DS-GA 3001 005 | Lecture 2

Reinforcement Learning

Jeremy Curuksu, PhD
NYU Center for Data Science
jeremy.cur@nyu.edu

February 7, 2024

DS-GA 3001 RL Curriculum

Reinforcement Learning:

- Introduction to Reinforcement Learning
- **Multi-armed Bandits**
- Dynamic Programming on Markov Decision Process
- ”Model-free” Reinforcement Learning
- Value Function Approximation (Deep RL)
- Policy Function Approximation (Actor-Critic)
- Planning from a Model of the Environment
- Examples of Industrial Applications
- Advanced Topics and Development Platforms

Multi-armed Bandit

Last week:

- What is Reinforcement Learning?
- Key components of Reinforcement Learning
- Introduction to the Gym Python library

Today:

- **Multi-armed Bandit with action values**
- **Upper Confidence Bound**
- **Bayesian Bandit**
- **Policy Gradient Bandit**

Multi-armed Bandit with action values

What is Multi-armed Bandit?

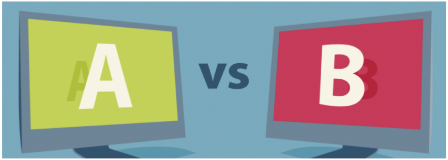
What is Multi-armed Bandit?

The Multi-armed Bandit problem

- Reinforcement learning uses data it receives to evaluate actions (correct actions are not given), which creates a need to explore
- A Bandit is a RL problem involving learning to act in only one situation: 1 state, k possible actions
- No sequential structure, past actions do not influence the future: the distribution of reward r_t given a_t is identical and independent across time

What is Multi-armed Bandit?

Example of Multi-armed Bandit problem

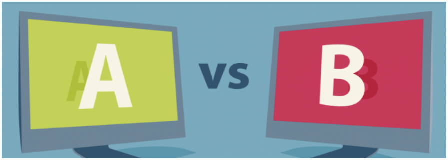


	action	\$ return
Week 1	A	
Week 2	B	
Week 3	?	

Which lever would you pull?

What is Multi-armed Bandit?

Example of Multi-armed Bandit problem



	action	$ return
Week 1	A	
Week 2	B	
Week 3	B	
Week 4	?	

How about now?

Exploration vs. Exploitation

Online decision-making involves a fundamental choice:

Exploitation:

Maximize performance using current knowledge

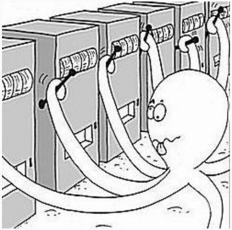
Exploration:

Increase knowledge

- The best strategy may involve short-term sacrifices
- The agent needs gather enough information to make the best overall decisions

Multi-armed Bandit Formalism

Problem Statement:



- The agent is faced repeateadly with a choice among k different actions (”*arms*”)
- At each step t the agent selects an action a_t
- After each choice it receives a numerical reward r_t that depends on the action selected
- The distribution $p(r|a)$ is fixed but unknown
- Goal is to maximize cumulative reward:

$$\sum_{i=1}^{t} r_i$$

Exploit knowledge with action value

Action value for action a is the expected reward:

$$q(a) \doteq \mathbb{E}(r|a) = \sum_{r \in (R)} p(r|a) \times r = \lim_{t \to +\infty} \frac{1}{t} \sum_{i=1}^{t} r_i | a$$

- An estimate is the average of the sampled rewards:

$$q_t(a) \doteq \frac{\textit{sum of rewards when a taken prior to t}}{\textit{number of times a taken prior to t}}$$

- With an estimate of $q(a)$, we can select an action:

Greedy policy: $a_t \doteq \arg\max_a q_t(a)$

Incremental implementation

The agent can learn online with a moving average:

For $a = a_t$:
$$q_t(a) = \frac{1}{t}\sum_{i=1}^{t} r_i | a$$

$$q_t(a) = q_{t-1}(a) + \frac{1}{t}\left(r_t - q_{t-1}(a)\right)$$

$\forall\, a \neq a_t$:
$$q_t(a) = q_{t-1}(a)$$

For non-stationary problems, the agent can *track* q(a):

$$q_t(a) = q_{t-1}(a) + \alpha\left(r_t - q_{t-1}(a)\right)$$

Explore new actions with ϵ-greedy

The agent must explore to learn q-values

- Greedy selection always exploits current knowledge on q-values to maximize reward, it never explore

- Alternative: Behave greedily most of the time, but every once in a while select a random action

- **ϵ-greedy algorithm:**
  - Select random action (explore) with $p = \epsilon$
  - Select greedy action (exploit) with $p = 1 - \epsilon$

- ϵ-greedy ensures all actions can be sampled indefinitely:

$$\lim_{t \to +\infty} q_t(a) = q(a)$$

Practice: k-armed Bandit Algorithm

k-armed Bandit both evaluates $q(a)$ and improves a:

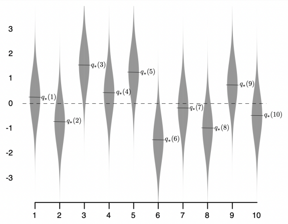
```
Initialize, for a = 1 to k:
  q(a) = 0
  n(a) = 0

Loop forever:
  a  = random action  with p = epsilon
  or = argmax q(a)    with p = 1 - epsilon
  Execute a, observe r
  n(a) = n(a) + 1
  q(a) = q(a) + 1/n(a) * (r - q(a))
```

Case Study: 10-armed testbed

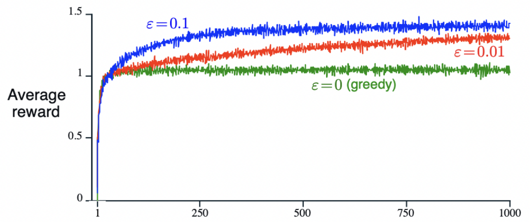
k-armed Bandit problem

Distribution of $q(a)$ vs. action a



Average performance of ϵ-greedy

Average reward over 2000 runs vs. time step



* Sutton and Barto, 1998

Total regret L_t

Analyzing regret in Multi-armed Bandit

- How can we reason about the exploration trade off?
- The (true) optimal value is: $v_* = \max_a q(a)$
- Regret is the opportunity loss at step t: $v_* - q(a_t)$
- Thus the best trade-off between exploration and exploitation is the one that minimizes total regret L_t:

$$L_t = \sum_{i=1}^{t} (v_* - q(a_i))$$

- The agent cannot measure regret directly, but regret can be used to analyze different RL algorithms on solved problems

Action Regret Δ_a

Analyzing regret in Multi-armed Bandit

- The action regret Δ_a for an action a is the difference between the optimal value and the true value of a:

$$\Delta_a = (v_* - q(a))$$

- Total regret can be defined by action regrets and action counts:

$$L_t = \sum_{i=1}^{t} (v_* - q(a_i)) = \sum_{a \in (A)} N_t(a)(v_* - q(a)) = \sum_{a \in (A)} N_t(a)\Delta_a$$

- Thus the best trade-off between exploration and exploitation is the one that ensures small count for actions with large regret

Upper Confidence Bound

Explore new actions with UCB

Upper Confidence Bound (UCB)

- For each action value $q(a)$, compute an upper confidence $u_t(a)$ such that $q(a) \leq q_t(a) + u_t(a)$
- Select action that maximizes this Upper Confidence Bound:

$$a_t = \underset{a \in (A)}{\arg\max} \, [q_t(a) + u_t(a)]$$

where:

$$u_t(a) = c\sqrt{\frac{\ln t}{N_t(a)}}$$

- The UCB algorithm can achieve logarithmic expected total regret (demonstration out of scope)

Explore new actions with UCB

Upper Confidence Bound (UCB)

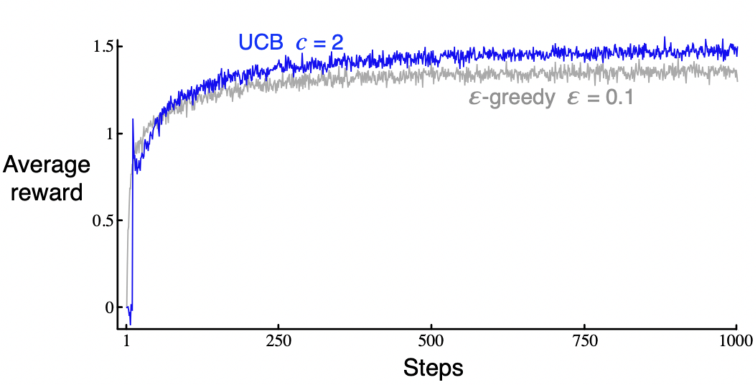
$$a_t = \underset{a \in (A)}{\arg\max} \left[q_t(a) + c \sqrt{\frac{\ln t}{N_t(a)}} \right]$$

- Uncertainty depends on number of times an action is selected:
  - **Small N_t(a)** $\Rightarrow$ **Large $u_t(a)$** $\Leftarrow$ Estimated q-value uncertain
  - **Large N_t(a)** $\Rightarrow$ **Small $u_t(a)$** $\Leftarrow$ Estimated q-value is accurate
- UCB favors an action because its estimated q-value is high, or because it has not been explored a lot relative to time elapsed
- UCB guarantees all actions will be explored without the need to manually predefine an ϵ-schedule

Case Study: 10-armed testbed

Average performance of ϵ-greedy and UCB algorithms

Average reward over 2000 runs vs. time step



Bayesian Bandit

The Bandit Model

A Bandit model is a reward transition function:

$$p\,(r\,|\,a) = p\,(r_{t+1} = r\,|\,a_t = a) \;\;\Leftrightarrow\;\; r(a) = \mathbb{E}(r, a)$$

$$\text{where}\;\; \mathbb{E}(r|a) = \sum_{r \in (R)} p(r|a) \times r = \lim_{t \to +\infty} \frac{1}{t} \sum_{i=1}^{t} r_i$$

The Bandit Model

A Bandit model is a reward transition function:

$$p\,(r\,|\,a) = p\,(r_{t+1} = r\,|\,a_t = a) \;\;\Leftrightarrow\;\; r(a) = \mathbb{E}(r, a)$$

$$\text{where}\;\; \mathbb{E}(r|a) = \sum_{r \in (R)} p(r|a) \times r = \lim_{t \to +\infty} \frac{1}{t} \sum_{i=1}^{t} r_i$$

- **Bandit model-based algorithm:** (Expectation model)

$$\hat{r}_t(a) = \hat{r}_{t-1}(a) + \alpha\,(r_t - \hat{r}_{t-1}(a))$$

The Bandit Model

A Bandit model is a reward transition function:

$$p\,(r \mid a) = p\,(r_{t+1} = r \mid a_t = a) \quad \Leftrightarrow \quad r(a) = \mathbb{E}(r, a)$$

$$\text{where} \quad \mathbb{E}(r|a) = \sum_{r \in (R)} p(r|a) \times r = \lim_{t \to +\infty} \frac{1}{t} \sum_{i=1}^{t} r_i$$

- **Bandit model-based algorithm:** (Expectation model)

$$\hat{r}_t(a) = \hat{r}_{t-1}(a) + \alpha\,(r_t - \hat{r}_{t-1}(a))$$

- **Bandit value-based algorithm:**

$$q_t(a) = q_{t-1}(a) + \alpha\,(r_t - q_{t-1}(a)) \quad \text{...Identical?}$$

Bayesian Bandit

Bayesian Bandit models the full distribution of rewards:

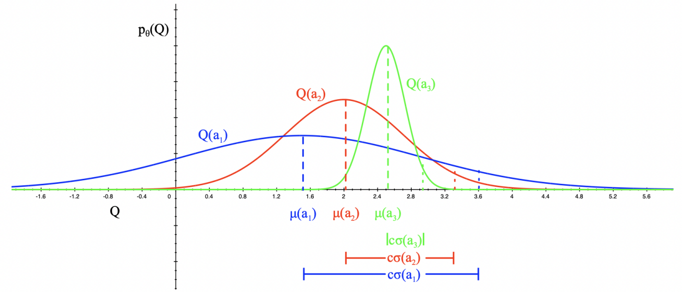
- Bayesian Bandit tracks a parameterized distribution function of expected reward $p(\mathbb{E}(r)|\theta, a)$, called likelihood function
- Selects actions based on $p(\mathbb{E}(r)|\theta, a)$ e.g., using UCB
- Uses reward observed to update posterior distributions of θ:

$$p_t(\theta|r) \propto p(\mathbb{E}(r)|\theta, a) \times p_{t-1}(\theta|r)$$

- For example, $\theta = (\mu, \sigma)_a$ if $p(\mathbb{E}(r)|\theta, a)$ are Gaussian distributions

Example: Bayesian Bandit with UCB

Apply UCB to a Bayesian Bandit model:



- Define Gaussian likelihood function: $p(\mathbb{E}(r)|\theta, a) = p_\theta(q(a))$ with mean $\mu_t(a)$ and standard deviation $\sigma_t(a)$ for each action
- Select greedy action with UCB: $a_t = \arg\max_a \left(q_t(a) + c\sigma_t(a)\right)$
- Adjust $\mu_t(a)$ and $\sigma_t(a)$ for a_t based on r_t actually observed

Bandit with Thompson Sampling

Bayesian model with Probability Matching:

- Instead of selecting actions from q-values with highest mean according to $p_\theta(q(a))$ with ϵ-greedy or UCB, Thompson sampling explicitly samples q-values from $p_\theta(q(a))$
- Thompson sampling selects action a according to probability that $q(a)$ is the maximum given the data sampled so far:

$$\pi_t(a) \doteq p\left(q(a) = \max_{a'} q(a') \mid \text{history}_{t-1}\right)$$

$$\pi_t(a) = \mathbb{E}\left(\mathcal{I}\left(q_t(a) = \max_{a'} q_t(a')\right) \mid \text{history}_{t-1}\right)$$

$$\pi_t(a) \simeq \frac{1}{t-1}\sum_{i=1}^{t-1}\left(\mathcal{I}\left(q_i(a) = \max_{a'} q_i(a')\right)\right)$$

$$\text{where} \quad \mathcal{I}(\text{True}) = 1\,, \quad \mathcal{I}(\text{False}) = 0$$

Toward Sequential RL and MDP...

Information State Space Bandit Model

- Bayesian Bandit tracks an evolving probability distribution of reward, which can be considered an *information state* s_t
- Each action a_t causes a transition to a new state s_{t+1} (by adding information), which is a sequential RL problem
- The tree of possible chains of events grows extremely rapidly, so approximate RL methods (lectures 5-7) are required

Toward Sequential RL and MDP...

Contextual Bandits

- Bandit with more than one state...
- If context on distinctive states are given to the agent, it can learn actions and values specific to each state
- In this case, the actions selected may depend on the state, but they do not affect which next states can be accessed later
- This is a simplified case of more general sequential RL problem where actions may affect next states and thus future possible rewards

Policy Gradient in Multi-armed Bandit

Policy Gradient Bandit

Can we learn a policy without learning values?

- Yes we can!
- Define a parameterized function $\pi_\theta(a) : a \mapsto p_\theta(a)$ and learn parameters θ that maximize a performance measure $J_{\pi_\theta}(a)$
- $\pi_\theta(a)$ can be arbitrary (just need distinguish possible actions)
- $J_{\pi_\theta}(a)$ can also be arbitrary (e.g. "always turn right in a maze")
- If $J_{\pi_\theta}(a)$ is unknown, it needs to be learned... It is often defined based on $q_t(a)$ = the *critic* in Actor-Critic algorithms:

$$\theta = \theta + \alpha \nabla_\theta \, q(a)$$

- **Out of scope for today (covered in depth in lecture 6)**

Today's Takeaways

Bandits is an RL problem where there is only one state

- **The fundamental problem is to balance exploration and exploitation to behave optimally**
- **To balance exploration and exploitation, the agent can use ϵ-greedy** which is a simple but efficient way to do it
- **...or UCB** which explicitly measures uncertainty to balance exploration and exploitation
- **...or a parameterized policy** with arbitrary objective measure J_θ
- **For example, J_θ can be the q-values parameterized by their means and standard deviations**, themselves updated based on the sampled rewards, as done in Thompson Sampling

Thank you!