



UPPSALA
UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

Machine learning – Block 8

Måns Magnusson
Department of Statistics, Uppsala University

Autumn 2022



UPPSALA
UNIVERSITET

This week's lectures

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- Introduction to Reinforcement learning



UPPSALA
UNIVERSITET

Introduction to Reinforcement Learning

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- Another type of Machine Learning:
 - Supervised Learning
 - Unsupervised Learning
 - Reinforcement learning



UPPSALA
UNIVERSITET

Introduction to Reinforcement Learning

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- Another type of Machine Learning:
 - Supervised Learning
 - Unsupervised Learning
 - Reinforcement learning
- Computational approach of learning from interaction
- Closest to human and animal learning: trial, error, and planning.
- The learner is *not* told which actions to take



UPPSALA
UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

Introduction to Reinforcement Learning

- Another type of Machine Learning:
 - Supervised Learning
 - Unsupervised Learning
 - Reinforcement learning
- Computational approach of learning from interaction
- Closest to human and animal learning: trial, error, and planning.
- The learner is *not* told which actions to take
- Connections to:
 - Game Theory
 - Control Theory
 - Multi-agent systems
 - Swarm intelligence
 - Information theory
 - Statistics



UPPSALA
UNIVERSITET

Introduction to Reinforcement Learning

- Introduction to Reinforcement Learning
 - Bandits
 - Markov Decision Processes
- **Goal:** maximize return over a sequence of actions



UPPSALA
UNIVERSITET

Introduction to Reinforcement Learning

- Introduction to Reinforcement Learning
 - Bandits
 - Markov Decision Processes
- **Goal:** maximize return over a sequence of actions
 - Three characteristics:
 1. Closed-loop: early actions affects later actions



UPPSALA
UNIVERSITET

Introduction to Reinforcement Learning

- Introduction to Reinforcement Learning
 - Bandits
 - Markov Decision Processes
- **Goal:** maximize return over a sequence of actions
 - Three characteristics:
 1. Closed-loop: early actions affects later actions
 2. No direct instructions



UPPSALA
UNIVERSITET

Introduction to Reinforcement Learning

- Introduction to Reinforcement Learning
 - Bandits
 - Markov Decision Processes
- **Goal:** maximize return over a sequence of actions
 - Three characteristics:
 1. Closed-loop: early actions affects later actions
 2. No direct instructions
 3. Reward signals over a long period of time



UPPSALA
UNIVERSITET

Recent Achievements

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- RL agent won over Lee Sedol in 2016: **AlphaGo**



UPPSALA
UNIVERSITET

Recent Achievements

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- RL agent won over Lee Sedol in 2016: **AlphaGo**
- Won over human champions in **DOTA2** in 2019



UPPSALA
UNIVERSITET

Recent Achievements

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- RL agent won over Lee Sedol in 2016: **AlphaGo**
- Won over human champions in **DOTA2** in 2019
- **ChatGPT**/InstructGPT in 2022



UPPSALA
UNIVERSITET

Recent Achievements

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- RL agent won over Lee Sedol in 2016: **AlphaGo**
- Won over human champions in **DOTA2** in 2019
- **ChatGPT**/InstructGPT in 2022
- Industry automation: RL is used to reduce the energy cost of datacenter cooling



UPPSALA
UNIVERSITET

Recent Achievements

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- RL agent won over Lee Sedol in 2016: **AlphaGo**
- Won over human champions in **DOTA2** in 2019
- **ChatGPT**/InstructGPT in 2022
- Industry automation: RL is used to reduce the energy cost of datacenter cooling
- Automated trading



UPPSALA
UNIVERSITET

Recent Achievements

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- RL agent won over Lee Sedol in 2016: **AlphaGo**
- Won over human champions in **DOTA2** in 2019
- **ChatGPT**/InstructGPT in 2022
- Industry automation: RL is used to reduce the energy cost of datacenter cooling
- Automated trading
- Elevator scheduling



UPPSALA
UNIVERSITET

Recent Achievements

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- RL agent won over Lee Sedol in 2016: **AlphaGo**
- Won over human champions in **DOTA2** in 2019
- **ChatGPT**/InstructGPT in 2022
- Industry automation: RL is used to reduce the energy cost of datacenter cooling
- Automated trading
- Elevator scheduling
- A/B testing and personalized recommendations



UPPSALA
UNIVERSITET

Recent Achievements

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- RL agent won over Lee Sedol in 2016: **AlphaGo**
- Won over human champions in **DOTA2** in 2019
- **ChatGPT**/InstructGPT in 2022
- Industry automation: RL is used to reduce the energy cost of datacenter cooling
- Automated trading
- Elevator scheduling
- A/B testing and personalized recommendations
- Board games such as backgammon, chess and checkers



UPPSALA
UNIVERSITET

The different parts in RL

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

1. The **Agent**: The learning agent.



UPPSALA
UNIVERSITET

The different parts in RL

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

1. The **Agent**: The learning agent.
2. The **Environment**: Where the agent performs actions.



UPPSALA
UNIVERSITET

The different parts in RL

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

1. The **Agent**: The learning agent.
2. The **Environment**: Where the agent performs actions.
3. **Actions**: Made by the agent and affects the environment.



UPPSALA
UNIVERSITET

The different parts in RL

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

1. The **Agent**: The learning agent.
2. The **Environment**: Where the agent performs actions.
3. **Actions**: Made by the agent and affects the environment.
4. **Reward**: The evaluation of an action. A scalar value.
Pleasure and pain.



- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

1. The **Agent**: The learning agent.
2. The **Environment**: Where the agent performs actions.
3. **Actions**: Made by the agent and affects the environment.
4. **Reward**: The evaluation of an action. A scalar value.
Pleasure and pain.
5. **Return**: The aggregated reward over a long period.



UPPSALA
UNIVERSITET

The different parts in RL

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

1. Agents:

- 1.1 Have a **goal** (maximize return)
- 1.2 **Sense** aspect of their environment
- 1.3 Choose **actions**
- 1.4 Possibility to **improve performance over time**



- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

1. Agents:

- 1.1 Have a **goal** (maximize return)
- 1.2 **Sense** aspect of their environment
- 1.3 Choose **actions**
- 1.4 Possibility to **improve performance over time**

2. Usually an **uncertainty** about the environment

3. Represent uncertainty of environment: **Probability**



UPPSALA
UNIVERSITET

Sub-elements of agents

1. **Policy**: How the agent choose actions. Determines behaviour.
2. **Model**: The agent's model of the environment. Used for planning

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes



UPPSALA
UNIVERSITET

Sub-elements of agents

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

1. **Policy**: How the agent choose actions. Determines behaviour.
2. **Model**: The agent's model of the environment. Used for planning
3. **Value function**: The long-term value (the expected long-term return following a policy)



Sub-elements of agents

1. **Policy**: How the agent choose actions. Determines behaviour.
 2. **Model**: The agent's model of the environment. Used for **planning**
 3. **Value function**: The long-term value (the expected long-term return following a policy)
- Outside agent: **Reward signal**: The instant value of an action



Sub-elements of agents

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

1. **Policy**: How the agent choose actions. Determines behaviour.
 2. **Model**: The agent's model of the environment. Used for planning
 3. **Value function**: The long-term value (the expected long-term return following a policy)
- Outside agent: **Reward signal**: The instant value of an action
 - Problem: **Balance** the trade-off between long-term and short-term rewards



UPPSALA
UNIVERSITET

Supervised, Unsupervised and Reinforcement Learning

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

1. Static vs. Dynamic



UPPSALA
UNIVERSITET

Supervised, Unsupervised and Reinforcement Learning

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

1. Static vs. Dynamic
2. No Gold Standard



UPPSALA
UNIVERSITET

Supervised, Unsupervised and Reinforcement Learning

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

1. Static vs. Dynamic
2. No Gold Standard
3. Multiple-Decision Process: Return vs. reward



UPPSALA
UNIVERSITET

Supervised, Unsupervised and Reinforcement Learning

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

1. Static vs. Dynamic
2. No Gold Standard
3. Multiple-Decision Process: Return vs. reward
4. Need for exploration



UPPSALA
UNIVERSITET

Supervised, Unsupervised and Reinforcement Learning

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

1. Static vs. Dynamic
2. No Gold Standard
3. Multiple-Decision Process: Return vs. reward
4. Need for exploration
5. Evaluates actions - not only instruct actions



UPPSALA
UNIVERSITET

Exploration vs Exploitation

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- **Goal:** Maximize the return (the total reward), i.e.



UPPSALA
UNIVERSITET

Exploration vs Exploitation

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- **Goal**: Maximize the return (the total reward), i.e.
- **Exploit** the best actions



UPPSALA
UNIVERSITET

Exploration vs Exploitation

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- **Goal**: Maximize the return (the total reward), i.e.
- **Exploit** the best actions
- **Explore** to know the best actions



UPPSALA
UNIVERSITET

Evolution vs Learning

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- Set a policy without learning: **Evolutionary** Methods
- Good when agent cannot sense the environment



UPPSALA
UNIVERSITET

Evolution vs Learning

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- Set a policy without learning: **Evolutionary** Methods
- Good when agent cannot sense the environment
- **Example:** Bacteria don't learn, they evolve



UPPSALA
UNIVERSITET

Setting the goal for the Agent

- Introduction to Reinforcement Learning
 - Bandits
 - Markov Decision Processes
- Setting the goal: **defining the reward** signal (reward function)
 - **Example:** If you want the agent to do something quick, give -1 per action.



Setting the goal for the Agent

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- Setting the goal: **defining the reward** signal (reward function)
- **Example:** If you want the agent to do something quick, give -1 per action.
- We should give rewards for correct **behaviour**
- Do **not** use reward to guide **how** to reach the goal
- **Be careful what you wish for...**



UPPSALA
UNIVERSITET

- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

Section 2

Bandits



UPPSALA
UNIVERSITET

The k -armed bandit problem

- **Goal:** Maximize the total or average reward after N actions

- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes



- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

The k -armed bandit problem

- **Goal:** Maximize the total or average reward after N actions
- **The actions:** Choose between k arms, i.e. $A_t \in \{1, \dots, k\}$



- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

The k -armed bandit problem

- **Goal:** Maximize the total or average reward after N actions
- **The actions:** Choose between k arms, i.e. $A_t \in \{1, \dots, k\}$
- The reward signal:

$$R_t \sim p(R_t|a),$$

where $\mathbb{E}(R_t|A_t = a) = q^*(a)$.



- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

The k -armed bandit problem

- **Goal:** Maximize the total or average reward after N actions
- **The actions:** Choose between k arms, i.e. $A_t \in \{1, \dots, k\}$
- The reward signal:

$$R_t \sim p(R_t|a),$$

where $\mathbb{E}(R_t|A_t = a) = q^*(a)$.

- $q^*(a)$ is **unknown**.



- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

The k -armed bandit problem

- **Goal:** Maximize the total or average reward after N actions
- **The actions:** Choose between k arms, i.e. $A_t \in \{1, \dots, k\}$
- The reward signal:

$$R_t \sim p(R_t|a),$$

where $\mathbb{E}(R_t|A_t = a) = q^*(a)$.

- $q^*(a)$ is **unknown**.
- The estimated (expected) value if action a at step t : $Q_t(a)$.



The k -armed bandit problem

- **Goal:** Maximize the total or average reward after N actions
- **The actions:** Choose between k arms, i.e. $A_t \in \{1, \dots, k\}$
- The reward signal:

$$R_t \sim p(R_t|a),$$

where $\mathbb{E}(R_t|A_t = a) = q^*(a)$.

- $q^*(a)$ is **unknown**.
- The estimated (expected) value if action a at step t : $Q_t(a)$.
- This is a **tabular** method/problem:
We can represent the actions in a table.
- Tabular methods works in small problems
e.g. A/B testing and dynamic web pages.



Exploration vs. Exploitation

- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

- Two types of actions:
 1. Exploitation: Choose the action with highest expected reward (short term)
 2. Exploration: Choose action to improve $Q_t(a)$, but reduces the reward (long term)
- The **conflict** between exploration and exploitation



UPPSALA
UNIVERSITET

- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

ϵ -greedy

- ϵ -greedy: $P(\text{exploration}) = \epsilon$



- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

ϵ -greedy

- ϵ -greedy: $P(\text{exploration}) = \epsilon$
 - Exploitation:

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



- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

ϵ -greedy

- ϵ -greedy: $P(\text{exploration}) = \epsilon$
 - Exploitation:

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

- Exploration:

$$A_t \sim U(1, \dots, k)$$



ϵ -greedy

- ϵ -greedy: $P(\text{exploration}) = \epsilon$

- Exploitation:

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

- Exploration:

$$A_t \sim U(1, \dots, k)$$

- $Q_1(a) = 0$ (or used to encourage initial exploration)



ϵ -greedy

- ϵ -greedy: $P(\text{exploration}) = \epsilon$

- Exploitation:

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

- Exploration:

$$A_t \sim U(1, \dots, k)$$

- $Q_1(a) = 0$ (or used to encourage initial exploration)
 - For any $\epsilon > 0$, $Q_t(a) \rightarrow q^*(a)$



ϵ -greedy

- ϵ -greedy: $P(\text{exploration}) = \epsilon$

- Exploitation:

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

- Exploration:

$$A_t \sim U(1, \dots, k)$$

- $Q_1(a) = 0$ (or used to encourage initial exploration)
- For any $\epsilon > 0$, $Q_t(a) \rightarrow q^*(a)$
- We estimate $q^*(a)$ using $Q_t(a)$ as

$$Q_T(a) = \frac{1}{N(a)} \sum_t^{T-1} R_{t, A_t=a},$$

where $N(a)$ is the total number of times action a has been taken.



ϵ -greedy

- ϵ -greedy: $P(\text{exploration}) = \epsilon$

- Exploitation:

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

- Exploration:

$$A_t \sim U(1, \dots, k)$$

- $Q_1(a) = 0$ (or used to encourage initial exploration)
- For any $\epsilon > 0$, $Q_t(a) \rightarrow q^*(a)$
- We estimate $q^*(a)$ using $Q_t(a)$ as

$$Q_T(a) = \frac{1}{N(a)} \sum_t^{T-1} R_{t, A_t=a},$$

where $N(a)$ is the total number of times action a has been taken.

- When should we explore?
 - Large $V(R_t)$



ϵ -greedy

- ϵ -greedy: $P(\text{exploration}) = \epsilon$

- Exploitation:

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

- Exploration:

$$A_t \sim U(1, \dots, k)$$

- $Q_1(a) = 0$ (or used to encourage initial exploration)
- For any $\epsilon > 0$, $Q_t(a) \rightarrow q^*(a)$
- We estimate $q^*(a)$ using $Q_t(a)$ as

$$Q_T(a) = \frac{1}{N(a)} \sum_t^{T-1} R_{t, A_t=a},$$

where $N(a)$ is the total number of times action a has been taken.

- When should we explore?
 - Large $V(R_t)$
 - Large \mathcal{A}



ϵ -greedy

- ϵ -greedy: $P(\text{exploration}) = \epsilon$

- Exploitation:

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

- Exploration:

$$A_t \sim U(1, \dots, k)$$

- $Q_1(a) = 0$ (or used to encourage initial exploration)
- For any $\epsilon > 0$, $Q_t(a) \rightarrow q^*(a)$
- We estimate $q^*(a)$ using $Q_t(a)$ as

$$Q_T(a) = \frac{1}{N(a)} \sum_t^{T-1} R_{t, A_t=a},$$

where $N(a)$ is the total number of times action a has been taken.

- When should we explore?
 - Large $V(R_t)$
 - Large \mathcal{A}
 - Non-stationarity



- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

Bandit example

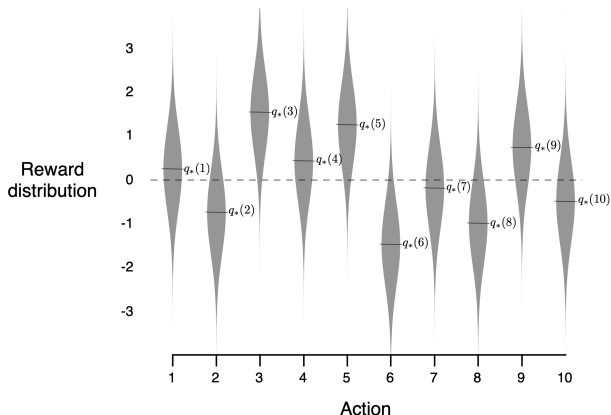


Figure: The 10-armed bandit environment (Sutton and Barto, 2017, Fig. 2.1)



UPPSALA
UNIVERSITET

- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

Bandit example

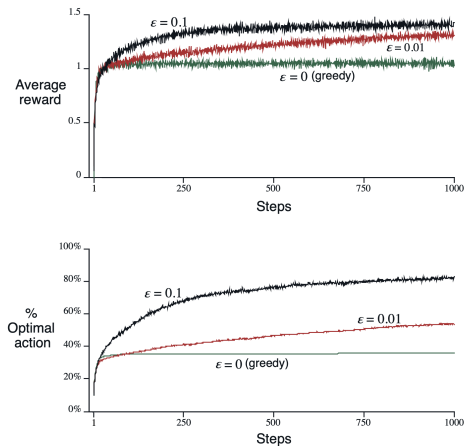


Figure: The ϵ -greedy algorithm result in the 10-armed bandit (Sutton and Barto, 2017, Fig. 2.2)



- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

Bandit example: Optimistic initialization

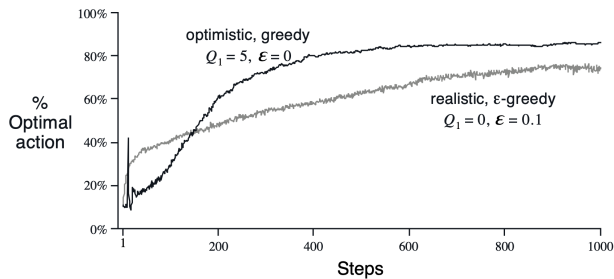


Figure: The ϵ -greedy algorithm and optimistic initialization (Sutton and Barto, 2017, Fig. 2.3)



- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

Efficient computation and non-stationarity

- Compute $Q_t(a)$ on the fly:

$$Q_T(a) = Q_{T-1} + \frac{1}{N_t(a)}(R_{t,A_t=a} - Q_{T-1}(a))$$



- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

Efficient computation and non-stationarity

- Compute $Q_t(a)$ on the fly:

$$Q_T(a) = Q_{T-1} + \frac{1}{N_t(a)}(R_{t,A_t=a} - Q_{T-1}(a))$$

- Handling non-stationarity:

$$Q_T(a) = Q_{T-1} + \alpha(t)(R_{t,A_t=a} - Q_{T-1}(a))$$



Efficient computation and non-stationarity

- Compute $Q_t(a)$ on the fly:

$$Q_T(a) = Q_{T-1} + \frac{1}{N_t(a)}(R_{t,A_t=a} - Q_{T-1}(a))$$

- Handling non-stationarity:

$$Q_T(a) = Q_{T-1} + \alpha(t)(R_{t,A_t=a} - Q_{T-1}(a))$$

- Examples:

- $\alpha(t) = 1$: $Q_T(a) = R_{t,A_t=a}$



Efficient computation and non-stationarity

- Compute $Q_t(a)$ on the fly:

$$Q_T(a) = Q_{T-1} + \frac{1}{N_t(a)}(R_{t,A_t=a} - Q_{T-1}(a))$$

- Handling non-stationarity:

$$Q_T(a) = Q_{T-1} + \alpha(t)(R_{t,A_t=a} - Q_{T-1}(a))$$

- Examples:

- $\alpha(t) = 1$: $Q_T(a) = R_{t,A_t=a}$
- $\alpha(t) = 0$: $Q_T(a) = Q_1(a)$



Efficient computation and non-stationarity

- Compute $Q_t(a)$ on the fly:

$$Q_T(a) = Q_{T-1} + \frac{1}{N_t(a)}(R_{t,A_t=a} - Q_{T-1}(a))$$

- Handling non-stationarity:

$$Q_T(a) = Q_{T-1} + \alpha(t)(R_{t,A_t=a} - Q_{T-1}(a))$$

- Examples:

- $\alpha(t) = 1$: $Q_T(a) = R_{t,A_t=a}$
- $\alpha(t) = 0$: $Q_T(a) = Q_1(a)$
- $\alpha(t) = \frac{1}{N_t(a)}$: Average reward



Efficient computation and non-stationarity

- Compute $Q_t(a)$ on the fly:

$$Q_T(a) = Q_{T-1} + \frac{1}{N_t(a)}(R_{t,A_t=a} - Q_{T-1}(a))$$

- Handling non-stationarity:

$$Q_T(a) = Q_{T-1} + \alpha(t)(R_{t,A_t=a} - Q_{T-1}(a))$$

- Examples:

- $\alpha(t) = 1$: $Q_T(a) = R_{t,A_t=a}$
- $\alpha(t) = 0$: $Q_T(a) = Q_1(a)$
- $\alpha(t) = \frac{1}{N_t(a)}$: Average reward
- $Q_T(a) \rightarrow q^*(a)$, if:



Efficient computation and non-stationarity

- Compute $Q_t(a)$ on the fly:

$$Q_T(a) = Q_{T-1} + \frac{1}{N_t(a)}(R_{t,A_t=a} - Q_{T-1}(a))$$

- Handling non-stationarity:

$$Q_T(a) = Q_{T-1} + \alpha(t)(R_{t,A_t=a} - Q_{T-1}(a))$$

- Examples:

- $\alpha(t) = 1$: $Q_T(a) = R_{t,A_t=a}$
- $\alpha(t) = 0$: $Q_T(a) = Q_1(a)$
- $\alpha(t) = \frac{1}{N_t(a)}$: Average reward

- $Q_T(a) \rightarrow q^*(a)$, if:

1. $\sum_t \alpha_t = \infty$
2. $\sum_t \alpha_t^2 < \infty$

- Where have we seen these criterias before?



- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

The ϵ -greedy algorithm

A simple bandit algorithm

Initialize, for $a = 1$ to k :

$$Q(a) \leftarrow 0$$

$$N(a) \leftarrow 0$$

Repeat forever:

$$A \leftarrow \begin{cases} \arg \max_a Q(a) & \text{with probability } 1 - \epsilon \quad (\text{breaking ties randomly}) \\ \text{a random action} & \text{with probability } \epsilon \end{cases}$$

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

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

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

Figure: The ϵ -greedy algorithm



UPPSALA
UNIVERSITET

The Upper-Confidence-Bound method

- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

- Explore based on our uncertainty of $Q_t(a)$



- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

- Explore based on our uncertainty of $Q_t(a)$
- The Upper-Confidence-Bound (UCB) method

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

An analogy:

$$A_t = \arg \max_a \left(Q_t + c \sqrt{\frac{\hat{\sigma}^2(a)}{N_t(a)}} \right)$$



UPPSALA
UNIVERSITET

- Introduction to Reinforcement Learning
- **Bandits**
- Markov Decision Processes

The UCB algorithm

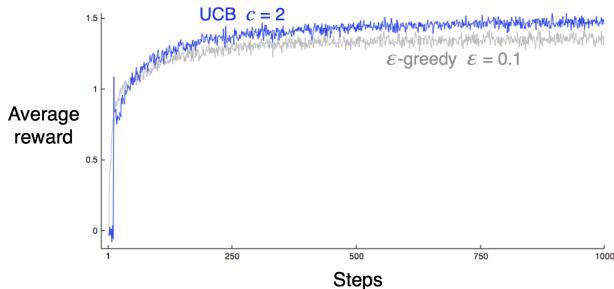


Figure: The UCB algorithm



The Bayesian Bandit: Thompson sampling

- A Bayesian Bandit
 1. Setup a likelihood for R , $p(R|a, \theta)$
 2. Setup a prior for θ , $p(\theta)$
 3. Compute posterior for θ , $p(\theta|R, a)$
 4. Choose action A_t proportional to

$$\int I[\mathbb{E}(R|\theta, a^*) = \arg \max_{a'} \mathbb{E}(R|\theta, a')] p(\theta|R, a) d\theta$$

where I is the indicator function.

- Repeat step 3-4



The Bayesian Bandit: Thompson sampling

- A Bayesian Bandit
 1. Setup a likelihood for R , $p(R|a, \theta)$
 2. Setup a prior for θ , $p(\theta)$
 3. Compute posterior for θ , $p(\theta|R, a)$
 4. Choose action A_t proportional to

$$\int I[\mathbb{E}(R|\theta, a^*) = \arg \max_{a'} \mathbb{E}(R|\theta, a')] p(\theta|R, a) d\theta$$

where I is the indicator function.

- Repeat step 3-4
- Monte Carlo approximation of step 4:
 1. Draw one sample from the posterior $\tilde{\theta}$

$$\tilde{\theta} \sim p(\theta|R, a)$$

2. Conditional on $\tilde{\theta}$, choose action A_t

$$A_t = \arg \max_{a'} \mathbb{E}(R|\tilde{\theta}, a')$$



UPPSALA
UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

Section 3

Markov Decision Processes



UPPSALA
UNIVERSITET

The Markov Decision process

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- Bandits does not have a *state*.



UPPSALA
UNIVERSITET

The Markov Decision process

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- Bandits does not have a *state*.
- An action might **change** the environment.
- An action might be different in different **states**



The Markov Decision process

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- Bandits does not have a *state*.
- An action might **change** the environment.
- An action might be different in different **states**
- **Example:** In chess, we want to make a move based on the current position of all pieces



The Markov Decision process

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- Bandits does not have a *state*.
- An action might **change** the environment.
- An action might be different in different **states**
- **Example:** In chess, we want to make a move based on the current position of all pieces
- To capture this we use a **Markov Decision process**
- One of the most important concepts in Reinforcement Learning



- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

The Markov Decision process

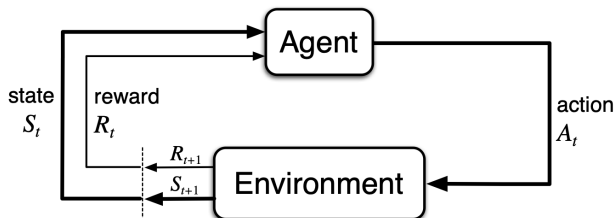


Figure: The (finite) Markov Decision Process (Sutton and Barto, 2017, Fig 3.1)

- States $S_t \in \mathcal{S}$: Basis for action
- Actions $A_t \in \mathcal{A}$
- Rewards $R_t \in \mathbb{R}$



UPPSALA
UNIVERSITET

The Markov Decision process

- Introduction to Reinforcement Learning
 - Bandits
 - Markov Decision Processes
- Boundry between Agent and Environment:
 - The **total control** of the action



The Markov Decision process

- Introduction to Reinforcement Learning
 - Bandits
 - Markov Decision Processes
- Boundry between Agent and Environment:
 - The **total control** of the action
 - Reward is **external** to agent: Pain and pleasure
 - The agent **should not be able to change the reward function**



- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

- Boundry between Agent and Environment:
 - The **total control** of the action
 - Reward is **external** to agent: Pain and pleasure
 - The agent **should not be able to change the reward function**
- The policy ($\pi(A_t|S_t = s)$):
 - We make an action given the current state S_t
- **The goal**: (Again) maximize return $G_t = R_{t+1} + \dots + R_T$



Return and discount

- Two type of interactions
 - **Episodic**: $T < \infty$, has terminal state
 - **Continuing**: $T = \infty$
- Discounting:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$



Return and discount

- Two type of interactions
 - **Episodic**: $T < \infty$, has terminal state
 - **Continuing**: $T = \infty$
- Discounting:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- Discount rate γ :
 - $0 \leq \gamma \leq 1$



Return and discount

- Two type of interactions
 - **Episodic**: $T < \infty$, has terminal state
 - **Continuing**: $T = \infty$
- Discounting:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- Discount rate γ :
 - $0 \leq \gamma \leq 1$
 - $\gamma = 1$: **No discount**
 - $\gamma = 0$: **Full discount**: Only next reward counts
 - $\gamma < 1$ and R_t is bounded: $G_t < \infty$



Return and discount

- Two type of interactions
 - **Episodic**: $T < \infty$, has terminal state
 - **Continuing**: $T = \infty$
- Discounting:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- Discount rate γ :
 - $0 \leq \gamma \leq 1$
 - $\gamma = 1$: **No discount**
 - $\gamma = 0$: **Full discount**: Only next reward counts
 - $\gamma < 1$ and R_t is bounded: $G_t < \infty$
- For episodic problem we assume $R_{T+i} = 0$ for all $i \in \mathbb{N}^+$



- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

The Markov Decision Process

- The Markov Decision process (MDP):

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

- Eq. (1) **fully specify** a MDP



- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

The Markov Decision Process

- The Markov Decision process (MDP):

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

- Eq. (1) **fully specify** a MDP
- Markov property:

$$P(S_{t+1} = s', R_{t+1} = r | S_1 = s, A_1 = a, \dots, S_t = s, A_t = a) = \\ P(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a)$$



The Markov Decision Process

- The Markov Decision process (MDP):

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

- Eq. (1) **fully specify** a MDP
- Markov property:

$$P(S_{t+1} = s', R_{t+1} = r | S_1 = s, A_1 = a, \dots, S_t = s, A_t = a) = \\ P(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a)$$

- The MDP is a good **approximation or model**:
All models are wrong, but some are useful.



The Markov Decision Process

- The Markov Decision process (MDP):

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

- Eq. (1) **fully specify** a MDP
- Markov property:

$$P(S_{t+1} = s', R_{t+1} = r | S_1 = s, A_1 = a, \dots, S_t = s, A_t = a) = P(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a)$$

- The MDP is a good **approximation or model**:
All models are wrong, but some are useful.
- From Eq. (1) we can get marginals of interest:
 - State-action rewards:

$$r(s, a) = \mathbb{E}(R_{t+1} | S_t = s, A_t = a)$$



The Markov Decision Process

- The Markov Decision process (MDP):

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

- Eq. (1) **fully specify** a MDP
- Markov property:

$$P(S_{t+1} = s', R_{t+1} = r | S_1 = s, A_1 = a, \dots, S_t = s, A_t = a) = \\ P(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a)$$

- The MDP is a good **approximation or model**:
All models are wrong, but some are useful.
- From Eq. (1) we can get marginals of interest:
 - State-action rewards:

$$r(s, a) = \mathbb{E}(R_{t+1} | S_t = s, A_t = a)$$

- State-transition probability:

$$p(s' | s, a) = P(S_{t+1} = s' | S_t = s, A_t = a)$$



- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

The Value function

- The value function $v_\pi(s)$:
the long-term value of s given a policy $\pi(a|s)$:

$$v_\pi(s) = \mathbb{E}_\pi(G_t | S_t = s) = \mathbb{E}_\pi \left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right)$$



- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

The Value function

- The value function $v_\pi(s)$:
the long-term value of s given a policy $\pi(a|s)$:

$$v_\pi(s) = \mathbb{E}_\pi(G_t | S_t = s) = \mathbb{E}_\pi \left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right)$$

- Informally: How "good" is a state for the agent with the policy π .



- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

The Value function

- The value function $v_\pi(s)$:
the long-term value of s given a policy $\pi(a|s)$:

$$v_\pi(s) = \mathbb{E}_\pi(G_t | S_t = s) = \mathbb{E}_\pi \left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right)$$

- Informally: How "good" is a state for the agent with the policy π .
- Estimating $v_\pi(s)$ is one of the most important problem in RL



The Value function

- The value function $v_\pi(s)$:
the long-term value of s given a policy $\pi(a|s)$:

$$v_\pi(s) = \mathbb{E}_\pi(G_t | S_t = s) = \mathbb{E}_\pi \left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right)$$

- Informally: How "good" is a state for the agent with the policy π .
- Estimating $v_\pi(s)$ is one of the most important problem in RL
- Value functions are **recursive**:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi(G_t | S_t = s) \\ &= \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a) (r + \gamma v_\pi(s')) \end{aligned}$$

- This is the **Bellman equation** for $v_\pi(s)$:
The relationship between the values of the state and its successor states.



The Value function

- The value function $v_\pi(s)$:
the long-term value of s given a policy $\pi(a|s)$:

$$v_\pi(s) = \mathbb{E}_\pi(G_t | S_t = s) = \mathbb{E}_\pi \left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right)$$

- Informally: How "good" is a state for the agent with the policy π .
- Estimating $v_\pi(s)$ is one of the most important problem in RL
- Value functions are **recursive**:

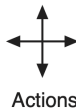
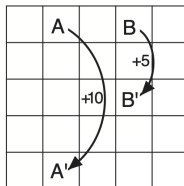
$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi(G_t | S_t = s) \\ &= \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a) (r + \gamma v_\pi(s')) \end{aligned}$$

- This is the **Bellman equation** for $v_\pi(s)$:
The relationship between the values of the state and its successor states.
- Bellman equation is the basis for computing $v_\pi(s)$ (not part of this course)



The value function

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

Figure: The gridworld equiprobable policy value function



The Optimal Policy

- A policy π is **better** than π' if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$.

- Introduction to Reinforcement Learning
- Bandits
- **Markov Decision Processes**



- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

The Optimal Policy

- A policy π is **better** than π' if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$.
- A policy that is better or equal to all other policies is the **optimal policy** π_\star .



- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

The Optimal Policy

- A policy π is **better** than π' if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$.
- A policy that is better or equal to all other policies is the **optimal policy** π_\star .
- The optimal value function: $v_{\pi_\star}(s) = v_\star(s)$



- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

The Optimal Policy

- A policy π is **better** than π' if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$.
- A policy that is better or equal to all other policies is the **optimal policy** π_\star .
- The optimal value function: $v_{\pi_\star}(s) = v_\star(s)$
- The optimal policy π_\star is greedy wrt $v_{\pi_\star}(s)$:
The best long term strategy



The Optimal Policy

- A policy π is **better** than π' if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$.
- A policy that is better or equal to all other policies is the **optimal policy** π_\star .
- The optimal value function: $v_{\pi_\star}(s) = v_\star(s)$
- The optimal policy π_\star is greedy wrt $v_{\pi_\star}(s)$:
The best long term strategy
- Computing optimal value function might be **impossible**:
we need to estimate/re-estimate/approximate it



The Optimal Policy

- A policy π is **better** than π' if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$.
- A policy that is better or equal to all other policies is the **optimal policy** π_\star .
- The optimal value function: $v_{\pi_\star}(s) = v_\star(s)$
- The optimal policy π_\star is greedy wrt $v_{\pi_\star}(s)$:
The best long term strategy
- Computing optimal value function might be **impossible**:
we need to estimate/re-estimate/approximate it
- One of the most important problem in RL



The Optimal Policy

- A policy π is **better** than π' if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$.
- A policy that is better or equal to all other policies is the **optimal policy** π_\star .
- The optimal value function: $v_{\pi_\star}(s) = v_\star(s)$
- The optimal policy π_\star is greedy wrt $v_{\pi_\star}(s)$:
The best long term strategy
- Computing optimal value function might be **impossible**:
we need to estimate/re-estimate/approximate it
- One of the most important problem in RL
- **Example**: Chess, we cannot compute the optimal long-term moves, we need to approximate/estimate (based on computational budget)



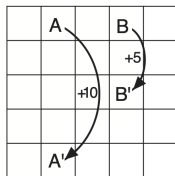
The Optimal Policy

- A policy π is **better** than π' if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$.
- A policy that is better or equal to all other policies is the **optimal policy** π_\star .
- The optimal value function: $v_{\pi_\star}(s) = v_\star(s)$
- The optimal policy π_\star is greedy wrt $v_{\pi_\star}(s)$:
The best long term strategy
- Computing optimal value function might be **impossible**:
we need to estimate/re-estimate/approximate it
- One of the most important problem in RL
- **Example**: Chess, we cannot compute the optimal long-term moves, we need to approximate/estimate (based on computational budget)
- We might also estimate $v_\star(s)$ better for commonly encountered states



The value function

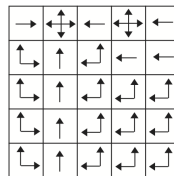
- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes



Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*



π_*

Figure: The gridworld optimal value function and policy