UPPSALA UNIVERSITET



# Machine learning – Block 8


Måns Magnusson
Department of Statistics, Uppsala University


Autumn 2022

# This week's lectures

- Introduction to Reinforcement learning

# Introduction to Reinforcement Learning

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

# 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

# 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

# Introduction to Reinforcement Learning

- Goal: maximize return over a sequence of actions

# Introduction to Reinforcement Learning

- Goal: maximize return over a sequence of actions
- Three characteristics:
    1. Closed-loop: early actions affects later actions

# Introduction to Reinforcement Learning

- Goal: maximize return over a sequence of actions
- Three characteristics:
  1. Closed-loop: early actions affects later actions
  2. No direct instructions

# Introduction to Reinforcement Learning

- 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

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# Recent Achievements

- RL agent won over Lee Sedol in 2016: AlphaGo

UPPSALA UNIVERSITET



# Recent Achievements

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

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# Recent Achievements

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

# Recent Achievements

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

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# Recent Achievements

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

# Recent Achievements

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

# Recent Achievements

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

# Recent Achievements

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

# The different parts in RL

1. The Agent: The learning agent.

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# The different parts in RL

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

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# The different parts in RL

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

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# The different parts in RL

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 singular value. Pleasure and pain.

# The different parts in RL

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 singular value. Pleasure and pain.
5. Return: The aggregated reward over a long period.

# The different parts in RL

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

# 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
2. Usually an uncertainty about the environment
3. Represent uncertainty of environment: Probability

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# 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

# 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)

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
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

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# 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
- Problem: Balance the trade-off between long-term and short-term rewards

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# Supervised, Unsupervised and Reinforcement Learning

1. Static vs. Dynamic

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# Supervised, Unsupervised and Reinforcement Learning

1. Static vs. Dynamic
2. No Gold Standard

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# Supervised, Unsupervised and Reinforcement Learning

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

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# Supervised, Unsupervised and Reinforcement Learning

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

# Supervised, Unsupervised and Reinforcement Learning

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

# Exploration vs Exploitation

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

# Exploration vs Exploitation

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

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# Exploration vs Exploitation

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

# Evolution vs Learning

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

# Evolution vs Learning

UPPSALA UNIVERSITET



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

# Setting the goal for the Agent

- 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



- 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...

# The $k$-armed bandit problem

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

# The $k$-armed bandit problem

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

# The $k$-armed bandit problem

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

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

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

UPPSALA UNIVERSITET

- 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. $A_t \in \{1, ..., k\}$
- The reward signal:

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

  where $\mathbb{E}(R_t|A_t = a) = q^\star(a)$.
- $q^\star(a)$ is unknown.

# The $k$-armed bandit problem

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

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

  where $\mathbb{E}(R_t|A_t = a) = q^\star(a)$.
- $q^\star(a)$ 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. $A_t \in \{1, ..., k\}$
- The reward signal:

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

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

UPPSALA
UNIVERSITET



# Exploration vs. Exploitation

- 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

# $\epsilon$-greedy

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

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# $\epsilon$-greedy

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

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

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# $\epsilon$-greedy

- $\epsilon$-greedy: $P(\text{exploration}) = \epsilon$
  - Exploitation:
  $$A_t = \arg\max_a Q_t(a)$$
  - Exploration:
  $$A_t \sim U(1, ...k)$$

# $\epsilon$-greedy

- $\epsilon$-greedy: $P(\text{exploration}) = \epsilon$
  - Exploitation:
$$A_t = \arg\max_a Q_t(a)$$
  - Exploration:
$$A_t \sim U(1, ...k)$$
- $Q_1(a) = 0$ (or used to encourage initial exploration)

# $\epsilon$-greedy

- $\epsilon$-greedy: $P(\text{exploration}) = \epsilon$
  - Exploitation:
$$A_t = \arg\max_a Q_t(a)$$
  - Exploration:
$$A_t \sim U(1, ...k)$$
- $Q_1(a) = 0$ (or used to encourage initial exploration)
- For any $\epsilon > 0$, $Q_t(a) \to q^\star(a)$

UPPSALA UNIVERSITET



# $\epsilon$-greedy

- $\epsilon$-greedy: $P(\text{exploration}) = \epsilon$
  - Exploitation:
  $$A_t = \arg\max_a Q_t(a)$$
  - Exploration:
  $$A_t \sim U(1, \ldots k)$$
- $Q_1(a) = 0$ (or used to encourage initial exploration)
- For any $\epsilon > 0$, $Q_t(a) \to q^\star(a)$
- We estimate $q^\star(a)$ 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.

# $\epsilon$-greedy

- $\epsilon$-greedy: $P(\text{exploration}) = \epsilon$
  - Exploitation:
    $$A_t = \arg\max_a Q_t(a)$$
  - Exploration:
    $$A_t \sim U(1, ...k)$$
- $Q_1(a) = 0$ (or used to encourage initial exploration)
- For any $\epsilon > 0$, $Q_t(a) \rightarrow q^\star(a)$
- We estimate $q^\star(a)$ 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 $V(R_t)$

UPPSALA UNIVERSITET



# $\epsilon$-greedy

- $\epsilon$-greedy: $P(\text{exploration}) = \epsilon$
  - Exploitation:
    $$A_t = \arg\max_a Q_t(a)$$
  - Exploration:
    $$A_t \sim U(1, \ldots k)$$
- $Q_1(a) = 0$ (or used to encourage initial exploration)
- For any $\epsilon > 0$, $Q_t(a) \to q^\star(a)$
- We estimate $q^\star(a)$ 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 $\mathcal{A}$

# $\epsilon$-greedy

- $\epsilon$-greedy: $P(\text{exploration}) = \epsilon$
  - Exploitation:
$$A_t = \arg\max_a Q_t(a)$$
  - Exploration:
$$A_t \sim U(1, ...k)$$
- $Q_1(a) = 0$ (or used to encourage initial exploration)
- For any $\epsilon > 0$, $Q_t(a) \to q^\star(a)$
- We estimate $q^\star(a)$ 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

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# Bandit example

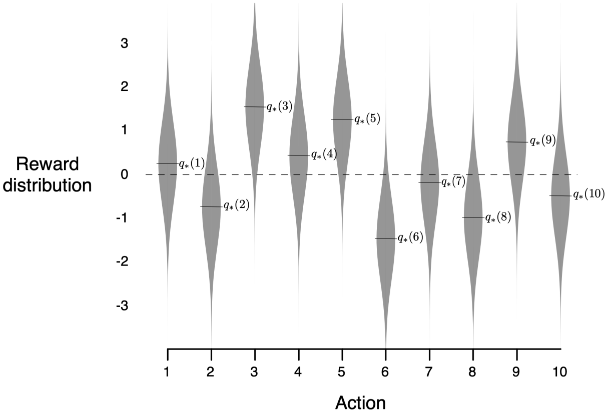



Figure: The 10-armed bandit environment (Sutton and Barto, 2017, Fig. 2.1)

# Bandit example

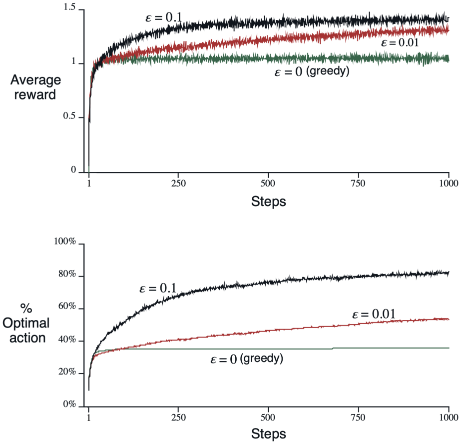



Figure: The $\epsilon$-greedy algorithm result in the 10-armed bandit (Sutton and Barto, 2017, Fig. 2.2)

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# Bandit example: Optimistic initialization

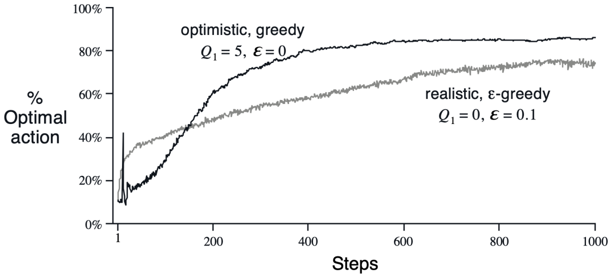



Figure: The $\epsilon$-greedy algorithm and optimistic initialization (Sutton and Barto, 2017, Fig. 2.3)

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# Efficient computation and non-stationarity

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

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

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

UPPSALA UNIVERSITET



# 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
- $Q_T(a) \to q^\star(a)$, 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
- $Q_T(a) \to q^\star(a)$, if:
  1. $\sum_t^\infty \alpha_t = \infty$
  2. $\sum_t^\infty \alpha_t^2 < \infty$
- Where have we seen these criterias before?

# The $\epsilon$-greedy algorithm

**A simple bandit algorithm**

Initialize, for $a = 1$ to $k$:

$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$

Repeat forever:

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

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

Figure: The $\epsilon$-greedy algorithm

# The Upper-Confidence-Bound method

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

# The Upper-Confidence-Bound method

- 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:

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

- Another (Bayesian) alternative is Thompson Sampling

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# The UCB algorithm

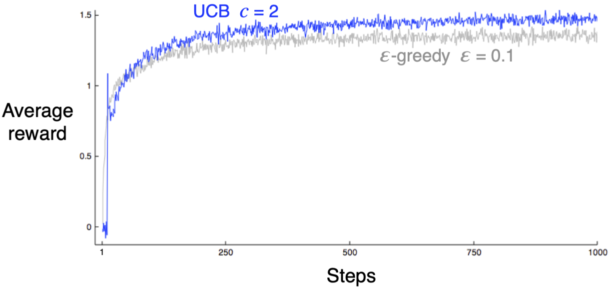



Figure: The UCB algorithm

# The Markov Decision process

- Bandit does not have a *state*.

UPPSALA UNIVERSITET



# The Markov Decision process

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

# The Markov Decision process



- Bandit 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

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# The Markov Decision process

- Bandit 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

# The Markov Decision process

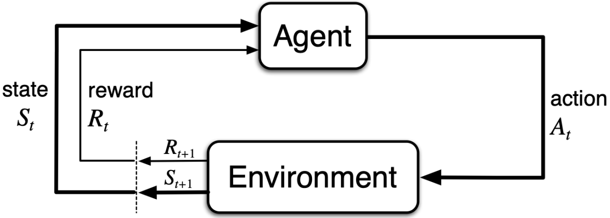



Figure: The (finite) Markov Decision Process (Sutton and Barto, 2017, Fig 3.1)

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

# The Markov Decision process

- Boundry between Agent and Environment:
  - The total control of the action

# The Markov Decision process

- 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

UPPSALA UNIVERSITET



# The Markov Decision process

- 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 $G_t = R_{t+1} + ... + R_T$

# Return and discount

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

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

# Return and discount

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

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

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

# Return and discount

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

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

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

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

- Discount rate $\gamma$:
  - $0 \leq \gamma \leq 1$
  - $\gamma = 1$: 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 $i \in \mathbb{N}^+$

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# The Markov Decision Process

- The Markov Decision process (MDP):

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

- Eq. (1) fully specify a MDP

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

# The Markov Decision Process

- The Markov Decision process (MDP):

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

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

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

# The Markov Decision Process

- The Markov Decision process (MDP):

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

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

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

- 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):

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

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

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

- 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):

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

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

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

- 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:

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

UPPSALA UNIVERSITET

- 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 $\pi(a|s)$:

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

UPPSALA UNIVERSITET



# 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 $\pi$.

# 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 $\pi$.
- Estimating $v_\pi(s)$ is one of the most important problem in RL

UPPSALA UNIVERSITET

- 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 $\pi$.
- 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 $\pi(a|s)$:

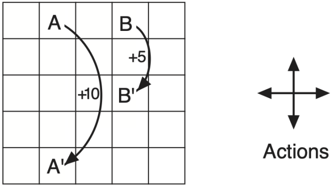
$$v_\pi(s) = \mathbb{E}_\pi(G_t|S_t = s) = \mathbb{E}_\pi\left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s\right)$$

- Informally: How "good" is a state for the agent with the policy $\pi$.
- 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




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

# The Optimal Policy

- A policy $\pi$ is better than $\pi'$ 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 $\pi_\star$.

# The Optimal Policy

- A policy $\pi$ is better than $\pi'$ 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 $\pi_\star$.
- The optimal value function: $v_{\pi_\star}(s) = v_\star(s)$

# The Optimal Policy

- A policy $\pi$ is better than $\pi'$ 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 $\pi_\star$.
- The optimal value function: $v_{\pi_\star}(s) = v_\star(s)$
- The optimal policy $\pi_\star$ is greedy wrt $v_{\pi_\star}(s)$:
  The best long term strategy

# The Optimal Policy

- A policy $\pi$ is better than $\pi'$ 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 $\pi_\star$.
- The optimal value function: $v_{\pi_\star}(s) = v_\star(s)$
- The optimal policy $\pi_\star$ 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

UPPSALA UNIVERSITET



# The Optimal Policy

- A policy $\pi$ is better than $\pi'$ 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 $\pi_\star$.
- The optimal value function: $v_{\pi_\star}(s) = v_\star(s)$
- The optimal policy $\pi_\star$ 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

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

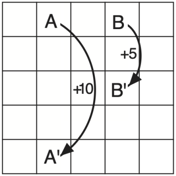
# The Optimal Policy

- A policy $\pi$ is better than $\pi'$ 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 $\pi_\star$.
- The optimal value function: $v_{\pi_\star}(s) = v_\star(s)$
- The optimal policy $\pi_\star$ 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 $\pi$ is better than $\pi'$ 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 $\pi_\star$.
- The optimal value function: $v_{\pi_\star}(s) = v_\star(s)$
- The optimal policy $\pi_\star$ 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

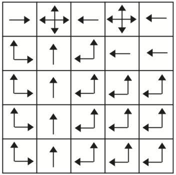
# The value function




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 |

$v_*$



$\pi_*$

Figure: The gridworld optimal value function and policy