Multi-arm Bandits

Corrado Possieri

Machine and Reinforcement Learning in Control Applications

Evaluative vs instructive feedback

Evaluation: how good the action taken is:

- active exploration;
- trial-and-error search.

Instruction: the correct action to take:

- independent of the action.

n-armed bandit

- Non-associative setting
  - does not involve learning to act in more than one situation.

An n-**armed bandit** problem:

- you are faced repeatedly with a choice among n actions;
- after each choice you receive a reward;
- your objective is to maximize the expected total reward.

Real life n-bandits

Slot machine: n different slot machines to play with.

- *Action:* which slot to play.
- *Reward:* payoff for hitting the jackpot.
- *Objective:* maximize winnings.

Experimental treatments: select treatment for ill patients.

- *Action:* treatment selection.
- *Reward:* survival or well-being of the patient.
- *Objective:* find new treatment.

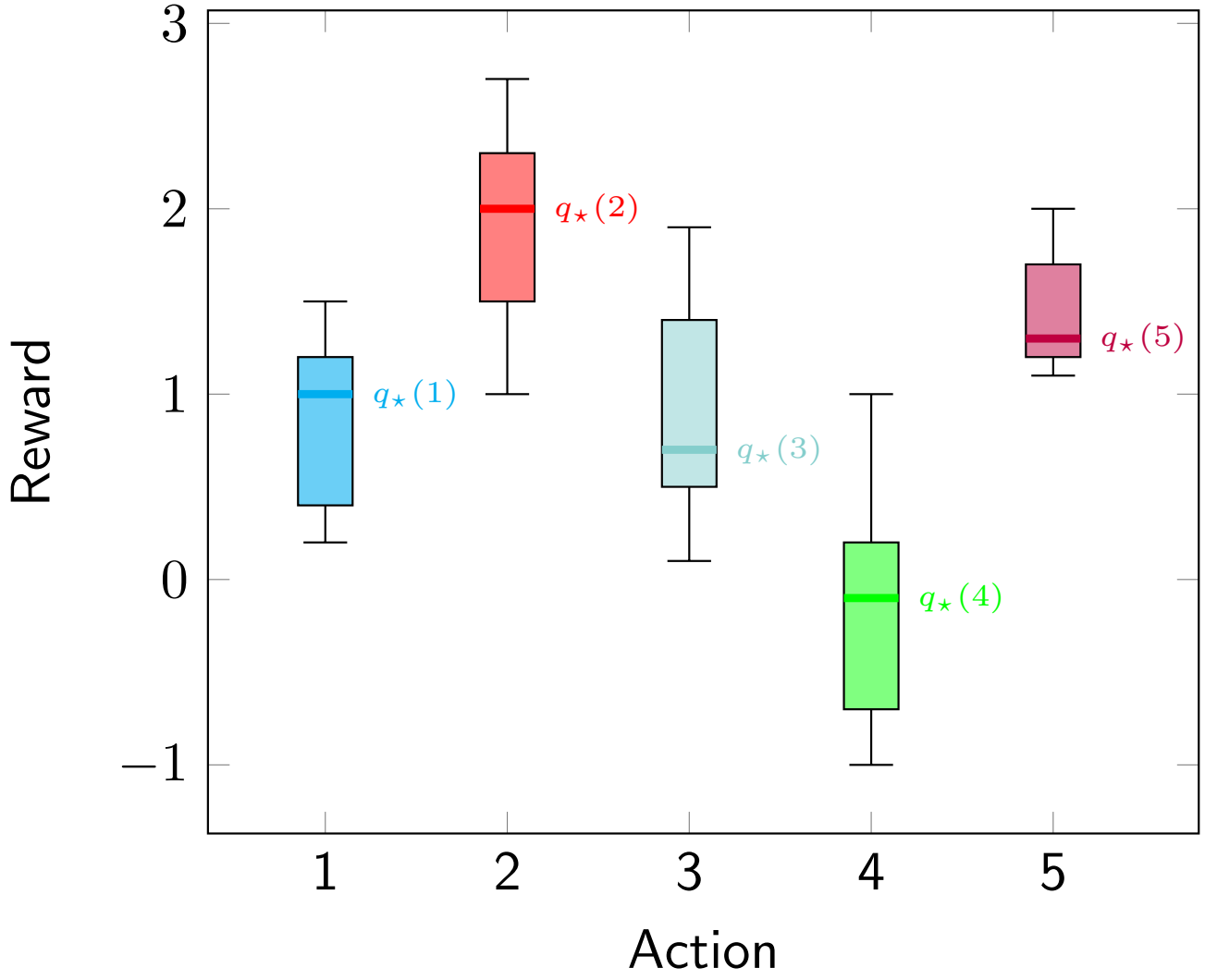
Mathematical setting

- $\mathcal{A} = \{1, \ldots, n\}$ is the set of all the actions.
- $A_t \in \mathcal{A}$ is the **action** selected at step t.
- R_t is the corresponding **reward**.
- The reward is chosen from a stationary probability distribution that depends on the action.
- The **value** of an action is the expected reward given that that action is selected:

$$q_\star(a) = \mathbb{E}[R_t | a = A_t].$$

- We do not know the action values with certainty, although we may have estimates.

Expected reward



Exploration vs exploitation

- A **greedy** action is an action whose estimated value is greater than all the others.
- Taking greedy actions means **exploiting** our knowledge.
- Taking non-greedy actions means **exploring**.
- Exploitation aims at maximizing the reward on one step.
- Exploration may produce greater reward in the long run.

Balancing exploration and exploitation

Reward is lower in the short run, during exploration, but higher in the long run because after you have discovered the better actions, you can exploit them many times.

Estimating the value of actions

Sample-average

- At time t, an estimate $Q_t(a)$ of $q(a)$ can be determined by averaging rewards (*sample-average*):
  - $N_t(a)$: times prior to t in which action a has been selected;
  - $R_1, R_2, \ldots, R_{N_t(a)}$: rewards gathered selecting action a;
  - $$\begin{cases} Q_t(a) = 0, & \text{if } N_t(a) = 0, \\ Q_t(a) = \dfrac{R_1 + R_2 + \cdots + R_{N_t(a)}}{N_t(a)}, & \text{if } N_t(a) > 0. \end{cases}$$
- $Q_t(a) \xrightarrow{t \to \infty} q(a)$, provided a is selected infinitely often.
- Greedy action:

$$A_t = \arg\max_a Q_t(a).$$

ε-greedy methods

- Greedy action selection exploits current knowledge to maximize immediate reward.
- An alternative is to select randomly from among all the actions with small probability ε:

$$A_t \in \begin{cases} \arg\max_a Q_t(a), & \text{with probability } 1-\varepsilon, \\ \mathcal{A}, & \text{with probability } \varepsilon. \end{cases}$$

- This guarantees that

$$N_t(a) \xrightarrow{t\to\infty} \infty,$$
$$Q_t(a) \xrightarrow{t\to\infty} q(a),$$

for all actions $a \in \mathcal{A}$.

Balance between exploration and exploitation

- Depends on the problem at hand.
- If the variance of the reward is
  - large: requires more exploration
    - larger values of ε are better;
  - small: greedy methods know the true value
    - smaller values of ε are better;
- Exploration is required even in the deterministic case.

Iterative implementation

- Sample average can be implemented iteratively.
- For some action a, let Q_k denote the estimate for its k-th reward (the average of its first $k-1$ rewards)

$$Q_{k+1} = \frac{1}{k}\sum_{i=1}^{k} R_i = \frac{1}{k}\left(R_k + \sum_{i=1}^{k-1} R_i\right)$$
$$= \frac{1}{k}\left(R_k + (k-1)Q_k\right) = Q_k + \frac{1}{k}\left(R_k - Q_k\right).$$

- Note that this equation has the form

$$\text{new\_estimate} = \text{old\_estimate} + \text{step\_size}\underbrace{\left(\text{target} - \text{old\_estimate}\right)}_{\text{error}}.$$

- Q_1 is the *initial guess*.

ε-greedy sample-average

A simple algorithm

Initialization

for $a = 1$ **to** n **do**

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Loop

$$A \leftarrow \begin{cases} \arg\max_a Q(a), & \text{with probability } 1-\varepsilon \\ \mathcal{A}, & \text{with probability } \varepsilon \end{cases}$$

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

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

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

Non-stationary problems

- If the environment is non-stationary, consider the update

$$Q_{k+1} = Q_k + \alpha \left(R_k - Q_k \right),$$

  where $\alpha \in (0, 1]$ is a constant.
- In this case, we have

$$\begin{aligned} Q_{k+1} &= \alpha R_k + (1 - \alpha) \left(\alpha R_{k-1} + (1 - \alpha) Q_{k-1} \right) \\ &= (1 - \alpha)^k Q_1 + \sum_{i=1}^{k} \alpha (1 - \alpha)^{k-i} R_i. \end{aligned}$$

- Q_{k+1} is a weighted average of past rewards and of Q_1.
- Older rewards have a smaller impact.

Convergence conditions

- The general update rule is

$$Q_{k+1} = Q_k + \alpha_k(a)\left(R_k - Q_k\right),$$

  where $\alpha_k(a)$ is the step size.

- Stochastic convergence in ensured if

$$\sum_{k=1}^{\infty} \alpha_k(a) = \infty, \qquad \sum_{k=1}^{\infty} \alpha_k(a)^2 < \infty.$$

- The step size $\alpha_k(a) = \frac{1}{k}$ satisfies the convergence condition.
- The constant step size does not
  - the update never completely converges.

Optimistic initialization

- Methods discussed so far depend on Q_1.
- The bias decreases over time.
- This bias can be exploited to enhance exploration:
  - Q_1 larger than expected outcome promotes exploration even under greedy actions;
  - whichever actions are initially selected, the reward is less than the starting estimates;
  - all actions are tried several times before the value estimates converge.

Upper confidence bound action selection

- ε-greedy action selection forces the non-greedy actions to be tried indiscriminately.
- It would be better to select among the non-greedy actions according to their potential for actually being optimal.
- **Upper confidence bound** action selection:

$$A_t = \arg\max_a \left\{ Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right\},$$

  where $c > 0$ controls the degree of exploration.
- The term $\sqrt{\frac{\ln t}{N_t(a)}}$ is a measure of the uncertainty.
- t increases at each time, whereas $N_t(a)$ does not
  - every action is selected infinitely often.

Combining UCB with optimistic initialization

UCB - optimistic initialization - constant step

Initialization

for $a = 1$ **to** n **do**

$Q(a) \leftarrow$ Large value

$N(a) \leftarrow 0$

Loop

$t \leftarrow t + 1$

$$A \leftarrow \arg\max_a \left\{ Q_t(a) + c\sqrt{\frac{\ln t}{N(a)}} \right\}$$

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

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

$Q(A) \leftarrow Q(A) + \alpha(R - Q(A))$

Preference updates

- Let $H_t(a)$ be the preference of selecting action a.
- Select actions according to a soft-max distribution

$$\Pr[A_t = a] = \frac{\exp(H_t(a))}{\sum_{b=1}^{n} \exp(H_t(b))} = \pi_t(a).$$

- Initially all preferences are the same.
- After selecting A_t and receiving R_t, update preferences as

$$\begin{cases} H_{t+1}(A_t) = H_t(A_t) + \alpha \left(R_t - \overline{R}_t\right)(1 - \pi_t(A_t)), & \text{and} \\ H_{t+1}(a) = H_t(a) - \alpha \left(R_t - \overline{R}_t\right)\pi_t(a), & \forall a \neq A_t, \end{cases}$$

where $\alpha > 0$ is a step size,
$\overline{R}_t$ is the average reward up to (including) t.

An interpretation of preference updates

- The average $\overline{R}_t$ acts as a *baseline* reward.
- If the gathered reward R_t is larger than the average $\overline{R}_t$
  - the probability of selecting A_t is increased.
- If the gathered reward R_t is smaller than the average $\overline{R}_t$
  - the probability of selecting A_t is decreased.
- The non-selected actions move in the opposite direction.
- The preference updates is a (stochastic) gradient ascent.

Associative tasks

- We have considered only non-associative tasks.
- In general problems we need to learn more than one situation.
- If each time a task is selected you have clues about its identity
  - the methods discussed so far can still be used
    - associate to each set of clues a policy.
- This cannot be done if actions affect the next situation.