Decision Making and Reinforcement Learning

Module 2: Bandit Problems

Tony Dear, Ph.D.

Department of Computer Science

School of Engineering and Applied Sciences

Topics

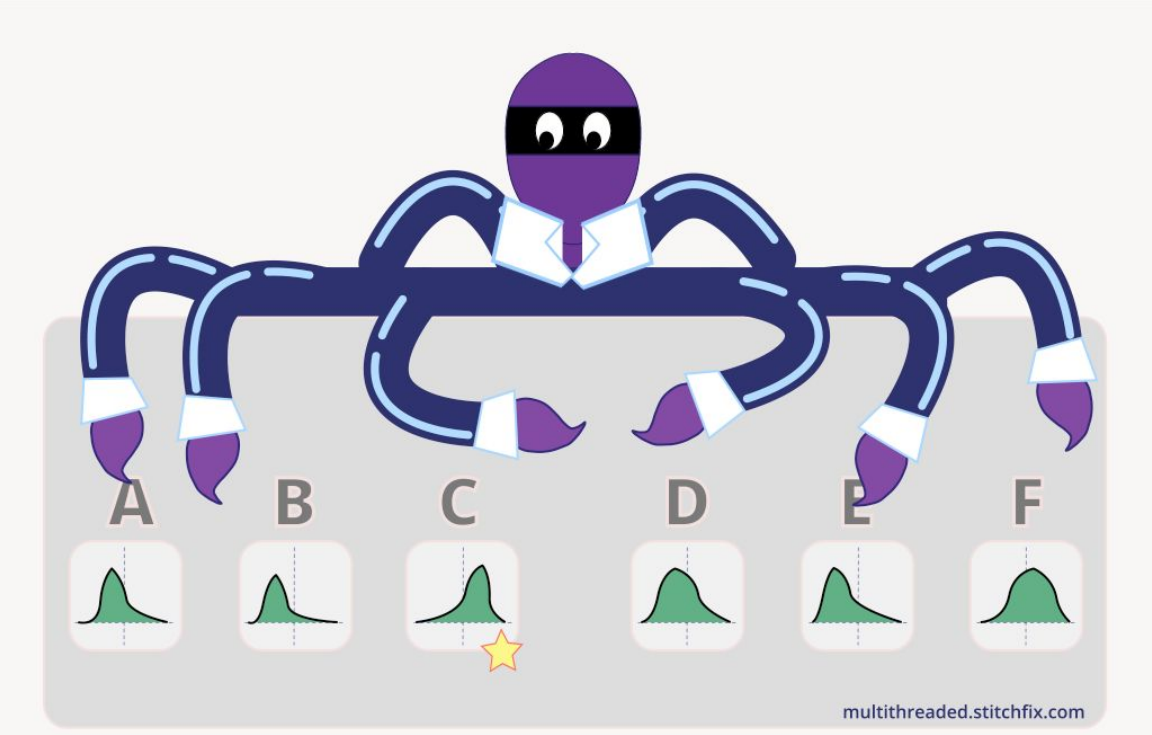
- Multi-armed bandit problems
- Action values and sample averaging
- ε-greedy action selection
- Upper confidence bound

Learning Objectives

- **Define** the multi-armed bandit problem and associated action values
- **Compute** action value updates using uniform and weighted averages
- **Understand** the exploration and exploitation tradeoff and **implement** ε-greedy selection policies
- **Understand and implement** upper confidence bound selection policies

Multi-Armed Bandits

- Suppose we have several slot machines with different rewards and odds
- We can only learn about the machine by trying them (taking actions)
- We want to maximize the overall rewards received



Multi-Armed Bandits

- How to optimize decision making for choosing actions?
- Fundamental tradeoff between **exploration** and **exploitation**

Multi-Armed Bandits

- How to optimize decision making for choosing actions?
- Fundamental tradeoff between **exploration** and **exploitation**

- Gather more information or maximize best rewards so far?
- How to determine when current knowledge is good enough?

Multi-Armed Bandits

- How to optimize decision making for choosing actions?
- Fundamental tradeoff between **exploration** and **exploitation**

- Gather more information or maximize best rewards so far?
- How to determine when current knowledge is good enough?

- Applications: Resource allocation for maximizing productivity, clinical trials to explore different treatments, financial portfolio design

Action Values

- Choosing a slot machine is choosing an *action* a
- A_t is the action chosen and R_t is the resultant reward at time t
- **Action value** is the *expected* reward of an action: $Q^*(a) = E[R_t|A_t = a]$

Action Values

- Choosing a slot machine is choosing an *action* a
- A_t is the action chosen and R_t is the resultant reward at time t
- **Action value** is the *expected* reward of an action: $Q^*(a) = E[R_t|A_t = a]$

- If we know all Q^*, optimal strategy would be to pick a with highest Q^*
- Idea: Build *estimates* of Q^* by trying different actions and recording results

Action Values

- Choosing a slot machine is choosing an *action* a
- A_t is the action chosen and R_t is the resultant reward at time t
- **Action value** is the *expected* reward of an action: $Q^*(a) = E[R_t | A_t = a]$

- If we know all Q^*, optimal strategy would be to pick a with highest Q^*
- Idea: Build *estimates* of Q^* by trying different actions and recording results

- First initialize all $Q_0(a)$, e.g. by trying each action once and recording reward

- *Sample-averaging*: $$Q_t(a) = \frac{\text{sum of rewards from taking } a \text{ prior to } t}{\text{number of times taking } a \text{ prior to } t}$$

Updating Action Values

- Sample averaging is simple but assumes that we store all rewards seen
- Not very efficient if we are updating *online* as we see rewards come in

Updating Action Values

- Sample averaging is simple but assumes that we store all rewards seen
- Not very efficient if we are updating *online* as we see rewards come in

- An incremental approach can be used to compute $Q_{t+1}(a)$ from $Q_t(a)$ without storing all previous rewards
- Suppose R_t is the nth reward from a; we have that

Updating Action Values

- Sample averaging is simple but assumes that we store all rewards seen
- Not very efficient if we are updating *online* as we see rewards come in

- An incremental approach can be used to compute $Q_{t+1}(a)$ from $Q_t(a)$ without storing all previous rewards
- Suppose R_t is the nth reward from a; we have that

$$Q_{t+1}(a) = \frac{1}{n}\big((n-1)Q_t(a) + R_t\big) = Q_t(a) + \frac{1}{n}\big(R_t - Q_t(a)\big)$$

- Update form: "new estimate" = "old estimate" + "step size" × "error"

Example: Updating Action Values

- Suppose that we see the following stream of reward values: $5, -1, 0, 4$

Example: Updating Action Values

- Suppose that we see the following stream of reward values: $5, -1, 0, 4$

- Simple sample averaging: $Q_1 = 5$

$$Q_2 = \frac{5 + (-1)}{2} = 2 \qquad Q_3 = \frac{5 + (-1) + 0}{3} = \frac{4}{3} \qquad Q_4 = \frac{5 + (-1) + 0 + 4}{4} = 2$$

Example: Updating Action Values

- Suppose that we see the following stream of reward values: $5, -1, 0, 4$

- Simple sample averaging: $Q_1 = 5$

$$Q_2 = \frac{5 + (-1)}{2} = 2 \qquad Q_3 = \frac{5 + (-1) + 0}{3} = \frac{4}{3} \qquad Q_4 = \frac{5 + (-1) + 0 + 4}{4} = 2$$

- Moving average:

Example: Updating Action Values

- Suppose that we see the following stream of reward values: $5, -1, 0, 4$

- Simple sample averaging: $Q_1 = 5$

$$Q_2 = \frac{5 + (-1)}{2} = 2 \qquad Q_3 = \frac{5 + (-1) + 0}{3} = \frac{4}{3} \qquad Q_4 = \frac{5 + (-1) + 0 + 4}{4} = 2$$

- Moving average: $Q_1 = 5$ $\qquad Q_2 = Q_1 + \frac{1}{2}(-1 - Q_1) = 5 + \frac{1}{2}(-1 - 5) = 2$

$$Q_3 = Q_2 + \frac{1}{3}(0 - Q_2) = \frac{4}{3} \qquad Q_4 = Q_3 + \frac{1}{4}(4 - Q_3) = 2$$

Nonstationary Problems

- $$Q_{t+1}(a) = \frac{1}{n}\big((n-1)Q_t(a) + R_t\big) = Q_t(a) + \frac{1}{n}\big(R_t - Q_t(a)\big)$$

- n increments each time action a is taken, so the step size shrinks over time
- Individual reward contributions become smaller as more samples are seen

Nonstationary Problems

- $$Q_{t+1}(a) = \frac{1}{n}\big((n-1)Q_t(a) + R_t\big) = Q_t(a) + \frac{1}{n}\big(R_t - Q_t(a)\big)$$

- n increments each time action a is taken, so the step size shrinks over time
- Individual reward contributions become smaller as more samples are seen

- For **nonstationary** problems in which reward distributions change over time, we can use a *constant* step size α between 0 and 1:

$$Q_{t+1}(a) = Q_t(a) + \alpha\big(R_t - Q_t(a)\big)$$

Exponential Recency-Weighted Average

- For constant α, the action value update rule ends up weighting all rewards, with weights on past rewards *decaying exponentially*

$$Q_{t+1}(a) = Q_t(a) + \alpha\big(R_t - Q_t(a)\big) = \alpha R_t + (1-\alpha)Q_t(a)$$

Exponential Recency-Weighted Average

- For constant α, the action value update rule ends up weighting all rewards, with weights on past rewards *decaying exponentially*

$$Q_{t+1}(a) = Q_t(a) + \alpha\big(R_t - Q_t(a)\big) = \alpha R_t + (1-\alpha)Q_t(a)$$

$$= \alpha R_t + (1-\alpha)\big(\alpha R_{t-1} + (1-\alpha)Q_{t-1}(a)\big)$$

Exponential Recency-Weighted Average

- For constant α, the action value update rule ends up weighting all rewards, with weights on past rewards *decaying exponentially*

$$Q_{t+1}(a) = Q_t(a) + \alpha\big(R_t - Q_t(a)\big) = \alpha R_t + (1-\alpha)Q_t(a)$$

$$= \alpha R_t + (1-\alpha)\big(\alpha R_{t-1} + (1-\alpha)Q_{t-1}(a)\big)$$

$$= \alpha R_t + (1-\alpha)\alpha R_{t-1} + (1-\alpha)^2 Q_{t-1}(a)$$

Exponential Recency-Weighted Average

- For constant α, the action value update rule ends up weighting all rewards, with weights on past rewards *decaying exponentially*

$$Q_{t+1}(a) = Q_t(a) + \alpha\big(R_t - Q_t(a)\big) = \alpha R_t + (1-\alpha)Q_t(a)$$

$$= \alpha R_t + (1-\alpha)\big(\alpha R_{t-1} + (1-\alpha)Q_{t-1}(a)\big)$$

$$= \alpha R_t + (1-\alpha)\alpha R_{t-1} + (1-\alpha)^2 Q_{t-1}(a)$$

$$= \cdots = (1-\alpha)^{t-1} R_1 + \alpha \sum_{i=2}^{t} (1-\alpha)^{t-i} R_i$$

Example: Updating Action Values

- Suppose that we see the following stream of reward values: $5, -1, 0, 4$
- What are the action value updates using $\alpha = 0.8$?

Example: Updating Action Values

- Suppose that we see the following stream of reward values: $5, -1, 0, 4$
- What are the action value updates using $\alpha = 0.8$?

$$Q_1 = 5 \qquad Q_2 = Q_1 + 0.8(-1 - Q_1) = 5 + 0.8(-1 - 5) = 0.2$$

$$Q_3 = Q_2 + 0.8(0 - Q_2) = 0.04 \qquad Q_4 = Q_3 + 0.8(4 - Q_3) = 3.208$$

Example: Updating Action Values

- Suppose that we see the following stream of reward values: $5, -1, 0, 4$
- What are the action value updates using $\alpha = 0.8$?

$$Q_1 = 5 \qquad Q_2 = Q_1 + 0.8(-1 - Q_1) = 5 + 0.8(-1 - 5) = 0.2$$

$$Q_3 = Q_2 + 0.8(0 - Q_2) = 0.04 \qquad Q_4 = Q_3 + 0.8(4 - Q_3) = 3.208$$

- Equivalent calculation using a *weighted average*:

$$Q_4 = 0.8\left(0.2^0(4) + 0.2^1(0) + 0.2^2(-1)\right) + 0.2^3(5) = 3.208$$

Initial Values and Exploration

- The choice of initial action values (even 0) provide *bias* to the estimates
- We can set them to reflect any prior knowledge we have about rewards

Initial Values and Exploration

- The choice of initial action values (even 0) provide *bias* to the estimates
- We can set them to reflect any prior knowledge we have about rewards

- *Optimistic* initial values can be used to encourage exploration
- Set all Q_0 much higher than 0, perhaps even higher than actual rewards

Initial Values and Exploration

-

Action Selection

- How to select which actions to try to gather samples for Q?
- If we had Q^*, the best strategy would be *greedy* action selection:

$$A_t = \text{argmax}_a \, Q^*(a)$$

Action Selection

- How to select which actions to try to gather samples for Q?
- If we had Q^*, the best strategy would be *greedy* action selection:

$$A_t = \text{argmax}_a \, Q^*(a)$$

- But we have Q_t instead of Q^*, whose estimates may not be accurate
- We need a strategy that tries different actions to improve Q_t estimates

Action Selection

- How to select which actions to try to gather samples for Q?
- If we had Q^*, the best strategy would be *greedy* action selection:

$$A_t = \text{argmax}_a \, Q^*(a)$$

- But we have Q_t instead of Q^*, whose estimates may not be accurate
- We need a strategy that tries different actions to improve Q_t estimates

- **ε-greedy** action selection: Select greedy action *most* of the time, but with small probability ε, pick a random action to *explore* instead

Example: ε-Greedy

- Suppose we have three actions with the following action values:
- $Q(a_1) = 3, Q(a_2) = -2, Q(a_3) = 5$

Example: ε-Greedy

- Suppose we have three actions with the following action values:
- $Q(a_1) = 3, Q(a_2) = -2, Q(a_3) = 5$

- Which action will be chosen?
- Greedy action selection: $a_3 = \text{argmax}_a\, Q(a)$ is *always* chosen

Example: ε-Greedy

- Suppose we have three actions with the following action values:
- $Q(a_1) = 3, Q(a_2) = -2, Q(a_3) = 5$

- Which action will be chosen?
- Greedy action selection: $a_3 = \text{argmax}_a\, Q(a)$ is *always* chosen

- ε-greedy action selection: a_3 is chosen with probability $1 - \varepsilon$, while a_1 and a_2 may be chosen each with probability 0.5ε

ε-Greedy Variations

- In the limit, following an ε-greedy strategy will ensure that Q_t values converge near Q^* values (though limit may be very large!)

ε-Greedy Variations

- In the limit, following an ε-greedy strategy will ensure that Q_t values converge near Q^* values (though limit may be very large!)

- Small ε: Q_t values end up closer to Q^*, but convergence may be slow
- Large ε: Q_t values are less accurate, but we update more values initially

ε-Greedy Variations

- In the limit, following an ε-greedy strategy will ensure that Q_t values converge near Q^* values (though limit may be very large!)

- Small ε: Q_t values end up closer to Q^*, but convergence may be slow
- Large ε: Q_t values are less accurate, but we update more values initially

- Other variations may vary the value of ε
- ***ε*-first**: Set $\varepsilon = 1$ for a fixed number of trials, then set $\varepsilon = 0$ afterward

ε-Greedy Variations

- In the limit, following an ε-greedy strategy will ensure that Q_t values converge near Q^* values (though limit may be very large!)

- Small ε: Q_t values end up closer to Q^*, but convergence may be slow
- Large ε: Q_t values are less accurate, but we update more values initially

- Other variations may vary the value of ε
- **ε-first**: Set $\varepsilon = 1$ for a fixed number of trials, then set $\varepsilon = 0$ afterward
- **ε-decreasing**: Set ε to be a high initial value and decrease it over time

Optimism Under Uncertainty

- While exploration is *necessary* to learn action values, it is **not** optimal for maximizing rewards received

- Example: If all rewards are deterministic, optimal strategy is to try each action just once and then act greedily thereafter

Optimism Under Uncertainty

- While exploration is *necessary* to learn action values, it is **not** optimal for maximizing rewards received

- Example: If all rewards are deterministic, optimal strategy is to try each action just once and then act greedily thereafter

- When exploring, ε-greedy picks actions completely randomly
- Better idea: Targeted exploration, to focus on more "promising" actions

Upper Confidence Bound

- **Upper confidence bound** combines both exploitation and exploration
  - t = current time step
  - $N_t(a)$ = number of times a taken prior to t
  - c = exploration parameter ≥ 0

$$UCB(a) = Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}$$

Upper Confidence Bound

- **Upper confidence bound** combines both exploitation and exploration
  - t = current time step
  - $N_t(a)$ = number of times a taken prior to t
  - c = exploration parameter ≥ 0

$$UCB(a) = Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}$$

- For proper initialization, we can take each possible action once so that each $N(a)$ starts with value equal to 1

- Subsequent action selection then follows $A_t = \text{argmax}_a UCB(a)$

Upper Confidence Bound

- $$UCB(a) = Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}$$

- UCB can be seen as a “boosted” action value due to second term
- Second term corresponds to exploration, weighted by choice of c

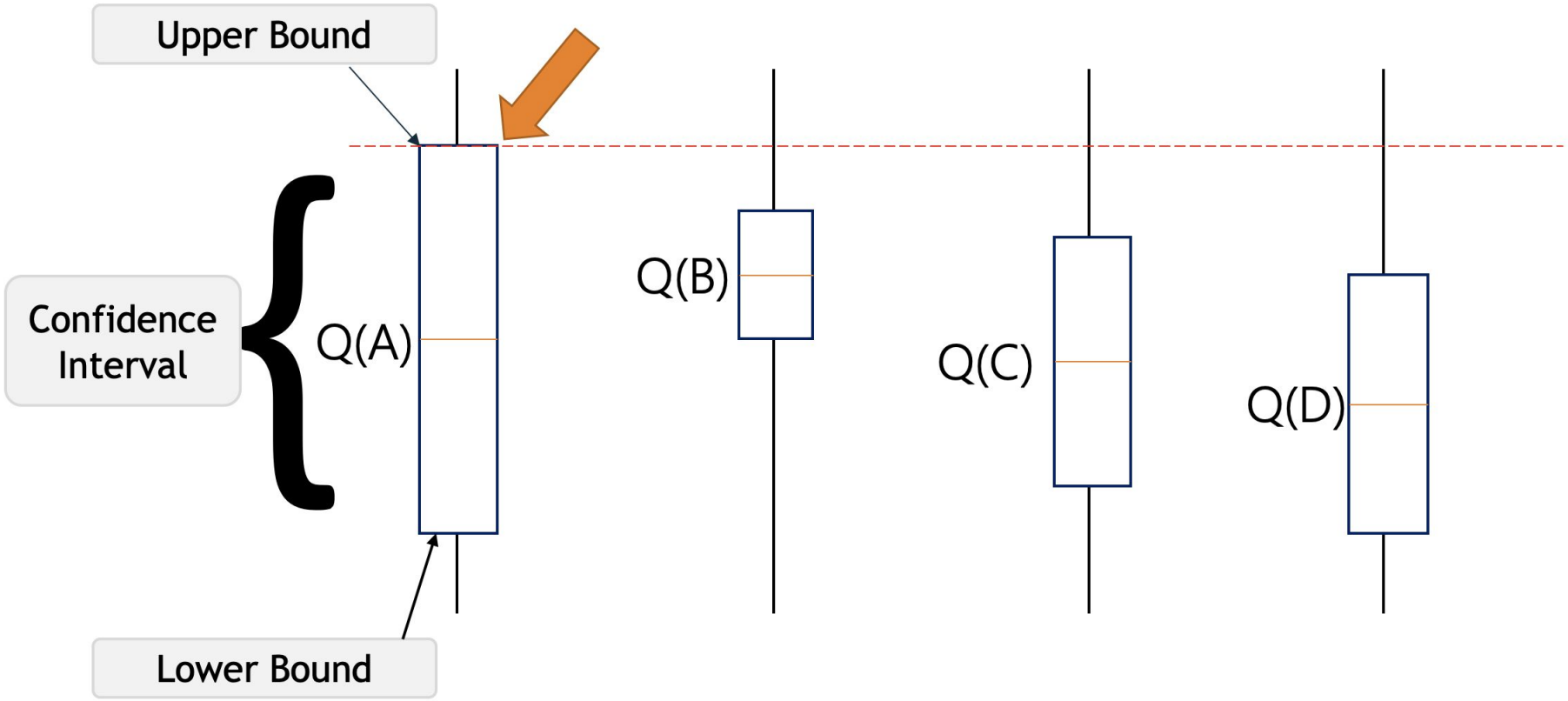
Upper Confidence Bound

- $$UCB(a) = Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}$$

- UCB can be seen as a "boosted" action value due to second term
- Second term corresponds to exploration, weighted by choice of c

- Exploration is more frequent when N_t is small and tapers out as N_t increases, leading to more exploitation
- $\ln(t)$ term *slowly* increases over time so that exploration is never ruled out

Visualization of UCB



https://www.geeksforgeeks.org/upper-confidence-bound-algorithm-in-reinforcement-learning/

UCB Implementation

- **Given**: Set of actions A to take, learning rate α, exploration parameter c

UCB Implementation

- **Given**: Set of actions A to take, learning rate α, exploration parameter c

- **For each** action $a \in A$, take a and assign associated reward to $Q(a)$
- **Initialize** $N(a) = 1$ for each $a \in A$

UCB Implementation

- **Given**: Set of actions A to take, learning rate α, exploration parameter c

- **For each** action $a \in A$, take a and assign associated reward to $Q(a)$
- **Initialize** $N(a) = 1$ for each $a \in A$

- **Loop:**
  - Take action $a = \text{argmax}_{a'} UCB(a')$ and observe reward r
  - Update $Q(a) \leftarrow Q(a) + \alpha\big(r - Q(a)\big)$ and $N(a) \leftarrow N(a) + 1$
  - Update α and/or c as desired

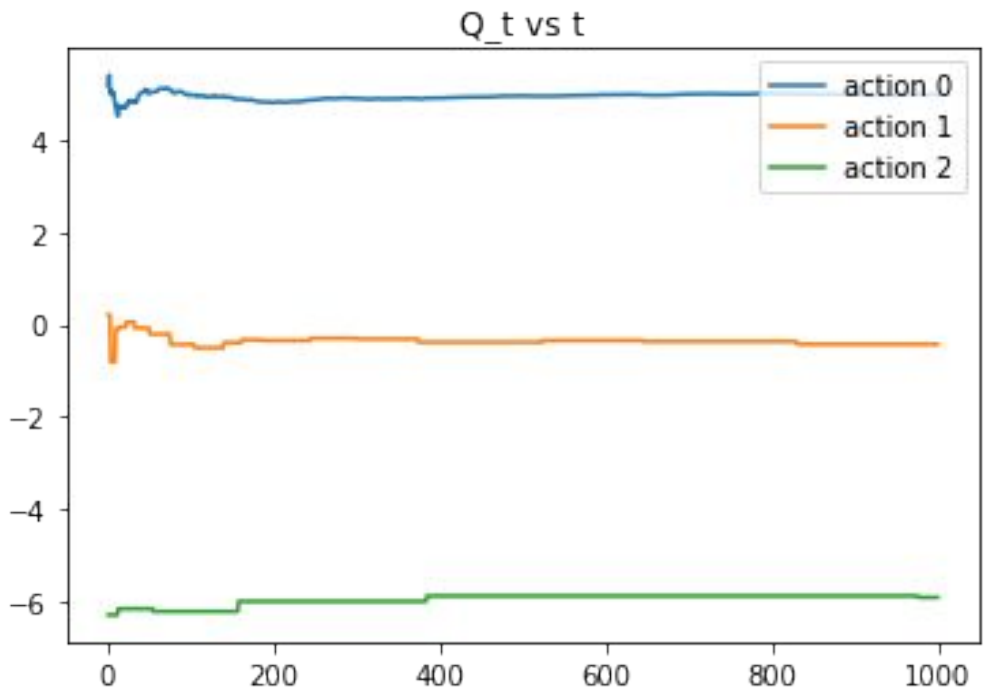
ε-Greedy Implementation

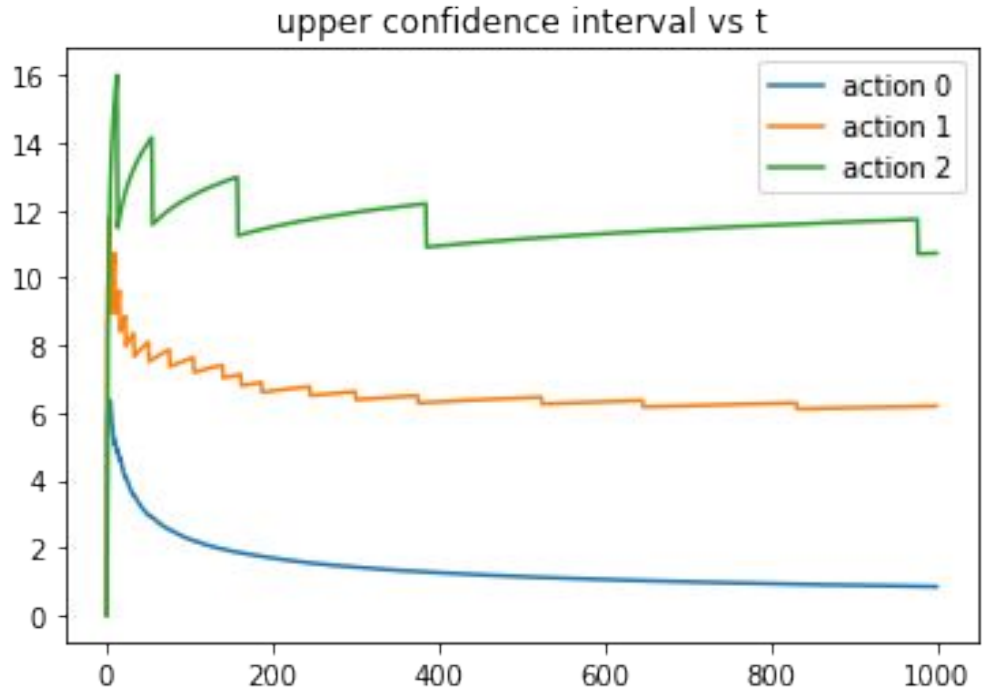
- **Given**: Set of actions A to take, learning rate α, exploration probability ε
- **Initialize** $Q(a) = 0$ (or any other desired value) for each $a \in A$
- **Loop:**

ε-Greedy Implementation

- **Given**: Set of actions A to take, learning rate α, exploration probability ε
- **Initialize** $Q(a) = 0$ (or any other desired value) for each $a \in A$
- **Loop:**
  - Determine a probabilistically using the value of ε
  - Take action a and observe reward r
  - Update $Q(a) \leftarrow Q(a) + \alpha\big(r - Q(a)\big)$
  - Update α and/or ε as desired

Example: UCB Simulation





Regret

- **Regret**: Expected difference in cumulative rewards received due to optimal (clairvoyant) strategy vs actual strategy taken (e.g., ε-greedy)
- One measure of regret comes from the "error" in the action values $|Q^*(a) - Q(a)|$

Regret

- **Regret**: Expected difference in cumulative rewards received due to optimal (clairvoyant) strategy vs actual strategy taken (e.g., ε-greedy)
- One measure of regret comes from the "error" in the action values $|Q^*(a) - Q(a)|$
- For UCB, it can be shown that that $\Pr\left[|Q^*(a) - Q(a)| > c\sqrt{\frac{\ln t}{N_t(a)}}\right] \leq t^{-2c^2}$
- $c\sqrt{\frac{\ln t}{N_t(a)}}$ acts as a "baseline" for the action value error $Q^*(a) - Q(a)$

Regret

- **Regret**: Expected difference in cumulative rewards received due to optimal (clairvoyant) strategy vs actual strategy taken (e.g., ε-greedy)
- One measure of regret comes from the "error" in the action values $|Q^*(a) - Q(a)|$

- For UCB, it can be shown that that $\Pr\left[|Q^*(a) - Q(a)| > c\sqrt{\frac{\ln t}{N_t(a)}}\right] \leq t^{-2c^2}$

- $c\sqrt{\frac{\ln t}{N_t(a)}}$ acts as a "baseline" for the action value error $Q^*(a) - Q(a)$

- Baseline decreases as N increases and increases (very slowly) as t increases
- *Likelihood* that error is larger than baseline decreases as t increases

Summary

- Multi-armed bandits model decision making with unknown consequences
- Fundamental tradeoff between exploration and exploitation

- We can keep track and summarize what we know using action values
- Action values can be efficiently updated online while decision making

- Exploration can be encouraged probabilistically, e.g. ε-greedy strategies
- Exploration can also take into account uncertainty in action value estimates, e.g. UCB strategies