Reinforcement Learning: Homework #2

Due on April 7, 2020 at 11:59pm

Professor Ziyu Shao

Tianyuan Wu
63305667

Problem 1

With the same format as bandit algorithms 1,2 and 3, write the pseudo-code of gradient bandit algorithm for this three-armed Bernoulli bandit problem.

Solution

Algorithm 1 Gradient Algorithm

Initialize $\bar{R}_t = 0, H(j) = 0, j \in \{1, 2, 3\}$

```
for t = 1, 2, 3, ..., N do
    π_t(j) ← e^{H(j)} / Σ_{k=1,2,3} e^{H(k)}, j ∈ {1, 2, 3}
    Sample I(t) with probability P(I(t) = j) = π_t(j)
    R̄_t ← (1/t) r_{I(t)} + ((t−1)/t) R̄_t
    if Baseline is set then
        B_t ← Baseline
    else
        B_t ← R̄_t
    end if
    H(I(t)) ← H(I(t)) + α(r_{I(t)} − B_t)(1 − π_t(I(t)))
    H(i) ← H(i) − α(r_{I(t)} − B_t)π_t(i), Forall i ∈ {1, 2, 3} and i ≠ I(t)
end for
```

Problem 2

Now suppose we obtain the Bernoulli distribution parameters from an oracle, which are shown in the following table below. Choose $N = 10000$ and compute the theoretically maximized expectation of aggregate rewards over N time slots. We call it the oracle value. Note that these parameters θ_j, $j = 1, 2, 3$ and oracle values are unknown to all bandit algorithms.

Arm_j	1	2	3
θ_j	0.9	0.8	0.7

Solution

We assume Arm_1 is chosen x times, Arm_2 is chosen y times, then Arm_3 is chosen $10000 - x - y$ times. By the expectation of binomial distribution:

$$E(X) = np,\ X \sim Bin(n, p)$$

We can obtain the expectation of aggregate reward $\mathbb{E}_{aggr}$

$$\mathbb{E}_{aggr} = 0.9x + 0.8y + 0.7(10000 - x - y),\ \ s.t.\ x, y \in \mathbb{N}, x + y \leq 10000$$

Hence, take $x = 10000$ and $y = 0$, we can get

$$\begin{aligned} \mathbb{E}_{aggr} &= 0.2x + 0.1y + 7000 \\ &\leq 2000 + 0 + 7000 \\ &= 9000 \end{aligned}$$

Thus, if $N = 10000$, the theoretically maximized expectation is 9000.

Problem 3

Implement classical bandit algorithms
Solution
Codes are in the jupyter-notebook document (.ipynb file).

Problem 4

Each experiment lasts for N = 5000 turns, and we run each experiment 1000 times. Results are averaged over these 1000 independent runs.
Solution

Algorithm	Parameters	Results			
		$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	Aggregate Reward
ϵ-greedy	$\epsilon = 0.1$	0.9002	0.8008	0.7010	4446.6
ϵ-greedy	$\epsilon = 0.5$	0.9002	0.7995	0.7000	4248.7
ϵ-greedy	$\epsilon = 0.9$	0.8995	0.8005	0.6990	4047.7
UCB	$c = 1$	0.9001	0.7966	0.6921	4386.7
UCB	$c = 5$	0.9000	0.7997	0.6991	4124.6
UCB	$c = 10$	0.8999	0.7998	0.6999	4062.8
TS	$\{1,1\},\{1,1\},\{1,1\}$	0.8998	0.7519	0.6282	4485.1
TS	$\{601,401\},\{401,601\},\{2,3\}$	0.6835	0.4002	0.6703	3776.2
Gradient	baseline=0	-	-	-	4425.2
Gradient	baseline=0.8	-	-	-	4431.1
Gradient	baseline=5	-	-	-	4200.5
Gradient	baseline=20	-	-	-	4109.6
Gradient	$\beta = 0.2$	-	-	-	4256.6
Gradient	$\beta = 1$	-	-	-	4428.4
Gradient	$\beta = 2$	-	-	-	4460.1
Gradient	$\beta = 5$	-	-	-	4481.0
Gradient	$\beta = \frac{5(N-t)}{t}$	-	-	-	4490.1

For the time-varient algorithm, I take $\beta = \frac{2(N-t)}{t}$. And for all gradient algorithms, I set $\alpha = 0.1$.

Problem 5

Solution

1. ϵ-greedy Algorithm
   Overall Regret

Table 1: ϵ-greedy Algorithm

Parameter	Accmulated Avg. Regret	Total percentage of optimals
ϵ=0.9	449.916	39.94%
ϵ=0.5	250.978	66.28%
ϵ=0.1	53.508	92.52%

Regret-time function

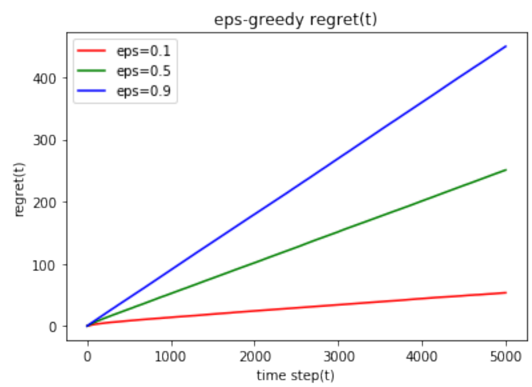

Figure 1: ϵ-greedy algorithm: regret-t

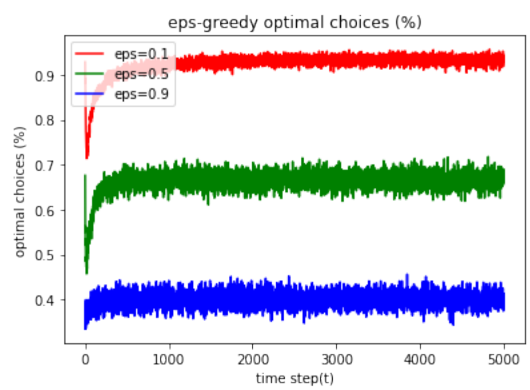

Figure 2: ϵ-greedy algorithm: Percentage of optimals

2. UCB Algorithm

Overall Regret

Table 2: UCB Algorithm

Parameter	Accmulated Avg. Regret	Total percentage of optimals
c=1	114.09	82.44%
c=5	374.568	47.28%
c=10	437.854	40.20%

Regret-time function

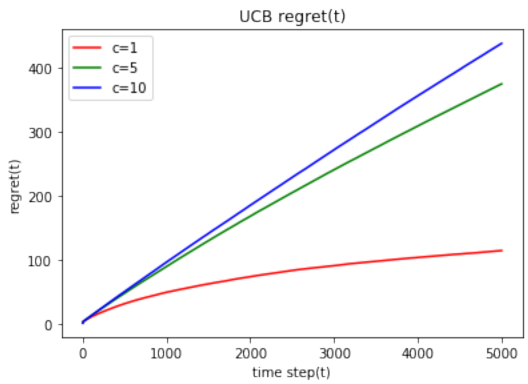

Figure 3: UCB algorithm: regret-t

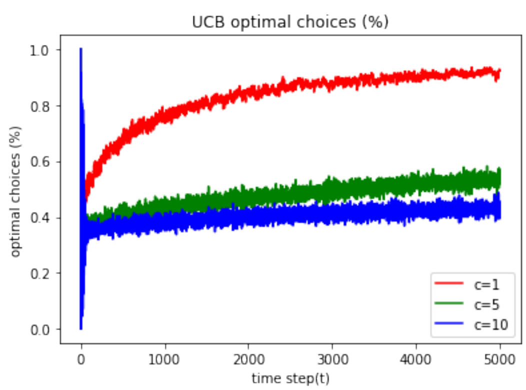

Figure 4: UCB algorithm: Percentage of optimals

3. TS Algorithm

Overall Regret

Table 3: TS Algorithm

Parameter	Accmulated Avg. Regret	Total percentage of optimals
{{1,1}, {1,1}, {1,1}}	15.985	97.55%
{{601,401}, {401,601}, {2,3}}	723.78	28.67%

Regret-time function

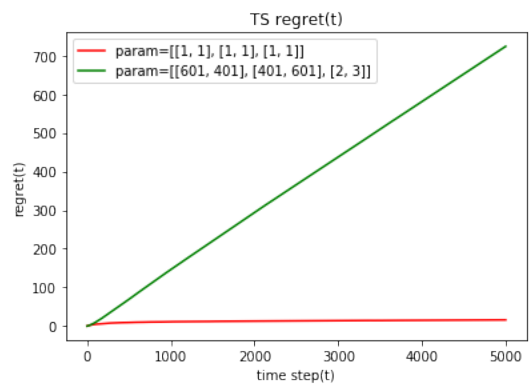

Figure 5: TS algorithm: regret-t

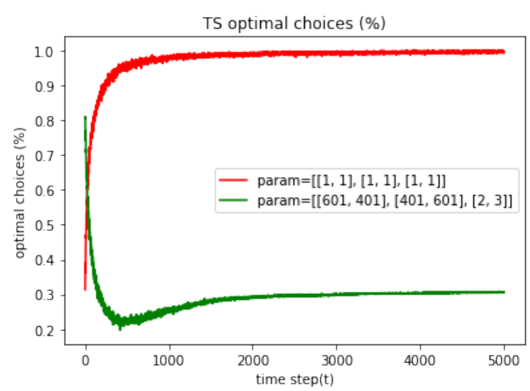

Figure 6: TS algorithm: Percentage of optimals

4. Gradient algorithm

Overall Regret

Table 4: Gradient Algorithm

Parameter	Accmulated Avg. Regret	Total percentage of optimals
baseline=0	74.8	89.83%
baseline=0.8	68.9	90.66%
baseline=5	299.5	60.14%
baseline=20	390.4	47.98%
$\beta = 0.2$	243.4	71.64%
$\beta = 1$	71.6	90.62%
$\beta = 2$	39.9	94.66%
$\beta = 5$	19.0	97.43%
time-varient	9.899	98.68%

Regret-time function

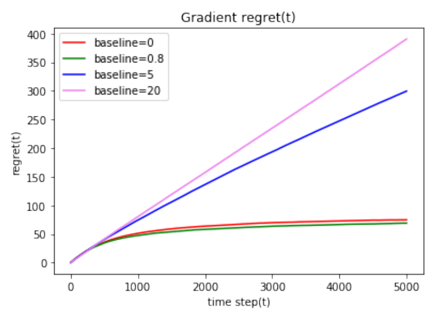

Figure 7: Gradient algorithm (baseline): regret-t

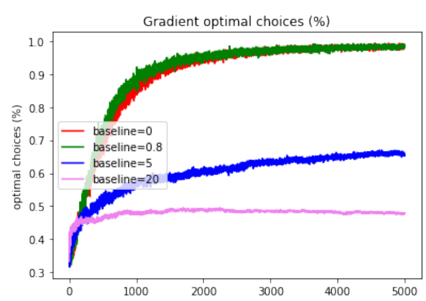

Figure 8: Gradient algorithm (baseline): Percentage of optimals

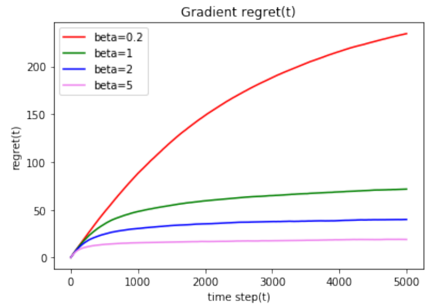

Figure 9: Gradient algorithm (parameter): regret-t

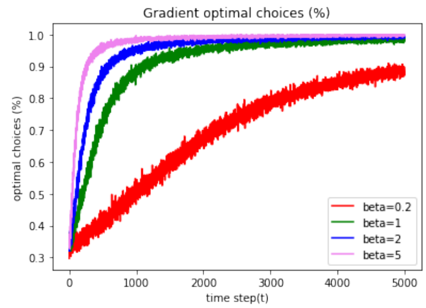

Figure 10: Gradient algorithm (parameter): Percentage of optimals

Figure 11: Gradient algorithm (time-varient): regret-t

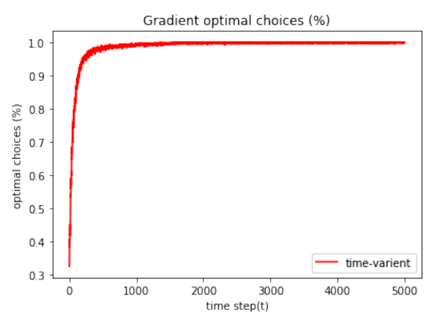

Figure 12: Gradient algorithm (time-varient): Percentage of optimals

Problem 6

Algorithm	Parameter	Gap
ϵ-greedy	$\epsilon = 0.1$	53.5
ϵ-greedy	$\epsilon = 0.5$	251.0
ϵ-greedy	$\epsilon = 0.9$	449.9
UCB	$c = 1$	114.1
UCB	$c = 5$	374.6
UCB	$c = 10$	437.8
TS	$\{1, 1\}, \{1, 1\}, \{1, 1\}$	15.9
TS	$\{601, 401\}, \{401, 601\}, \{2, 3\}$	723.8
Gradient	baseline=0	74.8
Gradient	baseline=0.8	68.9
Gradient	baseline=5	299.5
Gradient	baseline=20	390.4
Gradient	$\beta = 0.2$	243.4
Gradient	$\beta = 1$	71.6
Gradient	$\beta = 2$	39.9
Gradient	$\beta = 5$	19.0
Gradient	time-varient	9.9

The results are shown in Table.x, we can obverse that the gradient algorithm with time-varient β has the best performance. The impacts are:

1. ϵ-greedy
   When ϵ becomes larger, the exploitation increases, while explanition decreases.
   When ϵ becomes smaller, the exploitation decreases, while explanition increases.
   So, during the process that ϵ increases from a small value to 1, the gap first becomes smaller, then becomes larger.

2. UCB
   When c increases, the gap becomes larger.

3. TS
   When the parameters satisify $\frac{\alpha_i}{\alpha_i+\beta_i} = \theta_i$, the gap reaches the smallest value.

4. Gradient
   When baseline reaches the optimal value (0.9 in this situation), the gap reaches the smallest value. Also, during the process that β increases from a small value to a large value, the gap first becomes smaller, then becomes larger.

Problem 7

Give your understanding of the exploration-exploitation trade-off in bandit algorithms.
Solution
Exploration is to "try something new", is to explore if there are something better to choose, it may have better rewards, also may have worse rewards. In bandit algorithm, it's to choose a new bandit, it may different from the current optimal one. The algorithm doesn't know it will be better or worse. For example, in ϵ-greedy algorithm, the ϵ probability is to explore, prevent the algorithm from locking on the local optima. Exploitation is to choose the current optimal bandit based on observations/experiments. It's a greedy choice. So, if we only prefer exploitation, the algorithm will make choice randomly, cannot converge to the optimal value. But if we only consider about exploration, the algorithm may "lock" on some local optimals, cannot find the global optimal solution. Thus, we need to balance exploration and exploitation, try to explore more in the beginning, and then make greedy choices to converge to the optimal value.

Problem 8

We implicitly assume the reward distribution of three arms are independent. How about the dependent case? Can you design an algorithm to exploit such information to obtain a better result?
Solution
We can refer to TLP(two-level policy) algorithm, which is introduced in "multi-armed bandit problems with dependent arms - Sandeep Pandey, Deepayan Chakrebarti and Deepak Agarwal, Yahoo! research". For example, we have 3 arms and a prior that $\theta_1 - \theta_2 \leq 0.01$, then we can divide arm 1,2 into cluster1, and arm3 to cluster2. Thus, we can construct a model: n arms are divided into K clusters, by exploiting the dependence information. Then, for each step, we calculate the average reward $\bar{r_k}$ and variance $\sigma(r_k)$ for each cluster. Then call the independent bandit algorithm (e.g. UCB, gradient, etc.) to select a cluster. Then, in the selected group, call independent bandit algorithm again to get a choice. It can be descriped in pseudo-code:

Algorithm 2 TLP Algorithm

Divide $Arm_1, \ldots, Arm_n$ into K clusters $C_1, \ldots, C_K$ by dependence knowledge.
for $t = 1, 2, 3, \ldots, N$ **do**
    Foreach cluster C_i, calculate $\hat{\sigma}_{C_i}(t)$, $\hat{r}_{C_i}(t)$.
    Call independent bandit algorithm to choose a cluster C_t
    Call independent bandit algorithm in C_t, to choose Arm_t
end for

Problem 9

Please reproduce the proof of regret decomposition lemma.
Solution
For $Q(a_\tau) = \mathbb{E}[r_\tau | a_\tau]$ is a random variable, and by adam's rule,

$$\mathbb{E}[Q(a_\tau)] = \mathbb{E}[\mathbb{E}[r_\tau | a_\tau]] = \mathbb{E}[r_\tau]$$

Let $S_t = \sum_{\tau=1}^{t} Q(a_\tau)$, then we have:

$$\mathbb{E}(S_t) = \sum_{\tau=1}^{t} \mathbb{E}[Q(a_\tau)] = \mathbb{E}[\sum_{\tau=1}^{t} r_\tau]$$

Since $\sum_{a\in A} 1_{a_\tau=a} = 1$ for any fixed τ, then we have:

$$\begin{aligned}
\mathbb{E}(S_t) &= \mathbb{E}[\sum_{\tau=1}^{t} r_\tau] \\
&= \mathbb{E}[\sum_{\tau=1}^{t}\sum_{a\in A} r_\tau 1_{a_\tau=a}] \\
&= \sum_{a\in A}\sum_{\tau=1}^{t} \mathbb{E}[r_\tau 1_{a_\tau=a}]
\end{aligned}$$

On the other hand, $\sum_{\tau=1}^{t}\sum_{a\in A} 1_{a_\tau=a} = t$, so $\sum_{a\in A}\sum_{\tau=1}^{t} \mathbb{E}[1_{a_\tau=a}] = t$.
The total regret can be calculated as:

$$\begin{aligned}
L_t &= \mathbb{E}[\sum_{\tau=1}^{t} V^* - Q(a_\tau)] \\
&= tv^* - \mathbb{E}(\sum_{\tau=1}^{t} Q(a_\tau)) \\
&= tv^* - \mathbb{E}(S_t) \\
&= tv^* - \sum_{a\in A}\sum_{\tau=1}^{t} \mathbb{E}[r_\tau 1_{a_\tau=a}] \\
&= \sum_{a\in A}\sum_{\tau=1}^{t} \mathbb{E}[1_{a_\tau=a}]v^* - \sum_{a\in A}\sum_{\tau=1}^{t} \mathbb{E}[r_\tau 1_{a_\tau=a}] \\
&= \sum_{a\in A}\sum_{\tau=1}^{t} \mathbb{E}[(v^* - r_\tau)1_{a_\tau=a}]
\end{aligned}$$

Then, we calculate $\mathbb{E}[(v^* - r_\tau)1_{a_\tau=a}|a_\tau]$:

$$\begin{aligned}
&\mathbb{E}[(v^* - r_\tau)1_{a_\tau=a}|a_\tau] \\
&= 1_{a_\tau=a}\mathbb{E}[(v^* - r_\tau)|a_\tau] \\
&= 1_{a_\tau=a}(v^* - Q(a)) \\
&= 1_{a_\tau=a}\Delta_a
\end{aligned}$$

Thus,

$$\mathbb{E}[(v^* - r_\tau)1_{a_\tau=a}] = \mathbb{E}[1_{a_\tau=a}\Delta_a]$$

Then we have:

$$\begin{aligned}
L_t &= \sum_{a\in A}\sum_{\tau=1}^{t} \mathbb{E}[(v^* - r_\tau)1_{a_\tau=a}] \\
&= \sum_{a\in A}\sum_{\tau=1}^{t} \mathbb{E}[1_{a_\tau=a}\Delta_a] \\
&= \sum_{a\in A} \mathbb{E}[\sum_{\tau=1}^{t} 1_{a_\tau=a}]\Delta_a \\
&= \sum_{a\in A} \mathbb{E}[N_t(a)]\Delta_a
\end{aligned}$$

Problem 10

Please reproduce the derivation of gradient bandit algorithmem.

Solution

For the gradient ascent algorithm, for each iteration, we have:

$$w_{t+1} \leftarrow w_t + \nabla_w \mathbb{E}[f(w_t)]$$

And the objective function is $\max \mathbb{E}[R_t]$. We have:

$$\max \mathbb{E}[R_t] = \max \mathbb{E}[\mathbb{E}[R_t|E_t]] = \sum_x q_*(x)\pi_t(x)$$

where $\pi_t(x) = \frac{exp(H_t(x))}{\sum_{y=1}^k exp(H_t(y))}$. So, $\mathbb{E}[R_t]$ is a function of $H_t(x)$. And for each iteration, we have:

$$H_{t+1}(a) = H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)}$$

Then we prove the lemma:

$$\frac{\partial \pi(x)}{\partial H(a)} = \pi(x)(1_{\{x=a\}} - \pi(a))$$

Proof:

- a) If $x \neq a$:

$$\begin{aligned}
\frac{\partial \pi(x)}{\partial H(a)} &= \frac{\partial \frac{e^{H(x)}}{\sum_{y=1}^k e^{H(y)}}}{\partial H(a)} \\
&= \frac{0 - e^{H(a)}e^{H(x)}}{(\sum_{y=1}^k e^{H(y)})^2} \\
&= -\frac{e^{H(a)}}{\sum_{y=1}^k e^{H(y)}} \cdot \frac{e^{H(x)}}{\sum_{y=1}^k e^{H(y)}} \\
&= -\pi(a)\pi(x) \\
&= \pi(x)(0 - \pi(a))
\end{aligned}$$

- b) If $x = a$:

$$\begin{aligned}
\frac{\partial \pi(x)}{\partial H(a)} &= \frac{\partial \frac{e^{H(a)}}{\sum_{y=1}^k e^{H(y)}}}{\partial H(a)} \\
&= \frac{e^{H(a)} \cdot \sum_{y=1}^k e^{H(y)} - (e^{H(a)})^2}{(\sum_{y=1}^k e^{H(y)})^2} \\
&= \frac{e^{H(a)}}{\sum_{y=1}^k e^{H(y)}} \cdot \frac{\sum_{y=1}^k (e^{H(y)}) - e^{H(a)}}{\sum_{y=1}^k e^{H(y)}} \\
&= \pi(a)(1 - \pi(a)) \\
&= \pi(x)(1 - \pi(a))
\end{aligned}$$

Hence,

$$\begin{aligned}
\frac{\partial \pi(x)}{\partial H(a)} &= \begin{cases} \pi(x)(0 - \pi(a)), x \neq a \\ \pi(x)(1 - \pi(a)), x = a \end{cases} \\
&= \pi(x)(1_{\{x=a\}} - \pi(a))
\end{aligned}$$

Problem 10 continued on next page...

Hence,

$$\frac{\partial \pi_t(A_t)}{\partial H_t(a)} = \pi_t(x)(1_{\{A_t=a\}} - \pi_t(a))$$

and then:

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \sum_x q_*(x) \frac{\partial \pi_t(A_t)}{\partial H_t(a)}$$

Since $\sum_x \pi_t(x) = 1$, we have $\sum_x \frac{\partial \pi_t(A_t)}{\partial H_t(a)} = 0$.
Thus, we introduce a baseline B_t, which is not depend on x, satisifies $\sum_x B_t \frac{\partial \pi_t(A_t)}{\partial H_t(a)} = 0$.
Then we calculte the partial derivative $\frac{\partial \mathbb{E}(R_t)}{\partial H_t(a)}$

$$\begin{aligned}
\frac{\partial \mathbb{E}(R_t)}{\partial H_t(a)} &= \sum_x (q_*(x) - B_t) \frac{\partial \pi_t(A_t)}{\partial H_t(a)} \\
&= \sum_x \pi_t (q_*(x) - B_t) \frac{\partial \pi_t(A_t)}{\partial H_t(a)} \frac{1}{\pi_t} \\
&= \sum_x \pi_t (q_*(x) - B_t)(1_{x=a} - \pi_t(a)) \\
&= \mathbb{E}[(q_*(x) - B_t)(1_{x=a} - \pi_t(a))]
\end{aligned}$$

Now we choose $B_t = \bar{R}_t = \frac{1}{t}\sum_{i=1}^{t} R_i$.
Then we will show:

$$\mathbb{E}[q_*(A_t)(1_{x=a} - \pi_t(a))] = \mathbb{E}[R_t(1_{x=a} - \pi_t(a))]$$

Proof:

$$\begin{aligned}
&\mathbb{E}[(q_*(A_t))(1_{x=a} - \pi_t(a))] \\
&= \mathbb{E}[\mathbb{E}[R_t|A_t])(1_{A_t=a} - \pi_t(a))] \\
&= \mathbb{E}[\mathbb{E}[R_t(1_{A_t=a} - \pi_t(a))|A_t]] \\
&= \mathbb{E}[R_t(1_{A_t=a} - \pi_t(a))]
\end{aligned}$$

Thus we have:

$$\frac{\partial \mathbb{E}(R_t)}{\partial H_t(a)} = \mathbb{E}[(q_*(A_t) - \bar{R}_t)(1_{A_t=a} - \pi_t(a))] = \mathbb{E}[(R_t - \bar{R}_t)(1_{A_t=a} - \pi_t(a))]$$

Then, we have the gradient bandit algorithm:

$$H_{t+1}(a) = H_t(a) + \alpha(R_t - \bar{R}_t)(1_{A_t=a} - \pi_t(a)),\ \forall \alpha \in \{1, \dots, k\}$$

And for the policy gradient, we want to maximize $\mathbb{E}_x[f(x)]$, where $x \sim P_\theta$, P_θ is the policy, we have:

$$\begin{aligned}
\nabla_\theta \mathbb{E}[f(x)] &= \nabla_\theta \sum_x p(x) f(x) \\
&= \sum_x p(x) \frac{\nabla_\theta p(x)}{p(x)} f(x) \\
&= \sum_x p(x)(\nabla_\theta log(p(x)) f(x)) \\
&= \mathbb{E}[f(x) \nabla_\theta log(p(x))]
\end{aligned}$$