Reinforcement Learning

-Sarath Chandar

2. Immediate RL

2. Immediate Reinforcement Learning

Key characteristics of an RL problem :-

1. Learning to act in many situations.
2. Delayed rewards and credit assignment.
3. Exploration / Exploitation dilemma.

Exploration / Exploitation dilemma is a unique challenge in RL that does not exist in other learning paradigms. This is due to the nature of feedback to the RL agent.

Two types of feedback
- Instructive feedback
- Evaluative feedback.

Instructive feedback	Evaluative feedback
→ Instructs the right action a^*.	→ evaluates the action taken A_t by giving some reward.
→ Ignores the action taken.	→ Completely depends on the action taken.
→ Used in supervised learning.	→ Used in RL.

Evaluative feedback tells how good/bad the action taken is. But it does not tell us the optimal action to take. Hence the agent has to explore to find the optimal action.

Immediate RL: Simplest possible RL setting to study the problem of exploration vs. exploitation without worrying about the other two challenges (learning to act in many situations and delayed rewards).

Immediate RL aka k-armed bandit problem:-

You are faced repeatedly with a choice among k different options or actions.

After each choice, you receive a numerical reward chosen from a stationary probability distribution that depends on the action you selected.

Objective: to maximize the expected total reward over some time period.

Note: 1. Agent sees the same state (same set of actions) all the time.

2. Rewards are immediate.

Application: A/B testing. A website team wants to pick between placing a "buy it now" button on the top of page or bottom of the page. In traditional A/B testing, you randomly try two versions of the website (top_button / bottom_button) to two different set of users (say 10k users per group) and decide the choice based on the statistics).

This is not an adaptive process. The experiment does not utilize the incoming feedback during the course of this testing. One can pose this as a bandit problem.

arms = 2 versions of the same website.

reward = +1 if the user buys the product
-1 if the user does not buy the product.

Advantage: ① Each user feedback is utilized before the next action.
② The testing never needs to stop.

Each of the actions has an expected or mean reward given that action is selected. We will call that as the value of that action.

$$\text{value of action } a = q_*(a) = \mathbb{E}[R_t \mid A_t = a]$$

where A_t = action selected at timeStep t.

R_t = Reward obtained at timestep t.

The agent does not have access to $q_*(a)$.

If you have access to $q_*(a)$, then the optimal action a^* is given by

$$a^* = \underset{a \in \{a_1, a_2 \ldots a_k\}}{\operatorname{argmax}}\ q_*(a)$$

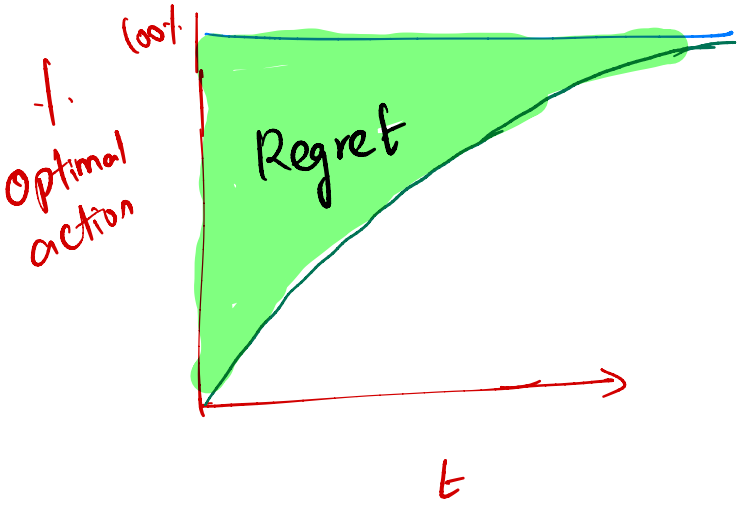
However, the agent can estimate the value of the actions based on interaction.

$Q_t(a)$ = estimated value of action 'a' at time 't'.

The goal of the agent is not just to explore and find the optimal action, but to do that as soon as possible so that it can maximize its expected total reward in the given time. This is captured by "regret". If the agent is given 'k' time steps, then

$$\text{regret} = k\, q_*(a^*) - \sum_{t=1}^{k} R_t$$

we dont have access to.



The goal is to design algorithms with least regret as possible.

Action-value methods:-

Estimate the values of actions and use them to make selection decisions.

$$Q_t(a) = \frac{\text{Sum of rewards when } a \text{ taken prior to } t}{\#\text{ of times } a \text{ taken prior to } t.}$$

$$= \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i = a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i = a}}$$

where $\mathbb{1}_{A_i = a} = \begin{cases} 1 & \text{if action 'a' was taken in timestep } i \\ 0 & \text{otherwise.} \end{cases}$

Note 1: If denominator is zero, we define $Q_t(a)$ to some default value like 0.

Note 2: As denominator goes to infinity, by law of large numbers, $Q_t(a)$ converges to $q_*(a)$.

Now the agent can exploit the action with best value based on the estimates.

$$A_t = \underset{a}{\operatorname{argmax}}\ Q_t(a)$$

This is a greedy action. However, the agent should explore other actions that look inferior to see if they are actually better.

Simple solution: Exploit most of the time. With a very small probability 'ϵ', also explore random actions.

↳ ϵ-greedy method.

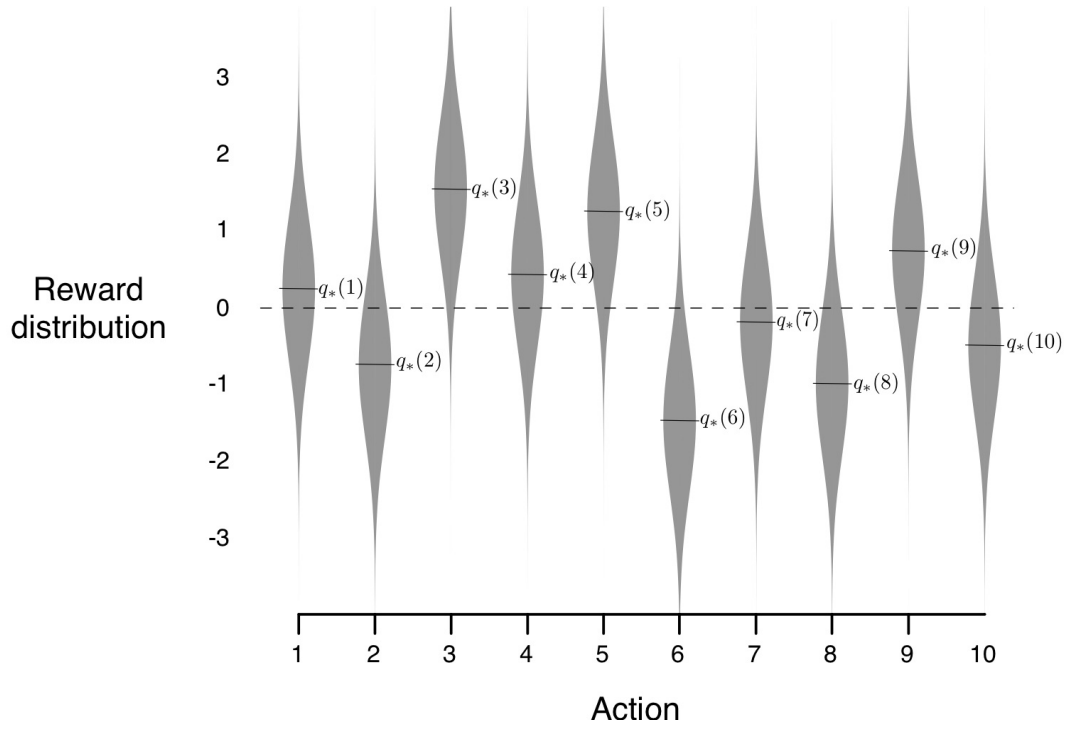
Advantage: In the limit, every action will be sampled infinite number of times, thus ensuring that $Q_t(a)$ converges to $q_*(a)$.

10 - armed test bed :-

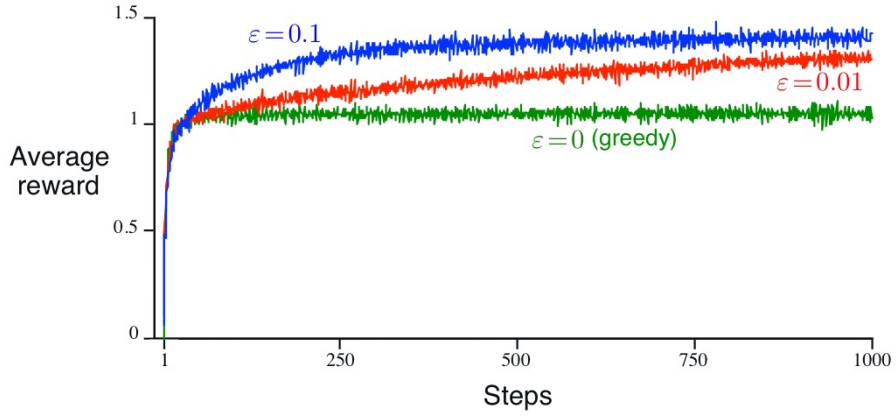
200 randomly generated k - armed bandits with $k = 10$.

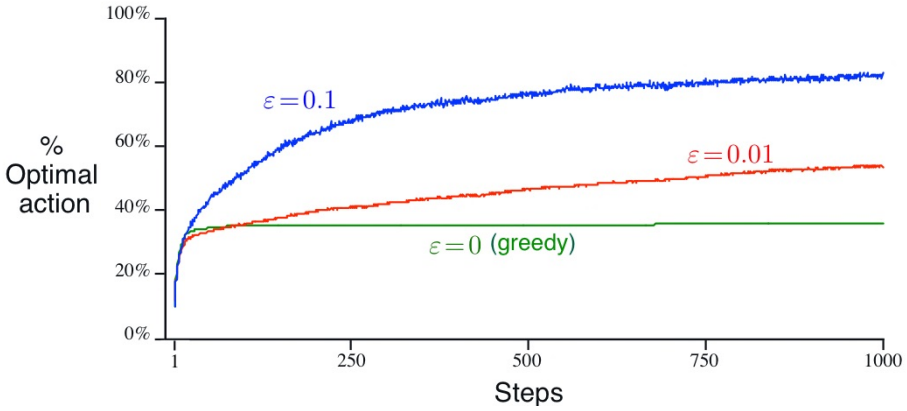
$q_*(a)$ are selected according to a Gaussian distribution with mean 0 and variance 1.

When the action is taken, then reward is sampled from a Gaussian distribution with mean $q_*(A_t)$ and variance 1.



Comparison of greedy and ϵ-greedy methods:-





The greedy method often gets stuck performing suboptimal actions.

Incremental implementation:-

Consider any action. Let R_i denote the reward received after the i^{th} selection of this action. Let Q_n denote the estimate of its action

Value after it has been selected $n-1$ times.

$$Q_n = \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}.$$

Naive implementation: Store all rewards and compute the average every time.

Incremental implementation:-

$$Q_{n+1} = \frac{1}{n} \sum_{i=1}^{n} R_i$$

$$= \frac{1}{n} \left(R_n + \sum_{i=1}^{n-1} R_i \right)$$

$$= \frac{1}{n} \left(R_n + (n-1) \frac{1}{(n-1)} \sum_{i=1}^{n-1} R_i \right)$$

$$= \frac{1}{n} \left(R_n + (n-1) Q_n \right)$$

$$= \frac{1}{n} \left(R_n + n Q_n - Q_n \right)$$

$$\boxed{Q_{n+1} = Q_n + \frac{1}{n} \left[R_n - Q_n \right]}$$

Note: for $n = 1$, $Q_2 = R_1$.

The general form of this update rule:

$$\text{NewEstimate} = \text{oldestimate} + \text{stepsize}\left[\text{Target} - \text{oldestimate}\right].$$

error in the estimate.

The error is reduced by taking a step towards the target. Target indicates a desirable direction to move, though it may be noisy.

Step size in our current setup is $\frac{1}{n}$.

A simple bandit algorithm

Initialize, for $a = 1$ to k:
$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$

Loop forever:
$A \leftarrow \begin{cases} \operatorname{argmax}_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$
$R \leftarrow bandit(A)$
$N(A) \leftarrow N(A) + 1$
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$

Tracking a non-stationary problem :-

Stationary bandits : reward probabilities do not change over time.

What if the reward distribution change over time ? In such non-stationary settings, it makes sense to give more weightage to recent rewards than long-past rewards.

One easy way to do that is by having a Constant step size parameter.

$$Q_{n+1} = Q_n + \alpha \left[R_n - Q_n \right]$$

where $\alpha \in (0, 1]$ is constant.

$$Q_{n+1} = Q_n + \alpha \left[R_n - Q_n \right]$$

$$= \alpha R_n + (1-\alpha) Q_n$$

$$= \alpha R_n + (1-\alpha) \left[\alpha R_{n-1} + (1-\alpha) Q_{n-1} \right]$$

$$= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1}$$

$$= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} +$$

$$\cdots + (1-\alpha)^{n-1} \alpha R_1 + (1-\alpha)^n Q_1$$

$$Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha (1-\alpha)^{n-i} R_i$$

This is weighted average because

$$(1-\alpha)^n + \sum_{i=1}^{n} \alpha (1-\alpha)^{n-i} = 1.$$

The weight decays exponentially according to the exponent on $1-\alpha$.

If $\alpha = 1$, all weights go to the last reward R_n.

This is called exponential recency-weighted average.

Let $\alpha_n(a)$ = step-size parameter used to process the reward received after the 'n'th selection of action a.

$\alpha_n(a) = \frac{1}{n}$ leads to sample average method.

Note: Convergence to true values is not guaranteed for all choices of $\alpha_n(a)$.

Conditions required to assure convergence with probability 1:

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \quad \text{and} \quad \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty.$$

Cond 1: to guarantee that the steps are large enough to eventually overcome any inital conditions or random fluctuations.

Cond 2: to guarantee that the steps eventually become small enough to assure convergence.

Note: Both conditions are met for $\alpha_n(a) = \frac{1}{n}$.

But for $\alpha_n(a) = \alpha$, second condition is not met.

Optimistic inital values :-

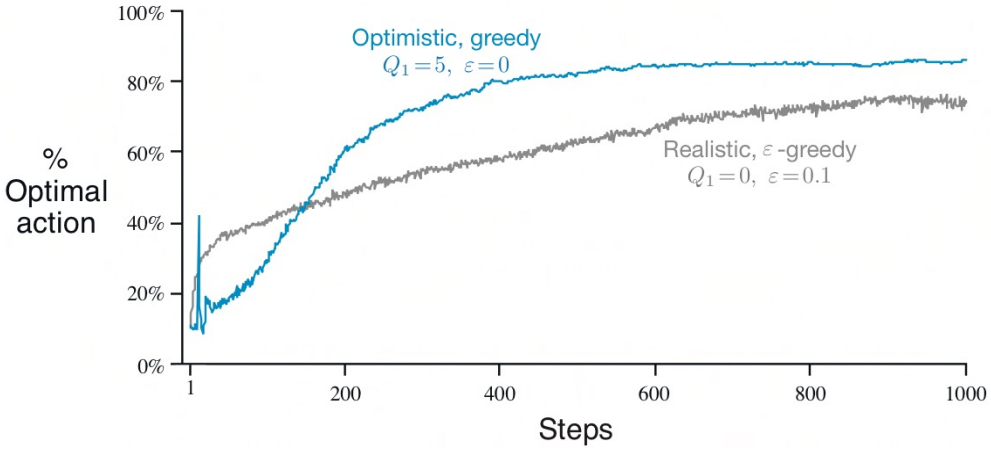
All these methods are dependent on the inital action-value estimates, $Q_1(a)$. They are biased by their initial estimates.

This bias decreases over time with more experience. This bias acts as an easy way to supply some prior knowledge about what level of rewards can be expected.

One can use the initial value estimates as a simply way to encourage exploration.

How? Instead of setting $Q_1(a) = 0$, set $Q_1(a) = +5$ in our 10-armed testbed. $q_*(a)$ was selected from $\mathcal{N}(0,1)$.

Hence $Q_1(a) = +5$ is wildly optimistic. This means, the agent will explore all the actions before converging to the best action even when $\epsilon = 0$. Let us call these methods as optimistic initial values.



Note: This simple trick does not work for non-stationary problems since exploration induced by optimistic initial values is

temporary and in the beginning of time, unlike ϵ-greedy which explores a bit throughout its lifetime.

Upper-Confidence-Bound (UCB) action selection:-

ϵ-greedy action selection forces the non-greedy actions to be tried, but indiscriminately with no preference for those that are nearly greedy or particularly uncertain.

It is better to select actions according to their potential for actually being optimal, taking into account both how close their estimates are to being maximal and the uncertainties in those estimates.

$$A_t = \underset{a}{\operatorname{argmax}} \left[Q_t(a) + c \sqrt{\frac{\ln t}{N_t(a)}} \right]$$

where $\ln t$ = Natural log of t.

$N_t(a)$ = # of times that action a has been selected prior to time t.

$c > 0$ controls the degree of exploration.

Note: If $N_t(a) = 0$, then a is considered to be a maximizing action.

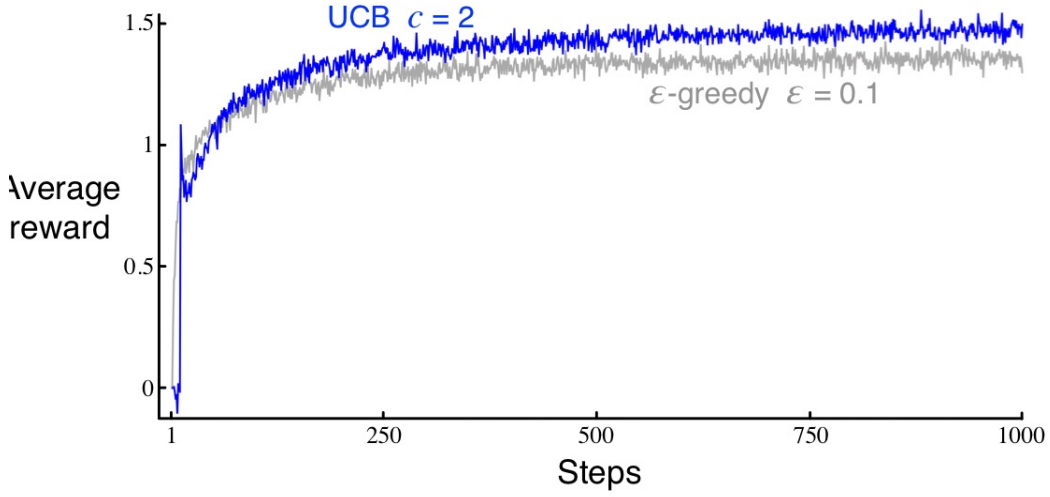
$\sqrt{\frac{\ln t}{N_t(a)}}$ is a measure of the uncertainty or variance in the estimate of a's value.

$Q_t(a) + c \sqrt{\frac{\ln t}{N_t(a)}}$ is an upper bound on the possible true value of action a.

Each time 'a' is selected $\rightarrow$ $N_t(a)$ increases and hence the uncertainty decreases.

Each time any other action is selected, $\ln t$ increases but $N_t(a)$ remains same. Hence uncertainty estimate increases (but only logarithmically).

All actions will eventually be selected, but the actions with lower value estimates or that have already been selected frequently, will be selected with decreasing frequency over time.



Note 1: Difficult to deal with non-stationarity using UCB.

Note 2: Difficult to deal with large state spaces and function approximations that we will see in the future.

Gradient bandit algorithms:-

So far we considered action value based methods. Now we consider learning a numerical preference for each action a, denoted as $H_t(a)$.

→ Larger the preference, more often the action is taken.

→ No reward or value interpretation.

Gibbs or Boltzmann distribution.

$$\Pr\{A_t = a\} = \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} = \pi_t(a)$$

$\pi_t(a)$ = Probability of taking action a.

Note 1: Adding 1000 to all $H_t(a)$ does not change the action probabilities.

Note 2: Initiall all actions are equally preferred.

$$H_1(a) = 0 \quad \text{for all } a.$$

How to learn the preferences?

In each step, after selecting action A_t and receiving the reward R_t, the action preferences are updated as follows:

$$H_{t+1}(A_t) = H_t(A_t) + \alpha (R_t - \bar{R}_t)(1 - \pi_t(A_t))$$

$$H_{t+1}(a) = H_t(a) + \alpha (R_t - \bar{R}_t)\, \pi_t(a)$$

$$\forall\, a \neq A_t$$

where $\alpha > 0$ is a step-size parameter.

$\bar{R}_t \in \mathbb{R}$ is the average of all rewards upto 't'.

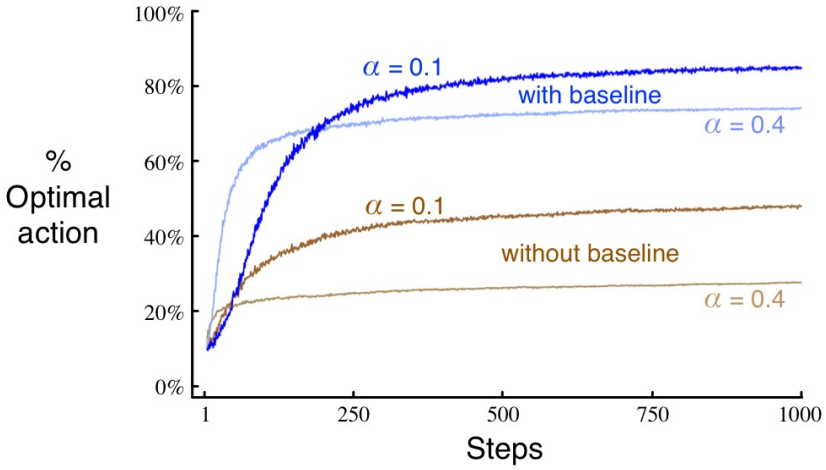
↑ baseline with which the reward is compared.

$R_t > \bar{R}_t$: Prob. of taking A_t in the future ↑

$R_t < \bar{R}_t$: Prob of taking A_t in the future ↓.

Non-selected actions move in the opposite direction.

10-armed test bed with mean of true reward set to 4 instead of zero:



The bandit gradient algorithm as stochastic gradient ascent :-

Gradient bandit algorithm is a stochastic approximation to gradient ascent.

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

$H_t(a)$ is incremented proportional to the increment's effect on performance.

$$\text{Performance} = \mathbb{E}[R_t] = \sum_x \pi_t(x)\, q_*(x)$$

It is not possible to implement this since we do not have access to $q_*(x)$.

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

$$= \sum_x q_*(x) \frac{\partial \pi_t(x)}{\partial H_t(a)}$$

$$= \sum_x \left(q_*(x) - B_t \right) \frac{\partial \pi_t(x)}{\partial H_t(a)}$$

where B_t, the baseline, can be any scalar that does not depend on x.

Note: $\sum_x \frac{\partial \pi_t(x)}{\partial H_t(a)} = 0$ so we can introduce the baseline.

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

$$= \sum_x \pi_t(x) \left(q_*(x) - B_t\right) \frac{\partial \pi_t(x)}{\partial H_t(a)} \Big/ \pi_t(x)$$

$$= \mathbb{E}\left[\left(q_*(x) - B_t\right) \frac{\partial \pi_t(A_t)}{\partial H_t(a)} \Big/ \pi_t(A_t) \right]$$

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \mathbb{E}\left[\left(R_t - \bar{R}_t\right) \frac{\partial \pi_t(A_t)}{\partial H_t(a)} \Big/ \pi_t(A_t) \right] \quad \text{(A)}$$

where $B_t = \bar{R}_t$.

R_t is a stochastic approximation of $q_*(x)$

Since

$$\mathbb{E}[R_t \mid A_t] = q_*(A_t).$$

Now let us compute $\frac{\partial \pi_t(x)}{\partial H_t(a)}$.

$$\frac{\partial \pi_t(x)}{\partial H_t(a)} = \frac{\partial}{\partial H_t(a)} \pi_t(x)$$

$$= \frac{\partial}{\partial H_t(a)} \left[\frac{e^{H_t(x)}}{\sum_{y=1}^{k} e^{H_t(y)}} \right]$$

$$= \frac{\sum_{y=1}^{k} e^{H_t(y)} \frac{\partial e^{H_t(x)}}{\partial H_t(a)} - e^{H_t(x)} \frac{\partial \sum_{y=1}^{k} e^{H_t(y)}}{\partial H_t(a)}}{\left(\sum_{y=1}^{k} e^{H_t(y)} \right)^2}$$

$$= \frac{\mathbb{1}_{a=x} e^{H_t(x)} \sum_{y=1}^{k} e^{H_t(y)} - e^{H_t(x)} e^{H_t(a)}}{\left(\sum_{y=1}^{k} e^{H_t(y)} \right)^2}$$

$$= \frac{\mathbb{1}_{a=x} e^{H_t(x)}}{\sum_{y=1}^{k} e^{H_t(y)}} - \frac{e^{H_t(x)} e^{H_t(a)}}{\left(\sum_{y=1}^{k} e^{H_t(y)} \right)^2}$$

$$= \mathbb{1}_{a=x} \pi_t(x) - \pi_t(x) \pi_t(a)$$

$$\frac{\partial \pi_t(x)}{\partial H_t(a)} = \pi_t(x) \left(\mathbb{1}_{a=x} - \pi_t(a) \right) \quad \text{---(1)}$$

Subs. (1) in (*).

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \mathbb{E}\left[(R_t - \bar{R}_t) \frac{\partial \pi_t(A_t)}{\partial H_t(a)} / \pi_t(A_t) \right]$$

$$= \mathbb{E}\left[(R_t - \bar{R}_t) \pi_t(A_t) \left(\mathbb{1}_{a=A_t} - \pi_t(a) \right) / \pi_t(A_t) \right]$$

$$= \mathbb{E}\left[(R_t - \bar{R}_t) \left(\mathbb{1}_{a=A_t} - \pi_t(a) \right) \right]$$

$$H_{t+1}(a) = H_t(a) + \alpha (R_t - \bar{R}_t) \left(\mathbb{1}_{a=A_t} - \pi_t(a) \right) \quad \forall a.$$

Contextual bandits :-

Bandits : Non-associative tasks.

i.e. No need to associate different actions with different situations.

Full RL : There are many situations. Goal is to learn a policy that maps situations to actions.

Contextual bandits : Associative task that is not as complex as full RL.

Assume there are 10 different k-armed bandit tasks and each step you confront one of these chosen at random.

→ Unlike bandits, there are 10 different situations to act.

→ Unlike RL, the rewards are immediate and actions do not affect the next state.

If you dont know the context, then this looks like a non-stationary bandit problem.

If the context is given, we can learn to associate Context to actions.

ex: Ad-placement. Given a set of ads, the goal is to place the most relevant ad for an user such that he/she will most likely click it. Context: User history