REINFORCEMENT LEARNING
344.101/2/56

Introduction

Silvan Peter
2022-10-06
Institute of Computational Perception

JOHANNES KEPLER UNIVERSITY LINZ

Lecturers

- Paul Primus
- Carlos Cancino-Chacón
- Silvan Peter

Exercise Team

- Khaled Koutini
- Shahed Masoudian
- Gustavo Escobedo Ticona
- Paul Primus
- Carlos Cancino-Chacón
- Jürgen Kieslich
- Pratheesh Nair
- Silvan Peter

Acknowledgements and Special Thanks

- Rainer Kelz
- Andreas Arzt
- Matthias Dorfer
- Florian Henkel

Organizational

Lecture Time:	**Thursday 13:45 - 15:15**
Room:	**HS 1**
Specifics:	KUSSS & **MOODLE**
	KUSSS for dates
	MOODLE for communication

Course Material

This lecture is based on the book:

Reinforcement Learning: An Introduction
Richard S. Sutton and Andrew G. Barto

Link

(and on the (video-) lectures of David Silver)

Throughout the slides, the book will be referred to simply as “book, chapter X”.

Semester Outlook

1. Introduction (X)
2. Multi-armed Bandits (X)
3. Finite Markov Decision Processes
4. Dynamic Programming (X)
5. Monte Carlo Methods (X)
6. Temporal-Difference Learning 1 & 2 (X)
7. Policy Gradient Methods (X)
8. Deep Reinforcement Learning - a Selection

(X) denotes an accompanying exercise

Exam for the Lecture

- will be on **MOODLE**
- will be **open book**
- is scheduled for **19. January 2023**

Exercises

there will be **complementary practical exercises** to improve the understanding of the material taught in the class — more on that in the exercise appointments themselves.

Communication

all communication will run through **MOODLE**, in particular:

- questions on **lecture** topics
- questions on **exercises**

Communication (in special cases)

You may also write us an email (silvan.peter@jku.at).

Subject:	**[RL2022]** Short description
Body:	Dear Silvan,
	...
	...
	Kind regards,
	Your Name
	MNR: **Your Immatriculation Number**

REINFORCEMENT LEARNING

Informal Description

*"The idea that we **learn by interacting with our environment** is probably the first to occur to us when we think about the nature of learning." — Book, Chapter 1*

Learning to walk

- you randomly move your limbs
- you get up, you stagger, you fall, you observe
- you experience pain and joy
- you want to feel good

Learning to walk (+ foreshadowing)

- you (randomly) move your limbs (**action**)
- you get up, you stagger, you fall, you observe (**state**)
- you experience pain and joy (**reward**)
- you want to feel good (**value**)

First Question on the Exam

Question:

What is Reinforcement Learning? Describe in one sentence.

Answer:

Reinforcement Learning is the computational approach to learning from interaction.

(Slightly) More Formal Problem Definition

Reinforcement learning problems involve learning “what to do”:

- by mapping **states** to **actions**
- while maximizing total **reward** over time

Three Central Characteristics

- **Closed-loop problems**: the actions of an agent *influence its later observations*.
- **No direct instructions**: the agent is not told which actions to take, *it must discover which actions to take*.
- **No initial knowledge**: the agent has to *figure out the consequences of its actions*, as they play out over an extended period of time.

Differences to other ML paradigms

We could **categorize machine learning into three main paradigms**:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Differences to other ML paradigms

We could **categorize machine learning into three main paradigms**:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning (← this course)

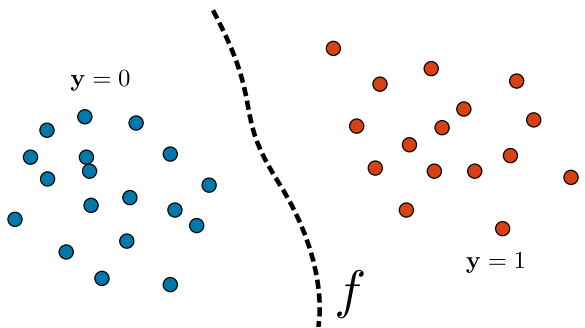
How does Reinforcement Learning differ from these other machine learning paradigms?

Supervised

Learning from a training set.

Given: a training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N} \in \mathcal{X} \times \mathcal{Y}$

Find: a function $f(\mathbf{x}) = \mathbf{y}$



Supervised

Supervised Learning: learning from a training set.

- Labeled examples provided by a supervisor
- Examples $\mathbf{x}$ along with a label $\mathbf{y}$

Challenge: find a $f(\mathbf{x})$ that generalizes to unseen observations that were not included in the training set.

Supervised

Supervised Learning: learning from a training set.

- Labeled examples provided by a supervisor
- Examples $\mathbf{x}$ along with a label $\mathbf{y}$

Difference to Reinforcement Learning:

- Agent must be able to **learn from its own experience**
- There is **no supervisor** that could tell the agent what to do

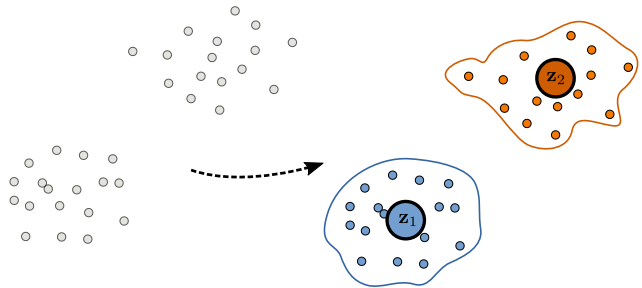
Unsupervised

Find structure in unlabeled data.

Given: a set of observations $\{\mathbf{x}_i\}_{i=1}^N \in \mathcal{X}$

a structuring criterion

Find: a summary $\{\mathbf{z}_c\}_{c=1}^K \in \mathcal{Z}$



Unsupervised

Unsupervised Learning: Find structure in unlabeled data.

- Define structuring criterion
- Define distance measure
- A set of observations $\mathbf{x}$

Challenge: Specify a clustering criterion and distance measure, so the summary $\mathbf{z}$ provides useful insights

Unsupervised

Unsupervised Learning: Find structure in unlabeled data.

- Define structuring criterion
- Define distance measure
- A set of observations $\mathbf{x}$

Difference to Reinforcement Learning:

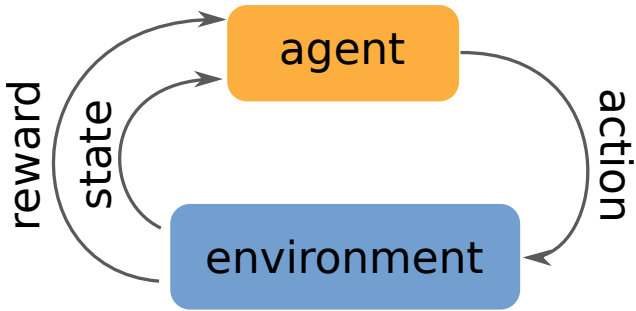
- There is **no predefined** data!
- Agent tries to **maximize reward**

Reinforcement

Maximize reward.

Given: an environment and a reward signal

Find: a behavior that maximizes total reward over time



Reinforcement

Reinforcement Learning: Maximize reward.

- Define an environment and a reward signal.
- Paths through state space $s_1, a_1, r_2, s_2, a_2, r_3, \ldots$

Challenge: Find good actions by exploring, exploit knowledge about good actions to maximize reward.

Difference Summary[1]

	from input $\mathbf{x}$, output:	
Supervised	prediction $\mathbf{y}$	$\rightarrow$ a label
Unsupervised	summary $\mathbf{z}$	$\rightarrow$ some clusters
Reinforcement	action $\mathbf{a}$	$\rightarrow$ a behavior

[1] adapted from http://www.argmin.net/2018/01/29/taxonomy/

Challenge in Reinforcement Learning

How do we **maximize** total reward?
How do we **find** highly rewarded actions?

Exploit

How do we **maximize** total reward?

- we need to **exploit** our knowledge about the past
- we need to take the **currently optimal** action

How do we know what is (currently) optimal?

Explore

How do we **find** highly rewarded actions?

- we need to **explore** the environment
- this means taking **random** actions from time to time

If we take random (suboptimal) actions, do we still maximize total reward?

The Dilemma

There is a special challenge in RL:
A **trade-off** between **exploration** and **exploitation**.

The Dilemma

There is a special challenge in RL:
A **trade-off** between **exploration** and **exploitation**.

- To obtain reward, an agent must favor actions that have proven to be beneficial in the past (**Exploitation**)
- To discover such actions, an agent has to try actions that it has not selected before (**Exploration**)

The Dilemma

There is a special challenge in RL:
A **trade-off** between **exploration** and **exploitation**.

- To obtain reward, an agent must favor actions that have proven to be beneficial in the past (**Exploitation**)
- To discover such actions, an agent has to try actions that it has not selected before (**Exploration**)

The **dilemma** is now, that neither exploration nor exploitation can be pursued exclusively without failing at the task.

EXAMPLES

Example: Gazelle Calf

A gazelle calf struggles to its feet minutes after it is born. Half an hour later it is running at

Example: Chess Player

A chess player makes a move. The choice is informed by **planning**, anticipating possible replies and counter replies, and **immediate, intuitive judgments** of particular positions and moves, assigning different **values** to different moves.

Example: Trash Collecting Robots



A mobile robot decides whether it should search more trash to collect or go recharge its batteries. It **makes its decision** based on the **current charge level** of its battery and **how quickly and easily** it has been able to find the recharger in the past. [2]

[2] Link 1, Link 2, Link 2

Example: Atari Games

Mnih et al. "Human-level control through deep reinforcement learning." Nature, 2015

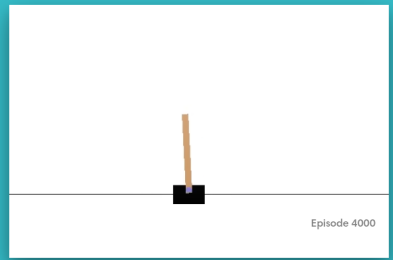
Example: OpenAI Gym

CartPole-v0

A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center.

CartPole-v0 defines "solving" as getting average reward of 195.0 over 100 consecutive trials.

This environment corresponds to the version of the cart-pole problem described by Barto, Sutton, and Anderson [Barto83].



Svalorzen's algorithm, Took 42 episodes to solve the environment.
200.00 ± 0.00 a year ago

The OpenAI Gym (`https://gym.openai.com`) is a toolkit for developing and comparing reinforcement learning algorithms.

Commonalities

- **interaction** between an agent and its environment
- agents seek to achieve a **goal** in their environments, despite **uncertainty** in the environment
- **actions affect the future** state of the environment, in turn affecting future options and opportunities

Commonalities

- optimal choices require taking into account **indirect, delayed consequences** of actions (need foresight and planning)
- consequences of actions can not be fully predicted, so the **agents** have to **frequently monitor** the **environment**
- agents use their **experience** to improve their performance
- **interaction** with the environment **is essential** for adjusting behavior

ELEMENTS OF REINFORCEMENT LEARNING

Environments

Examples for environments:

- the universe
- the streets only
- a coffee machine
- a chess game
- a simple 2D gridworld
- . . .

States

The state **represents** the environment's **current condition**:

- the precise position, mass and velocity of all atoms
- multiple images of the street surrounding a car
- water temperature and pressure in a coffee machine
- chess board, together with pieces
- the position in a simple 2D gridworld
- . . .

Actions

For each state, there is a **set of actions** that the agent can **choose** from:

- attend this lecture
- turning the steering wheel
- increase / decrease current to the heating element
- select a piece and make a legal move
- move in a particular direction on the grid
- . . .

Policy

An agent's policy **completely** determines its behavior:

- "avoid doing what you would blame others for doing"[3]
- if the car leaves the street, steer in the other direction
- if pressure too high, decrease current to heating element
- if a piece is threatened, move to protect it
- move to a position with high expected future reward
- ...

[3] Thales, somewhen between 624 and 546 BCE (and many others)

Reward

Reward is given out from the environment, and encodes how good the agent is doing currently.

- how happy we are, or if we are in pain
- car stays on the street, does not crash
- coffee is drinkable, pressure vessel still intact
- winning or losing the game
- move cost / abstract desirable / undesirable states
- . . .

Value Functions

A value function is compressed knowledge about the future, it encodes an agent's experience.

- eat cake, feel good
- turn the steering wheel D degrees, avoid a crash
- stop heating at temperature T, no explosion, good coffee
- making this move now will mate in M moves
- move to a state with more expected future reward
- ...

Four main elements

- Policy
- Reward Signal
- Value Function
- (Model of the Environment)

Policy

The behavior of an agent is called its **policy**.

The policy:

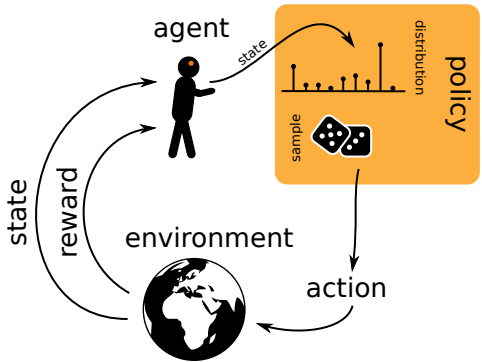
- maps a **state** to a probability distribution over **actions**
- **samples** from this distribution, to select the **action**

We can say that an RL agent:

- follows a **policy** (it behaves a certain way)
- updates its policy (to try and **maximize reward**)

Policy

The **policy** completely determines its behavior, deciding which **action** to take, based on the current **observation** of the **state**.



Policy Implementation

The policy is a **mapping** from perceived **states** of the environment to **actions**. It could be implemented as:

- a **lookup-table**
- a simple **function**
- a **search process**
- a **deep neural network**
- . . .
- a combination of all of the above

The policy is **sufficient** to determine behavior.

Reward Signal

Defines the **goal** in a RL problem. The reward:

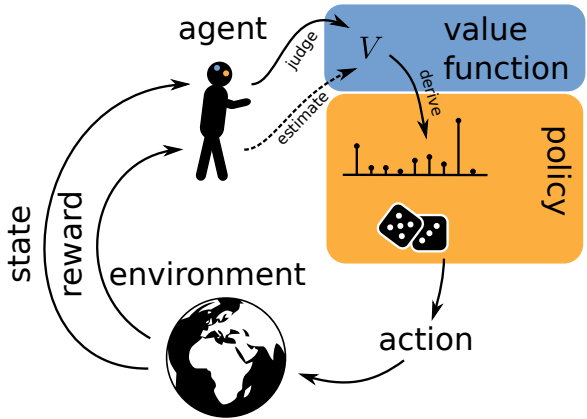
- is a **single, real number**
- is perceived by the agent **at each time step**
- defines what is **good** and what is **bad** for the agent
- is **immediate** and defines features of the problem
- is **the primary basis** for changing the policy

Value Function

Defines what is **good** in the **long run**. The value of a state:

- is the **total amount** of **reward** an agent can **expect** to **accumulate** in the **future**, starting from that state
- is usually an **estimate**
- is often used to **choose an action**

Value Function



Reward vs. Value

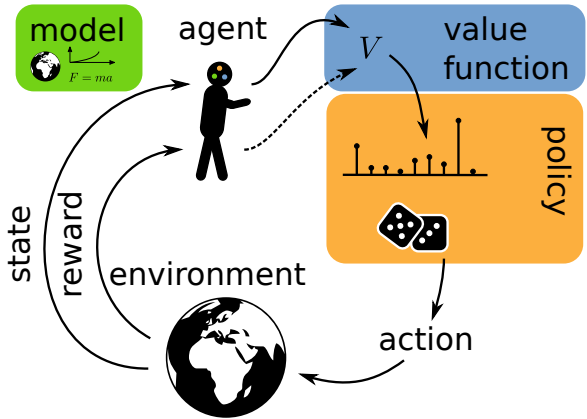
- rewards are **immediate**, part the environment
- values are predictions of **future reward**, part of the agent
- rewards are **used**, in order to estimate value
- the only purpose of estimating values is to **get more reward** in the long run
- most RL algorithms focus on **efficient value estimation**
- a state might yield a low immediate reward but still have a high value because it is usually followed by beneficial states

Model of Environment

Mimics the behavior of the environment.

- some RL methods are **model-based**
- others are simpler **model-free** methods: trial and error learners
- models can be used for **planning**: given state and action the model predicts next state and next reward.

Model of Environment

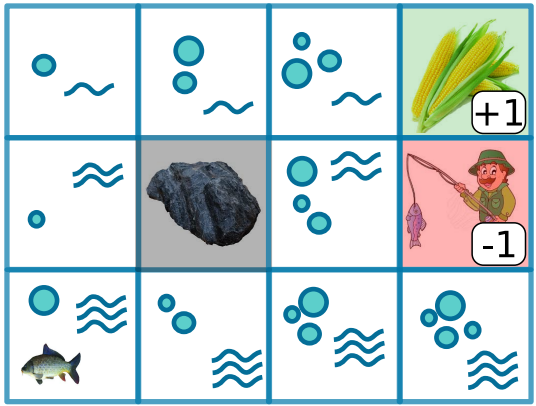


Boundaries?

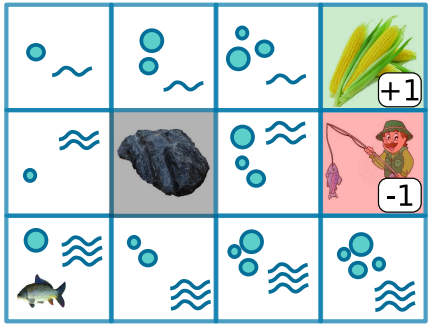
Quick Question: where is the **boundary** between the agent and the environment?

GET FED, NOT EATEN

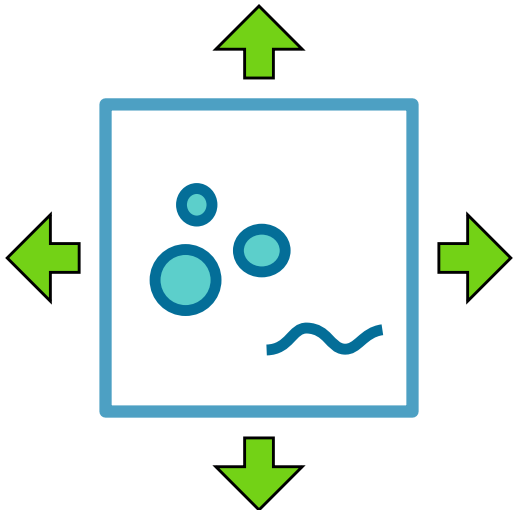
Gridworld Environment



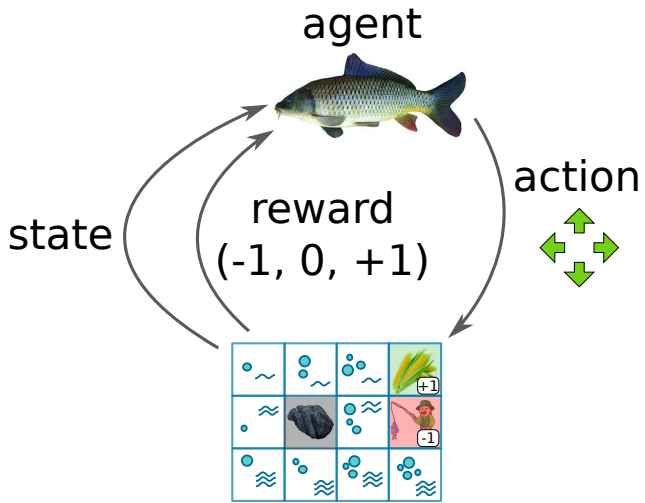
States



Actions



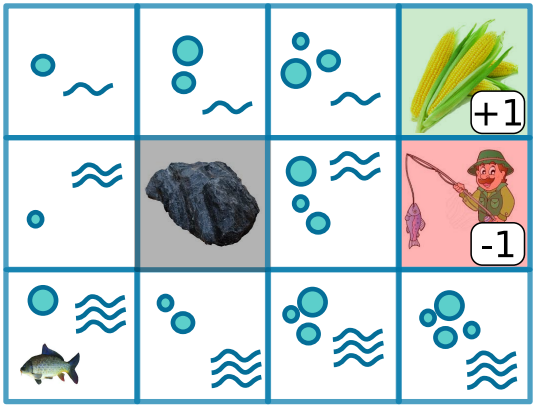
The Loop



How to get food and survive?

Approaching the problem using a value function:

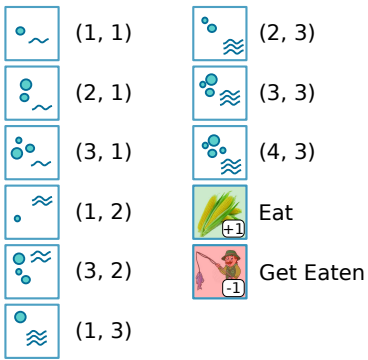
1. Set up a table of numbers, one row for each possible world state.



How to get food and survive?

Approaching the problem using a value function:

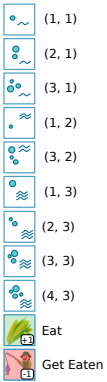
1. Set up a table of numbers, one row for each possible world state.



How to get food and survive?

Approaching the problem using a value function:

1. Set up a table of numbers, one row for each possible world state.



How to get food and survive?

Approaching the problem using a value function:

1. Set up a table of numbers, one row for each possible world state.

STATE	VALUE
 (1, 1)	0
 (2, 1)	0
 (3, 1)	0
 (1, 2)	0
 (3, 2)	0
 (1, 3)	0
 (2, 3)	0
 (3, 3)	0
 (4, 3)	0
 Eat	1
 Get Eaten	-1

How to get food and survive?

Approaching the problem using a value function:

1. Set up a table of numbers, one row for each possible world state.
2. Estimate: **state value** V (use a table as a **value function**).

How to get food and survive?

Approaching the problem using a value function:

1. Set up a table of numbers, one row for each possible world state.
2. Estimate: **state value** V (use a table as a **value function**).
3. Initialize value function (eat: 1, get eaten: -1, else: 0)

How to get food and survive?

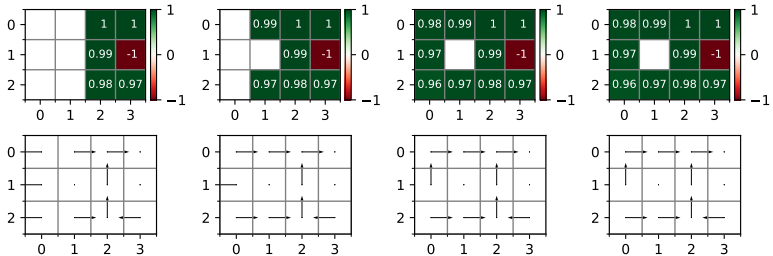
Approaching the problem using a value function:

1. Set up a table of numbers, one row for each possible world state.
2. Estimate: **state value** V (use a table as a **value function**).
3. Initialize value function (eat: 1, get eaten: -1, else: 0)
4. Move, eat or die, repeat. Collect experience. Learn.

How to get food and survive?

Approaching the problem using a value function:

1. Set up a table of numbers, one row for each possible world state.
2. Estimate: **state value** V (use a table as a **value function**).
3. Initialize value function (eat: 1, get eaten: -1, else: 0)
4. Move, eat or die, repeat. Collect experience. Learn.



How to get food and survive?

Approaching the problem using a value function:

1. Set up a table of numbers, one row for each possible world state.
2. Estimate: **state value** V (use a table as a **value function**).
3. Initialize value function (eat: 1, get eaten: -1, else: 0)
4. Move, eat or die, repeat. Collect experience. Learn.
5. **Select** an **action** based on the **value** of the **next state**:
   - Most of the time we select **greedily**, choosing the move that leads to the state with the **greatest value** - we are **exploiting** our collected knowledge.
   - Occasionally we select **randomly** among actions - we are **exploring** and collect new knowledge.

Gridworld - Learning

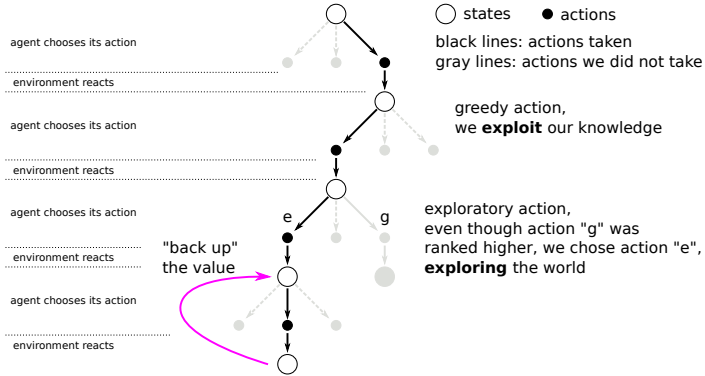


Figure: Sequence of moves in the gridworld

Gridworld - Learning

While we are navigating the waters of the gridworld over and over again, we change the values of states visited during the game. These value estimates should become more accurate estimates of how much **cumulative reward** we expect in the **future**.

Gridworld - Learning

While we are navigating the waters of the gridworld over and over again, we change the values of states visited during the game. These value estimates should become more accurate estimates of how much **cumulative reward** we expect in the **future**.

Procedure:

- "back up" value of the state after a greedy action to value before the action. The current value $V(s)$ of the earlier state s is adjusted to be closer to the value $V(s')$ of the later state s'.
- $V(s) \leftarrow V(s) + \alpha \left[V(s') - V(s) \right]$
- This update rule (using *step-size* / *learning rate* α) is an example of **temporal-difference learning**.

Gridworld - Learning

Example outlines key features of RL methods:

- Emphasis on learning while interacting with an environment.
- There is a clear goal, defined by the reward signal.
- Correct behavior requires planning and taking into account delayed effects of choices through backups of the value function estimates.

SUMMARY

Summary

You should now be familiar with ...

- the concept of RL which is the **computational approach to learning from interaction**.
- its **distinction** to supervised and unsupervised learning.
- the trade-off between **exploration** and **exploitation**.
- the terms Agents, Actions, Environment, States and Rewards.
- the definition of Policy, Reward, Values and Models of the Environment.