# Quantathon

TEAM SIMMONS

February 2022

## §1 Preliminaries

Given the probability of getting heads is $p$, then the probability that we have $H_n$ heads and $T_n$ tails in $n$ flips is $p^{H_n}(1-p)^{T_n}$. Therefore, the expected value of the $n+1$ flip would be

$$\begin{aligned}\frac{\int_0^1 p(p^{H_n}(1-p)^{T_n})dp}{\int_0^1 p^{H_n}(1-p)^{T_n}dp} &= \frac{\frac{(H_n+1)!(T_n)!}{(n+2)!}}{\frac{(H_n)!(T_n)!}{(n+1)!}} \\ &= \frac{H_n+1}{n+2}\end{aligned}$$

**Extension:** we thought it would be fruitful to consider greedily optimize for expected final outcome instead of the outcome of next throw, and we computed confidence intervals for the conditional value of $p$. We evaluated $\mathrm{Var}(p) = \mathbb{E}(p^2) - (\mathbb{E}(p))^2$, noting that $\mathbb{E}(p^2) =$

$$\begin{aligned}\frac{\int_0^1 p^2(p^{H_n}(1-p)^{T_n})dp}{\int_0^1 p^{H_n}(1-p)^{T_n}dp} &= \frac{\frac{(H_n+2)!(T_n)!}{(n+3)!}}{\frac{(H_n)!(T_n)!}{(n+1)!}} \\ &= \frac{(H_n+2)(H_n+1)}{(n+2)(n+3)}\end{aligned}$$

This gave $\mathrm{Var}(p) = \frac{(H_n+2)(H_n+1)}{(n+2)(n+3)} - (\frac{H_n+1}{n+2})^2 = \frac{(1+H_n)(1-H_n+n)}{(n+2)^2(n+3)}$, thus we get a standard deviation $\sigma = \sqrt{\frac{(1+H_n)(1-H_n+n)}{(n+2)^2(n+3)}}$. Our modified strategy then utilizes $\frac{H_n+1}{n+2} + \varepsilon\sigma$ as an upper bound for a confidence interval given some parameter $\varepsilon$. We suggest using these values to decide which coin to flip, rather than the conditional probabilities themselves.

## §2 Introduction to Algorithm

To find the best strategy for an arbitrary large $n$, we wanted to prioritize getting as much information as possible for both coins, and for small $n$, we want to be greedy as possible so that we can select the better coin with probability greater $\frac{1}{2}$, or at least better than the other coin using updated probabilities.

There are two ways we thought of doing this. The first approach was to try some fixed set of trials for both coins, and then apply a greedy algorithm. However, this only gives a fixed amount of information for both coins, and then is just as bad as a greedy algorithm the rest of the way, especially in the presence of a uniform prior. Therefore, heuristically, it's only better than greedy by a constant amount of information.

As can be seen from the additional computation on variance after problem 1, there is so much inherent variability with the statistics $p$ and $q$ that greedy algorithms could cause the search to stabilize around attractor points in the event space and potentially miss out on situations where the other coin might be better.

We found improvements on the naive greedy algorithm incorporating some $p_{change}$ factor into our greedy algorithm, so that the algorithm chooses greedily with probability $1 - p_{change}$, and chooses to get information about the other coin with probability $p_{change}$. This is better since we are getting an amount of information that varies with the amount of trials, which is better than a greedy algorithm by a factor of the amount of trials we have.

In addition, note we don't want epsilon to be constant, since as we have more flips of $P$, our estimate for the probability of getting a heads on the next flip approaches $p$, and similarly for $Q$. Therefore, the value of $p_{change}$ should decrease as $n \to \infty$. since we don't need to test the coins as often. Therefore, we scale $p_{change}$ so that as $n \to \infty$, $p_{change} \to 0$.

To test the effect of introducing randomness to our greedy algorithm, we used the formula $p_{\text{change}} = \varepsilon$. This improved our runtime and we sought to make improvements on $p_{change}$.

## §2.1 Scaling $p_{change}$ with respect to sample standard deviation

We introduced the hyperparameter $\varepsilon > 0$ for our purposes described below.

As a more educated guess we took $p_{change} = \frac{\varepsilon}{\sqrt{\text{total number of throws on the better coin}}}$ to scale the probability of change with the sample standard deviation.

We implemented this algorithm in `R`, and plotted $\epsilon$ against the error rates for the model for fixed n. (the `R` file is attached separately with the graphs).

When we regressed these points using a quadratic model, we found that for large $n$, the $\epsilon$ corresponding with the lowest error rates were near 0.5, and as $n$ became smaller, the $\epsilon$ corresponding with the lowest error rates were much smaller than 0.5.

However, we found this $p_{change}$ to switch too frequently and have too high of error rates of picking the right coin. In addition, we wanted to incorporate a chance to explore the other coin for larger values of $n$ rather than solely exploiting the coin with the higher probability. Therefore, we decided to include the variance of our distributions of $P$ and $Q$ into our $p_{change}$ function to improve our knowledge of both coins.

## §3 The Upper Confidence Bound

We improved our deterministic strategy by adapting the commonly used upper confidence bound method in the literature in our simulations (see reference for an example). This method takes into account how much risk the user is willing to take while playing this game by incorporating a risk factor $\varepsilon$ into comparing coins, where instead of comparing the conditional probability of the next head we compare the upper confidence interval, i.e. in the greedy procedure instead of comparing the statistic $\frac{H_n+1}{n+2}$ across the coins, we compare the statistic $\frac{H_n+1}{n+2} + \sqrt{\frac{(1+H_n)(1-H_n+n)}{(n+2)^2((n+3)}}$ and saw improvements from the initial greedy algorithm in our simulation as well (see R file).

However, our $p_{change}$ function didn't include the number of trials, and couldn't differentiate between the exploration and exploitation periods. Therefore, we wanted to switch to the lower variance distribution earlier in our tests and stay on the higher expected probability later in our tests. Therefore, we introduced the Simmons Model.

## §4 Simmons Model

We combined both approaches and reconstructed our $p_{change}$ for the process to make our new model.

We experimented with a parameter $\varepsilon$, and used a comparison function defined as

$$\varepsilon(\frac{\sigma_{worst}}{\sigma_{best}+\sigma_{worst}})(1-f(\frac{m_{cur}}{m_{tot}}))$$

where $\sigma_{best}$ is the conditional standard deviation of the current optimum, $\sigma_{worst}$ is that of the other coin, $m_{cur}$ is the number of moves which have past, and $m_{tot}$ is the total number of moves. Futhermore, we defined the filtration function $f : [0,1] \to \mathbb{R}$ via $f(x) = \frac{e^{-\frac{1}{x}}}{e^{-\frac{1}{x}}+e^{-\frac{1}{1-x}}}$. This function was useful due to its initially slow increase which grows much more significant as the input approaches 1. Furthermore, the ratio of the standard deviations ensures that for earlier tests, we could explore the coin with higher variance rather than stick to the coin with higher expected probability. Simmon's Model achieves significantly lower error rates for $n \geq 12$ ($> 2\%$ improvement of performance against the better of the two previous approaches).

Citation: Peter Auer. 2003. Using confidence bounds for exploitation-exploration trade-offs. J. Mach. Learn. Res. 3, null (3/1/2003), 397–422.