Summary of
Reinforcement Learning: An Introduction
Chapter 2: Multi-armed Bandits

Justin Chen[*] chenjus@bu.edu

Tyrone Hou[*] tyroneh@bu.edu

January 15, 2018

Introduction

In supervised learning, the label is independent of the algorithm's hypothesis which does not give meaningful signals for how good a particular hypothesis is. In contrast, reinforcement learning pushes the agent to actively discover good behavior by *evaluating actions* rather than simply *correcting* them.

2.1 A k-armed Bandit Problem

Imagine you have k slot machines (k states-action pairs to consider) with a single lever (action). Each turn the agent chooses an action; i.e. which lever to pull. For the chosen action, the agent receives a scalar **reward** (winnings for pulling lever) from a **stationary probability distribution** associated with that action.

- **Goal:** Maximize total reward over n steps in the future
- **Non-associative, evaluative feedback problem** - Actions are associated with a *single situation* and agent is given quality feedback for their selected action
- **Value** - The mean of the reward distribution associated with an action. For action a, the optimal value is $q_*(a) = \mathbb{E}[R_t | A_t = a]$.
  - The agent does not know the true value function, and instead computes an **estimate** $Q_t(a)$ at time t.
  - **Greedy action** - action a* that maximizes the current value estimate
  - $a_* = \underset{a}{argmax}\ Q_t(a)$
  - Selecting **greedy actions** exploits knowledge of current action values
  - Selecting **non-greedy actions** explores action values for which we may not have a good estimate
  - Not immediately rewarding, but allows agent to discover a more optimal strategy, which it can repeatedly exploit

2.2 Action-value Methods

- How do we estimate **value function** $Q_t(a)$? That is, the value of taking action a.
- **Sample-average method**

[*]These authors contributed equally to this work

- Average over all rewards seen when taking an action a
  $Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to t}}{\text{number of times } a \text{ taken prior to t}} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$
- Where $\mathbb{1}_{A_i=a}$ is a random variable that equals **1** if action a is selected at time i and **0** otherwise.
- How do we formulate an **action-selection** algorithm?
  * Use the value function to choose actions
- *Greedy action-selection*
  * Select the action giving you the highest value at time t
    $A_t \doteq \underset{a}{argmax} Q_t(a)$
  * 100% exploitation; never explores actions with low estimates
  * Most of the time act greedily
  * With small probability, ϵ, select an action at random
  * Given infinite time, all actions are sampled and value estimates converge to true values

- **Takeaway**: Learn from past experience

2.3 The 10-armed Testbed

- Test suite of multiple 10-armed bandit algorithms for evaluating action selection methods
  - 2000 randomly-generated bandit problems
  - Actions selected for 1000 steps for each problem
  - True value $q(a)$ for each action chosen using normal distribution $\mathcal{N}(0,1)$
  - Reward for time step t and action a also given by normal distribution $\mathcal{N}(q(a),1)$
- **Performance results**
  - $\epsilon = 0$ (greedy), $\epsilon = 0.1$, $\epsilon = 0.01$ methods tested
  - Asymptotically, $\epsilon = 0.01$ should produce the most value
  - Greedy algorithm performs the worst over time
  - Generally, **lower non-zero epsilons**
    * *perform better* over time (select optimal action more often)
    * but explore *more slowly*
- **Non-stationary problems** - the action values change over time
  - Most reinforcement learning problems are non-stationary
  - Exploration is even more important as optimal actions may become less optimal over time
  - Solving reinforcement learning requires balancing exploration and exploitation strategies

2.4 Incremental Implementation

- How to efficiently compute sample averages of observed rewards?
- Action-value estimate after it has been selected $n-1$ times:

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

- Could naively store all rewards over time, but would be space inefficient

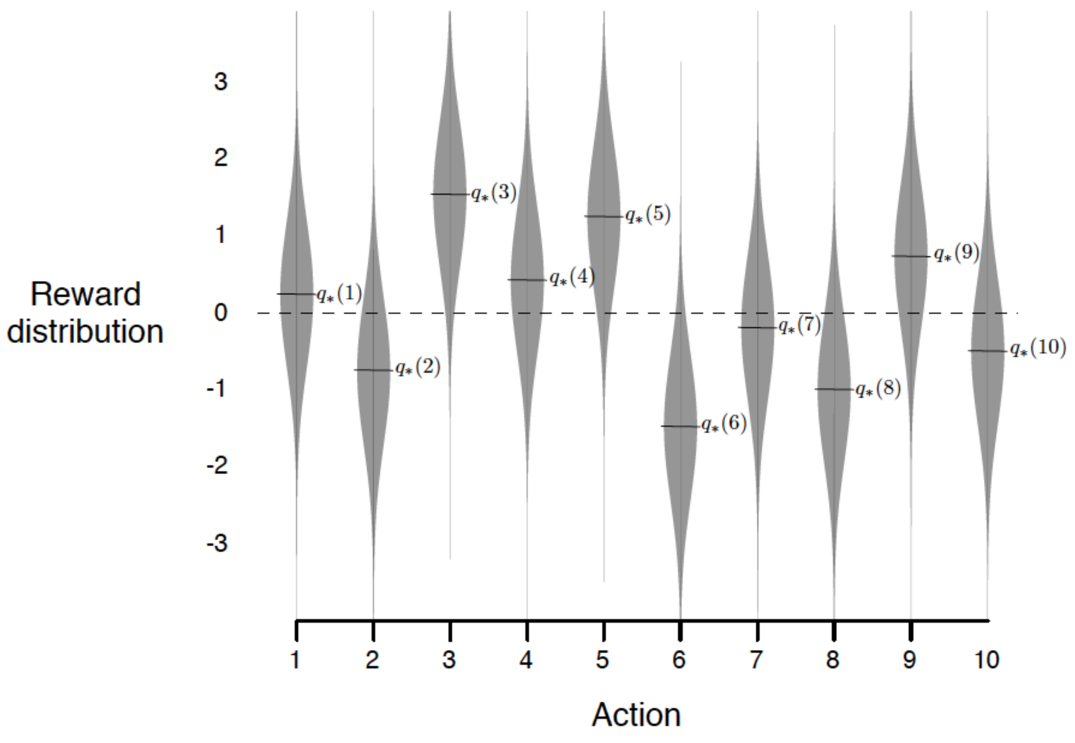

Figure 2.1: An example bandit problem from the 10-armed testbed. The true value $q_*(a)$ of each of the ten actions was selected according to a normal distribution with mean zero and unit variance, and then the actual rewards were selected according to a mean $q_*(a)$ unit variance normal distribution, as suggested by these gray distributions.

- Instead, compute:

$$Q_{n+1} = Q_n + \tfrac{1}{n}[R_n - Q_n]$$

- This only requires storing Q_n and n, and only computing each new reward

Simple Bandit Algorithm

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

Repeat forever:
    With probability $1 - \epsilon$ (break ties evenly):
        $A \leftarrow \underset{a}{argmax}\ Q(a)$
    With probability $\epsilon$:
        $A \leftarrow$ random action
    $R \leftarrow bandit(a)$
    $N(a) \leftarrow N(A) + 1$
    $Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)]$
```

- General update form - similar to gradient descent

$$NewEstimate \leftarrow OldEstimate + StepSize[Target - OldEstimate]$$

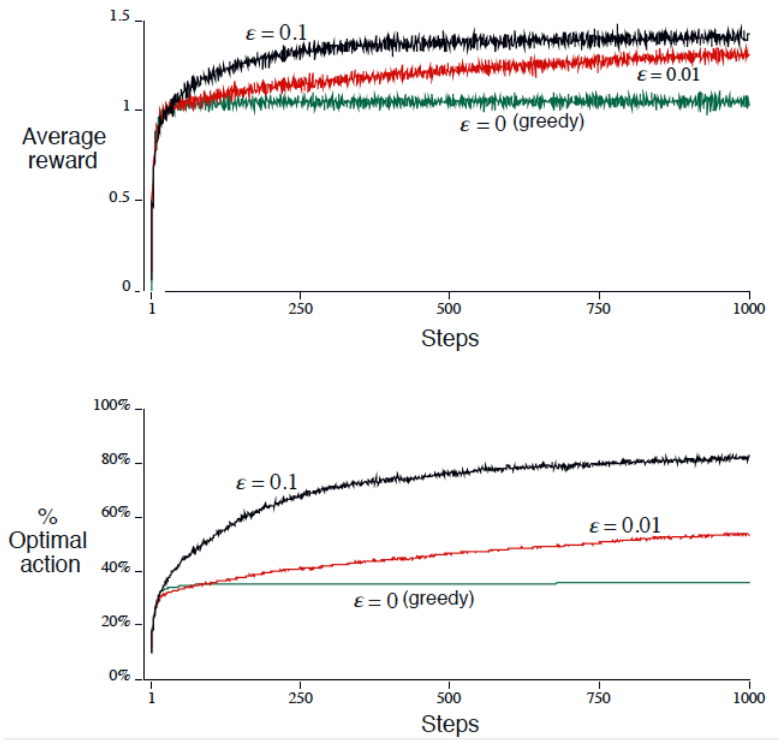

Figure 2.2: Average performance of ϵ-greedy action-value methods on the 10-armed testbed. These data are averages over 2000 runs with different bandit problems. All methods used sample averages as their action-value estimates.

- StepSize changes each timestep:

$$\alpha = \alpha_t(a) = \frac{1}{n}$$

- Reduce error between target and old estimate by taking actions towards the target

2.5 Tracking a Nonstationary Problem

- Most reinforcement learning problems involve nonstationary distributions
- Using a constant step size changes the incremental update rule to

$$\begin{aligned}
Q_{n+1} &\doteq Q_n + \alpha[R_n - Q_n], \alpha \in (0,1] \\
&= \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)[\alpha R_{n-1} + (1-\alpha)Q_{n-1}] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \cdots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i
\end{aligned}$$

- Weights sum to 1:

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

- Puts more weight on more recent events
- Sample-average method with step size of $\alpha_n(a) = \frac{1}{n}$ is guaranteed to converge by the *Law of Large Numbers*
- Not all choices of sequences of step sizes guarantee that the reward function converges
- Satiating conditions from Stochastic Approximation Theory guarantee convergence with probability 1:
  - Step Size Convergence Condition 1:
    - Guarantees steps are large enough to eventually overcome initial conditions or random fluctuations

$$\sum_{n=1}^{\infty} a_n(a) = \infty$$

  - Step Size Convergence Condition 2:
    - Guarantees that eventually the steps become small enough to assure convergence

$$\sum_{n=1}^{\infty} a_n^2(a) < \infty$$

- Constant step-size does not meet second condition
  - Estimates never completely converge and continue to vary in response to the most recently received rewards. This is beneficial for nonstationary environments.
  - Often converges faster than Sample-average method

$$a_n(a) = \alpha$$

2.6 Optimistic Initial Values

- Action selection methods are biased by their ***initial value estimates***
  - No bias when all actions have been chosen at least once
  - Initial values represent prior knowledge about expected reward levels
  - Default often sets initial results to 0
- What happens if we use *very optimistic (large)* initial settings?
  - ***Optimistic Initial Values*** exploration technique
  - Algorithm is encouraged to **explore** actions that have not been taken
  - If task changes (nonstationary), must explore policy space from scratch
- Example:
  - We let $Q_0(a) = 5$ instead of initializing to 0, where $q_*(a) < 5$ for all actions and select greedily
  - Say the first action chosen has reward 3. The other actions will have higher estimated values, so they will be chosen next
  - Performs better than $\epsilon = 0.1$ method initialized to zero on 10-armed testbed
- Advantages
  - Effective for stationary problems
  - Simple to implement and use
  - Often useful in practice
- Disadvantages
  - Does not generalize well to non-stationary tasks
  - Exploration only occurs at the beginning

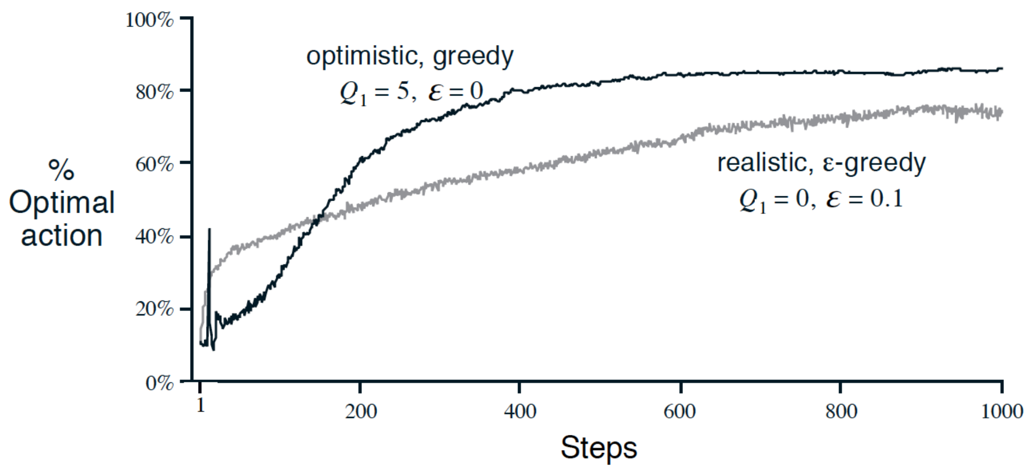

Figure 2.3: The effect of optimistic initial action-value estimates on the 10-armed testbed. Both methods used a constant step-size parameter, $\alpha = 0.1$.

2.7 Upper-Confidence-Bound Action Selection

- How do we strategically explore actions that are more likely to give high values?
- We can select actions according to

$$A_t \doteq \underset{a}{argmax} \left[Q_t(a) + c\sqrt{\frac{lnt}{N_t(a)}} \right]$$

- t= current timestep
- c (confidence level) = hyperparameter controlling degree of exploration
- $N_t(a)$ = number of times that action a has been selected before time t
  - If equals 0, a is a maximizing action
- Square-root term measures uncertainty or variance in the estimate of action's value
- Quantity inside of argmax is upper bounds true value of action
- Uncertainty decreases as more actions are taken
- Uncertainty increases as suboptimal actions are taken - actions that do not maximize the expression inside the brackets
- Natural logarithm scales down predicted value of rewards for timesteps further in the future, which is desirable since they can be uncertain
- Another property is that all possible actions will eventually be explored and suboptimal actions and frequently selected actions will be selected less over time
- This approach struggles with large state spaces and nonstationary problems

2.8 Gradient Bandit Algorithms

- Method for exploration, which can be used in lieu of ϵ-greedy
- ϵ-greedy selects a suboptimal a over optimal a_* each timestep. Then selects among the suboptimal choices uniformly.
- $H_t(a)$ - numerical **preference** for action a

- Does not provide interpretation for rewards
- Measures *relative* preference of one action over another

- **Action probabilities** distributed according to softmax distribution

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

  - Notice that e is raised to a different preference in the numerator and denominator and that $\pi_t(a)$ is the preference of the action in the numerator
  - All actions are initialized to $H_1(a) = 0$ so that each action has an equal chance of being chosen
  - Notation: $\pi_t(a)$ is probability of taking action a at time t

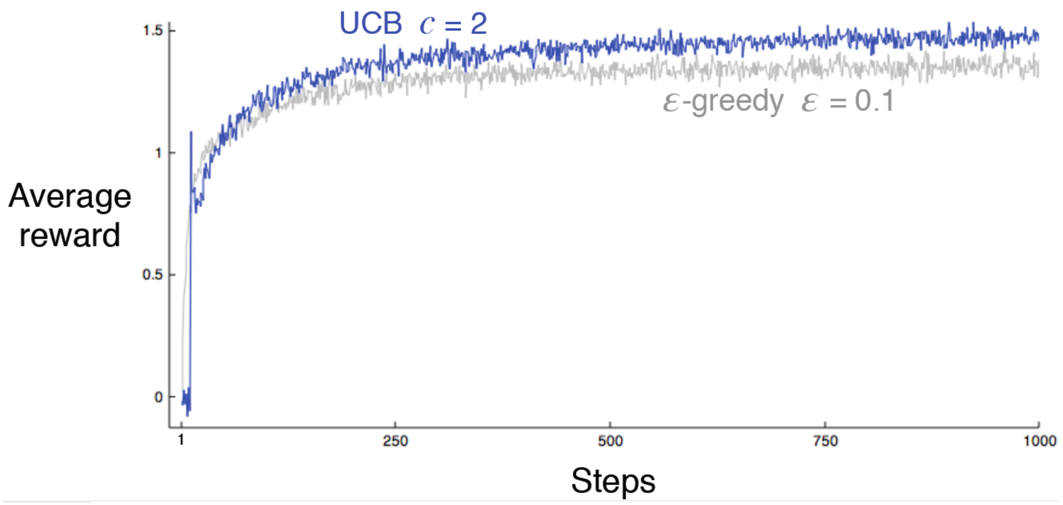

Figure 2.4: Average performance of UCB action selection on the 10-armed testbed. As shown, UCB generally performs better than ϵ-greedy action selection, except in the first k steps, when it selects randomly among the as-yet-untried actions.

Let $H_t(a)$ be a numerical **preference** for the agent to take one action over another, where higher preferences mean the action is more likely to be taken. Then we define the Gradient Bandit Algorithm as follows:

Gradient Bandit Algorithm

Preference update for selecting action A_t, which gets reward R_t:

$$\begin{aligned} H_{t+1}(A_t) &\doteq H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t)), && \textit{and} \\ H_{t+1}(a) &\doteq H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(A_t), && \forall a \neq A_t \end{aligned}$$

- **Reward baseline** - $\bar{R}_t \in \mathbb{R}$ is the average of all the rewards up through and including time t
  - Baseline used to compare with reward received at current timestep t, R_t
  - $\alpha \cdot \pi_t(A_t) =$ step size scaled by preference
  - If $(R_t - \bar{R}_t) > 0$, then $\pi_t(a)$ (probability of selecting A_t) increases, else if $(R_t - \bar{R}_t) < 0$ then $\pi_t(a)$ decreases
    - Simple sign change, add or subtract from current preference
    - Notice that the second equation above says $\forall a \neq A_t$. A_t was the selected action, so that second equation is adjusting (increasing or decreasing) the value of *all other* actions! Then the softmax is recomputed and because it's a probability distribution, all probabilities must sum to 1, so decreasing all other probabilities increases the probability of A_t.

- Expected reward measures performance where as would typically use loss function with gradient descent:

$$\mathbb{E}[R_t] = \sum_b \pi_t(b) q_*(b)$$

However, we do not know $q_*(b)$ (optimal value for b), which is what we're trying to learn, so we use this equation to derive a representation that does not depend on it.

$$\begin{aligned}\frac{\partial \mathbb{E}\left[R_t\right]}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)}\left[\sum_b \pi_t(b) q_*(b)\right] \\ &= \sum_b q_* \frac{\partial \pi_t(b)}{\partial H_t(a)} \\ &= \sum_b (q_* - X_t) \frac{\partial \pi_t(b)}{\partial H_t(a)}\end{aligned}$$

X_t represents any arbitrary scalar that **does not depend on b**.

- Including X_t is an algebraic trick. $\sum_b \frac{\partial \pi_t(b)}{\partial H_t(a)} = 0$ because $\pi_t(b)$ is a probability distribution and probabilities must sum to 1, so changing probability of one action reactively changes the probabilities of all other actions proportionately, so the derivative (rate of change) of all the action probabilities must sum to 0, so this is really just adding zero! Easier if we rearrange the terms so it's more obvious:

$$\sum_b q_*(b) \frac{\partial \pi_t(b)}{\partial H_t(a)} - X_t \sum_b \frac{\partial \pi_t(b)}{\partial H_t(a)},$$
$$\text{where } X_t \sum_b \frac{\partial \pi_t(b)}{\partial H_t(a)} = 0$$

Now another simple algebraic trick to expose variables that we can use to further convert the derivative into a representation that does not involve $q_*(b)$.

$$\begin{aligned}\frac{\partial \mathbb{E}\left[R_t\right]}{\partial H_t(a)} &= \sum_b \pi_t(b)(q_* - X_t) \frac{\partial \pi_t(b)}{\partial H_t(a)} / \pi_t(b) \\ &= \mathbb{E}\left[(q_*(A_t) - X_t) \frac{\partial \pi_t(A_t)}{\partial H_t(a)} / \pi_t(A_t)\right] \\ &= \mathbb{E}\left[(R_t - \bar{R}_t) \frac{\partial \pi_t(A_t)}{\partial H_t(a)} / \pi_t(A_t)\right]\end{aligned}$$

- The weighted sum of optimal value function $\sum_b \pi_b(b) q_*(b)$ is an expected value, so can rewrite as the expected value of the random variable $\mathbb{E}[q_*(A_t)]$ instead.
- X_t doesn't change after distributing $\pi_t(b)$ because X_t is an arbitrary scalar
- Derivative changes from $\frac{\partial \pi_t(b)}{\partial H_t(a)}$ to $\frac{\partial \pi_t(A_t)}{\partial H_t(a)}$ because we're now considering the expectation of the random variable A_t
- Differentiating w.r.t. a, the action that is being considered and is associated with reward, R_t
- Can replace X_t with $\bar{R}_t$ because X_t is arbitrary, and so can now replace $q_*(A_t)$ with R_t thus removing the optimal value function assumption

Fact: $\frac{\partial \pi_t(b)}{\partial H_t(a)} = \pi_t(b)\big(\mathbb{1}_{a=b} - \pi_t(a)\big)$

Standard quotient rule for derivatives:

$$\frac{\partial}{\partial x}\left[\frac{f(x)}{g(x)}\right] = \frac{\frac{\partial f(x)}{\partial x} g(x) - f(x)\frac{\partial g(x)}{\partial x}}{g(x)^2}$$

Derivative of e^x:

$$\frac{\partial e^x}{\partial x} = e^x dx$$

Derivation:

$$\begin{aligned}
\frac{\partial \pi_t(b)}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)} \pi_t(b) \\
&= \frac{\partial}{\partial H_t(a)}\left[\frac{e^{H_t(b)}}{\sum_{c=1}^{k} e^{H_t(c)}}\right] \\
&= \frac{\frac{\partial e^{H_t(b)}}{\partial H_t(a)} \sum_{c=1}^{k} e^{H_t(c)} - e^{H_t(b)} \frac{\partial \sum_{c=1}^{k} e^{H_t(c)}}{\partial H_t(a)}}{\left(\sum_{c=1}^{k} e^{H_t(c)}\right)^2} && \text{(by the quotient rule)} \\
&= \frac{\mathbb{1}_{a=b} e^{H_t(b)} \sum_{c=1}^{k} e^{H_t(c)} - e^{H_t(b)} e^{H_t(a)}}{\left(\sum_{c=1}^{k} e^{H_t(c)}\right)^2} && \text{(because derivative of } e^x\text{)} \\
&= \frac{\mathbb{1}_{a=b} e^{H_t(b)}}{\sum_{c=1}^{k} e^{H_t(c)}} - \frac{e^{H_t(b)} e^{H_t(a)}}{\left(\sum_{c=1}^{k} e^{H_t(c)}\right)^2} \\
&= \mathbb{1}_{a=b}\pi_t(b) - \pi_t(b)\pi_t(a) \\
&= \pi_t(b)\big(\mathbb{1}_{a=b} - \pi_t(a)\big)
\end{aligned}$$

- $\frac{\partial e^{H_t(b)}}{\partial H_t(a)} = \mathbb{1}_{a=b} e^{H_t(b)}$ because only considering $e^{H_t(b)}$ if action b is selected, else the gradient is 0.
- $\frac{\partial \sum_{c=1}^{k} e^{H_t(c)}}{\partial H_t(a)} = \frac{e^{H_t(a)}}{\partial H_t(a)}$ because summation is considering all possible actions c, which effectively behaves as just saying "some arbitrary action a", so can replace $\sum_{c=1}^{k} e^{H_t(c)}$ with $e^{H_t(a)}$ and then differentiate as $\frac{\partial e^x}{x} = e^x$

Continuing from above:

$$\begin{aligned}
\frac{\partial \mathbb{E}\left[R_t\right]}{\partial H_t(a)} &= \mathbb{E}\left[(R_t - \bar{R}_t)\frac{\partial \pi_t(A_t)}{\partial H_t(a)} / \pi_t(A_t)\right] \\
&= \mathbb{E}\left[(R_t - \bar{R}_t)\pi_t(A_t)\big(\mathbb{1}_{a=A_t} - \pi_t(a)\big) / \pi_t(A_t)\right] \\
&= \mathbb{E}\left[(R_t - \bar{R}_t)\big(\mathbb{1}_{a=A_t} - \pi_t(a)\big)\right]
\end{aligned}$$

Thus our performance gradient update equation for the Bandit Gradient algorithm becomes:

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

- Gradient bandit algorithm is equal to the gradient of expected reward
- This algorithm is an instance of stochastic gradient ascent
- Assures robust convergence properties
- Only reward property required is that reward baseline only depends on selected action
- Choice of baseline does not affect algorithm's expected update

2.9 Associative Search (Contextual Bandits)

- **Nonassociative tasks** - tasks in which there is no need to associate an action with many different situations so the agent learns to be greedy with stationary tasks or learns to find the best action over time as the task changes in nonstationary tasks
- Full reinforcement learning has multiple situations and goal is to learn best policy
- **Associative Search Task/Contextual Bandits** - trial-and-error learning to *search* for the best actions, and *association* of the actions with the situations in which they are best.
  - Each action affects only the immediate reward in associative search tasks
  - Full reinforcement learning task is when actions affect both the current reward and next situation

2.10 Summary

- Gradient bandit algorithms estimate action preferences not action values
- Initializing estimates optimistically causes even greedy methods to explore significantly

References

Sutton, Richard S., and Andrew G. Barto. "Multi-armed Bandits." Reinforcement Learning: An Introduction, The MIT Press, 2018, pp. 19–35.