Advanced Computational Neuroscience - Week 4 — Two Armed Bandit Problem

Sam Suidman

March 25, 2022

1 Introduction

The two armed bandit problem is as follows. In a room there are two gambling machines next to each other. When pulling, each machine either pays off 1 or 0 with its own fixed probability (p_1,p_2). This probability, however, is not known to the agent. Furthermore the agent is allowed to have in total h pulls and can pick a different machine each time if he wants. The goal of the agent is to get as big a profit as possible. What is the best strategy the agent could choose?

2 Theory

2.1 Exploration and exploitation

Before diving into the problem it is good to understand that in principle the strategy should have two components. There is the exploration part, where the agent tries to get a sense of the probabilities of the machine. But also there is the exploitation part, where the agent is cashing out its profit from what he thinks is the best machine. These phases are not necessarily right after each other, but of course in the beginning there is more exploration and in the end more exploitation.

2.2 Dynamic programming

To solve this programming we use dynamic programming. This can best be illustrating by looking at Figure 1 where the situation is depicted as a graph. There are different paths you can go to get from point A to point J each with its own cost. An examples is $\left[A \xrightarrow{2} B \xrightarrow{6} G \xrightarrow{3} I \xrightarrow{4} J\right]$. This gives you a total cost of $C = 2 + 6 + 3 + 4 = 15$. The goal in the end is to find the path of minimal cost from A to J.

To do this you start all the way to the end at point J and set this cost to 0 using the notation $C(J) = 0$. From this you calculate the costs $C(H)$, $C(I)$ from the previous layer, which is the previous time step. They are given by $C(H) = 3 + C(J) = 3 + 0 = 3$ and $C(I) = 4 + C(J) = 4 + 0 = 4$, so quite straight forward so far. In the next (or previous time) step you calculate $C(E)$,$C(F)$,$C(G)$. If we take for example $C(E)$ we now got 2 choices: $C_1(E) = 1 + C(H) = 1 + 3 = 4$ and $C_2(E) = 4 + C(I) = 4 + 4 = 8$. The value we assign to it is the minimum of the options, which is $C(E) = \min[1 + C(H), 4 + C(I)] = \min[4, 8] = 4$. The same can now be done for $C(F)$,$C(G)$. Then you calculate the previous time step before that and so on. In the end you get to a cost $C(A) = 11$. This is not necessarily a unique path. In this case is for example given by $\left[A \xrightarrow{4} C \xrightarrow{3} E \xrightarrow{1} H \xrightarrow{3} J\right]$, $\left[A \xrightarrow{3} D \xrightarrow{4} E \xrightarrow{1} H \xrightarrow{3} J\right]$ and $\left[A \xrightarrow{3} D \xrightarrow{1} F \xrightarrow{3} I \xrightarrow{4} J\right]$. Formally this solution is governed by the Bellman equation, which is just a mathematical way to right down the process just described above.

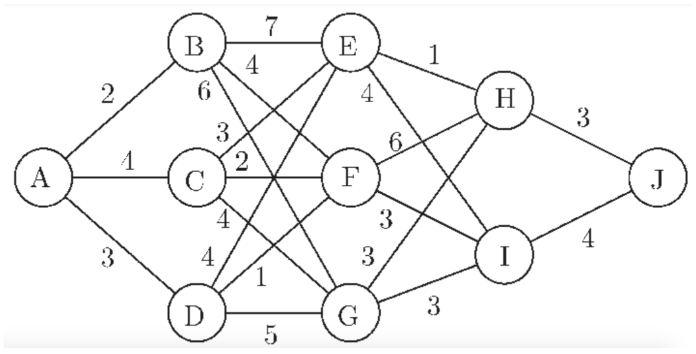

Figure 1: A graph of states with costs between each point.

2.3 Bayesian approach

To use Dynamic programming for the two armed bandit problem we use a Bayesian approach, but first we want the probability of w_i payoffs after pulling n_i times if the machine $i = 1, 2$ you are pulling from has a probability p_i for paying off. This is given by the binomial distribution given in equation **??**

$$P(w_i|p_i, n_i) = \binom{n_i}{w_i} p_i^{w_i} (1 - p_i)^{n_i - w_i} \tag{1}$$

If we use Bayes rule with a flat prior $P(p_i|n_i) = P(p_i) = 1$ we get:

$$P(p_i|w_i, n_i) = \frac{P(w_i|p_i, n_i)P(p_i|n_i)}{\int P(w_i|p_i, n_i)P(p_i|n_i)dp_i} = \frac{(n_i + 1)!}{(w_i)!(n_i - w_i)!} p_i^{w_i} (1 - p_i)^{n_i - w_i} \tag{2}$$

This is the beta distribution and has an expected value of

$$\langle p_i \rangle = \frac{w_i + 1}{n_i + 2} \tag{3}$$

This is something we can use for the two armed bandit problem in dynamic programming context. To understand this we introduce a new variable $V^*(n_1, w_1, n_2, w_2)$. This variable is the expected payoff for n_1 pulls from machine 1 with w_1 payoffs and n_2 pulls from machine 2 with w_2 payoffs. Of course we want V^* to be as big as possible. We get V^* using the backward propagation used in dynamic programming with the maximization of V^*. At a point in time $t < h$ (actually a certain pull), we look at the next step $t + 1$ when either machine 1 or 2 is pulled. For example machine 1 pays off 1 with expected value $\langle p_1 \rangle$ and then also gets the remaining profit $V^*_{t+1}(n_1 + 1, w_1 + 1, n_2, w_2)$. If it does not pay off (with probability $1 - \langle p_1 \rangle$), then it only get the remaining profit $V^*_{t+1}(n_1 + 1, w_1, n_2, w_2)$ and 0 payoff. This is all summarized in the equations 4 and 5.

$$V^*_{t,1}(n_1, w_1, n_2, w_2) = \langle p_1 \rangle \left[1 + V^*_{t+1}(n_1 + 1, w_1 + 1, n_2, w_2)\right] + (1 - \langle p_1 \rangle) \left[0 + V^*_{t+1}(n_1 + 1, w_1, n_2, w_2)\right] \tag{4}$$

$$V^*_{t,2}(n_1, w_1, n_2, w_2) = \langle p_2 \rangle \left[1 + V^*_{t+1}(n_1, w_1, n_2 + 1, w_2 + 1)\right] + (1 - \langle p_2 \rangle) \left[0 + V^*_{t+1}(n_1, w_1, n_2 + 1, w_2)\right] \tag{5}$$

The actual value V^*_t is the maximum of these 2 and is given by equation 6.

$$V^*_t(n_1, w_1, n_2, w_2) = \max\left[V^*_{t,1}, V^*_{t,2}\right] \tag{6}$$

If there are in total h pulls then we start (at the end) with $V^*_{t=h} = 0$ then we can backpropagate this all the way to $V^*_{t=0}$ and have then basically for all possible states (n_1, w_1, n_2, w_2) an expected remaining payoff V^*_t.

2.4 Start pulling

With this in mind we can now start pulling from the machines starting in the state $(n_1, w_1, n_2, w_2) = (0, 0, 0, 0)$. Which machine we pull from depends on its expected payoffs $V^*_{(t=0),1}$, $V^*_{(t=0),2}$. In the beginning we know nothing (also $V^*_1 = V^*_2$), so we may as well choose machine 1. If we get a payoff then we are in the state $(1, 1, 0, 0)$, if not we are in $(1, 0, 0, 0)$. Depending on which state we are in, we look at the new expected values for the next time step $V^*_{(t+1),2}$, $V^*_{(t+1),2}$. Again, we pull from the machine with the highest expected value. Remember that each expected value $V^*(n_1, w_1, n_2, w_2)$ is known since we calculated them in the beginning. In the end we got a series of pulls from machine 1 and machine 2 and total payoff of $w = w_1 + w_2$.

2.5 Amount of possible states

It is possible to calculate the total amount of possible states N based on the number of pulls h. There are in total $t = 0, ..., h$ time steps. Each time step you can pull $n_1 = 0, ..., t$ times machine 1 and $t - n_2$ times machine 2. There are then $w_1 = 0, ..., n_1$ possible payoffs for machine 1 and $w_2 = 0, ..., n_2$ for machine 2. Hence, the total amount of states N can be calculated with equation 7.

$$N = \sum_{t=0}^{h} \sum_{n_1=0}^{t} \sum_{w_1=0}^{n_1} \sum_{w_2=0}^{t-n_1} 1 = \sum_{t=0}^{h} \sum_{n_1=0}^{t} (n_1 + 1)(t - n_1 + 1) \tag{7}$$

To evaluate this we take use of equations 8, 9 and 10.

$$\sum_{i=0}^{n} = i = \frac{n(n+1)}{2} \tag{8}$$

$$\sum_{i=0}^{n} = i^2 = \frac{n(n+1)(2n+1)}{6} \tag{9}$$

$$\sum_{i=0}^{n} = i^3 = \left[\sum_{i=0}^{n} i\right]^2 = \left[\frac{n(n+1)}{2}\right]^2 \tag{10}$$

In the end the total amount of states is:

$$N_h = \sum_{t=0}^{h} \left[\frac{1}{6}t^3 + t^2 + \frac{11}{6}t + 1\right] = \frac{h^4 + 10h^3 + 35h^2 + 50h + 24}{24} \tag{11}$$

If we don't count all last states where $V^*_{t=h} = 0$, then we get:

$$N_{h-1} = \sum_{t=0}^{h-1} \left[\frac{1}{6}t^3 + t^2 + \frac{11}{6}t + 1\right] = \frac{h^4 + 6h^3 + 11h^2 + 6h}{24} \tag{12}$$

3 Results

3.1 $V^*(n_1, w_1, n_2, w_2)$ for different h

Following the approach of the theory section $V^*(n_1, w_1, n_2, w_2)$, simplified as $(n_1 w_1 n_2 w_2)$ could be calculated and all N_{h-1} states are written down for $h = 1, 2, 3, 4, 5$.

$h = 1$, $N_{h-1} = 1$
(0000)=0.50

$h = 2$, $N_{h-1} = 5$
(0000)=1.08 (0010)=0.50 (0011)=0.67 (1000)=0.50 (1100)=0.67

$h = 3$, $N_{h-1} = 15$
(0000)=1.67 (0010)=1.00 (0011)=1.33 (0020)=0.50 (0021)=0.50 (0022)=0.75 (1000)=1.00 (1010)=0.33 (1011)=0.67
(1100)=1.33 (1110)=0.67 (1111)=0.67 (2000)=0.50 (2100)=0.50 (2200)=0.75

$h = 4$, $N_{h-1} = 35$
(0000)=2.28 (0010)=1.53 (0011)=2.03 (0020)=1.00 (0021)=1.08 (0022)=1.50 (0030)=0.50 (0031)=0.50 (0032)=0.60
(0033)=0.80 (1000)=1.53 (1010)=0.72 (1011)=1.33 (1020)=0.33 (1021)=0.50 (1022)=0.75 (1100)=2.03 (1110)=1.33
(1111)=1.39 (1120)=0.67 (1121)=0.67 (1122)=0.75 (2000)=1.00 (2010)=0.33 (2011)=0.67 (2100)=1.08 (2110)=0.50
(2111)=0.67 (2200)=1.50 (2210)=0.75 (2211)=0.75 (3000)=0.50 (3100)=0.50 (3200)=0.60 (3300)=0.80

$h = 5$, $N_{h-1} = 70$
(0000)=2.89 (0010)=2.06 (0011)=2.72 (0020)=1.50 (0021)=1.67 (0022)=2.25 (0030)=1.00 (0031)=1.03 (0032)=1.20
(0033)=1.60 (0040)=0.50 (0041)=0.50 (0042)=0.50 (0043)=0.67 (0044)=0.83 (1000)=2.06 (1010)=1.11 (1011)=2.00
(1020)=0.67 (1021)=1.00 (1022)=1.50 (1030)=0.33 (1031)=0.40 (1032)=0.60 (1033)=0.80 (1100)=2.72 (1110)=2.00
(1111)=2.12 (1120)=1.33 (1121)=1.33 (1122)=1.52 (1130)=0.67 (1131)=0.67 (1132)=0.67 (1133)=0.80 (2000)=1.50
(2010)=0.67 (2011)=1.33 (2020)=0.25 (2021)=0.50 (2022)=0.75 (2100)=1.67 (2110)=1.00 (2111)=1.33 (2120)=0.50
(2121)=0.50 (2122)=0.75 (2200)=2.25 (2210)=1.50 (2211)=1.52 (2220)=0.75 (2221)=0.75 (2222)=0.75 (3000)=1.00
(3010)=0.33 (3011)=0.67 (3100)=1.03 (3110)=0.40 (3111)=0.67 (3200)=1.20 (3210)=0.60 (3211)=0.67 (3300)=1.60
(3310)=0.80 (3311)=0.80 (4000)=0.50 (4100)=0.50 (4200)=0.50 (4300)=0.67 (4400)=0.83

3.2 Some trials

Now we can run some trials to look at the total profit and the probability of each machine, which we can estimate via $p_i = \frac{w_i}{n_i}$. The Figures 2 to 10 show this for several values of h and p_1,p_2, where each time about 100 runs have been completed. Normalized histograms of the p_1,p_2-estimates and of the profit are plotted.

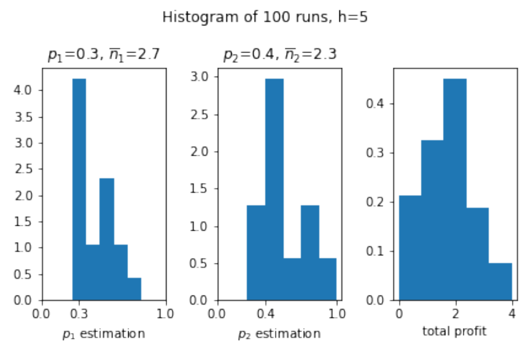

Figure 2: Histogram for h=5, p_1=0.3, p_2=0.4 that shows the estimates of p_1, p_2 and the total profit over 100 runs.

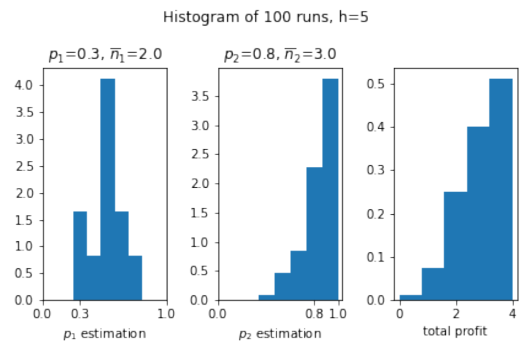

Figure 3: Histogram for h=5, p_1=0.3, p_2=0.8 that shows the estimates of p_1, p_2 and the total profit over 100 runs.

3.3 Time per iteration

In equation 11 the total amount of states N_h is calculated, but you can also see between brackets how many states per time step there are. When calculating the values of V_t^* for bigger t the iterations take longer and should, with some factor, look like the formula given in brackets. Figure 11 shows the measured time per iteration and the estimated time per iteration where the constant factor is calculated as the mean time per iteration. You see that the factor before it makes it not totally overlapping, but the shape of the estimated line follows very well the measured one.

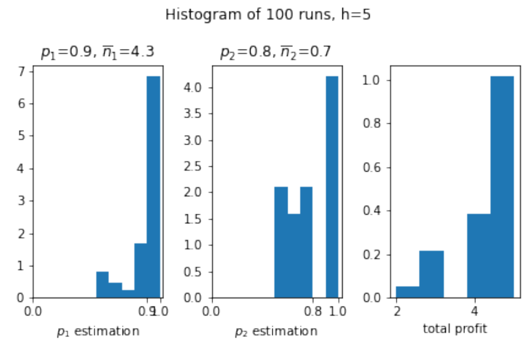

Figure 4: Histogram for h=5, p_1=0.9, p_2=0.8 that shows the estimates of p_1, p_2 and the total profit over 100 runs.

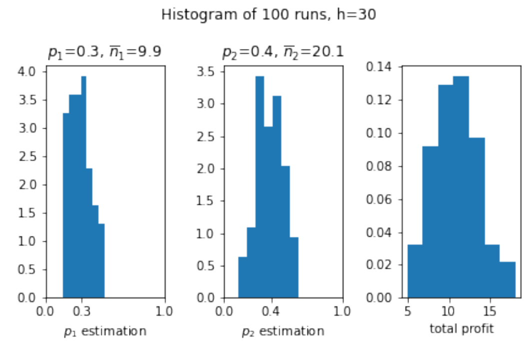

Figure 5: Histogram for h=40, p_1=0.3, p_2=0.4 that shows the estimates of p_1, p_2 and the total profit over 100 runs.

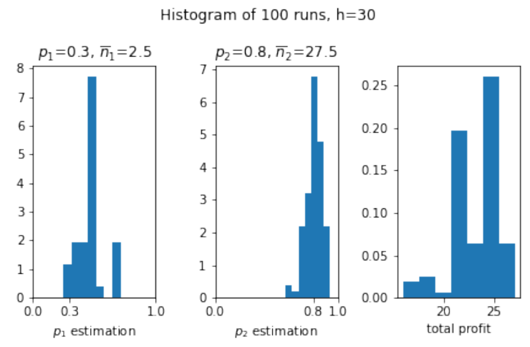

Figure 6: Histogram for h=30, p_1=0.3, p_2=0.8 that shows the estimates of p_1, p_2 and the total profit over 100 runs.

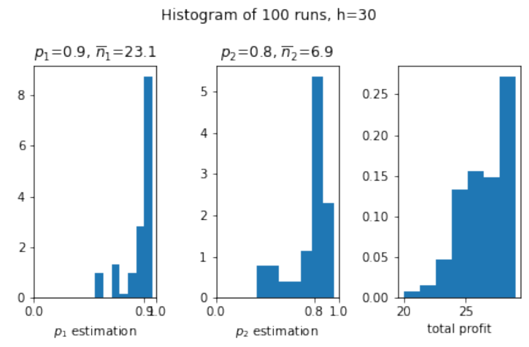

Figure 7: Histogram for h=30, p_1=0.9, p_2=0.8 that shows the estimates of p_1, p_2 and the total profit over 100 runs.

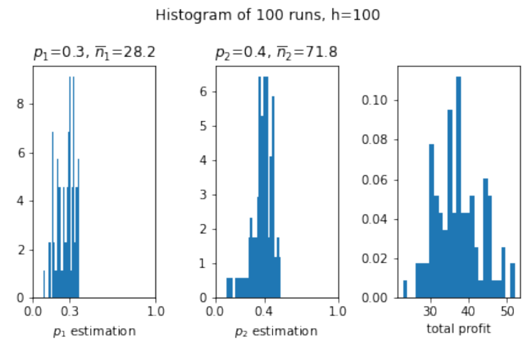

Figure 8: Histogram for h=100, p_1=0.3, p_2=0.4 that shows the estimates of p_1, p_2 and the total profit over 100 runs.

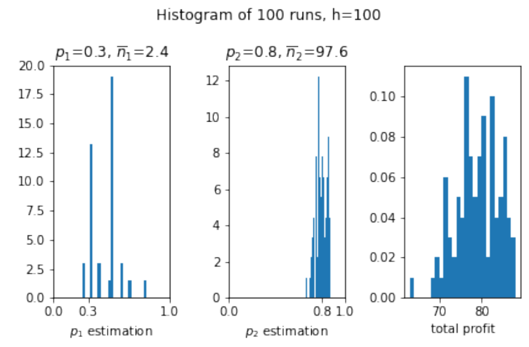

Figure 9: Histogram for h=100, p_1=0.3, p_2=0.8 that shows the estimates of p_1, p_2 and the total profit over 100 runs.

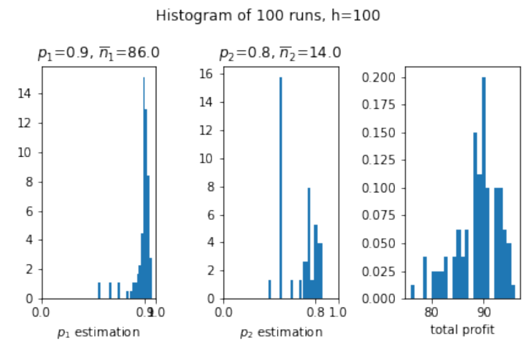

Figure 10: Histogram for h=100, p_1=0.9, p_2=0.8 that shows the estimates of p_1, p_2 and the total profit over 100 runs.

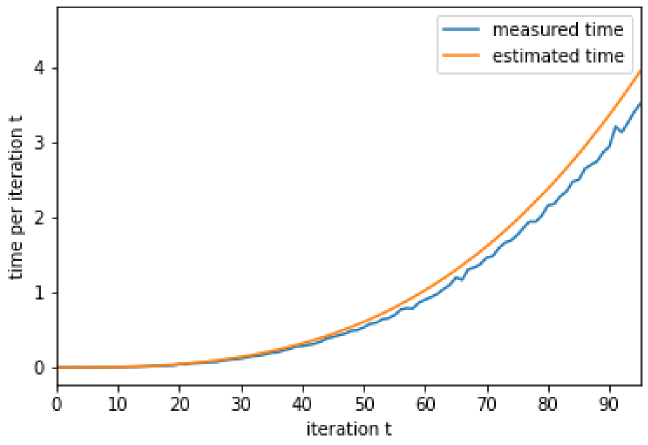

Figure 11: Plot of the measured time per iteration and the estimated time per iteration. On the x-axis the iteration is shown.

4 Conclusion

We have seen that the two armed bandit problem can be solved using dynamic programming. The number of states of $V^*(n_1 w_1 n_2 w_2)$ grows here with $\sim h^4$. Furthermore the values of p_1, p_2 can be estimated via this approach and the total profit can be calculated.

5 Appendix

The code to run this can be found in the attached Jupyter Notebook and python file.