore, our current project has been based on the assumption of a uniform distribution of reward boxes. However, we have not yet explored the potential implications of varying the distribution of reward boxes to follow different probability distributions. This exploration could unveil valuable insights into how the distribution pattern affects the optimal strategy and expected gain in our problem scenario.

In this project, we analysed different methods to maximise our expected gain, exploring different intuitions. Moreover, the implementation of backtracking algorithm has introduced us to a new domain of optimisation techniques.

A Appendix

A.1 Backtracking Algorithm (C++)

```
#include <bits/stdc++.h>
using namespace std;


map<vector<int>, double> memMap;
double gain(int x, int y, int C, int V, int N) {
    vector<int> key {x,y,C,N};
    if (memMap.find(key) != memMap.end())
        return memMap[key];

    if (x >= C || y > N - C)
        return memMap[key] = 0;

    double p = static_cast<double>(C - x) / (N - x - y);
    double g = p * (V - 1 + gain(x + 1, y, C, V, N))
                 + (1 - p) * (gain(x, y + 1, C, V, N) - 1);
    return memMap[key] = max(g, 0.0);
}

int main(){
    cout << gain(0,0,5,5,20);
}
```

A.2 Cutoff Algorithm

```
#include <bits/stdc++.h>
#include "gain.h"
using namespace std;

int main() {
    int k=1;
    for (int c = 1; c <= 1000; ++c) {
        int n = k;
        double G = 4 * n;
        while (G > 0) {
            ++n;
            G = gain(0, 0, c, 5, n);
        }
        k = n;
        cout << "[" << c << "," << n - 1 << "]" << endl;
    }
}
```

A.3 Expectation of Derived Strategy

```
#include <bits/stdc++.h>
using namespace std;

int profit(const vector<int>& l) {
    int p = 15 - l[2];
    if (20 - l[2] > 15) return p;

    p += (5 - l[3] + l[2]);
    if (20 - l[3] > 8) return p;

    p += (5 - l[4] + l[3]);
```

```
    return p;
}

int main() {
    vector<vector<int>> L;
    vector<int> P;

    for (int a = 1; a <= 16; ++a) {
        for (int b = a + 1; b <= 17; ++b) {
            for (int c = b + 1; c <= 18; ++c) {
                for (int d = c + 1; d <= 19; ++d) {
                    for (int e = d + 1; e <= 20; ++e) {
                        L.push_back({a, b, c, d, e});
                    }
                }
            }
        }
    }

    int totalProfit = 0;
    for (const auto& i : L) {
        P.push_back(profit(i));
        totalProfit += P.back();
    }

    cout << static_cast<double>(totalProfit) / P.size() << endl;
}
```

A.4 Cutoffs

i	n_i	i	n_i	i	n_i	i	n_i	i	n_i	i	n_i	i	n_i	i	n_i
1	8	36	202	71	387	106	569	141	750	176	930	211	1110	246	1289
2	15	37	208	72	392	107	574	142	755	177	935	212	1115	247	1294
3	21	38	213	73	397	108	579	143	760	178	940	213	1120	248	1299
4	27	39	218	74	402	109	584	144	765	179	945	214	1125	249	1304
5	33	40	224	75	407	110	589	145	770	180	950	215	1130	250	1309
6	39	41	229	76	413	111	595	146	775	181	955	216	1135	251	1314
7	45	42	234	77	418	112	600	147	780	182	961	217	1140	252	1320
8	50	43	240	78	423	113	605	148	786	183	966	218	1145	253	1325
9	56	44	245	79	428	114	610	149	791	184	971	219	1151	254	1330
10	62	45	250	80	434	115	615	150	796	185	976	220	1156	255	1335
11	67	46	255	81	439	116	620	151	801	186	981	221	1161	256	1340
12	73	47	261	82	444	117	626	152	806	187	986	222	1166	257	1345
13	78	48	266	83	449	118	631	153	811	188	991	223	1171	258	1350
14	84	49	271	84	454	119	636	154	817	189	997	224	1176	259	1355
15	89	50	276	85	460	120	641	155	822	190	1002	225	1181	260	1361
16	95	51	282	86	465	121	646	156	827	191	1007	226	1186	261	1366
17	100	52	287	87	470	122	651	157	832	192	1012	227	1192	262	1371
18	106	53	292	88	475	123	657	158	837	193	1017	228	1197	263	1376
19	111	54	298	89	480	124	662	159	842	194	1022	229	1202	264	1381
20	117	55	303	90	486	125	667	160	847	195	1027	230	1207	265	1386
21	122	56	308	91	491	126	672	161	853	196	1033	231	1212	266	1391
22	128	57	313	92	496	127	677	162	858	197	1038	232	1217	267	1396
23	133	58	319	93	501	128	682	163	863	198	1043	233	1222	268	1401
24	138	59	324	94	506	129	688	164	868	199	1048	234	1227	269	1407
25	144	60	329	95	512	130	693	165	873	200	1053	235	1233	270	1412
26	149	61	334	96	517	131	698	166	878	201	1058	236	1238	271	1417
27	154	62	339	97	522	132	703	167	883	202	1063	237	1243	272	1422
28	160	63	345	98	527	133	708	168	889	203	1068	238	1248	273	1427
29	165	64	350	99	532	134	713	169	894	204	1074	239	1253	274	1432
30	170	65	355	100	537	135	719	170	899	205	1079	240	1258	275	1437
31	176	66	360	101	543	136	724	171	904	206	1084	241	1263	276	1442
32	181	67	366	102	548	137	729	172	909	207	1089	242	1268	277	1447
33	186	68	371	103	553	138	734	173	914	208	1094	243	1274	278	1453
34	192	69	376	104	558	139	739	174	919	209	1099	244	1279	279	1458
35	197	70	381	105	563	140	744	175	925	210	1104	245	1284	280	1463