Reinforcement Learning Specialization - Lecture Notes

Course 1 - Fundamentals of Reinforcement Learning

Module 2

New terms:

short/long-term reward

policies

planning methods

dynamic programming

reward

time steps

Video: Sequential Decision Making with Evaluative Feedback

Action-Value function

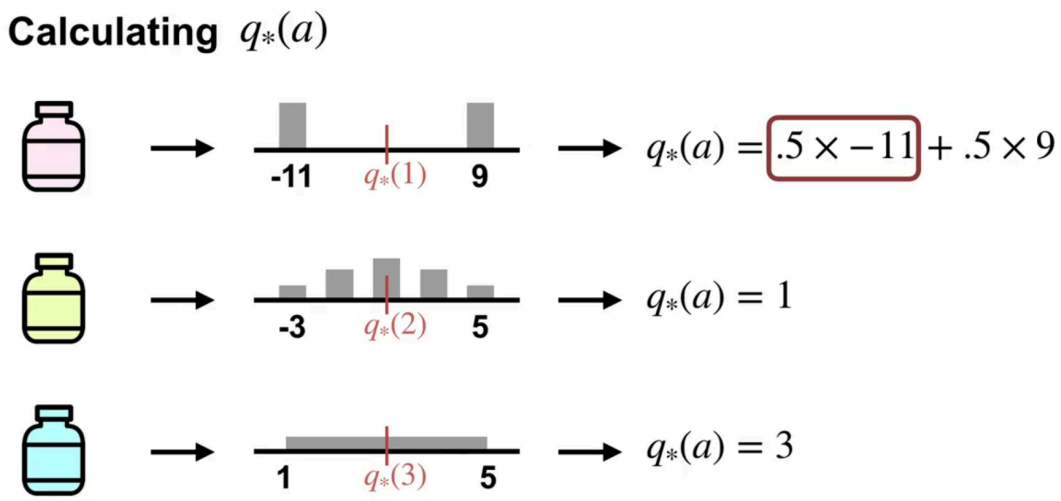
- Giá trị của hành động (q_*) là giá trị kỳ vọng của tất cả các giá trị khả thi khi thực hiện hành động *a*

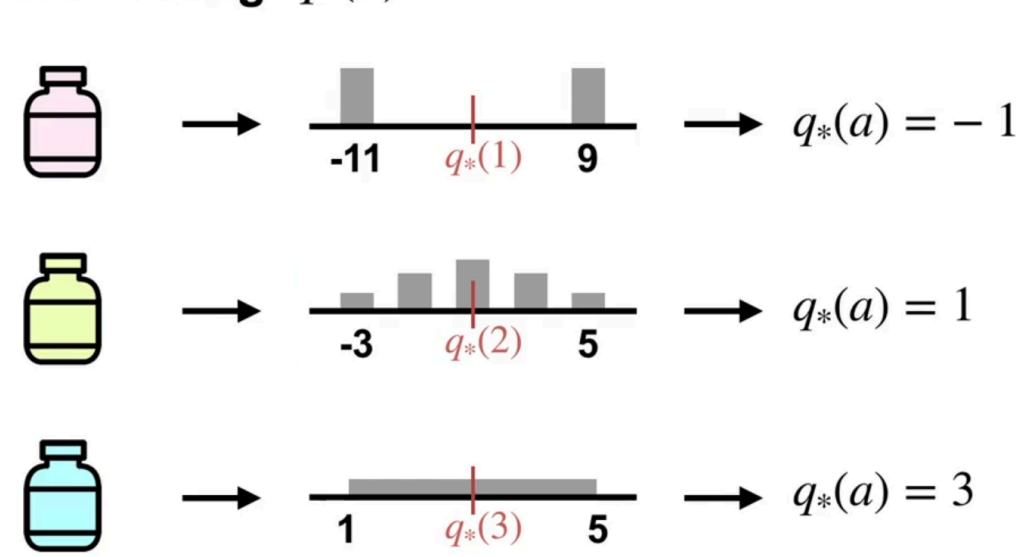
$$q_*(a) \doteq \mathrm{E}[R_t \mid A_t = a] \quad \forall a \in \{1, \dots, k\}$$
$$= \sum_r p(r \mid a)\, r$$

  Giá trị của hành động q là số chưa biết -> cần được ước tính! q_*: giá trị kỳ vọng thực sự q: giá trị kỳ vọng ước tính

- Mục tiêu là chọn hành động *a* để tối đa hóa phần thưởng/giá trị kỳ vọng của hành động

$$\arg\max_a q_*(a)$$





How is the bandit problem similar to or different from the supervised learning problem?

Vietnamese

Giống: cả 2 đều có mục tiêu đạt được kết quả tối ưu được đo lường bởi 1 hàm số (supervised: loss function, bandit problem: q*/reward function), supervised được train trên 1 tập data hữu hạn và cố định, bandit problem thì có số lượng action là 1 tập hữu hạn các action (K).

Khác: supervised learning tối đa hóa hàm mất mát trên 1 tập data cố định, ko thay đổi, label là cố định; bandit problem thì có label là giá trị kỳ vọng

trên 1 phân phối xác suất.

English

Similarities

1. **Optimization Objective**
    - Both aim to optimize a measurable function:
        - *Supervised Learning*: Minimizes a **loss function** (e.g., cross-entropy, MSE).
        - *Bandit Problems*: Maximizes a **reward function** (e.g., Q*-value, expected reward).
2. **Finite Action Space**
    - Supervised learning uses a fixed, finite dataset.
    - Bandit problems assume a finite set of **K actions** (e.g., choosing between K ad variants).

Key Differences

Aspect	Supervised Learning	Bandit Problems
Data Dynamics	Static dataset with fixed labels	Dynamic, stochastic rewards from a distribution
Feedback Type	Full feedback (labels for all inputs)	Partial feedback (reward only for chosen action)
Exploration Strategy	No exploration needed (deterministic training)	Requires **exploration-exploitation trade-off** (e.g., ε-greedy, UCB)
Objective	Generalize to unseen data	Maximize cumulative reward over interactions

Video: Learning Action Values

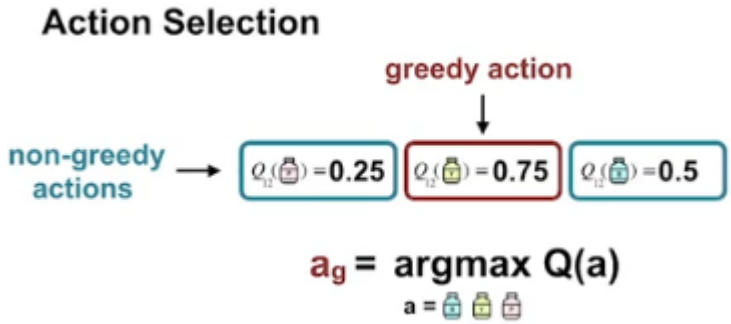
Sample-Average Method

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

$$= \frac{\sum_{i=1}^{t-1} R_i}{t-1}$$

Greedy action

Method of choosing action: choosing the **greedy action** a.k.a the action currently has the largest estimated value



Video: Estimating Action Values Incrementally

Incremental update rule

Recall

$$Q_n = \frac{1}{n-1}\sum_{i=1}^{n-1} R_i$$

Incremental update rule

$$Q_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i$$

$$= \frac{1}{n}(R_n + \sum_{i=1}^{n-1} R_i)$$

$$= \frac{1}{n}(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i)$$

$$= \frac{1}{n}(R_n + (n-1)Q_n)$$

$$= \frac{1}{n}(R_n + nQ_n - Q_n)$$

$$= Q_n + \frac{1}{n}(R_n - Q_n)$$

$$NewEstimate \leftarrow OldEstimate + StepSize\,[Target - OldEstimate]$$

$$Q_{n+1} = Q_n + \alpha_n (R_n - Q_n)$$
$$\alpha_n \rightarrow [0, 1]$$

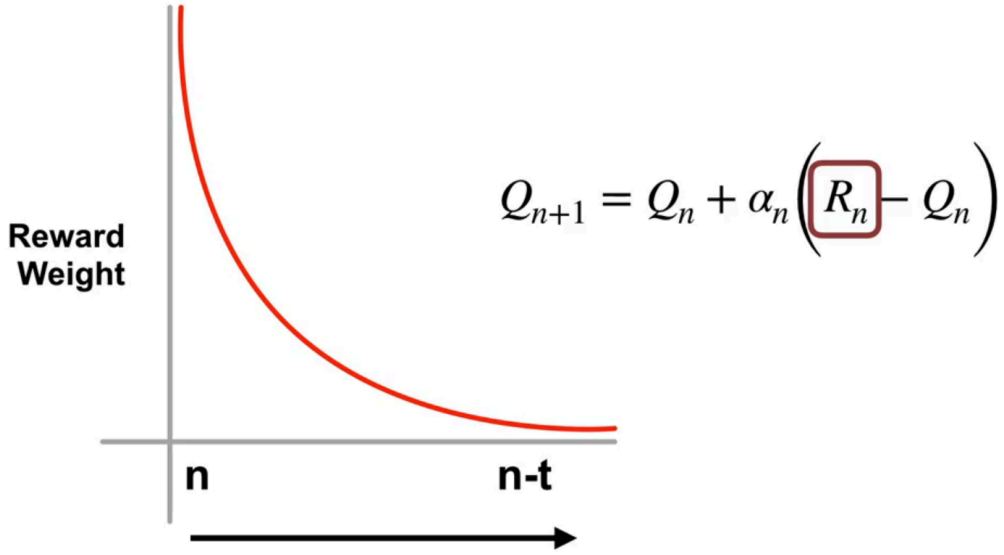
Sample average method

$$\alpha_n = \frac{1}{n}$$

α (step size), là 1 hằng số

Non-stationary bandit problem (rewad ko cố định và có thể thay đổi theo thời gian)

Trong các bài toán non-stationary (có tính thay đổi theo thời gian, học trực tuyến a.k.a online learning), stepsize là 1 hằng số cố định sẽ giúp estimate và điều chỉnh q tốt và nhanh hơn là 1 step size giảm dần theo thời gian (decaying: $\frac{1}{n}$)



the most recent rewards affect the estimate more than older rewards (các reward mới nhất ảnh hưởng đến giá trị ước lượng hơn các reward ở các bước xa hơn)

Decaying past rewards

$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha_n (R_n - Q_n) \\
&= \alpha_n R_n + Q_n - \alpha_n Q_n \\
&= \alpha_n R_n + (1 - \alpha_n) Q_n \\
&= \alpha_n R_n + (1 - \alpha_n) \left[\alpha_n R_{n-1} + (1 - \alpha_n) Q_{n-1}\right] \\
&= \alpha_n R_n + (1 - \alpha_n) \alpha_n R_{n-1} + (1 - \alpha_n)^2 Q_{n-1} \\
&= \alpha_n R_n + (1 - \alpha_n) \alpha_n R_{n-1} + (1 - \alpha_n)^2 \alpha_n R_{n-2} + \cdots \\
&\quad + (1 - \alpha_n)^{n-1} \alpha_n R_1 + (1 - \alpha_n)^n Q_1 \\
&= (1 - \alpha)^n Q_1 + \sum_{i=1}^{n} \alpha (1 - \alpha)^{n-i} R_i
\end{aligned}
$$

$$
Q_1 \rightarrow \textbf{initial action-value}
$$

Đóng góp của Q_1 với giá trị ước tính tại bước $n + 1$ giảm dần theo cấp lũy thừa theo thời gian, các giá trị reward ở các bước cũ đóng góp theo cấp lũy thừa ít hơn. Sự ảnh hưởng của giá trị khởi tạo (Q_1) tiến gần về 0 khi càng có thêm nhiều data, các giá trị mới nhất quyết định giá trị ước tính hiện tại (Q_{n+1})

Exploration vs. Exploitation Tradeoff

Video: What is the trade-off?

Exploration and Exploitation

- **Exploration** - **improve** knowledge for **long-term** benefit
- **Exploitation** - **exploit** knowledge for **short-term** benefit (being greedy w.r.t estimated values, may not actually get the most reward)
- Round Robin fashion: tuần tự theo chu kỳ

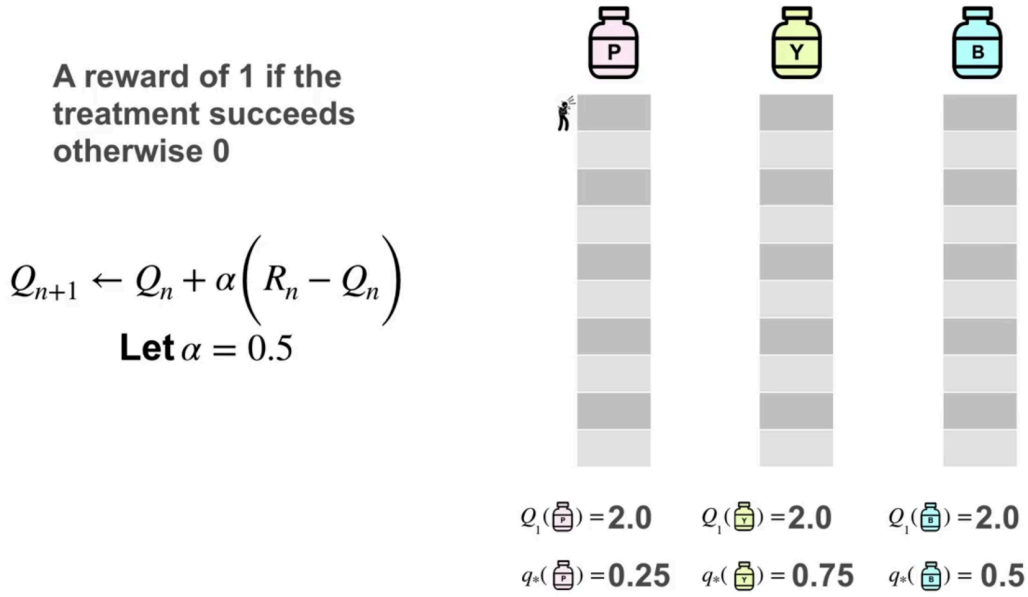
Các phương pháp để cân bằng giữa Exploration và Exploitation

Epsilon-Greedy Action Selection

$$A_t \leftarrow \begin{cases} \arg\max_a Q_t(a) & \text{with probability } 1-\epsilon \\ a \sim \text{Uniform}(\{a_1, \ldots, a_k\}) & \text{with probability } \epsilon \end{cases}$$

Lựa chọn khám phá (explore) ngẫu nhiên 1 hành động với xác suất ϵ từ xác suất phân phối đều, và lựa chọn greedy hành động có ước tính phần thưởng Q_t lớn nhất trong số các hành động (exploit)

Video: Optimistic Initial Values

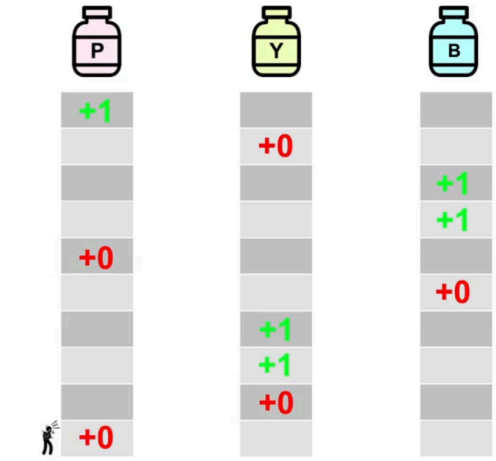


Khởi tạo giá trị kỳ vọng ước lượng khởi đầu cao và thực hiện chiến thuật lựa chọn tham lam (greedy selection) giúp agent có thể explore các lựa chọn khác nhau ở các timestep đầu tiên và update dần về giá trị hành động thực tế

A reward of 1 if the treatment succeeds otherwise 0

$$Q_{n+1} \leftarrow Q_n + \alpha\left(R_n - Q_n\right)$$

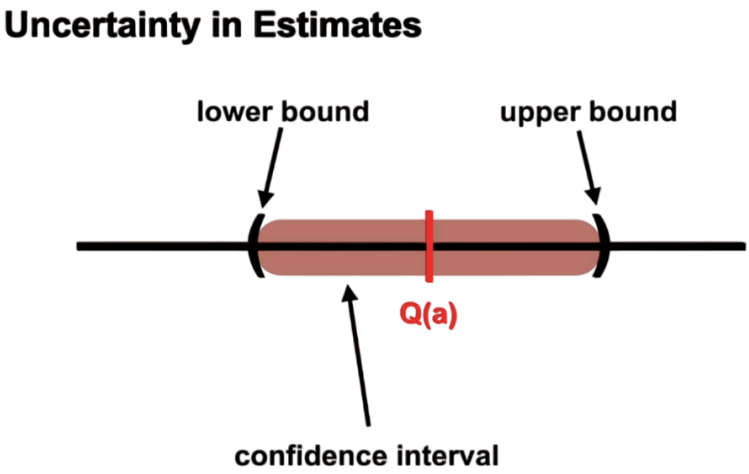
Let $\alpha = 0.5$



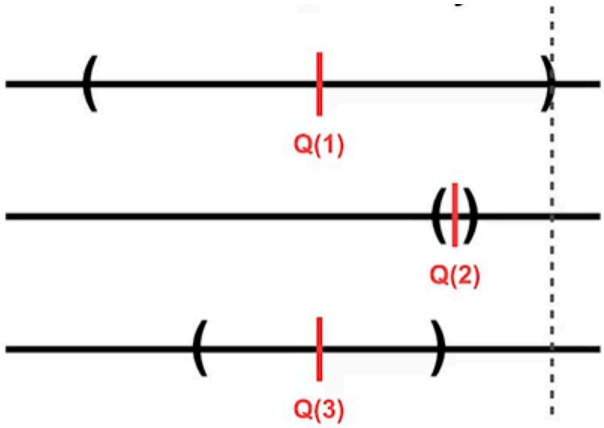
Giới hạn:

- Chỉ explore ở các bước đầu tiên, sau khi đến bước nào đó sẽ dừng explore
- Ko phù hợp với các bài toán có reward thay đổi theo thời gian (non-stationary problems)
- Không thể biết được giá trị khởi đầu lạc quan nên để là bao nhiêu (vì không biết giá trị tối đa của reward)

Video: Upper-Confidence Bound (UCB) Action Selection



Optimism in the Face of Uncertainty



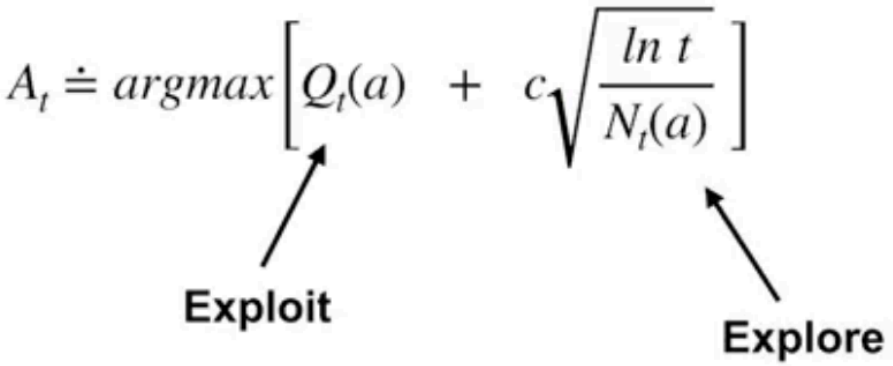
Upper-Confidence Bound (UCB) Action Selection

chọn action có cận trên của giá trị hành động là cao nhất□□□

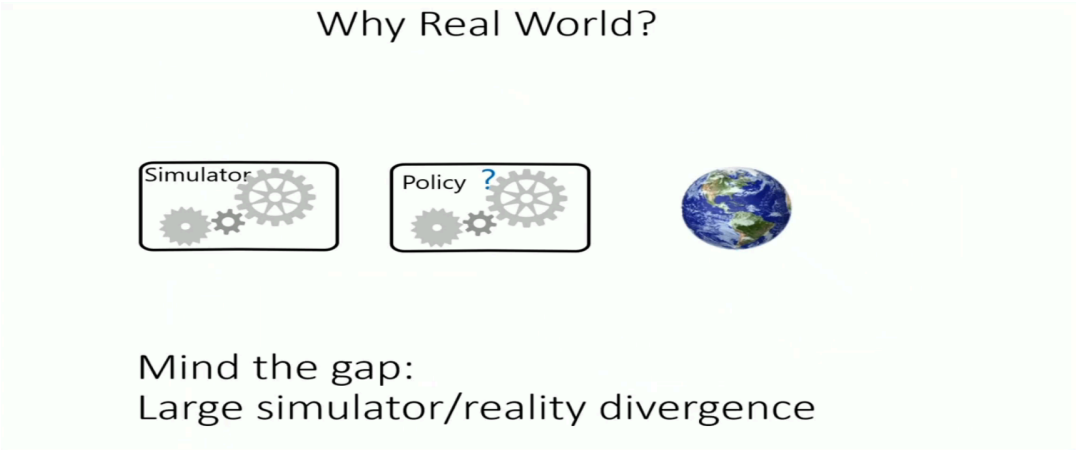
$$A_t \doteq \arg\max_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

$c\sqrt{\frac{\ln t}{N_t(a)}}$ is upper-confidence bound (UCB) exploration term

c is user-specified parameter that controls the amount of exploration

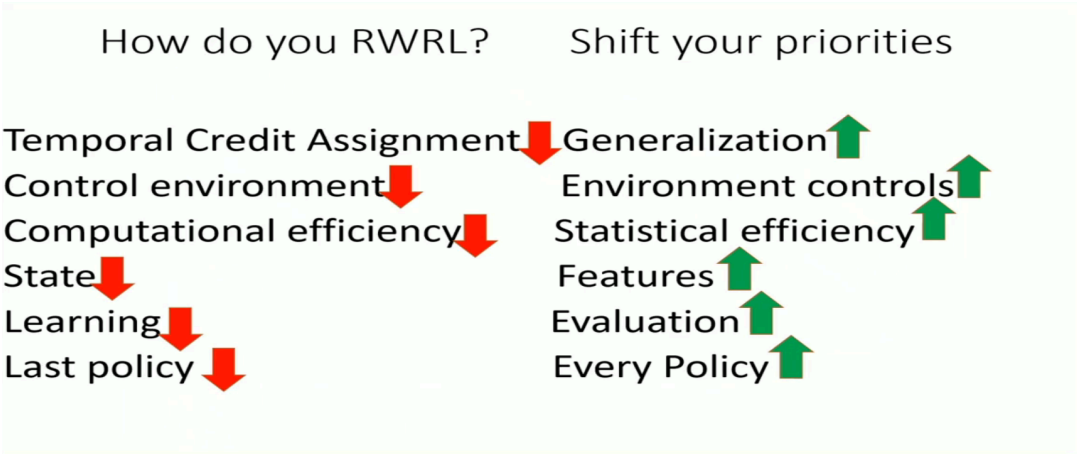


Video: Jonathan Langford: Contextual Bandits for Real World Reinforcement Learning

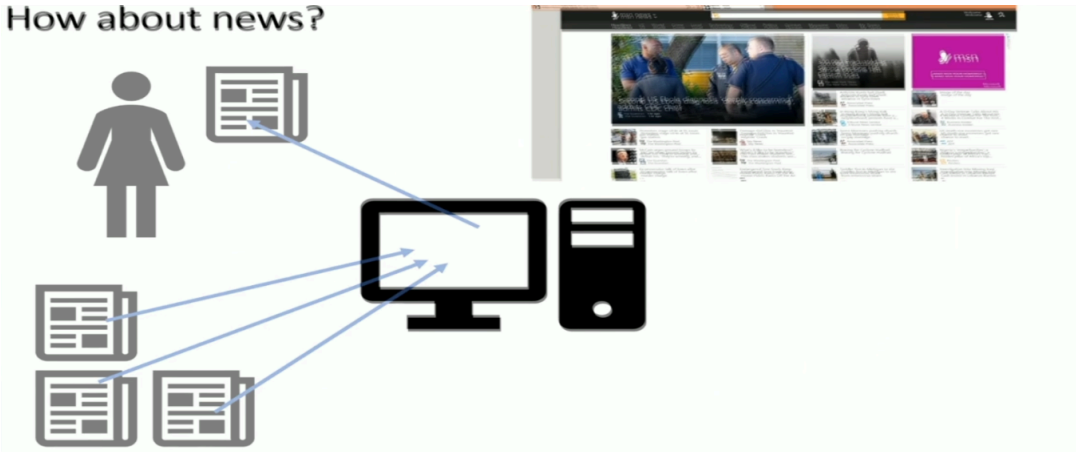


There's is a gap between the simulator and the reality.

Real World Reinforcement Learning



An example: Contextual Bandits



Having a set of possible news articles in interest today, suggest news article for new user come to website that they maybe have interest in, get

feedback about whether or not they actually read the article.

Repeatedly:

1. Observe features x (user geolocation, past profile behaviour, features of available actions,...)
2. Choose action $a \in A$
3. Observe reward r

Goal: Maximize reward

Major caveat: No credit assignment (for selected action, what's the reward if we had chosen a different action?)

Video: Week 1 Summary

Exploration vs Exploitation strategies

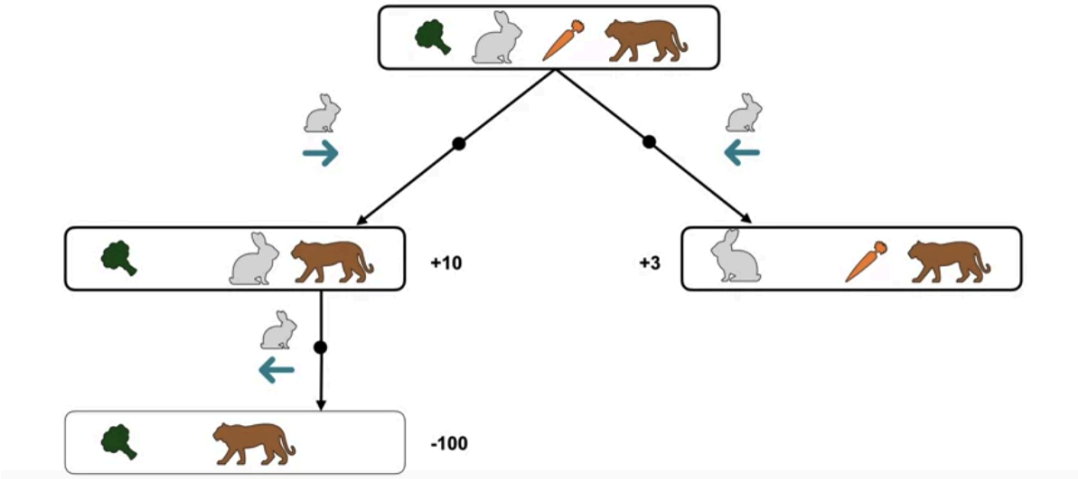
- ε-Greedy Action Selection
- Optimistic Initial Values
- Upper-Confidence Bound (UCB) Action Selection

Module 3

Introduction to Markov Decision Processes

Video: Markov Decision Processes

- Các tình huống khác nhau sẽ xuất hiện các lựa chọn hành động khác nhau
- Hành động dẫn đến trạng thái của thế giới thay đổi action -> state



Markov Decision Processes (MDPs) là các khung toán học được sử dụng để mô hình hóa các vấn đề ra quyết định, trong đó một tác nhân tương tác với một môi trường trong một chuỗi các bước thời gian riêng biệt (discrete time steps).

Markov Decision Processes terms:

- **Agent**: là thực thể mà chúng ta đang huấn luyện để đưa ra các quyết định chính xác.
- **Environment**: môi trường xung quanh mà agent tương tác.
- **State**: (thông tin) trạng thái hiện tại của agent trong môi trường để đưa ra quyết định.
- **Action**: lựa chọn mà agent thực hiện tại mỗi bước thời gian hiện tại.
- **Policy**: Policy là quá trình suy nghĩ hoặc chiến lược đứng sau việc lựa chọn một action.
- **P (Transition Probability):** `P(s'|s, a)` cho biết xác suất chuyển sang trạng thái `s'` khi thực hiện hành động `a` tại trạng thái `s`. Điều này thể hiện tính ngẫu nhiên của môi trường.
- **R (Reward Function):** `R(s, a, s')` xác định phần thưởng nhận được ngay lập tức sau khi chuyển từ trạng thái `s` sang trạng thái `s'` do thực hiện hành động `a`.

Sometimes:

- **γ (Discount Factor):** Một giá trị trong khoảng từ 0 đến 1, quyết định mức độ quan trọng của các phần thưởng trong tương lai.

How MDPs Work

Ở mỗi bước thời gian:

- Agent quan sát trạng thái hiện tại.
- Chọn một action dựa trên policy (một ánh xạ từ state đến action).
- Environment chuyển sang một state mới dựa trên xác suất chuyển tiếp (transition probabilities).
- Agent nhận được reward tương ứng với action đã thực hiện.

Mục tiêu là tìm ra một **chính sách tối ưu** để tối đa hóa tổng phần thưởng kỳ vọng (expected cumulative reward) theo thời gian.

The dynamics of an MDP (Markov property)

Khi agent thực hiện hành động `a` trong trạng thái `s`, sẽ có nhiều trạng thái kế tiếp s' và phần thưởng r có thể xảy ra. Hàm động lực chuyển tiếp $P(s', r|s, a)$ mô tả xác suất của từng kết quả (s', r).

Công thức:

$$P(s', r|s, a) = \mathrm{P}(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a)$$

$p(s', r|s, a)$ là xác suất xảy ra trạng thái s' và nhận được phần thưởng r với hành động a từ trạng thái s

Giải thích:

- Hàm này định nghĩa "luật chơi" của môi trường:
  - **Đầu vào:** Trạng thái hiện tại `s` + hành động `a`.
  - **Đầu ra:** Phân phối xác suất cho tất cả cặp `(s', r)` có thể.
- Ví dụ: Robot di chuyển nhưng có 10% khả năng trượt (sang trạng thái khác ngoài dự định).

Tính chất:

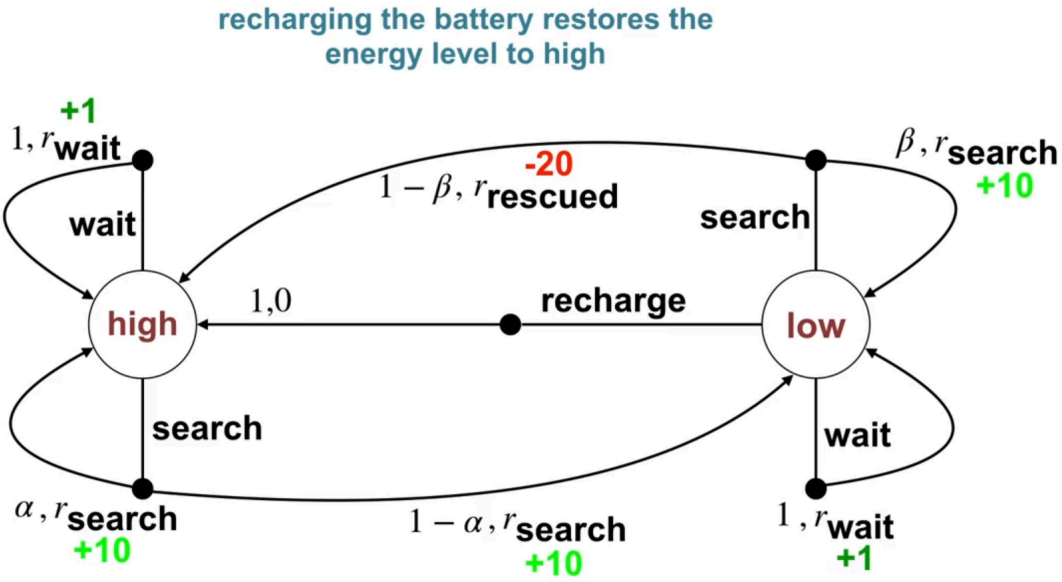
- $\sum_{s',r} P(s', r \mid s, a) = 1$ (tổng xác suất bằng 1).

$$p : \mathrm{S} \times \mathrm{R} \times \mathrm{S} \times \mathrm{A} \to [0, 1]$$
$$\sum_{s' \in \mathrm{S}} \sum_{r \in \mathrm{R}} p(s', r \mid s, a) = 1, \forall s \in \mathrm{S},\ a \in \mathrm{A}(s)$$

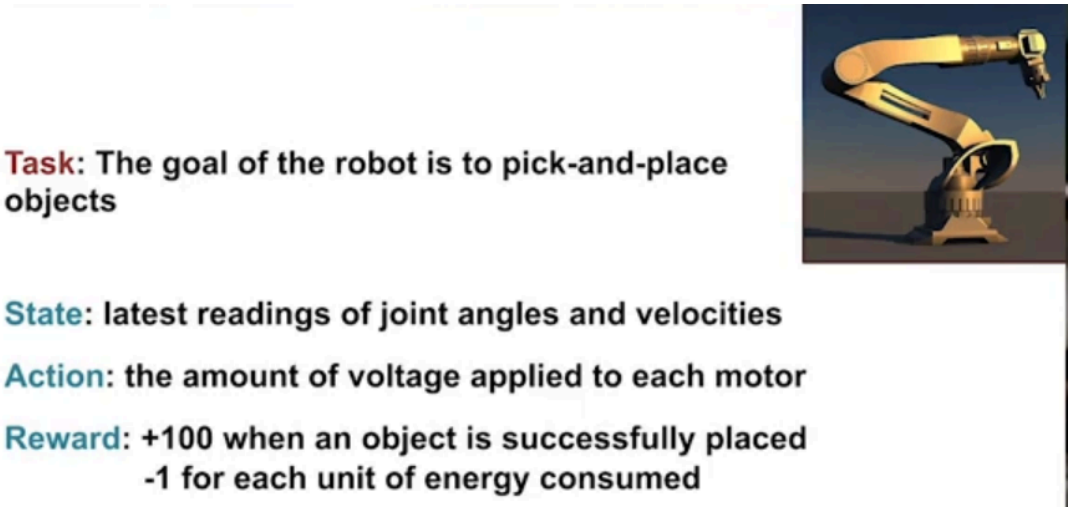
- Môi trường **deterministic** là trường hợp đặc biệt: P(s', r | s, a) = 1 cho một cặp `(s', r)` duy nhất.
- Tính chất Markov: trạng thái tương lai chỉ phụ thuộc vào trạng thái và hành động hiện tại, không phải vào chuỗi các sự kiện trước nó (ghi nhớ về các trạng thái trước đó không giúp cải thiện cho dự đoán về tương lai)

Video: Examples of MDPs

Dynamics of the Recycling Robot



state, action có thể đơn giản, chi tiết (low-level) hoặc abstract (high-level)



Goal of Reinforcement Learning

Video: The Goal of Reinforcement Learning

Goal of an Agent : Formal definition

Mục tiêu của agent là tối đa hóa phần thưởng nhận được trong tương lai

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \dots$$

G_t là 1 biến ngẫu nhiên

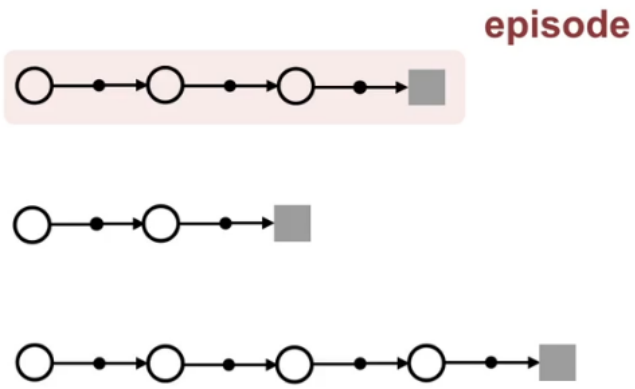
$$\mathrm{E}[G_t] = \mathrm{E}[\, R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T \,]$$

maximize the expected return

Tối đa hóa giá trị mong đợi cho phần thưởng nhận được trong tương lai (tổng các phần thưởng phải là 1 số hữu hạn)

T là thời điểm kết thúc (final timestep)

Epsodic Tasks (tập hành động)



Các tương tác được chia thành các đoạn nhỏ gọi là các episode, mỗi episode bắt đầu độc lập với kết thúc của episode trước đó, khi 1 episode kết thúc, agent được đặt lại về trạng thái bắt đầu, mỗi episode sẽ có 1 trạng thái kết thúc.

Ví dụ:

- Chơi một ván cờ (mỗi ván là một episode) -> trạng thái kết thúc: thắng, hòa, đầu hàng
- Một lần robot di chuyển từ vị trí xuất phát đến đích

Discussion Prompt: Is the reward hypothesis sufficient?

- Multi-objective tasks: Can’t balance goals like speed and safety in self-driving cars.
- Risk-sensitive tasks: Ignores risk, e.g., investors want to avoid rare catastrophic losses, not just maximize average return.
- Modal/counterfactual tasks: Can’t reason about what could have happened, e.g., a robot must maintain the ability to avoid collisions.
- Dynamic/subjective preferences: Can’t adapt to changing or conflicting values, e.g., recommenders may need to shift from engagement to well-being.
- Non-Markovian dependencies: Can’t encode history-based requirements, e.g., loan systems must ensure fairness over time.

Video: Michael Littman: The Reward Hypothesis

Whence Behavior?

- give a man a fish and he'll eat for a day
    - Programming (GOFAI - Good-old fashioned AI)
- teach a man to fish and he'll eat for a lifetime
    - Examples (LfD)
- give a man a taste for fish and he'll figure out how to get fish, even if the details change!
    - Optimization (RL)

Goals as Rewards

- 1 for goal, 0 otherwise
    - goal-reward representation
- -1 for not goal, 0 once goal reached
    - action-penalty representation