UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

Machine learning – Block 8

Måns Magnusson
Department of Statistics, Uppsala University

Autumn 2025

UPPSALA UNIVERSITET

- Introduction to Reinforcement Learning
- Bandits
- Markov Decision Processes

This week's lectures

- Introduction to reinforcement learning

UPPSALA UNIVERSITET

Introduction to Reinforcement Learning

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

UPPSALA UNIVERSITET

Introduction to Reinforcement Learning

- Another type of machine learning:
  - Supervised Learning
  - Unsupervised Learning
  - Reinforcement learning
- Computational approach of learning from interaction

UPPSALA UNIVERSITET

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.

UPPSALA UNIVERSITET

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

UPPSALA UNIVERSITET

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

- Goal: maximize return over a sequence of actions

UPPSALA UNIVERSITET

Introduction to Reinforcement Learning

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

UPPSALA UNIVERSITET

Introduction to Reinforcement Learning

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

UPPSALA UNIVERSITET

Introduction to Reinforcement Learning

- Goal: maximize return over a sequence of actions
- Three characteristics:
  1. Closed-loop: early actions affect later actions
  2. No instructions
  3. Reward signals over a long period of time

UPPSALA UNIVERSITET

Recent Achievements

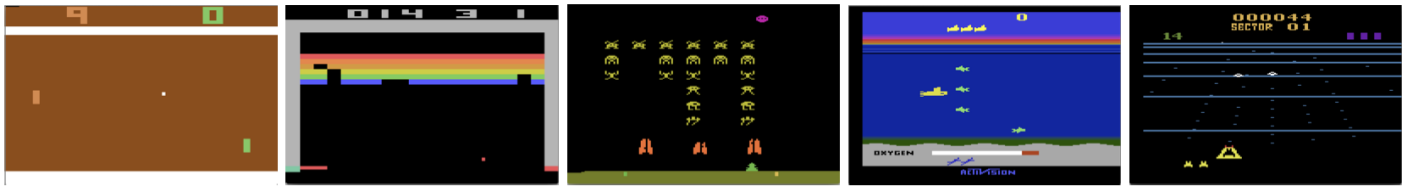


Figure: Mnih et al (2013) "Playing Atari with Deep Reinforcement Learning"

UPPSALA UNIVERSITET

Recent Achievements

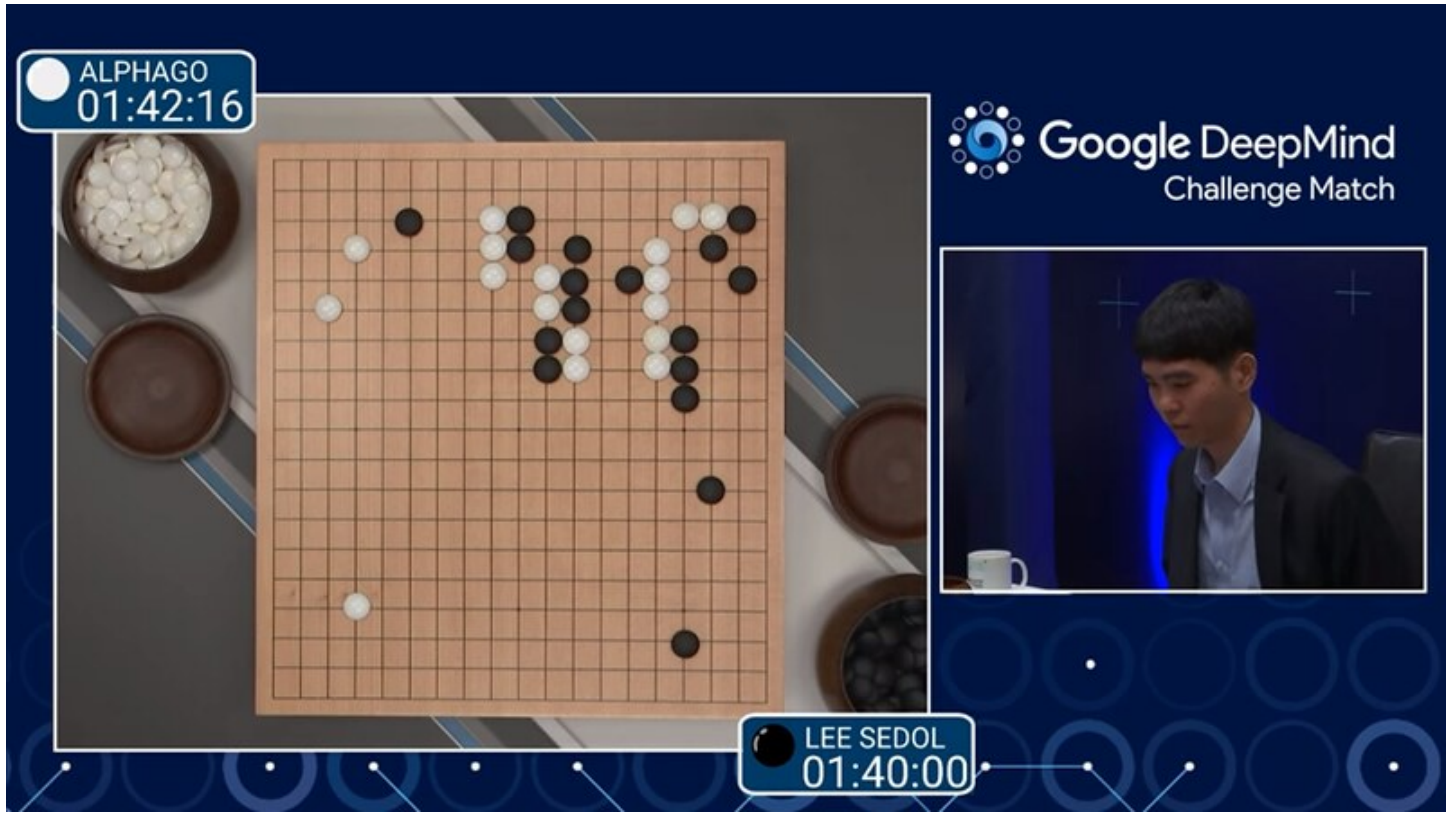


Figure: Lee Sedol vs. Alpha Go in 2016

UPPSALA UNIVERSITET

Recent Achievements, also

- Industry automation: RL is used to reduce the energy cost of datacenter cooling

UPPSALA UNIVERSITET

Recent Achievements, also

- Industry automation: RL is used to reduce the energy cost of datacenter cooling
- Automated trading

UPPSALA UNIVERSITET

Recent Achievements, also

- Industry automation: RL is used to reduce the energy cost of datacenter cooling
- Automated trading
- Elevator scheduling

UPPSALA UNIVERSITET

Recent Achievements, also

- 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

The different parts in RL

1. The Agent: The learning agent.

UPPSALA UNIVERSITET

The different parts in RL

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

UPPSALA UNIVERSITET

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 affect the environment.

UPPSALA UNIVERSITET

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 affect the environment.
4. Reward: The evaluation of an action. A scalar value. "Pleasure and pain" of actions.

UPPSALA UNIVERSITET

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 affect the environment.
4. Reward: The evaluation of an action. A scalar value. "Pleasure and pain" of actions.
5. Return: The aggregated reward over a long period.

UPPSALA UNIVERSITET

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

UPPSALA UNIVERSITET

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

UPPSALA UNIVERSITET

Sub-elements of agents

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

UPPSALA UNIVERSITET

Sub-elements of agents

1. Policy: How the agent chooses 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 chooses 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 the agent: Reward signal: The instant value of an action

UPPSALA UNIVERSITET

Sub-elements of agents

1. Policy: How the agent chooses 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 the agent: Reward signal: The instant value of an action
- Problem: Balance the trade-off between long-term and short-term rewards

UPPSALA UNIVERSITET

Key Distinctions in Reinforcement Learning

Reward vs. Return

- *Reward* R_{t+1}:
  - Immediate, scalar feedback from the environment.
  - Measures short-term desirability of an action.
- *Return* G_t:
  - Cumulative (possibly discounted) future reward.
  - Objective the agent seeks to maximize.

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

Policy vs. Value Function

- *Policy* $\pi(a \mid s)$:
  - Defines how the agent chooses actions.
  - A probability distribution over actions given a state.
- *Value function* $v_\pi(s)$:
  - Evaluates how good a state is under a policy.
  - Expected return starting from state s and following π.

UPPSALA UNIVERSITET

Supervised, Unsupervised and Reinforcement Learning

1. Static vs. Dynamic

UPPSALA UNIVERSITET

Supervised, Unsupervised and Reinforcement Learning

1. Static vs. Dynamic
2. No Gold Standard

UPPSALA UNIVERSITET

Supervised, Unsupervised and Reinforcement Learning

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

UPPSALA UNIVERSITET

Supervised, Unsupervised and Reinforcement Learning

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

1. Static vs. Dynamic
2. No Gold Standard
3. Multiple-Decision Process: Return vs. reward
4. Need for exploration
5. Evaluates actions rather than being explicitly instructed which actions to take

UPPSALA UNIVERSITET

Exploration vs Exploitation

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

UPPSALA UNIVERSITET

Exploration vs Exploitation

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

UPPSALA UNIVERSITET

Exploration vs Exploitation

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

UPPSALA UNIVERSITET

Evolution vs Learning

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

UPPSALA UNIVERSITET

Evolution vs Learning

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

UPPSALA UNIVERSITET

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.

UPPSALA UNIVERSITET

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...

UPPSALA UNIVERSITET

Bandits vs. Markov Decision Processes

Two levels of sequential decision-making

Bandits

- No state: each action is independent of past actions.
- No dynamics: actions do not change the environment.
- Feedback is myopic: reward depends only on the chosen action.
- Objective: identify and exploit the best action.

UPPSALA UNIVERSITET

Bandits vs. Markov Decision Processes

Two levels of sequential decision-making

Bandits

- No state: each action is independent of past actions.
- No dynamics: actions do not change the environment.
- Feedback is myopic: reward depends only on the chosen action.
- Objective: identify and exploit the best action.

Markov Decision Processes (MDPs)

- Explicit state describing the environment.
- Actions influence future states and rewards.
- Feedback is delayed: consequences unfold over time.
- Objective: plan a sequence of actions to maximize long-term return.

UPPSALA UNIVERSITET

Section 2

Bandits

UPPSALA UNIVERSITET

The k-armed bandit problem

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

UPPSALA UNIVERSITET

The k-armed bandit problem

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

UPPSALA UNIVERSITET

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:

$$R_t \sim p(\cdot \mid A_t = a), \qquad \mathbb{E}[R_t \mid A_t = a] = q^\star(a).$$

UPPSALA UNIVERSITET

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:

$$R_t \sim p(\cdot \mid A_t = a), \qquad \mathbb{E}[R_t \mid A_t = a] = q^\star(a).$$

- $q^\star(a)$ is unknown.

UPPSALA UNIVERSITET

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:

$$R_t \sim p(\cdot \mid A_t = a), \qquad \mathbb{E}[R_t \mid A_t = a] = q^\star(a).$$

- $q^\star(a)$ is unknown.
- Estimated expected reward of action a at time t.

UPPSALA UNIVERSITET

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:

$$R_t \sim p(\cdot \mid A_t = a), \qquad \mathbb{E}[R_t \mid A_t = a] = q^\star(a).$$

- $q^\star(a)$ is unknown.
- Estimated expected reward of action a at time t.

- 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

UPPSALA UNIVERSITET

ϵ-greedy

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

UPPSALA UNIVERSITET

ϵ-greedy

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

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

UPPSALA UNIVERSITET

ϵ-greedy

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

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

  - Exploration:

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

UPPSALA UNIVERSITET

ϵ-greedy

- ϵ-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)

UPPSALA UNIVERSITET

ϵ-greedy

- ϵ-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 fixed $\epsilon > 0$, under stationarity and with sample-average updates,
$$Q_t(a) \xrightarrow{\text{a.s.}} q^\star(a) \quad \text{as } t \to \infty.$$

UPPSALA UNIVERSITET

ϵ-greedy

- We estimate $q^\star(a)$ using $Q_t(a)$ as

$$Q_T(a) = \frac{1}{N_T(a)} \sum_{t=1}^{T} R_t \, \mathbf{1}\{A_t = a\}.$$

where $N_T(a)$ is the total number of times action a has been taken at time point T.

UPPSALA UNIVERSITET

ϵ-greedy

- We estimate $q^\star(a)$ using $Q_t(a)$ as

$$Q_T(a) = \frac{1}{N_T(a)} \sum_{t=1}^{T} R_t \, \mathbf{1}\{A_t = a\}.$$

  where $N_T(a)$ is the total number of times action a has been taken at time point T.
- When should we explore?
  - Large $V(R_t)$

UPPSALA UNIVERSITET

ϵ-greedy

- We estimate $q^\star(a)$ using $Q_t(a)$ as

$$Q_T(a) = \frac{1}{N_T(a)} \sum_{t=1}^{T} R_t \, \mathbf{1}\{A_t = a\}.$$

  where $N_T(a)$ is the total number of times action a has been taken at time point T.
- When should we explore?
  - Large $V(R_t)$
  - Large $\mathcal{A}$

UPPSALA UNIVERSITET

ϵ-greedy

- We estimate $q^\star(a)$ using $Q_t(a)$ as

$$Q_T(a) = \frac{1}{N_T(a)} \sum_{t=1}^{T} R_t \, \mathbf{1}\{A_t = a\}.$$

  where $N_T(a)$ is the total number of times action a has been taken at time point T.
- When should we explore?
  - Large $V(R_t)$
  - Large $\mathcal{A}$
  - Non-stationarity

UPPSALA UNIVERSITET

The ϵ-greedy algorithm

A simple bandit algorithm

Initialize, for $a = 1$ to k:
$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$

Repeat forever:
$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$
$R \leftarrow bandit(A)$
$N(A) \leftarrow N(A) + 1$
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$

Figure: The ϵ-greedy algorithm

Bandit example

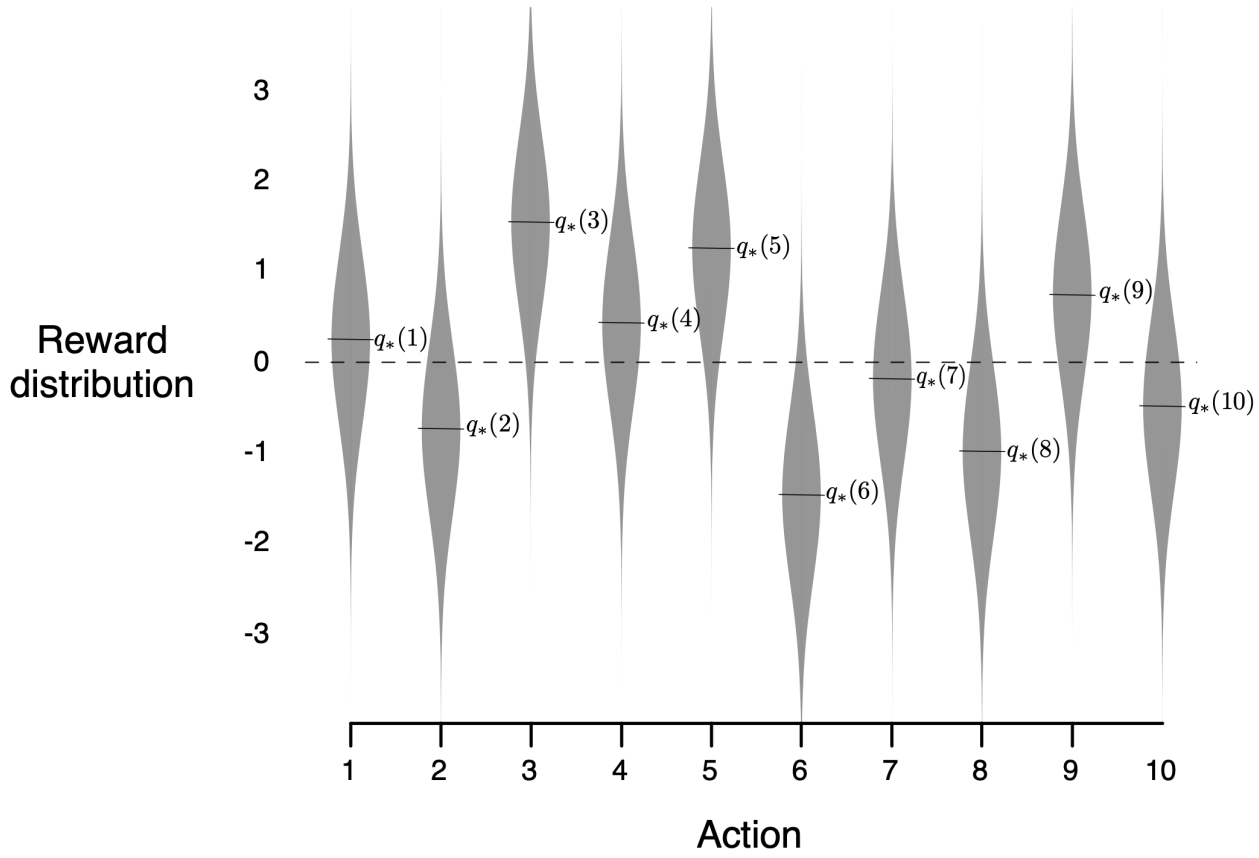


Figure: The 10-armed bandit environment (Sutton and Barto, 2017, Fig. 2.1)

UPPSALA UNIVERSITET

Bandit example

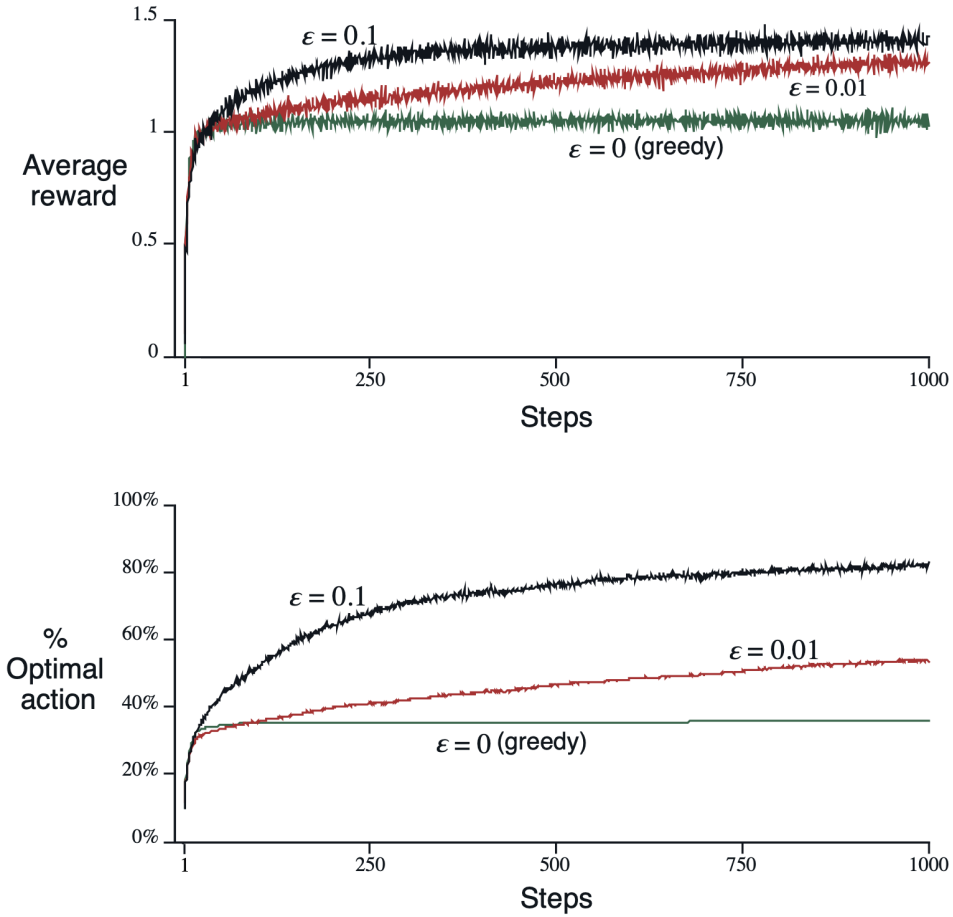


Figure: The ϵ-greedy algorithm result in the 10-armed bandit (Sutton and Barto, 2017, Fig. 2.2)

UPPSALA UNIVERSITET

Bandit example: Optimistic initialization

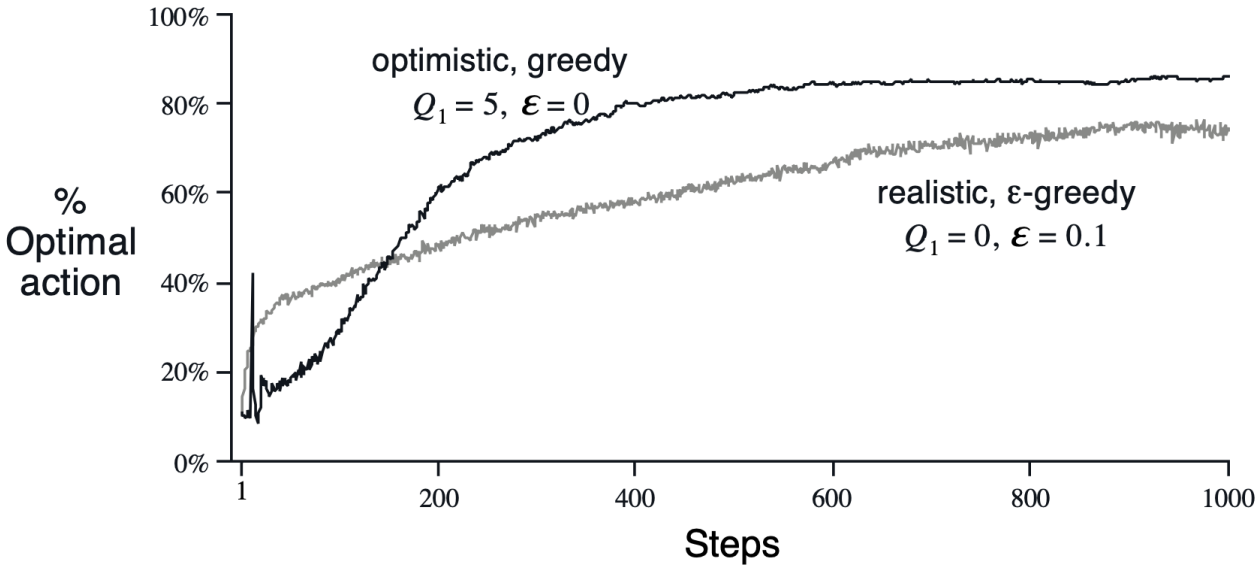


Figure: The ϵ-greedy algorithm and optimistic initialization (Sutton and Barto, 2017, Fig. 2.3)

UPPSALA UNIVERSITET

Efficient computation and non-stationarity

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

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

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:

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

UPPSALA UNIVERSITET

Efficient computation and non-stationarity

$$Q_{t+1}(a) = Q_t(a) + \alpha_t \big(R_{t+1} - Q_t(a)\big).$$

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

UPPSALA UNIVERSITET

Efficient computation and non-stationarity

$$Q_{t+1}(a) = Q_t(a) + \alpha_t \big(R_{t+1} - Q_t(a)\big).$$

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

UPPSALA UNIVERSITET

Efficient computation and non-stationarity

$$Q_{t+1}(a) = Q_t(a) + \alpha_t \big(R_{t+1} - Q_t(a)\big).$$

- 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

UPPSALA UNIVERSITET

Efficient computation and non-stationarity

$$Q_{t+1}(a) = Q_t(a) + \alpha_t \big(R_{t+1} - Q_t(a)\big).$$

- 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

UPPSALA UNIVERSITET

Stochastic Approximation Conditions

$$Q_{t+1}(a) = Q_t(a) + \alpha_t\big(R_{t+1} - Q_t(a)\big).$$

Robbins–Monro conditions

For convergence of stochastic approximation algorithms, the learning rates must satisfy

$$\sum_{t=1}^{\infty} \alpha_t = \infty, \qquad \sum_{t=1}^{\infty} \alpha_t^2 < \infty.$$

Interpretation

- $\sum_t \alpha_t = \infty$: ensures continued learning and prevents premature convergence.
- $\sum_t \alpha_t^2 < \infty$: controls the variance of the stochastic noise and prevents divergence.

UPPSALA UNIVERSITET

Stochastic Approximation Conditions

$$Q_{t+1}(a) = Q_t(a) + \alpha_t\big(R_{t+1} - Q_t(a)\big).$$

$$\sum_{t=1}^{\infty} \alpha_t = \infty, \qquad \sum_{t=1}^{\infty} \alpha_t^2 < \infty.$$

- These are *not heuristics*, but results from stochastic approximation theory (Robbins & Monro, 1951).
- They guarantee convergence of value estimates under stationarity, sufficient exploration, and bounded rewards.
- Many RL algorithms (e.g. bandits, TD learning, Q-learning) rely on these conditions.

Examples

- Sample-average updates: $\alpha_t = \frac{1}{t}$ (satisfies both conditions).
- Constant step-size: $\alpha_t = \alpha$ (violates $\sum_t \alpha_t^2 < \infty$; useful in non-stationary settings).

UPPSALA UNIVERSITET

The Upper-Confidence-Bound method

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

UPPSALA UNIVERSITET

The Upper-Confidence-Bound method

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

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

where $c > 0$ is a hyperparameter controlling exploration.

UPPSALA UNIVERSITET

The Upper-Confidence-Bound method

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

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

where $c > 0$ is a hyperparameter controlling exploration. An analogy:

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

UPPSALA UNIVERSITET

The UCB algorithm

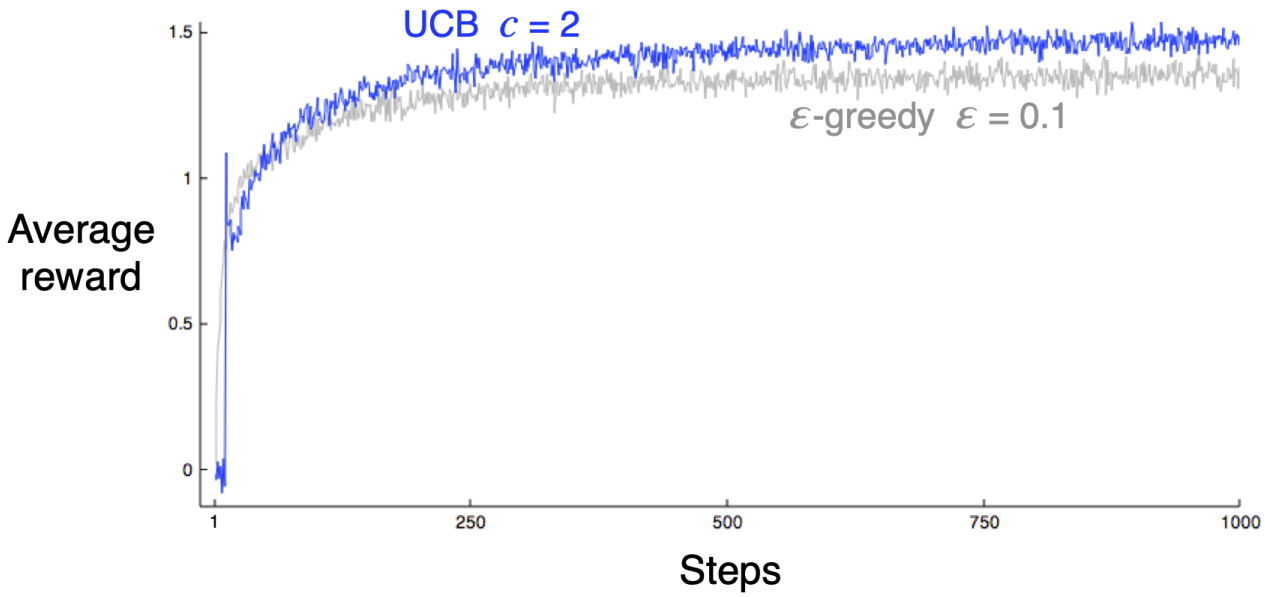


Figure: The UCB algorithm

UPPSALA UNIVERSITET

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, $p(\theta|D, a)$
  4. Choose action A_t by

$$\theta \sim p(\theta \mid D), \qquad A_t = \arg\max_a \mathbb{E}[R \mid a, \theta]\,,$$

     where $D = \{(A_1, R_1), \ldots, (A_{t-1}, R_{t-1})\}$.
  5. Collect reward R_t.
- Repeat step 3-5. Step 4 can be approximated using Monte Carlo.

UPPSALA UNIVERSITET

Section 3

Markov Decision Processes

UPPSALA UNIVERSITET

The Markov Decision process

- Bandits don't have a *state*.

UPPSALA UNIVERSITET

The Markov Decision process

- Bandits don't have a *state*.
- An action might change the environment.
- An action might be different in different states

UPPSALA UNIVERSITET

The Markov Decision process

- Bandits don't 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

The Markov Decision process

- Bandits don't 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

UPPSALA UNIVERSITET

The Markov Decision process

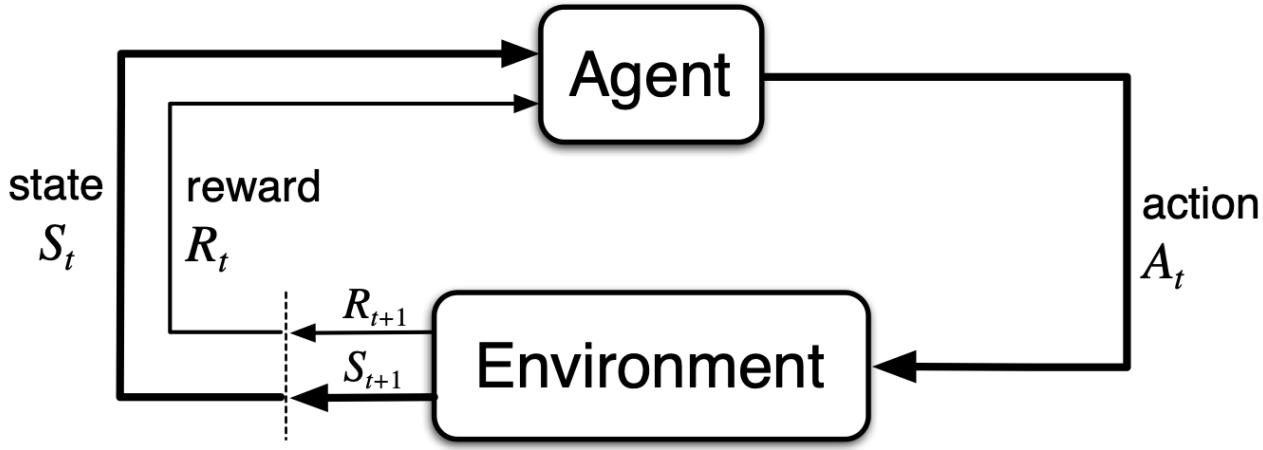


Figure: The (finite) Markov Decision Process (Sutton and Barto, 2017, Fig 3.1)

- 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

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

UPPSALA UNIVERSITET

The Markov Decision process

- Boundary 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

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

UPPSALA UNIVERSITET

Return and discount

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

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

UPPSALA UNIVERSITET

Return and discount

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

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

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

UPPSALA UNIVERSITET

Return and discount

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

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

- 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: $G_t < \infty$

UPPSALA UNIVERSITET

Return and discount

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

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

- 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 $i \in \mathbb{N}^+$

UPPSALA UNIVERSITET

The Markov Decision Process

- The Markov Decision process (MDP):

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

- An MDP is fully specified by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$.

UPPSALA UNIVERSITET

The Markov Decision Process

- The Markov Decision process (MDP):

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

- An MDP is fully specified by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$.
- 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.

UPPSALA UNIVERSITET

The Markov Decision Process Marginals

- The Markov Decision process (MDP):

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

- From Eq. (1) we can get marginals of interest:
  - State-action rewards:

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

UPPSALA UNIVERSITET

The Markov Decision Process Marginals

- The Markov Decision process (MDP):

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

- 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

The Value function

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

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

UPPSALA UNIVERSITET

The Value function

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

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

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

UPPSALA UNIVERSITET

The Value function

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

UPPSALA UNIVERSITET

The Value function

- 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.

UPPSALA UNIVERSITET

The Value function

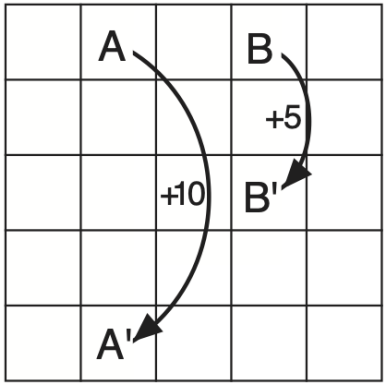
- Estimating $v_\pi(s)$ is one of the most important problem in RL
- Value functions are recursive:

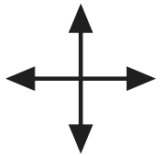
$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi(G_t|S_t = s) \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)(r + \gamma v_\pi(s')) \end{aligned}$$

- 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)

UPPSALA UNIVERSITET

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

UPPSALA UNIVERSITET

The Optimal Policy

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

UPPSALA UNIVERSITET

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 $\pi_\star$.

UPPSALA UNIVERSITET

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 $\pi_\star$.
- The optimal value function: $v_{\pi_\star}(s) = v_\star(s)$

UPPSALA UNIVERSITET

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 $\pi_\star$.
- 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

UPPSALA UNIVERSITET

The Optimal Policy

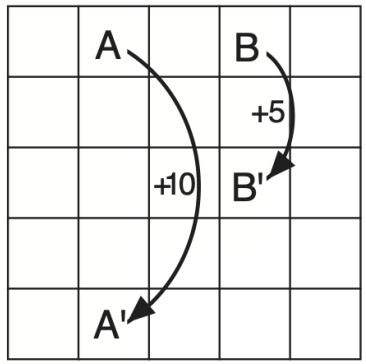
- Computing optimal value function might be impossible: we need to estimate/re-estimate/approximate it
- Example: Chess, we cannot compute the optimal long-term moves, we need to approximate/estimate (based on computational budget)

UPPSALA UNIVERSITET

The Optimal Policy

- Computing optimal value function might be impossible: we need to estimate/re-estimate/approximate it
- 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

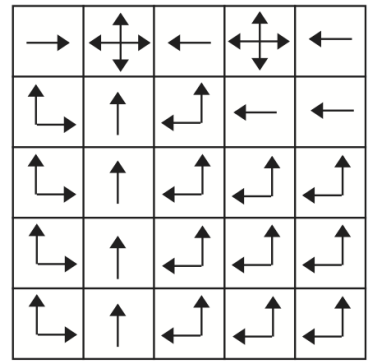
The value function



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_*



π_*

Figure: The gridworld optimal value function and policy