

Lecture Schedule

Dynamical programming

- 1 The finite-horizon decision problem
2 February
- 2 Dynamical Programming
9 February
- 3 DP reformulations and introduction to Control
16 February

Control

- 4 Discretization and PID control
23 February
- 5 Direct methods and control by optimization
1 March
- 6 Linear-quadratic problems in control
8 March
- 7 Linearization and iterative LQR
15 March

Syllabus: <https://02465material.pages.compute.dtu.dk/02465public>
Help improve lecture by giving feedback on DTU learn

Reinforcement learning

- 8 **Exploration and Bandits**
22 March
- 9 Policy and value iteration
5 April
- 10 Monte-carlo methods and TD learning
12 April
- 11 Model-Free Control with tabular and linear methods
19 April
- 12 Eligibility traces and value-function approximations
26 April
- 13 Q-learning and deep-Q learning
3 May

Reading material:

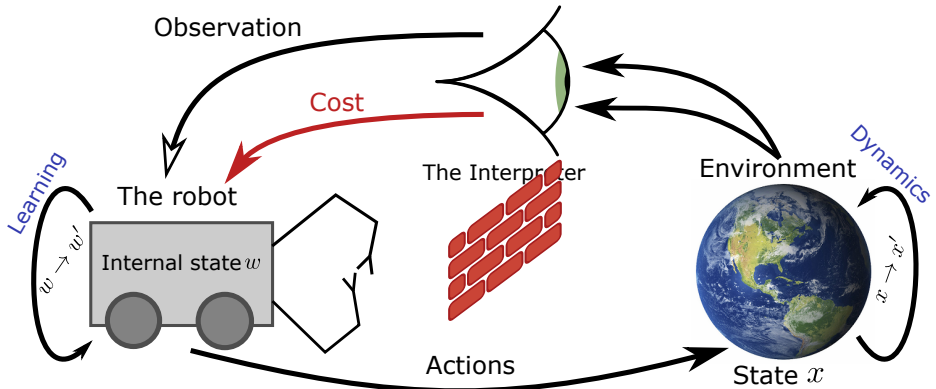
- [SB18, Chapter 1; Chapter 2-2.7; 2.9-2.10] Only as background

Learning Objectives

- Exploration/exploitation problem
- Bandits as a simplified reinforcement learning setting
- Formalizing the bandit problem
- Algorithms for solving the bandit problem

Housekeeping

- Project evaluations are online
- Many singleton groups – try to merge groups (use Discord or the classrooms)
 - No advantage to working alone.
- Sorry for messing up the quiz last time; **Option a** was correct (I remembered it as false and didn't read it during class.); I have uploaded an improved version of the quiz in the slides and will refrain from double-negation.

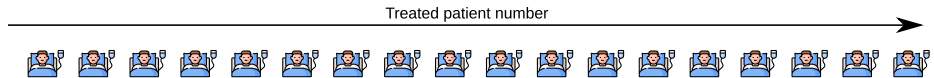


- Dynamics of world not known
- Simultaneously **learn the environment** and **maximize expected reward**
- Balance **exploration** and **exploitation**

Bandit studies this in an idealized setting

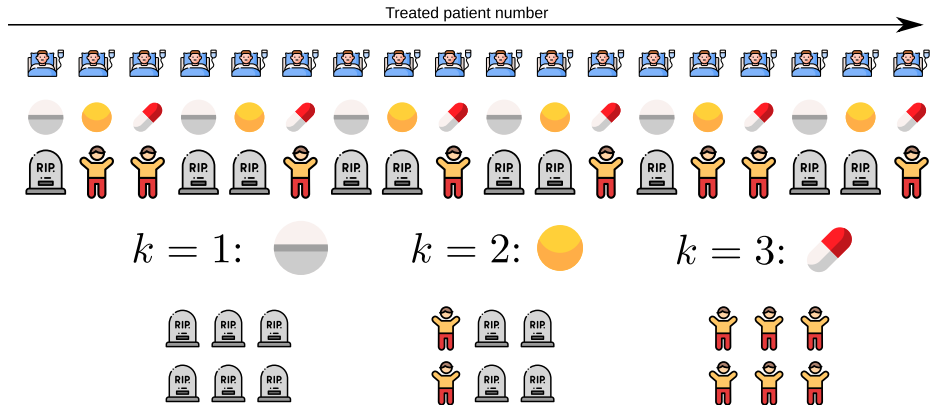
Bandits, examples

- Suppose you have a large number of patients $t = 1, 2, \dots$ with the same disease
- You have access to k drugs $a = 0, 1, \dots, k - 1$ with different outcome probabilities
- Outcome of treatment is either that the patient recovers, $R_t = 1$, or not $R_t = 0$
- Goal is to maximize $\sum_{t=1}^T R_t$



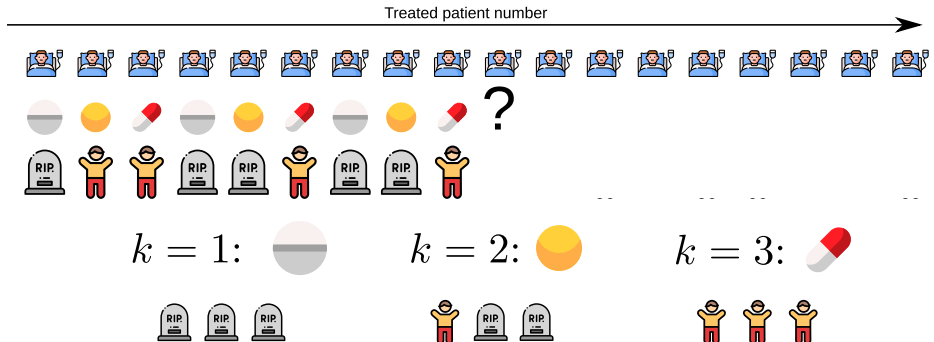
Idea 1: Statistics!

- Divide first T patients into K groups of $S = \frac{T}{K}$ patients
- Administer drugs to each group
- observe results



Bandit approach

- After $t - 1$ choices of actions A_1, \dots, A_{t-1} and observed rewards R_1, \dots, R_{t-1}
- Decide next action A_t to maximize reward
- **Bandit assumption:** Action A_t only affects R_t
 - Personalized medicine
 - Evaluating similar, approved drugs (low risk)
 - SMART trials/JITAs



Example: An opinion columnist

Suppose you are writing for a major newspaper which relies on social media to get as many reads as possible. You can choose between 5 headlines, and your job is to get as many clicks as possible:

- $k = 0$: "With less destructive nukes on the way, it's time for the left to say good-bye to those annoying non-proliferation treaties. "
- $k = 1$: "Opinion | The upside of nuclear war? Making popcorn without a microwave."
- $k = 2$: "Joe Biden has prevented a nuclear holocaust. But how will that play with suburban moms this fall?"
- $k = 3$: "Opinion | Nuclear war may not be woke. But it's not a war crime."
- $k = 4$: "Opinion | With rising temperatures, would a nuclear winter really be that bad?"

But which one to choose?

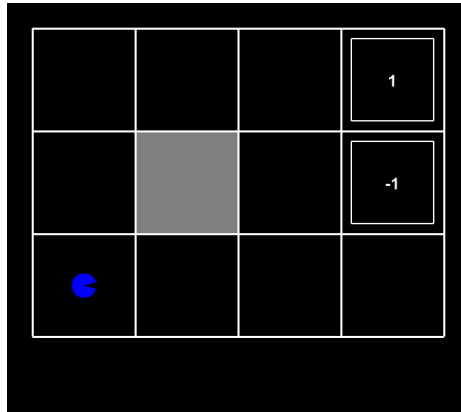
Example: An opinion columnist

- For each exposure $t = 1, 2, \dots$ on twitter, selects a headline $A_t = 0, \dots, k - 1$
- Observe whether the user clicks the story $R_t \in \{0, 1\}$
- Use this to select the next headline for the next user $A_{t+1} = a$
- **You want to maximize total clicks, knowing the story has a finite lifespan:**

$$\overset{\text{2-3 days?}}{\sum_{t=1}} R_t$$

Looking ahead: Reinforcement learning

- In a state s , select optimal action a , then observe what reward we get
- It is like a bandit problem in each state (but more about that in a few weeks)



Many types of bandits

Sequentially take decisions A_1, A_2, \dots and observe rewards R_1, R_2, \dots

Stationary In a stationary bandit the reward distribution does not change

Nonstationary The environment can change (but not as consequence of our actions)

Contextual You get a bit of information to make your decision

Structured Reward of different arms can be inferred from each other (Bayesian black box optimization)

Stationary bandits

- Action at time step $t = 1, 2, \dots$ is A_t
- Reward is R_t
- Observations available to make action at t :

$$H_t = (A_1, R_1, A_2, R_2, \dots, A_{t-1}, R_{t-1})$$

- Actions are generated from a **policy** π which we learn based on H_t :

$$A_t \sim \pi_t(\cdot)$$

- **Value** of an action is

$$q_*(a) = \mathbb{E}[R_t | A_t = a], \quad a = 0, \dots, K - 1$$

- Optimal strategy at t is to select action with highest value
- Our learned estimate of $q_*(a)$ at time t is $Q_t(a)$

Exploit Select action a with **highest** estimate of $Q_t(a)$

Explore Do something else to **learn** more about $Q_t(a)$

- Note bandit methods can be classified according to what they learn about $Q_t(a)$

Bandit objective and definitions

Objective 1: Average reward at time t and total reward up to time T

$$\mathbb{E}_\pi [q_*(a_t)], \quad \sum_{t=1}^T \mathbb{E}_\pi [q_*(a_t)]$$

Optimal value and optimal action

$$V^* = \max_a [q_*(a)], \quad a_t^* = \arg \max_a [q_*(a)]$$

Objective 2: Fraction optimal actions

$$P_\pi(A_t = a_t^*)$$

Gab

$$\Delta_a = V^* - q_*(a)$$

Objective 3: Cumulative regret

$$l_t = \mathbb{E} [V^* - q_*(a_t)], \quad L_T = \sum_{t=1}^T l_t$$

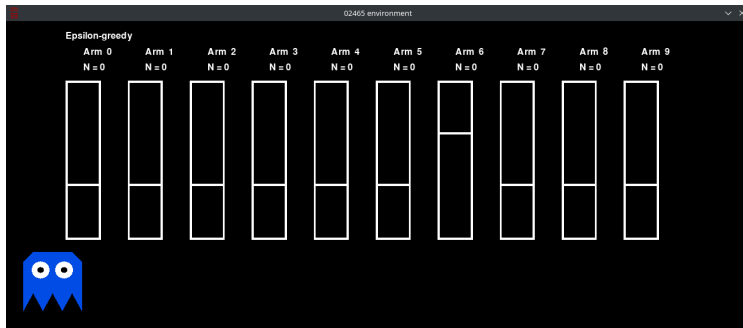
Goal is to maximize cumulative reward \leftrightarrow minimize total regret

Quiz: What is the regret?

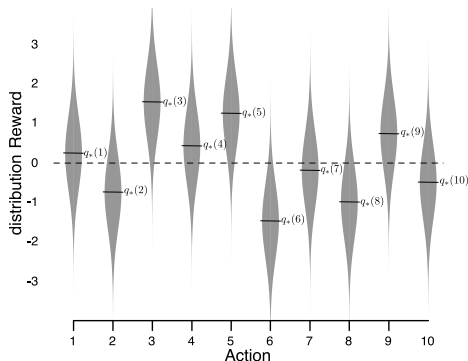
- Reward $R_t = 1$ on win and $R_t = 0$ on loss.
- The win probabilities are shown by horizontal lines
- What is the regret for a policy which always select $a = 3$? ($\pi(a = 3) = 1$)

$$l_t = \mathbb{E} [V^* - q_*(a_t)], \quad V^* = \max_a [q_*(a)]$$

- It is a random quantity (either zero or 1)
- It depends on how many actions we have taken
- It is about $\frac{1}{3}$
- It is about $-\frac{2}{3}$



The $k = 10$ -armed testbed



- Let $k = 10$ and select each $q_*(a) \sim N(\mu = 0, \sigma^2 = 1)$
- for each action a , select reward

$$R_t|a \sim \mathcal{N}(\mu = q_*(a), \sigma^2 = 1)$$

- Let each agent interact for a number of **steps** ~ 1000
- Repeat procedure for 2000 runs to calculate average agent performance

Making it practical: A bandit problem

```
1  # bandits.py
2  class BanditEnvironment(Env):
3      def __init__(self, k : int):
4          super().__init__()
5          self.observation_space = Discrete(1) # Dummy observation space with a single o
6          self.action_space = Discrete(k) # The arms labelled 0,1,...,k-1.
7          self.k = k # Number of arms
8
9
10     def reset(self):
11         raise NotImplementedError("Implement the reset method")
12
13     def bandit_step(self, a):
14         reward = 0 # Compute the reward associated with arm a
15         regret = 0 # Compute the regret, by comparing to the optimal arms reward.
16         return reward, regret
17
18     def step(self, action):
19         reward, average_regret = self.bandit_step(action)
20         info = {'average_regret': average_regret}
21         return None, reward, False, False, info
```

Action-value method

Idea: approximate $q_*(a)$ by keeping track of $Q_t(a)$

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbf{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbf{1}_{A_i=a}} = \frac{S_t(a)}{N_t(a)}$$

Explore with probability ϵ

- Action selection π
 - With probability ϵ select random action
 - With probability $1 - \epsilon$ select $a^* = \arg \max_a Q_t(a)$
- As only one entry A_t of Q_t change at a time track number of times a was selected $n = N(a)$:

$$Q_n \doteq \frac{R_1 + R_2 + \dots + R_{n-1}}{n-1} = \frac{S_n(a)}{N(a)} \quad (1)$$

One can show that:

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

- Given observed $a = A_t$, $r = R_t$ update:

Simple action-value bandit algorithm

A simple bandit algorithm

Initialize, for $a = 1$ to k :

$$Q(a) \leftarrow 0$$

$$N(a) \leftarrow 0$$

Loop forever:

$$A \leftarrow \begin{cases} \operatorname{argmax}_a Q(a) & \text{with probability } 1 - \varepsilon \quad (\text{breaking ties randomly}) \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

$$R \leftarrow \text{bandit}(A)$$

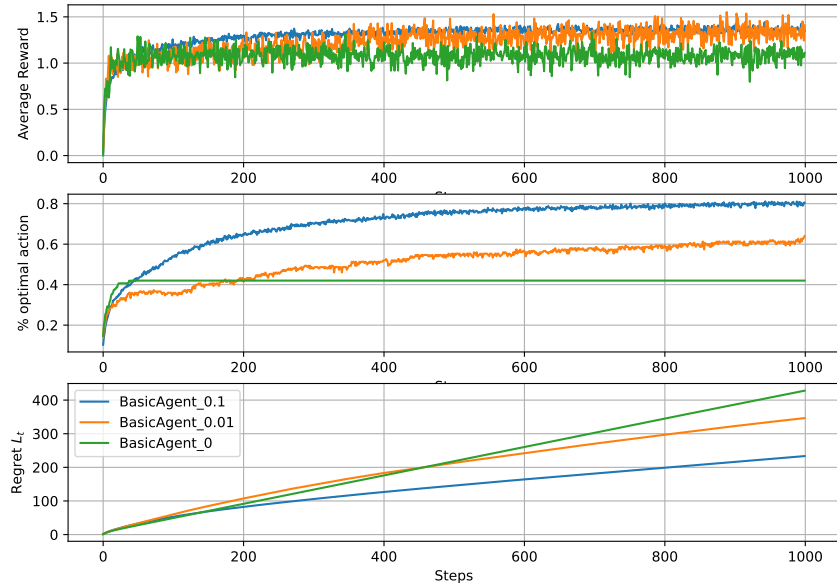
$$N(A) \leftarrow N(A) + 1$$

$$Q(A) \leftarrow Q(A) + \frac{1}{N(A)} [R - Q(A)]$$

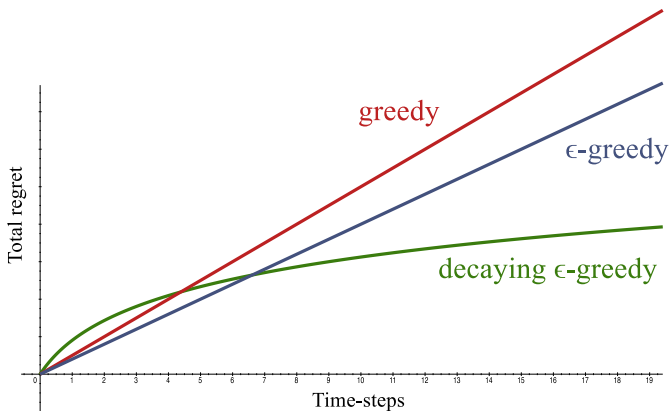
$$Q_{n+1} = Q_n + \frac{1}{n} [R_n - Q_n]$$

Results: action-value bandit

Evaluated on StationaryBandit_0 for 150 episodes



Regret asymptotics



- Fixed- ϵ algorithms have linear regret
- With decreasing ϵ it is possible to get sub-linear regret, **but only by assuming we know things about the reward distribution**
 - Theoretically best possible bandit method has logarithmic regret.

Confidence-bound methods

- Estimate an upper confidence bound $\hat{U}_t(a)$ for $q_*(a)$ st.

$$q_*(a) \leq \hat{U}_t(a) + Q_t(a)$$

with high probability

- Generally
 - If $N_t(a)$ low $\rightarrow \hat{U}_t(a)$ high
 - If $N_t(a)$ high $\rightarrow \hat{U}_t(a)$ low
- Select actions to minimize

$$\arg \min_a \left[\hat{U}_t(a) + Q_t(a) \right]$$

UCB1

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

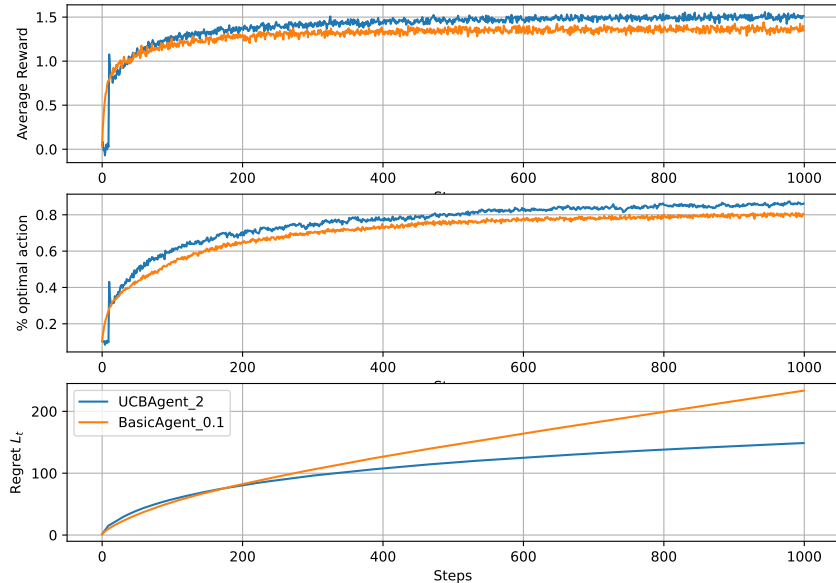
Asymptotic logarithmic regret when $R_t \in [0, 1]$

$$\lim_{t \rightarrow \infty} L_t \leq \sum_{a \neq a^*, \Delta_a > 0} \left(\frac{4 \ln t}{\Delta_a} + 2\Delta_a \right)$$

- The variant UCB-normal obtains logarithmic regret on normal reward distributions

Results

Evaluated on StationaryBandit_0 for 2000 episodes



Quiz: How does UCB explore?

Consider the update rule for UCB1:

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

Which one of the following statements is true about UCB1?

- a.** UCB1 requires that the rewards are positive
- b.** If one arm give a much higher reward than the other, UCB1 will eventually only select this arm
- c.** If one arm is much, much worse than the others, UCB1 will eventually stop selecting that arm
- d.** It is possible to predict which arms UCB1 will select k steps in the future
- e.** At least one of the upper-confidence estimates $\hat{U}_t(a)$ will converge to 0.
- f.** Don't know.

Non-stationary bandits

- These is a (hidden) state S_t which evolves as:

$$P(S_{t+1}, R_t | S_t = s, A_t = a) = P(S_{t+1} | S_t = s) P(R_t | S_t = s, A_t = a)$$

- Example: Add normal noise to $q_*(a)$ at each time step
- One idea is to replace $\frac{1}{n}$ with $\alpha_t(a)$ and use scheduling:

$$\text{Previous update: } Q_{n+1} = Q_n + \frac{1}{n} [R_n - Q_n]$$

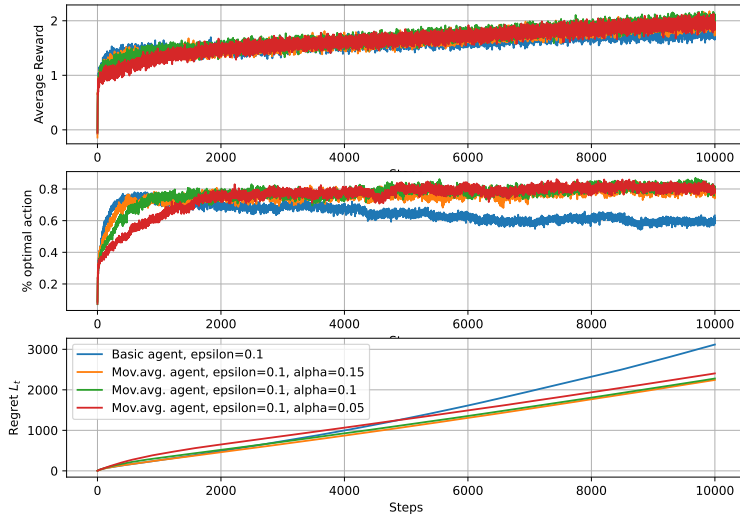
$$\text{New update: } Q_{n+1} = Q_n + \alpha [R_n - Q_n]$$

- Constant α means fast adaption but no convergence
- Typically chose

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

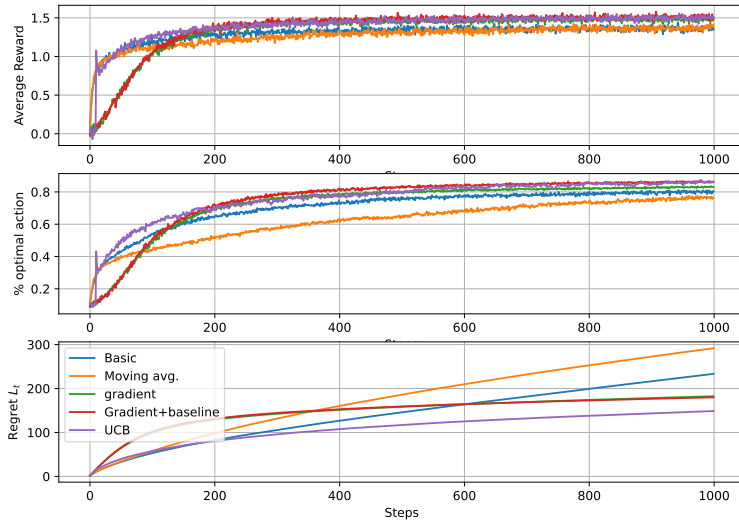
Results

Evaluated on NonstationaryBandit_0_0.01 for 400 episodes



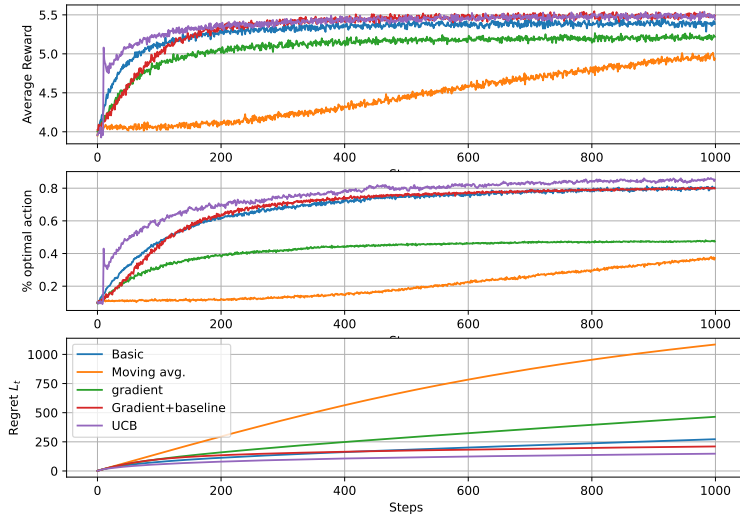
Results


Stationary bandit (no offset)



Results

Stationary bandit (with offset)



 Richard S. Sutton and Andrew G. Barto.
Reinforcement Learning: An Introduction.
The MIT Press, second edition, 2018.
(Freely available online).

Appendix: Probability-matching methods

- Our goal is to find the optimal probability distribution π
- We can parameterize any distribution as

$$\pi(a) = \frac{e^{H_a}}{\sum_{b=1}^k e^{H_b}}$$

for a weight-vector $H \in \mathbb{R}^k$

- Optimal π is the one maximizing expected reward

$$\mathbb{E}_\pi [R_t] = \sum_a \pi_t(a; H) q_*(a) = E(H)$$

- This is a function of H
- Let's just do gradient descent, WCGW?

$$H_{t+1} \leftarrow H_t - \alpha \nabla_H E(H)$$

$$\frac{\partial}{\partial H} E(H) = \sum_a \pi(a; H) q^*(a) \frac{\partial \log \pi(a; H)}{\partial H} \quad (2)$$

We can sample from $\pi(a)$ and then our environment will give an estimate of $q^*(a)$

$$\sum_a \pi(a; H) q^*(a) \frac{\partial \log \pi(a; H)}{\partial H} \approx \frac{1}{S} \sum_{s=1}^S R_t(a_s) \frac{\partial \log \pi(a_s; H)}{\partial H} \quad (3)$$

- Nobody has told us we cannot use $S = 1$

$$\nabla E(H) \approx R_t \frac{\partial \log \pi(a_t; H)}{\partial H}$$

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha R_t (1 - \pi_t(A_t)), \quad \text{and}$$

$$H_{t+1}(a) \doteq H_t(a) - \alpha R_t \pi_t(a), \quad \text{for all } a \neq A_t$$



Kullback-Leibner divergence Given discrete probability distribution p and q :

$$\text{KL}[p; q] = \sum_{i=1}^n p(x_i) \log \frac{q(x_i)}{p(x_i)}$$

The logarithm trick for $q(x, \theta) > 0$

$$\frac{\partial}{\partial \theta} \int q(x, \theta) f(x) dx = \int q(x, \theta) \frac{\partial \log q(x, \theta)}{\partial \theta} f(x) dx$$

- Let \bar{R}_t be the average reward over $0, \dots, t-1$
- Update weights as

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha (R_t - \bar{R}_t) (1 - \pi_t(A_t)), \quad \text{and}$$
$$H_{t+1}(a) \doteq H_t(a) - \alpha (R_t - \bar{R}_t) \pi_t(a), \quad \text{for all } a \neq A_t$$

- Why? **legal** because they do not change the gradient, **sensible** because they can reduce variance/promote exploration
- To my knowledge, no theoretical analysis exists
- This gradient-trick is basis of **policy gradient** methods for reinforcement learning

Evaluated on StationaryBandit_4 for 100 episodes

