Quantathon 2022

Team: Buy High Sell Low

19 February 2022

1 Problem 1

Let θ be the probability of a head where θ is random with prior pdf $f(\theta) = 1$.

Given that we have obtained H_n heads and T_n tails where $n = H_n + T_n$, we obtain the posterior pdf for θ as follows:

$$\begin{aligned} f(\theta \mid H_n, T_n) &= \frac{p(H_n, T_n \mid \theta) f(\theta)}{p(H_n, T_n)} && \text{(Bayes' Theorem)} \\ &= \frac{p(H_n, T_n \mid \theta) f(\theta)}{\int_0^1 p(H_n, T_n \mid \theta) f(\theta) d\theta} \\ &= \frac{\binom{n}{H_n} \theta^{H_n} (1-\theta)^{T_n}}{\int_0^1 \binom{n}{H_n} \theta^{H_n} (1-\theta)^{T_n} d\theta} \\ &= \frac{\theta^{H_n} (1-\theta)^{T_n}}{\int_0^1 \theta^{H_n} (1-\theta)^{T_n} d\theta} \end{aligned}$$

Now,

$$\begin{aligned} \int_0^1 \theta^{H_n} (1-\theta)^{T_n} d\theta &= \frac{\Gamma(H_n + 1)\Gamma(T_n + 1)}{\Gamma(H_n + T_n + 2)} \\ &= \frac{H_n! T_n!}{(n+1)!} && (\star) \end{aligned}$$

Thus, the probability of a head on the $(n+1)$th toss is

$$\begin{aligned}
p(H \mid H_n, T_n) &= \int_0^1 p(H \mid \theta) f(\theta \mid H_n, T_n) d\theta \\
&= \int_0^1 \frac{\theta^{H_n+1}(1-\theta)^{T_n}}{\int_0^1 \theta^{H_n}(1-\theta)^{T_n} d\theta} d\theta \\
&= \int_0^1 \frac{\theta^{H_n+1}(1-\theta)^{T_n}}{\frac{H_n! T_n!}{(n+1)!}} d\theta && \text{(by } \star\text{)} \\
&= \frac{(n+1)!}{H_n! T_n!} \int_0^1 \theta^{H_n+1}(1-\theta)^{T_n} d\theta \\
&= \frac{(n+1)!}{H_n! T_n!} \cdot \frac{(H_n+1)! T_n!}{(n+2)!} && \text{(by } \star\text{)} \\
&= \frac{H_n+1}{n+2}
\end{aligned}$$

□

As this is an estimate of the probability, we denote it $\hat{P}$.

2 Problem 2

2.1 Complete Search

We begin by writing a complete search in Python to provide us with intuition for the problem.

To maximize the probability of flipping a head:

1. For the final toss, toss the coin with the highest probability of heads (using the formula for problem 1).
2. Otherwise, toss the coin that results in the highest probability of heads on the fourth toss. That is, for each coin, we calculate the final probabilities of heads given that tossing the coin results in heads / tails. We then weigh the probabilities based on the current probability of a head, and choose the coin with the highest weighted average.

```
from functools import lru_cache

def p(h, t):
    return (h + 1) / (h + t + 2)

@lru_cache(maxsize=None)
def f(l, c1, c2):
    pc1 = p(c1[0], c1[1])
    pc2 = p(c2[0], c2[1])
    if l == 1:
        return max(pc1, pc2)
    v1 = pc1 * f(l - 1, (c1[0] + 1, c1[1]), c2) + (1 -
        pc1) * f(l - 1, (c1[0], c1[1] + 1), c2)
    v2 = pc2 * f(l - 1, c1, (c2[0] + 1, c2[1])) + (1 -
        pc2) * f(l - 1, c1, (c2[0], c2[1] + 1))
    return max(v1, v2)
```

The answers for small n are as follows

n	f(n)	approx
1	0.5	1/2
2	0.5833333333333333	7/12
3	0.5833333333333333	7/12
4	0.6111111111111112	11/18
5	0.6166666666666667	37/60
6	0.6250000000000001	5/8
7	0.627777777777778	113/180
8	0.6341269841269843	799/1260
9	0.635952380952381	2671/4200

Table 1: Answers for small n

In particular, the highest probability achievable for $n = 4$ is $f(n) = \dfrac{11}{18}$.
In addition, it was observed that $f(n) \to \frac{2}{3}$ as $n \to \infty$.

2.2 Optimal Strategy

In this section, we present the optimal strategy for $n = 4$.

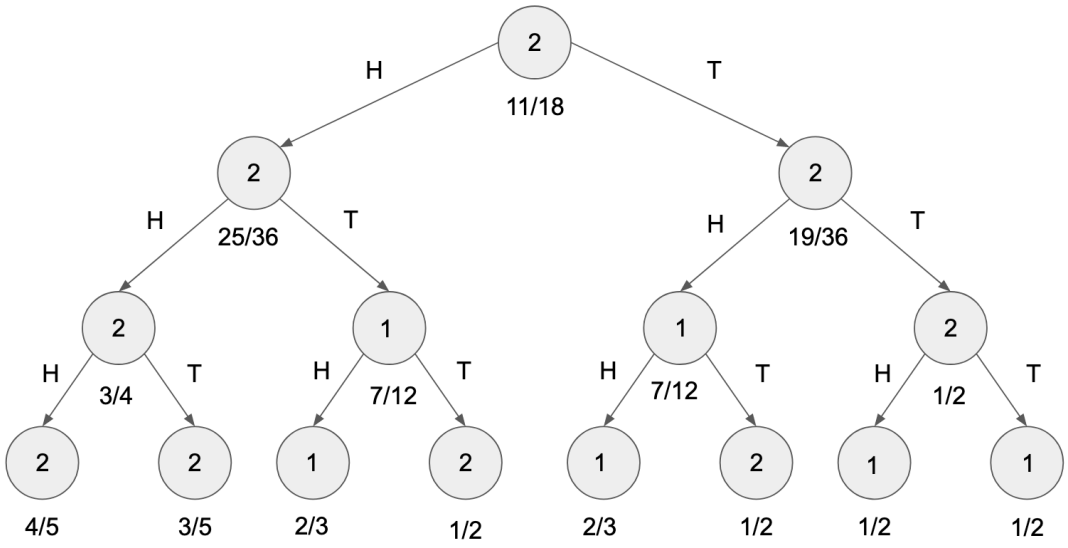


Figure 1: Decision Tree

In Figure 1, we start from the root node. The numbers within each node represent the coin that should be tossed next (WLOG let coin 1 be the nickel and coin 2 be the dime). The numbers below each node represent the probability that the fourth toss is a head (assuming the optimal strategy), given the results of the tosses so far.

Depending on the outcome of the toss, follow the corresponding edge (**H**eads / **T**ails) to the next node.

It can be observed that the optimal strategy is to always toss the dime twice at the start. Interestingly, when the first two tosses are tails, we toss the dime for a third time instead of tossing the nickel for the first time.

To see why,

- Assume we toss the dime for a third time.

  If the outcome is a head (with probability 1/4), the next toss now has probability 2/5 of being a head.

  If the outcome is a tail (with probability 3/4), the next toss now has probability 1/5 of being a head.

  Regardless, it would be optimal to toss the nickel for the fourth toss, as it has probability 1/2 of being a head (vs the dime's 1/4 or 1/5).

  So in this case, we have probability 1/2 of tossing a head on the fourth toss.

- Assume we toss the nickel for the first time.

  If the outcome is a head (with probability 1/2), the next toss now has probability 2/3 of being a head. So it would be optimal to toss the nickel for the fourth toss, as it has probability 2/3 of being a head (vs the dime's 1/4).

  If the outcome is a tail (with probability 1/2), the next toss now has probability 1/3 of being a tail. So it would be optimal to toss the nickel for the fourth toss, as it has probability 1/3 of being a head (vs the dime's 1/4).

  Since both outcomes are equally likely, we have probability 1/2 of tossing a head on the fourth toss.

- Thus, regardless of which coin is tossed for the third toss, we will have probability 1/2 of tossing a head on the fourth toss.

3 Problem 3

As $n \to \infty$, our estimates $\hat{P}$ and $\hat{Q}$ would converge to P and Q respectively. The highest probability we can achieve would therefore be

$\max\left\{\hat{P}, \hat{Q}\right\} \to \max\{P, Q\}$ Then,

$$\begin{aligned}
\lim_{n\to\infty} \pi_n &= \iint\limits_{[0,1]\times[0,1]} \max\{P,Q\} \underbrace{f(P,Q)}_{=1} \,\mathrm{d}P\,\mathrm{d}Q \\
&= \int_0^1 \int_0^P P \,\mathrm{d}Q\,\mathrm{d}P + \int_0^1 \int_0^Q Q \,\mathrm{d}P\,\mathrm{d}Q \\
&= \int_0^1 P^2 \,\mathrm{d}P + \int_0^1 Q^2 \,\mathrm{d}Q \\
&= \frac{2}{3}
\end{aligned}$$

4 Sources

- `https://www.wolframalpha.com/input?i=integrate+x%5Ey+%281-x%29%5Ez+dx+from+0+to+1`
- `https://ocw.mit.edu/courses/mathematics/18-05-introduction-to-probability-and-statistics-spring-2014/readings/MIT18_05S14_Reading13a.pdf`

5 Appendix

5.1 Strategies that we initially tried

We have come up with two main strategies to choose which coin (the nickel or the dime) to flip.

Greedy Strategy

- On the first flip we will flip either the dime (D) or the nickel (N). WLOG assume we flip the nickel (N).
- We look at the result of the first flip. If the first flip is a head we have a $\frac{2}{3}$ chance of getting a head if we flip N and a $\frac{1}{2}$ of getting a head if we flip D. Hence we will flip N.
- If we got a tail on the first flip, we will have a $\frac{1}{3}$ chance of getting a head if we flip N and a $\frac{1}{2}$ chance of getting a head if we flip D. Hence we will flip D.

- We repeat each flip based on the calculated conditional probability of getting a head if we flip N or D and flip the coin where we have a higher probability of getting a head.

Information + Greedy (Hybrid) Strategy

- We based this strategy off the idea that we will be able to get more information on P and Q after flipping N and D respectively. We want to maximize the information gained at each flip.
- For the first 2 flips, we will flip N and D once (the order does not matter). This is the key difference between the first strategy and this one in that we always flip both coins once in the first 2 flips.
- Based on the results of the first 2 flips we implement the greedy strategy for the subsequent flips.

Notice that since we are highly constrained by the number of flips we have in Q2 (4) doing a hybrid strategy does not make sense since we may be wasting 1 flip at the start and we may end up in the situation where we did not uncover much information from the first 2 flips. It would make more sense to follow the greedy strategy for the case where n = 4.

5.2 Entropy Strategy

We attempted to rigorously define the notion of 'information' using the concept of **differential entropy**.

After throwing a particular coin n times and getting H_n heads and T_n tails, we can model the probability distribution of the coin's probability P with a probability density function $f : [0, 1] \to \mathbb{R}$:

$$f(P) = \frac{P^{H_n}(1-P)^{T_n}}{H_n!T_n!/(n+1)!} = (n+1)\binom{n}{H_n}P^{H_n}(1-P)^{T_n}$$

The differential entropy of this distribution (in bits) is given by:

$$H = -\int_0^1 f(P)\log_2 f(P)\,\mathrm{d}P$$

After tossing a coin, we have a probability P of getting heads and a probability $1 - P$ of getting tails. This allows us to update our distribution, and obtain a new entropy for that distribution H'_H if we get heads and H'_T if we get tails. Then, the expected value of performing this coin toss, in bits, would be given by:

$$E[\Delta H_f] = PH'_H + (1-P)H'_T - H$$

How this strategy could be implemented is as follows: determine which coin to toss gives a higher expected value of increasing your entropy.

For instance, tossing the first coin always gives an entropy of 0.721 bits. If we get heads, then tossing the same coin allows us to get 0.962 bits (if we get heads) or 2.40 bits (if we get tails), thus yielding an expected gain of 0.96 bits. On the other hand, tossing the second coin gives us an entropy of 0.72 bits. Then, we would choose to toss the first coin again.

Repeating this process up until the $(n-1)$th coin, we would have the distribution with the highest entropy.

The limitation of this solution is that it does not result in the desired outcome of 'decreasing marginal utility' should we flip the same coin many, many times. We note that in the limit that $f(P) \to \delta$, where δ is the Dirac delta distribution (i.e. a discrete distribution), $H \to -\infty$, and hence this approach does not have the desired properties.

5.3 Mean Squared Error Strategy

Another strategy we considered was to search across all possible strategies and find the one which minimizes the **mean squared error** between our predicted probabilities $\hat{P}$ and $\hat{Q}$.

Consider the search space for various n:

- $n = 3$: Then the choice for the second coin is a function $f : \{H, T\} \to \{N, D\}$, i.e. based on the outcome of the first coin determine which coin to throw for the second one. The search space is $2^2 = 4$.
- $n = 4$: We add a function to choose the third coin to flip, $f : \{H, T\} \times \{ND\} \times \{H, T\} \to \{N, D\}$. The search space is $2^8 = 512$.

For each function in the search space we would have to perform an integral to determine the **mean squared error**, which makes implementing this strategy too arduous.

$$\begin{aligned}
\hat{P} &= \int_0^1 \frac{x^{H_n+1}(1-x)^{T_n}}{\frac{H_n! \cdot T_n!}{(n+1)!}} \, \mathrm{d}x \\
&= \frac{\frac{(H_n+1)! \cdot T_n!}{(n+2)!}}{\frac{H_n! \cdot T_n!}{(n+1)!}} \\
&= \frac{H_n+1}{n+2}
\end{aligned}$$