Advanced Topics in Reinforcement Learning

Lecture 2: Bandits

Josiah Hanna
University of Wisconsin — Madison

Announcements

- Waiting list is starting to clear; talk to me if you’re not in yet.
- Reading Sign-Ups: https://docs.google.com/spreadsheets/d/1Fy0wqktiVnjp-_JIl4QneGdItEQkKZME_O7Bqs8h3h8/edit?usp=sharing
- Background survey: https://forms.gle/sB854sD9g7TjKCsH6
- First reading response due at 12pm. Grace given if not yet enrolled.

Learning Outcomes

After today's lecture, you will:

1. Be able to define the essential aspects of a multi-armed bandit problem.

2. Be able to formalize real-world decision-making applications as multi-armed bandit problems.

3. Be able to explain the exploration-exploitation trade-off.

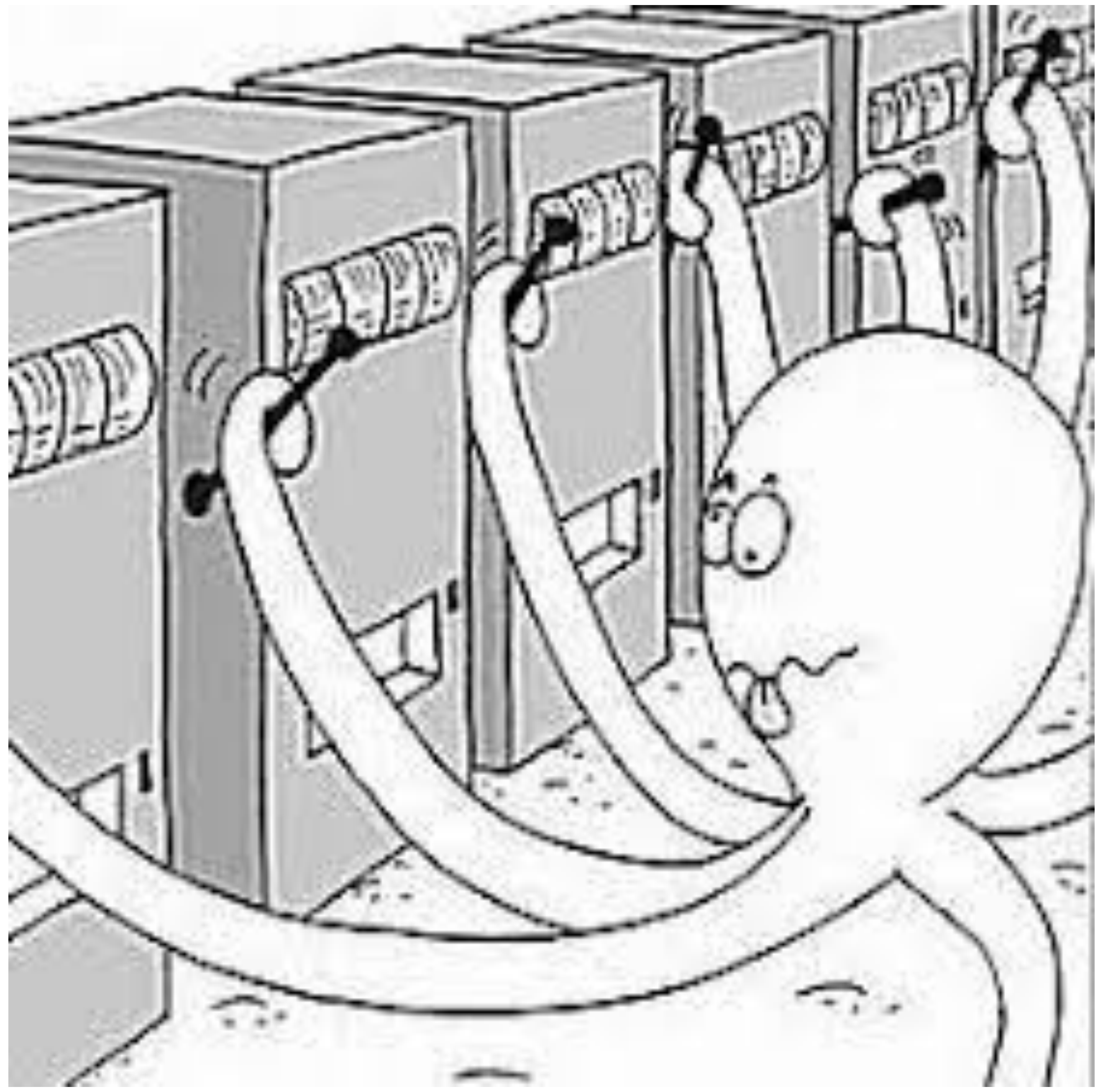
4. Be able to explain how to learn in bandits.

Today's Outline

- Do I have a bandit problem?
- Estimating action-values.
- Exploration vs. Exploitation.
- Policy-based learning.

Why Study Bandits?

- Simplest model of sequential decision-making.
- Build intuition for important concepts; many concepts extend to the more complex decision processes we focus on in this course.
- Today's lecture scratches the surface of a deep topic with extensive research, many applications, and many variations.
    - https://tor-lattimore.com/downloads/book/book.pdf



Full Reinforcement Learning

- States: $s \in \mathcal{S}$
- Actions: $a \in \mathcal{A}$
- Rewards: $R \sim r(s, a)$
- State transitions: $S \sim p(s, a)$
- Goal: Find a policy, $\pi : \mathcal{S} \rightarrow \mathcal{A}$, that maximizes cumulative reward.

Bandit Problems

- ~~States: $s \in \mathcal{S}$~~ No state (or equivalently $|\mathcal{S}| = 1$)
- Actions: $a \in \mathcal{A}$ (also called "arms")
- ~~Rewards: $R \sim r(s, a)$~~ $R \sim r(a)$ with expected value $q(a)$.
- ~~State transitions: $S \sim P(s, a)$~~ Actions do not affect future decisions.
- ~~Goal: Find a policy, $\pi : \mathcal{S} \rightarrow \mathcal{A}$, that maximizes cumulative reward.~~
- Goal: Find the action with highest expected reward.

Contextual Bandit Problems

- States: $s \in \mathcal{S}$ Now we have multiple states.
- Actions: $a \in \mathcal{A}$ (still called "arms")
- Rewards: $R \sim r(s, a)$ with expected value $q(s, a)$.
- ~~State transitions: $S \sim P(s, a)$~~ Actions do not affect future state probability.
- Goal: Find a policy, $\pi : \mathcal{S} \rightarrow \mathcal{A}$, that maximizes cumulative reward.

Attractions and Limitations

- Bandits are a simple model that requires solving explore-exploit trade-off.
- Widely applicable to real world problems.
- No state — take an action and immediately faced with the same situation.
  - No need for planning or reasoning about delayed rewards.
- Immediate pay-off for action choice.
  - No need for credit assignment

Is my application a bandit?

- Hyper-parameter optimization for machine learning.
- Recommend ads and web content.
- Recommend medical treatments.
- Sending push notifications to promote app engagement.

Questions:

1. What are the actions and rewards?
2. Are rewards stochastic or deterministic?
3. Do you see any issues treating the application as a MAB (without context or state)?
4. How might we address any such problem without modeling state?
5. What trade-offs might there be between different algorithms in these applications?

Action-Values

- Need to estimate expected rewards for each arm. Denote the estimate as $Q_t(a)$.

- Each time we pull an arm, we update $Q_t(a)$.

- $Q_t(a) = \dfrac{\sum_{i=0}^{t} R_t \cdot I\{A_t = a\}}{N_t(a)}$.

- $Q_t(a)$ is the estimated *action-value* for action a at time t.

- $A_{t+1} \leftarrow \arg\max_{a \in \mathscr{A}} Q_t(a)$

Action-Values

- What is time and memory complexity of action-value updates?

$$O(t)$$

- $Q_t(a) = \frac{\sum_{i=0}^{t} R_t \cdot I\{A_t = a\}}{N_t(a)} = Q_{t-1}(a) + \frac{1}{N_t(a)}(R_t - Q_{t-1}(a))$

- New Estimate ← Old Estimate + Step Size x (Target - Old Estimate)

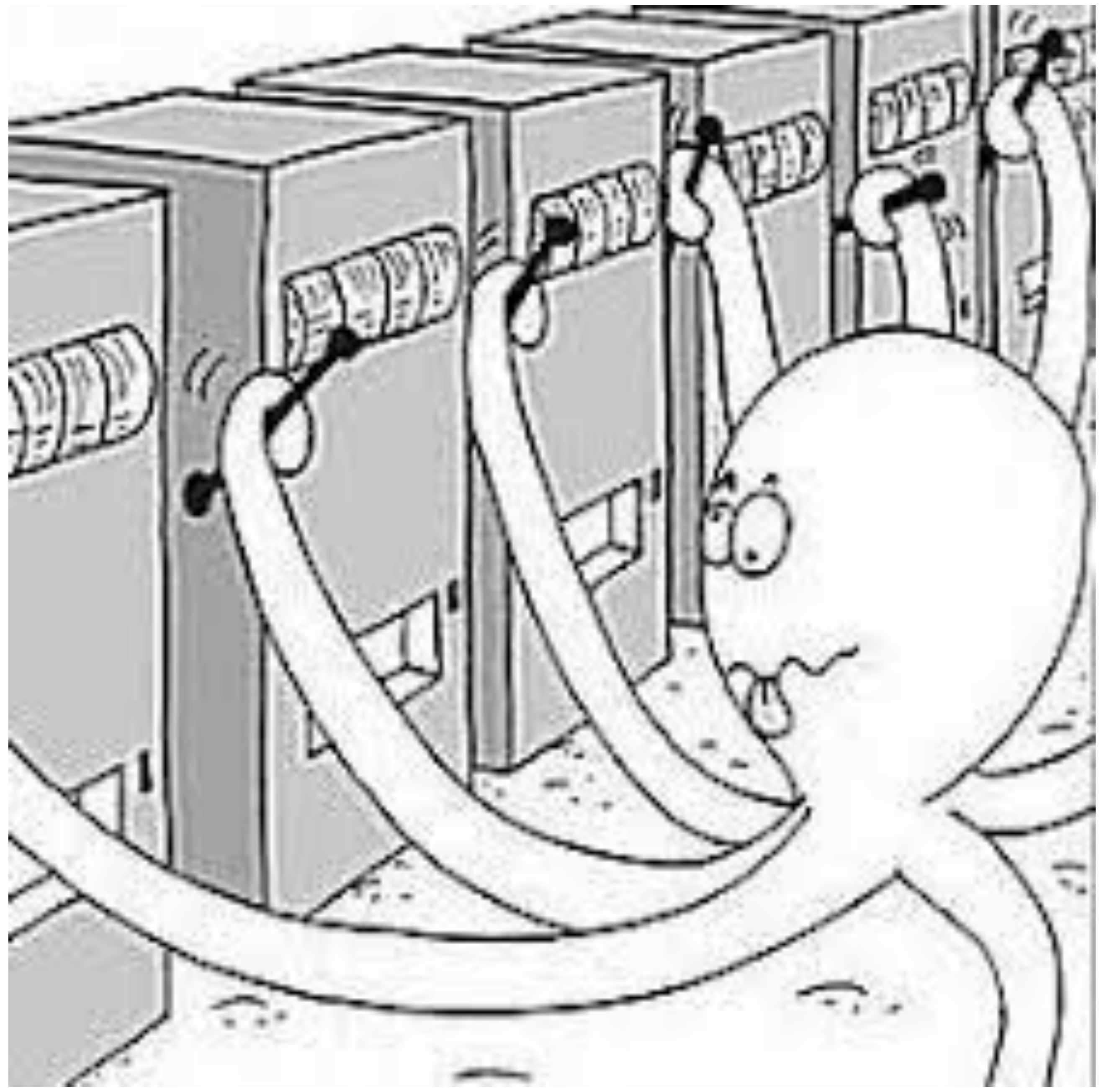
Step-Size Selection

- How does the choice of step-size affect algorithm behavior?
- New Estimate <— Old Estimate + Step Size * (Target - Old Estimate)
- When might you want a big step size? Small step-size?

Exploration vs Exploitation

"..the problem [exploration-exploitation] was proposed [by British scientist] to be dropped over Germany so that German scientists could also waste their time on it."

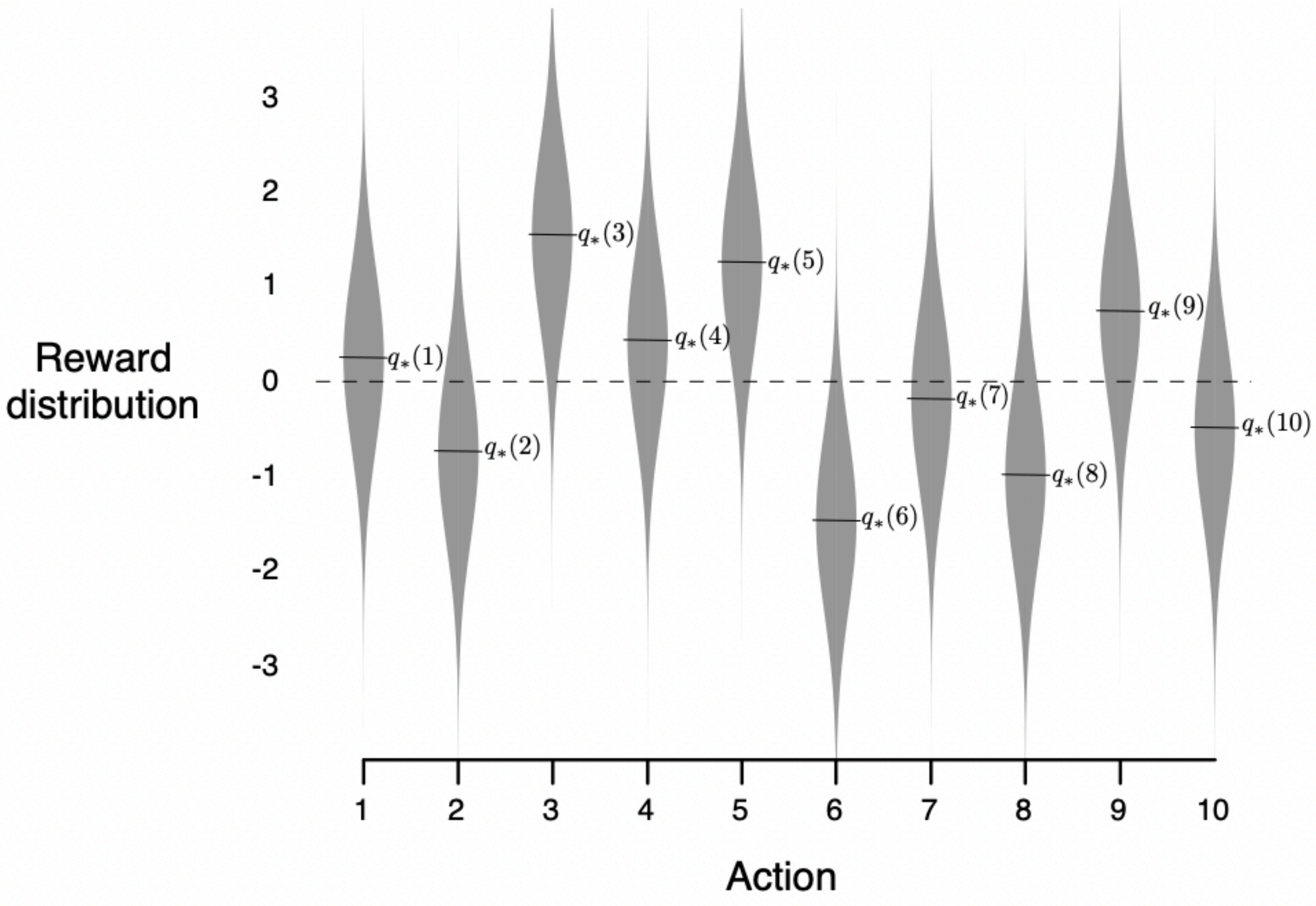
- Peter Whittle



Naive Exploration

- Greedy action selection: $A_{t+1} \leftarrow \arg\max_{a \in \mathcal{A}} Q_t(a)$
- What might go wrong?
- Simple Solution: pull the arm with the best **estimated** reward with probability $1 - \epsilon$, otherwise pull a random arm.
- The value of ϵ controls how much exploration we do.

Naive Exploration



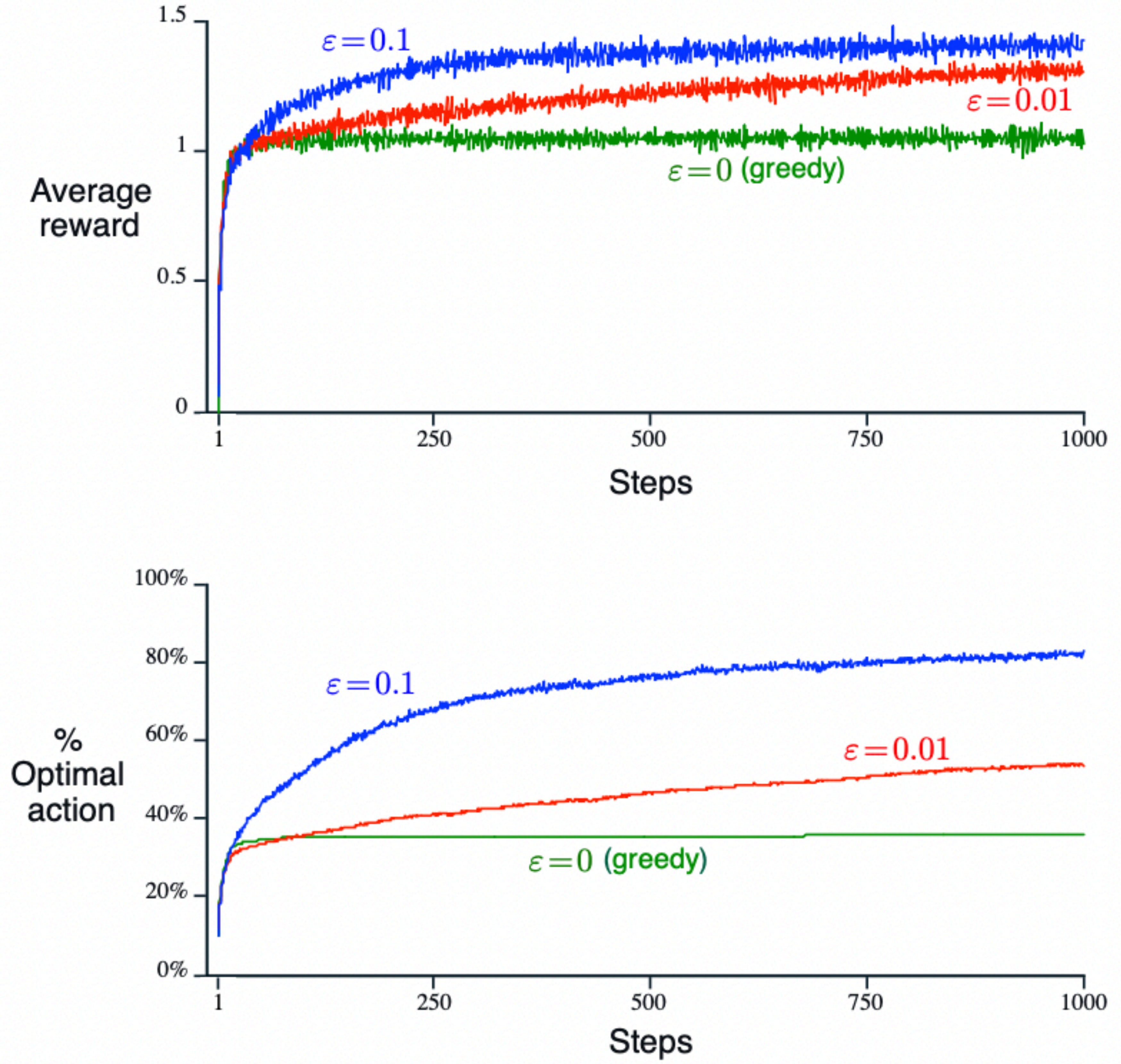
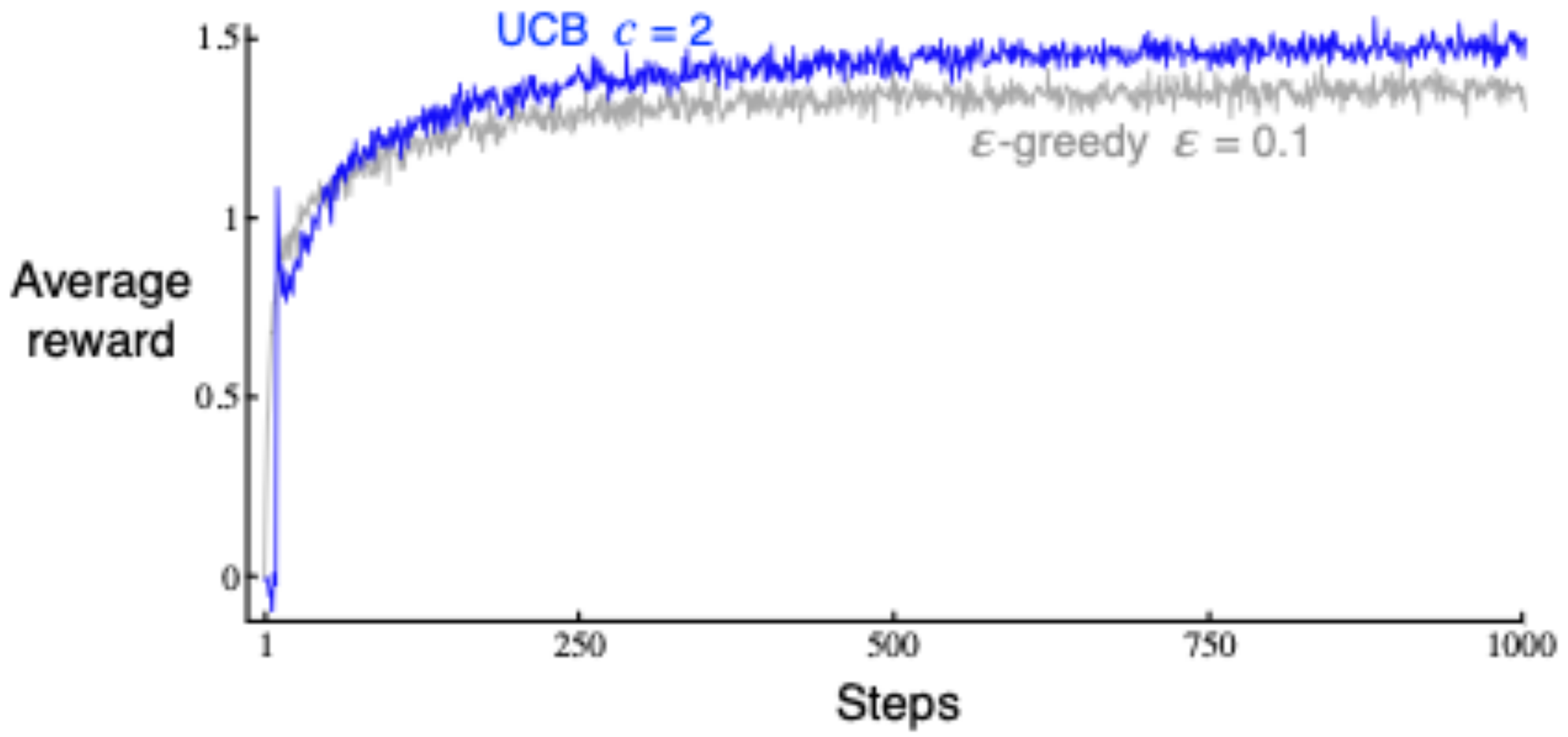


Figure 2.2: Average performance of ε-greedy action-value methods on the 10-armed testbed. These data are averages over 2000 runs with different bandit problems. All methods used sample averages as their action-value estimates.

UCB: Advanced Exploration

- Epsilon-greedy never stops exploring arms **even after clearly sub-optimal.**
- The upper confidence bound algorithm only explores actions that could have the highest expected reward.
- $$A_{t+1} = \arg\max Q_t(a) + c\sqrt{\frac{\ln(t)}{N_t(a)}}$$

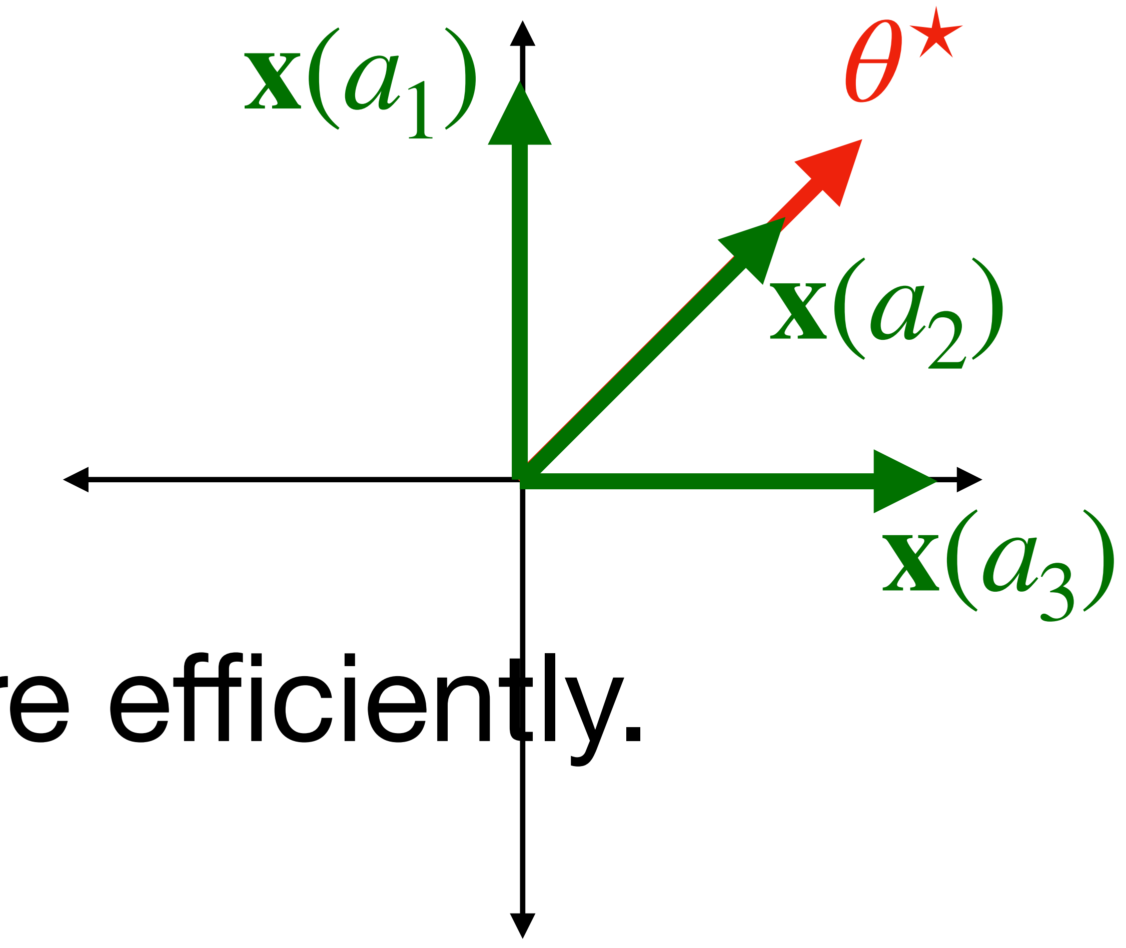


- The parameter c controls exploration vs. exploitation.

LinUCB: Exploration with Many Arms

- Imagine we have a large number of arms (e.g., 10^3). How does this affect UCB?

- Linear bandits: each arm has a feature vector $\mathbf{x}(a) \in \mathbb{R}^d$ and $q(a) = \theta^\star \mathbf{x}(a)^\top$ for unknown parameter $\theta^\star \in \mathbb{R}^d$.



- Goal is to estimate $\hat{\theta} \approx \theta^\star$.

- LinUCB exploits the structure in the action space to explore efficiently.

- Estimates a confidence ellipse around the estimate $\hat{\theta}$ and pulls arms that reduce volume of the ellipse.

Many more advanced algorithms and work going beyond the linearity assumption!

Policy-based Learning

- Ultimately, we just want action selection!
- Instead of estimating action-values, maintain action preferences.
- Let $H_t(a)$ be the preference for action a at time t.
- Select action with softmax probability: $\Pr(A_{t+1} = a) := \dfrac{\exp H_t(a)}{\sum_b \exp H_t(b)} = \pi(a)$
- Update preferences:

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

$$H_{t+1}(a) \leftarrow H_t(a) - \alpha(R_t - \overline{R}_t)\pi_t(a) \qquad a \neq A_t$$

Learning with a baseline

Obtained by following the book's derivation with the baseline set to zero

- Update rule without baseline: $H_{t+1}(A_t) \leftarrow H_t(A_t) + \alpha R_t(1 - \pi_t(A_t))$
- How does this update change the preferences? What happens when all rewards are positive? All negative?
- With a baseline: $H_{t+1}(A_t) \leftarrow H_t(A_t) + \alpha(R_t - \overline{R}_t)(1 - \pi_t(A_t))$

  Update in proportion to how much better than average instead of how much above zero.
- Value of $\overline{R}_t$ doesn't change expected update **as long as independent of A_t.**

Summary

- Multi-armed bandits provide a simplified setting for studying sequential decision-making.
- Estimating action-values provides a means to optimal action selection.
- Must sufficiently explore to find maximum reward action.
- Key ideas: action-values, incremental updates, step-sizes, epsilon-greedy exploration, gradient-based learning.

Action Items

- Join Piazza!
- Join Gradescope!
- Background survey: https://forms.gle/sB854sD9g7TjKCsH6
- Read Chapters 4 and 5 of the course textbook.
- Send a reading response by 12 pm on Monday.
- Sign-up for a presentation (link on Piazza).