Reinforcement Learning
Lecture 1

Georg Martius

Distributed Intelligence / Autonomous Learning Group, Uni Tübingen, Germany

October 15, 2024

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Organizational structure of the lecture

- Lecturer: Prof. Georg Martius
  - 29.10.2024: by Dr. René Geist
  - 12.11.2024: by Dr. Claire Vernade
- Teaching language is English, although you can ask questions in German
- Tuesdays 14:15 – 15:45 Lectures
- Tuesdays 16:00 – 17:30 (or 16:15 – 17:45) Recitations/Übungen

Tutors:

Pavel Kolev

René Geist

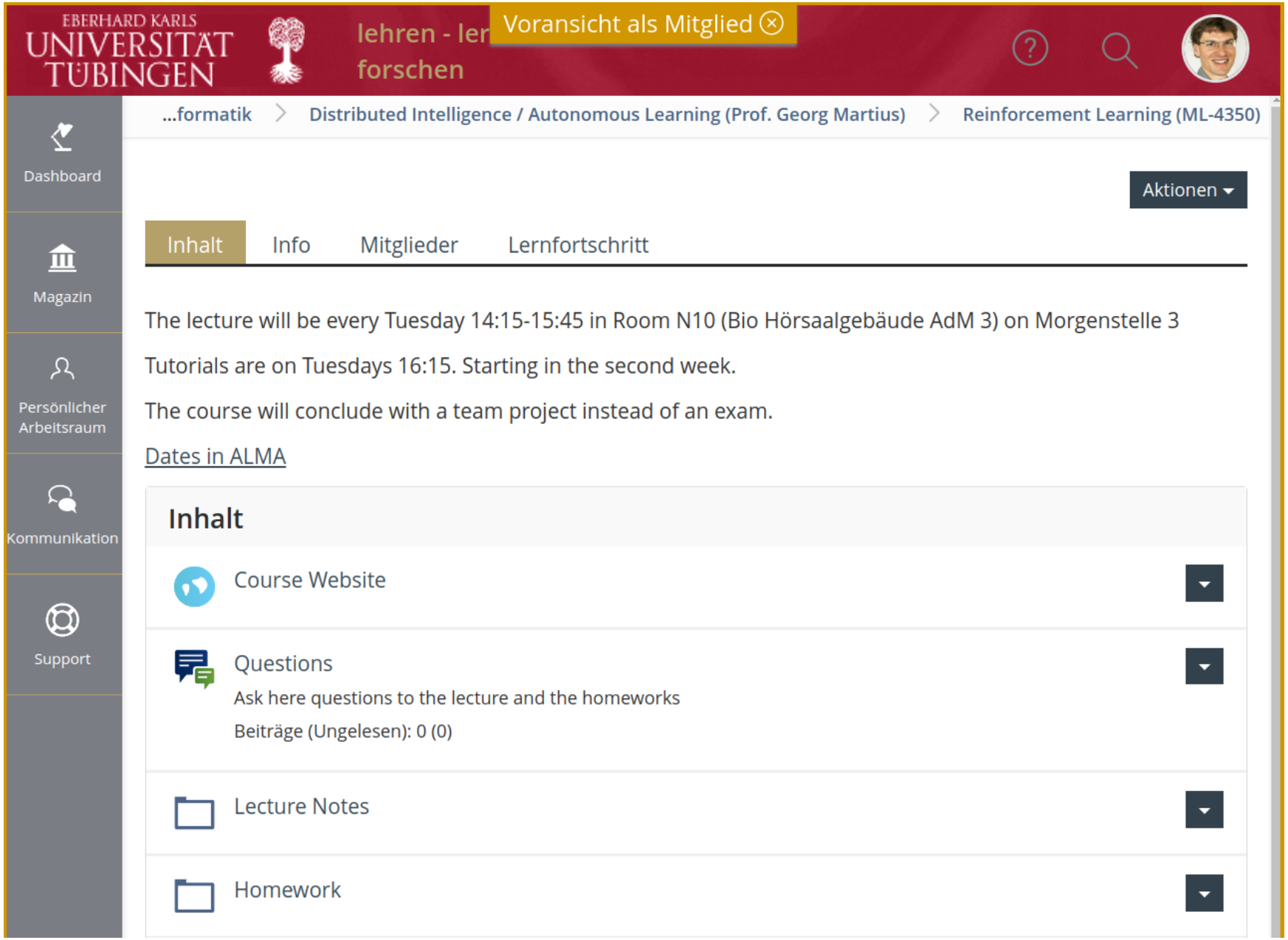
Mikel Zhobro

Pierre Schumacher

Organizational structure of the lecture

- Assignments:
  - homework sheets have to be uploaded in the following week to ILIAS
  - teams with max 3 members are allowed
  - no individual feedback per homework is provided ($\rightarrow$ recitations sessions)
  - at least 5 out of 9 sheets are required for passing the course
  - passing means: having demonstrated non-negligable work on the sheet
- Lecture notes: slides and background material to read
- Exam equivalent: final project with a short report + program performance + mini-presentation (also in teams with up to 3 members)
  24.02.2025 deadline for program / start of competition
  25.02.2025 report + presentation video due
  kein Nachprüfungstermin, later hand-in only with doctor's certificate
- Webpage:
  `https://uni-tuebingen.de/de/271554`
  (linked from ALMA, ILIAS, and Uni Webpage)
- Registration: register latest until 29th of October in ILIAS

ILIAS – place for material and homeworks



About me

- 2005 Diploma in Computer Science
  - studied in Leipzig and Edinburgh
- 2009 PhD in Computational Neuroscience/Robotics, Göttingen
- Post-docs at Max-Planck-Institutes for
  - Dynamics and Self-organization (Göttingen)
  - Mathematics in the Science (Leipzig)
- 2015 Fellow at IST-Austria
- 2017 Max-Planck Group Leader
  @MPI for Intelligent Systems in Tübingen
- 2023 Prof. at Uni Tübingen

Leipzig

Research interests: Machine learning for robotics, autonomous learning, haptic sensing, optimization, …

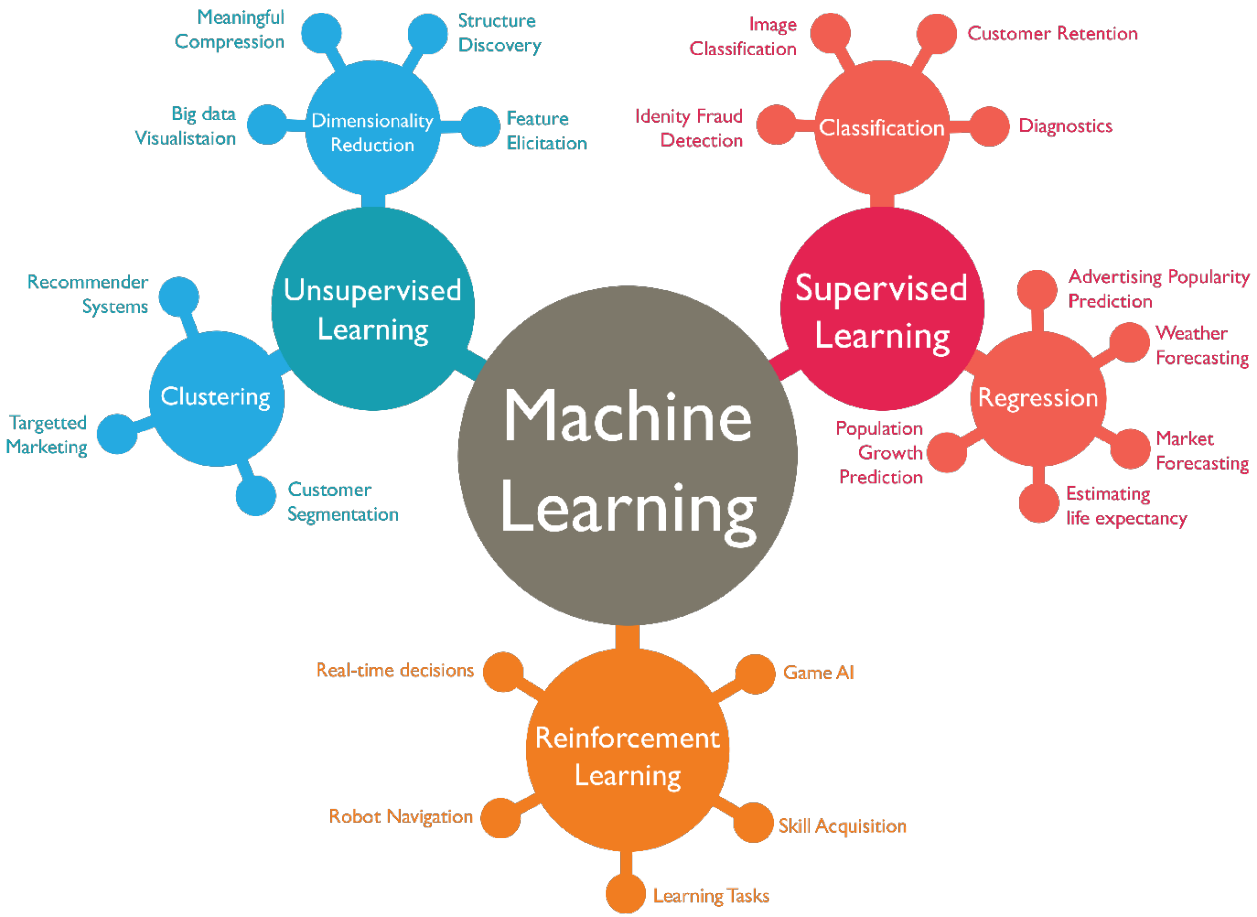
Plan for today

- Overview of machine learning
- Synopsis of the course
- Motivation and history of Reinforcement Learning (RL)
- Introduction to RL
- Markov Decision Processes (start)

Machine Learning Overview

Machine learning is not voodoo,
it is about automatically finding a function that best solves a given task.

Three different classes of tasks:



Supervised Learning

given: pairs of data (x, y)
with data point $x \in \mathbb{R}^n$ and label $y \in \mathcal{Y}$ coming from a data distribution $\mathcal{D}$
What to find function $h(\cdot)$ such that

$$h(x) = y \quad \forall (x, y) \sim D$$

To measure quality of h and to be able to optimize something:
Define loss function:

$$J(h) = \mathbb{E}_{\mathcal{D}}[\text{dist}(y, h(x))]$$

dist: distance between true label y and predicted label $h(x)$

Task: find function that minimized loss: $h^* = \arg\min_h J(h)$

Math can be so easy ;-)

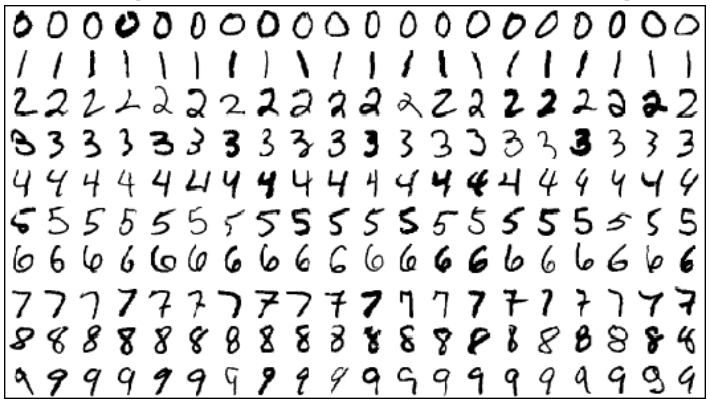
This is not so easy in practice.

Supervised Learning – Examples

Classification: $\mathcal{Y}$ is discrete

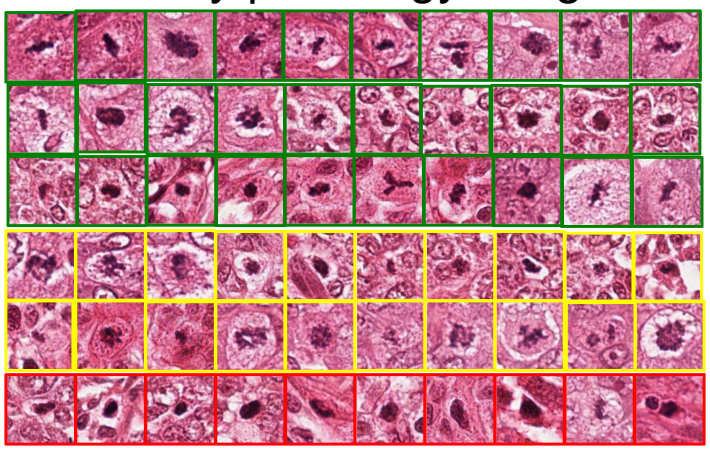
Examples:

Recognize handwritten digits:



(MNIST)

Classify pathology images:

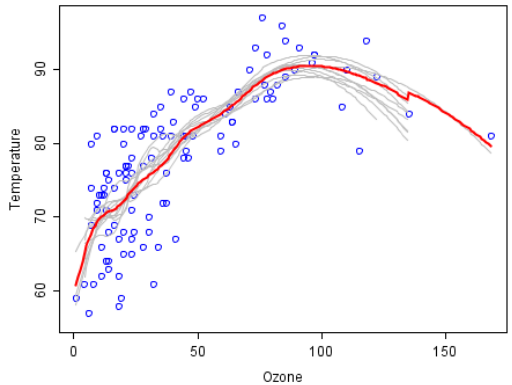


(Mitosis in breast cancer)

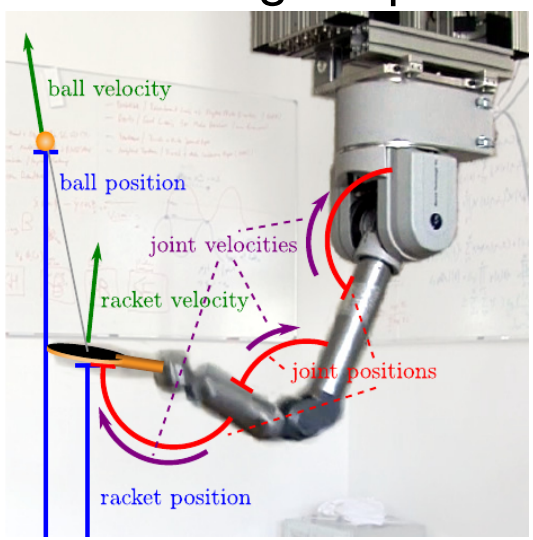
Regression: $\mathcal{Y}$ is continuous

Examples:

Predicting Ozon levels

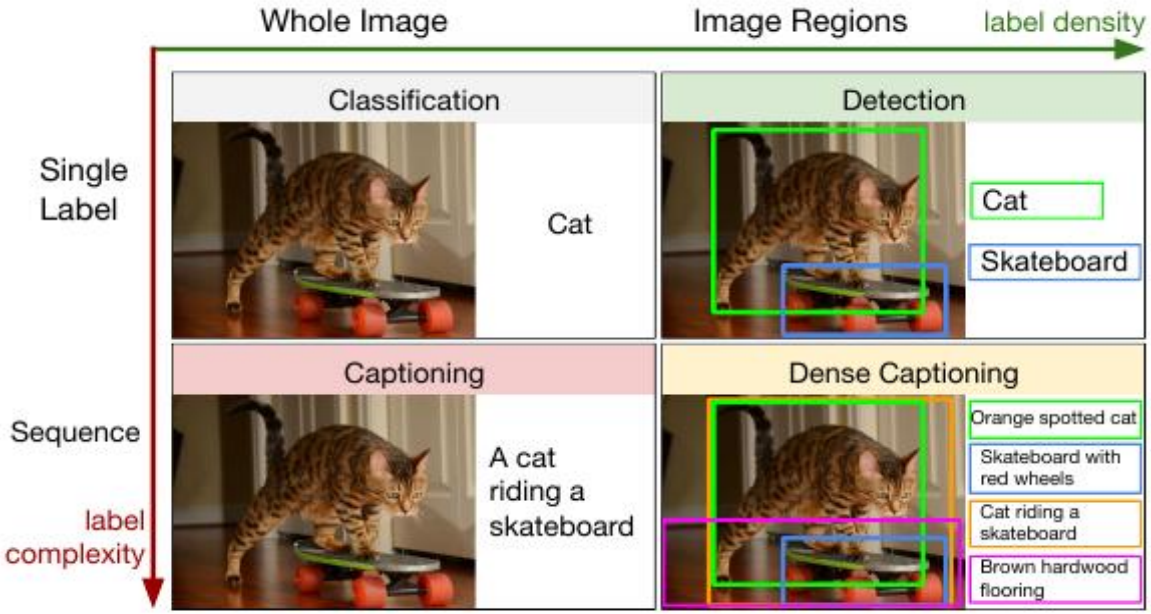


Predicting torques



Supervised Learning – Examples

Image recognition and captioning: data of images and captions creates automatic image captioning function



DenseCap, Johnson et al 2015

Unsupervised Learning

given: data points (x) with $x \in \mathbb{R}^n$
What to find function $h(\cdot)$ such that $h(x) = y$ where y low dimensional,
e.g. a cluster number

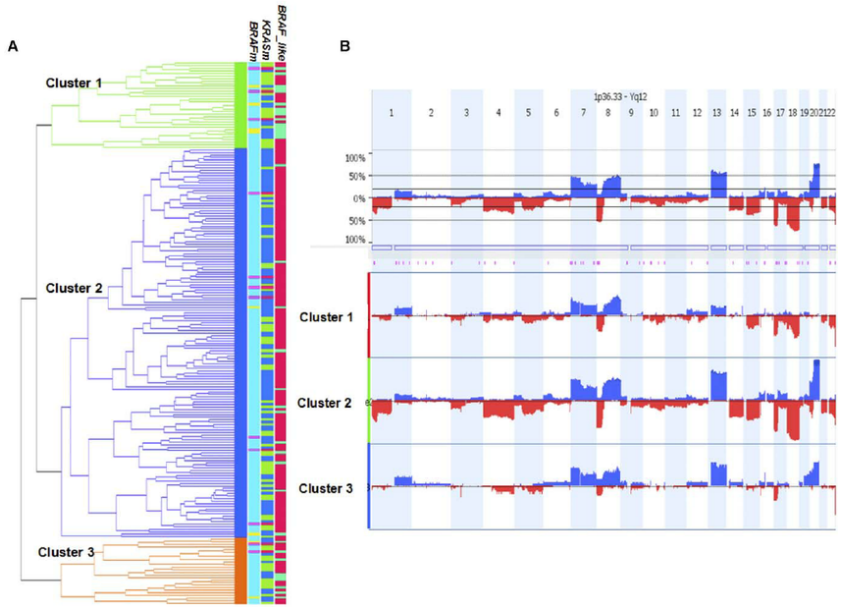
- Much less clear what is the objective.
- Many algorithms but no unifying theory.

Unsupervised Learning – Examples

Clustering: discrete y

Examples:

Genome comparison:



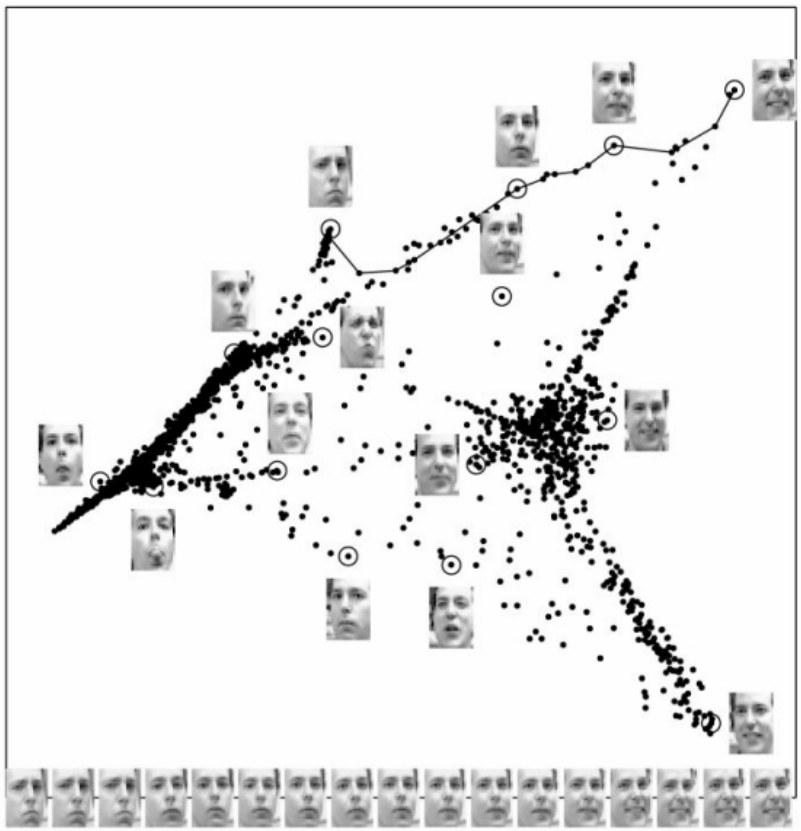
(by Tao Xie)

Both cases are expecially useful for high-dimensional data

Dim. reduction: continuous y

Examples:

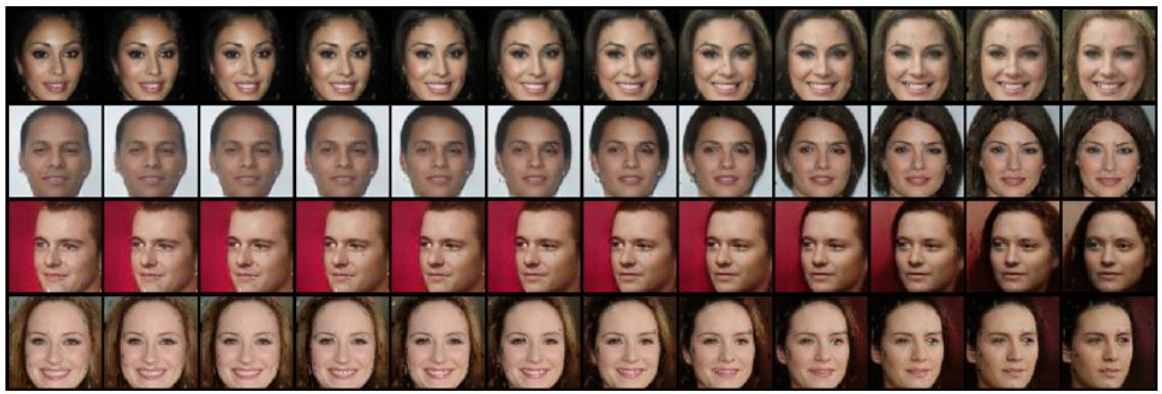
Finding descriptors for face expressions



(by Sam T Roweis)

Unsupervised Learning – Examples

Finding disentangled generating factors: Take images of faces and find few discriptive variables. (can also generate new images)



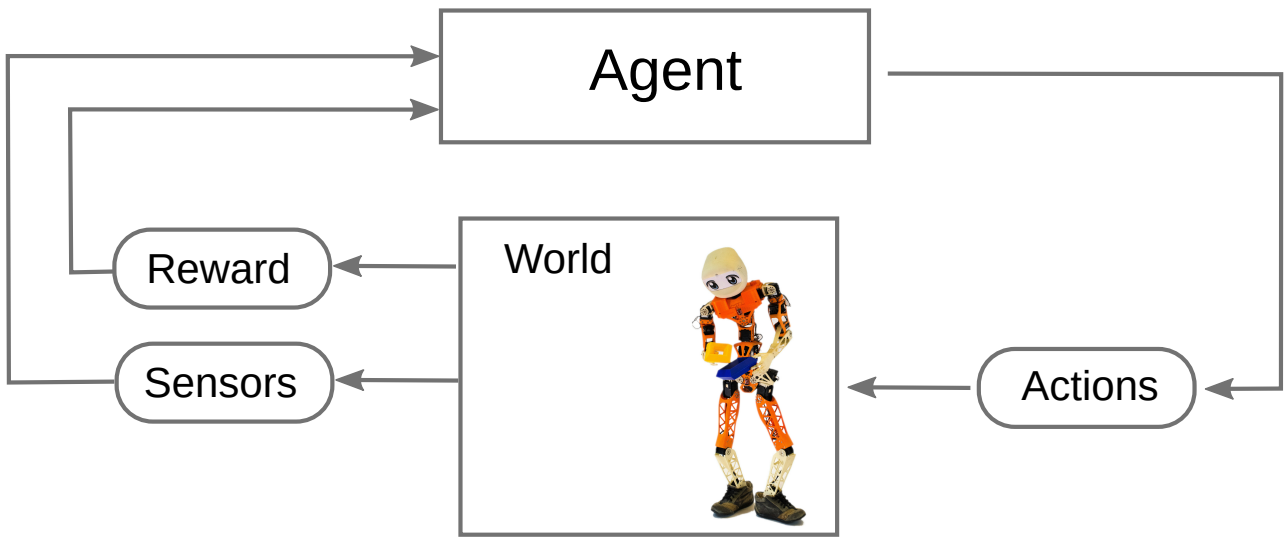
Khan et al 2018

Reinforcement Learning

given:

- system to interact with: $s_{t+1} = \mathrm{Sys}(a_t, s_t, t)$ where s_t is the state and a_t is the action.
- reward/utility function: $r_t = U(a_t, s_t)$

What to find function $\pi(\cdot)$ (policy) such that $a = \pi(s)$ and expected reward ($\mathbb{E}[r]$) is maximized (simplified).



Reinforcement Learning

given:

- system to interact with: $s_{t+1} = \mathrm{Sys}(a_t, s_t, t)$ where s_t is the state and a_t is the action.
- reward/utility function: $r_t = U(a_t, s_t)$

What to find function $\pi(\cdot)$ (policy) such that $a = \pi(s)$ and expected reward ($\mathbb{E}[r]$) is maximized (simplified).

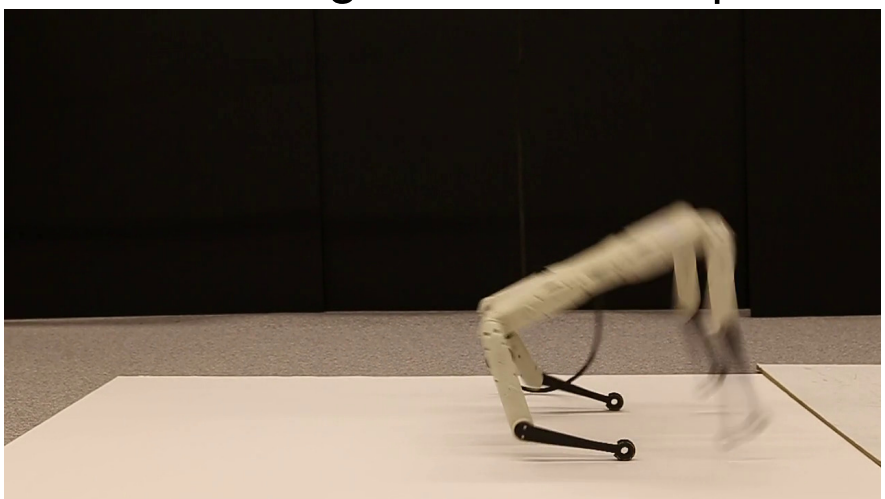
In general: stochastic systems formulated as Markov Decision Processes.

- Need to simultaneously learn π and potentially models of Sys and U.
- Reward can be sparse
  (e.g. only at the end of an long action sequence)

Reinforcement Learning – Examples

Robot Control

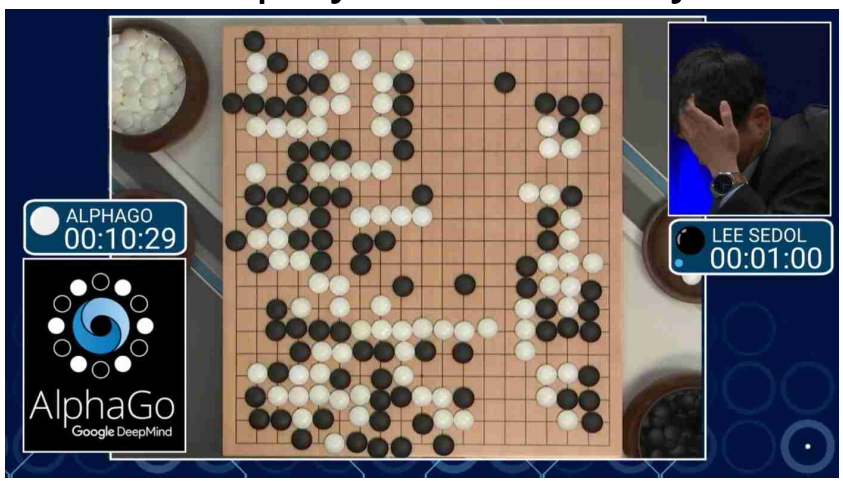
Learning to do a backflip



(my Lab 2022)

Deepmind AlphaGo

Learn to play Go extremely well



(go-baduk-weiqi.de)

Improve performance by learning from experience
and exploring new strategies.

Plan of the course

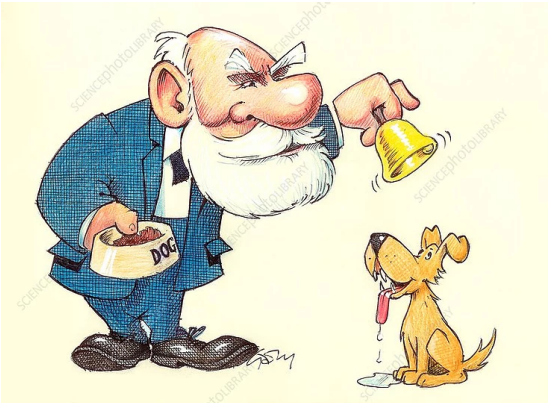
- Reinforcement learning basics
  - Markov Decision Processes (MDPs) and background
  - Bellman equations, Value functions, TD learning, Q-Learning, . . .
  - Exploration vs exploitation
- Reinforcement learning in large and/or continuous spaces
  - Actor-Critic
  - Reinforcement Learning with parametrized policies
  - Advanced topics: goal-based RL, intrinsic motivation etc.
  - Optimal control/trajectory optimization and model predictive control
  - Model-based RL
  - Off-line RL

Why Reinforcement Learning

- Many interesting systems are agents:
  - Animals and humans
  - Machines: in particlar robots
  - Societies
  - ...
- Decision making problem: performing actions to obtain favorable outcome in the future
- Many real-world problem are decision making problems:
  - control of systems: autonomous robots, autonomous driving, nuclear fusion reactors, ...
  - playing games
  - law-making
  - investments
- Perhaps the most natural setting to study self-learning systems and intelligence

Main directions

Conditioning



Ivan Pavlov, 1920ies

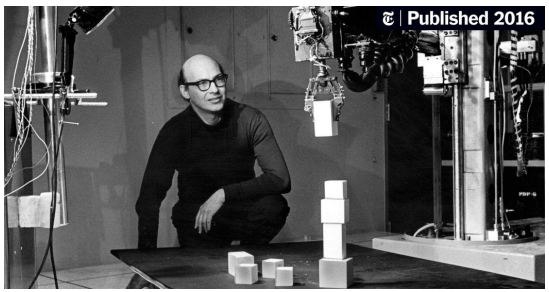
Temporal Difference

Artur Samuel, 1956

Optimal Control

Richard E. Bellman, 1953

Trial and Error Learning



Marvin Minsky, 1961

History

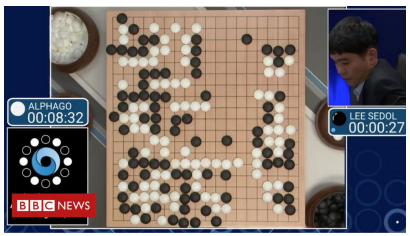
- 1977 Witten: Temporal Difference Learning Algorithm
- 1981 Sutton & Barto: Actor-Critic Architecture


- 1991 Chris Watkins (with Peter Dayan): Q-learning
  Merge of temporal difference learning, trial and error and optimal control


[Image: ualberta.ca, umass.edu,rhul.ac.uk,mpg.de]

History

- 1991 Gerald Tesauro: TD-Gammon: AI to master Backgammon
- 2013 Playing Atari games with Deep Reinforcement Learning
- 2016 Deepmind's AlphaGo defeats Ke Jie (#1 Go player in the world)
- 2019 OpenAI: Shadow-Hand learns to manipulate a cube
- 2021 ETH: 4-legged robot learns versatile and robust locomotion

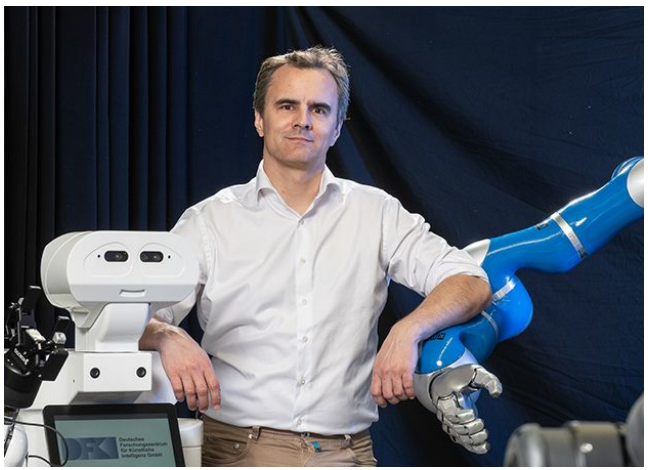


[Image: wikipedia.com,deepmind.com,openai.com,ethz.ch]

Research directions and some researchers

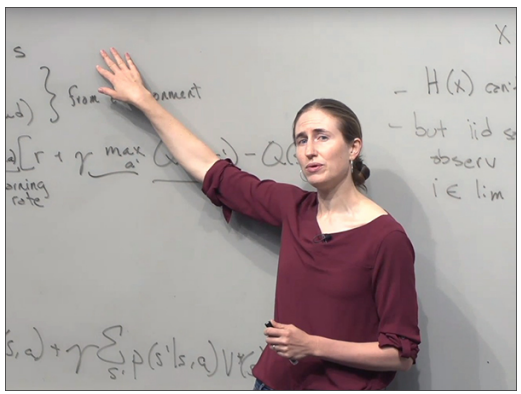
- robot control
- autonomous driving
- safety critical environments
- games
- human computer interaction
- meta-learning
- biological reinforcement learning
- ...

Katja Hofmann @ MS-R



Jan Peters @ TU-Darmstadt

C. Finn, S. Levine and P. Abbeel

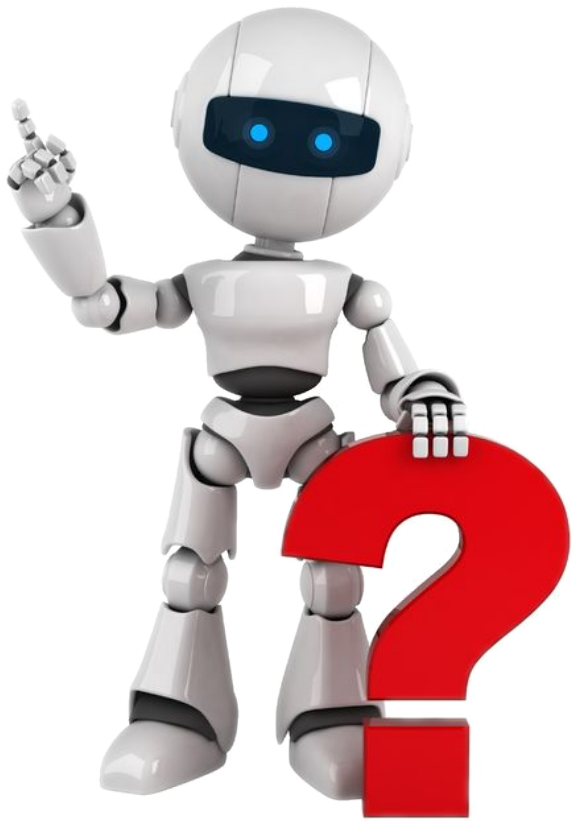


Emma Brunskill @ Stanford

[Image: nytimes.com, acm.com, cs.washington.edu,dfki,de]

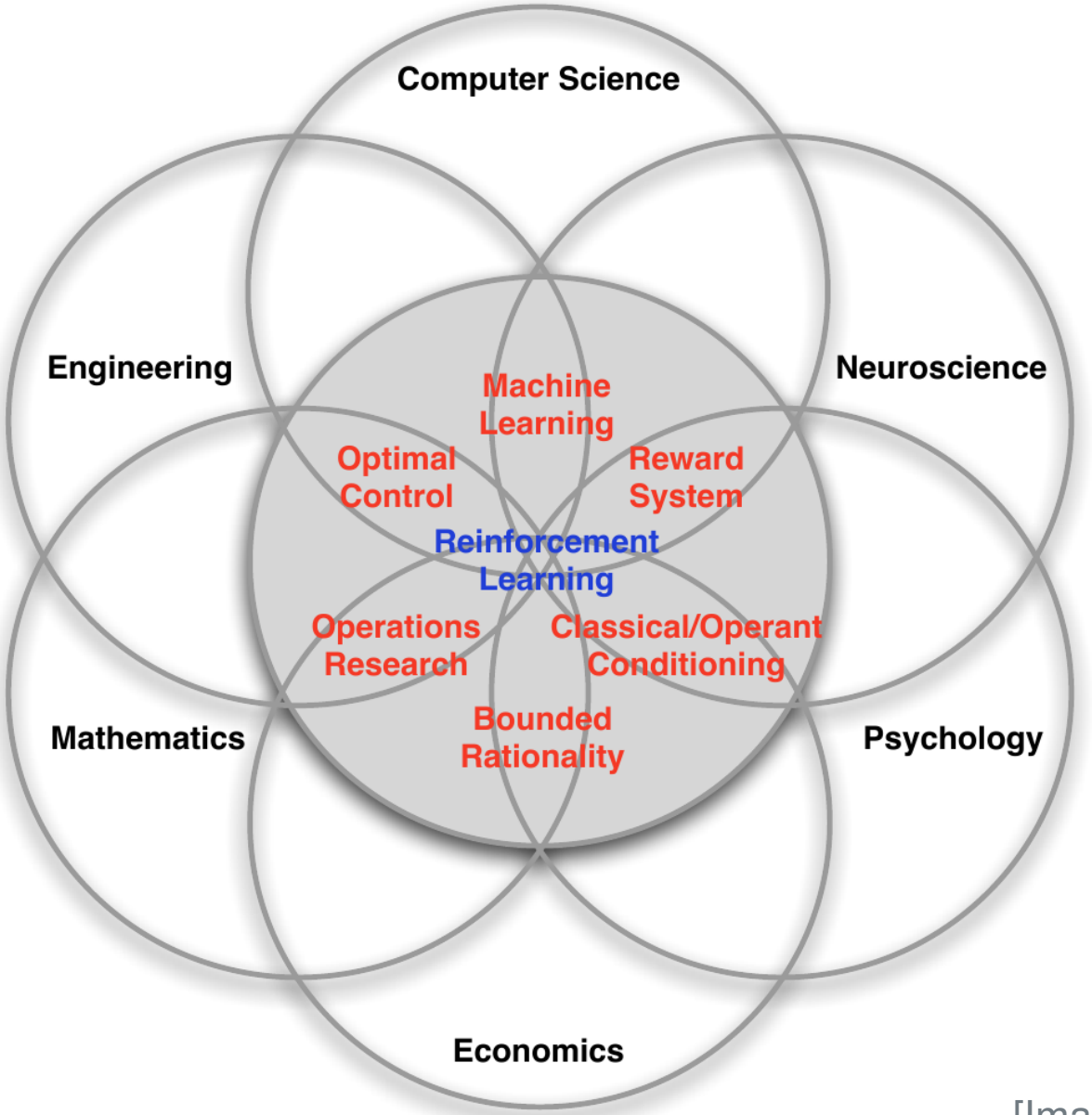
What interests you the most?

Discuss with your neighbors for 3 minutes.



[Image: globalrobots.com]

Many Types and Areas of Reinforcement Learning



[Image: David Silver]

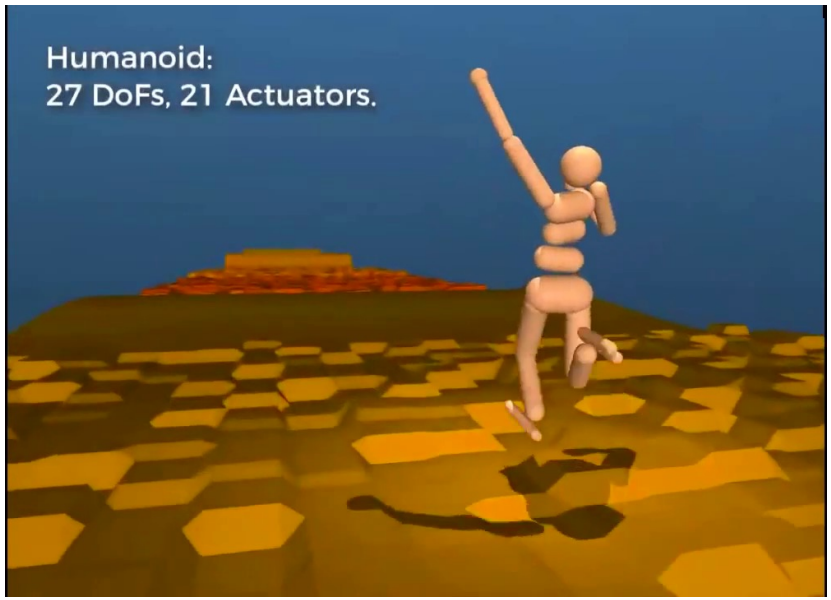
Features of Reinforcement Learning

- data is generated / influenced by the Agent's actions
  - in contast to supervised and unsupervised machine learning
- only indirect supervision: reward signal
- feedback is delayed, not instantaneous (attribution needs to be made)
- time really matters: sequential, non i.i.d.[1] data

[1] independently and identically distributed

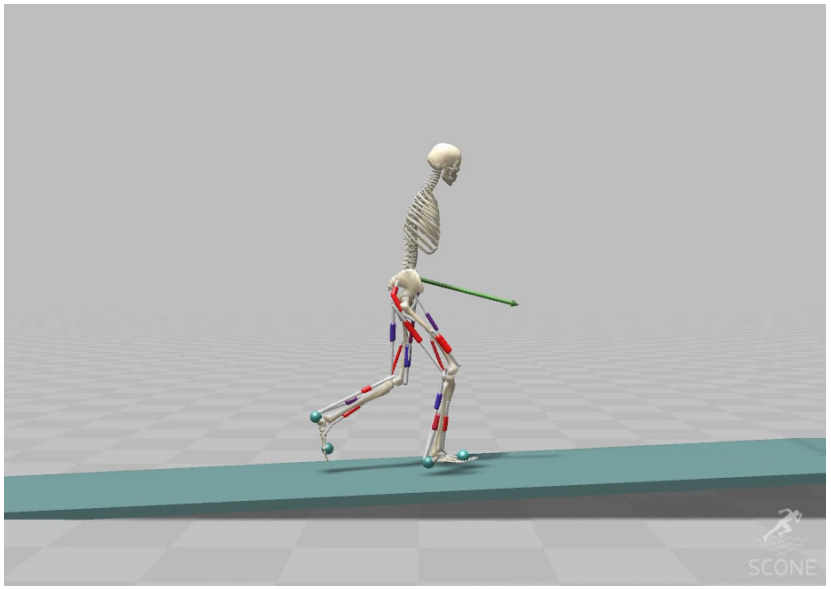
Reinforcement Learning – Examples

Learning to run



Deepmind: Heess et al 2017

Learning to run with muscles



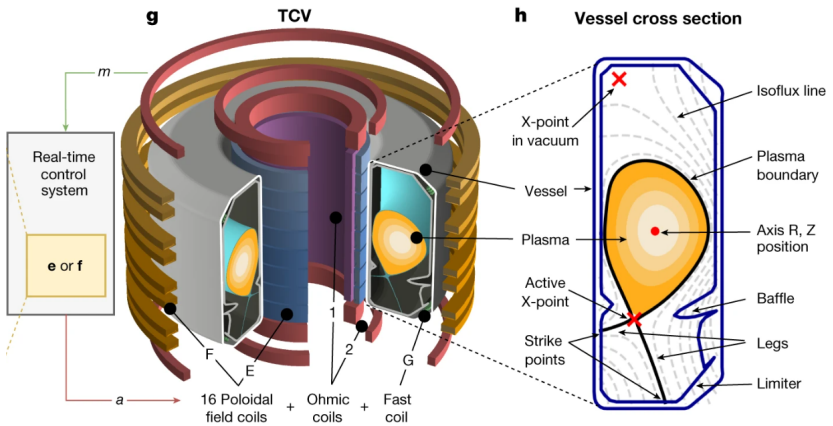
My lab: Schumacher et al 2022

Reinforcement Learning – Examples

Learning to walk in wild terrain

Hutter lab, ETH, 2021

Control plasma in fusion reactor



Deepmind + EPFL, 2022

Reinforcement Learning – Examples

Computer games: ATARI

https://www.youtube.com/watch?v=Vr5MR5lKOc8

Computer games: Starcraft II

https://www.youtube.com/watch?v=jtlrWblOyP4

- Making trading decisions
- many more...

Rewards

- A reward r_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximize cumulative reward

Reinforcement learning is based on the reward hypothesis

Definition (Reward Hypothesis)

All goals can be described by the maximization of expected cumulative reward

Do you agree with this statement?

[Slide adapted from David Silver]

Rewards in our Examples

- Make a robot walk
  - +ve reward for moving with desired speed
  - -ve reward for falling over
- Playing Go
  - +/-ve reward for winning/losing a game
- Control of plasma in tokamak
  - +ve reward for high plasma density in center
  - -ve reward for plasma wall collision, exceeding safety thresholds...
- Play computer games
  - +/-ve reward for increasing/decreasing score

Sequential Decision Making

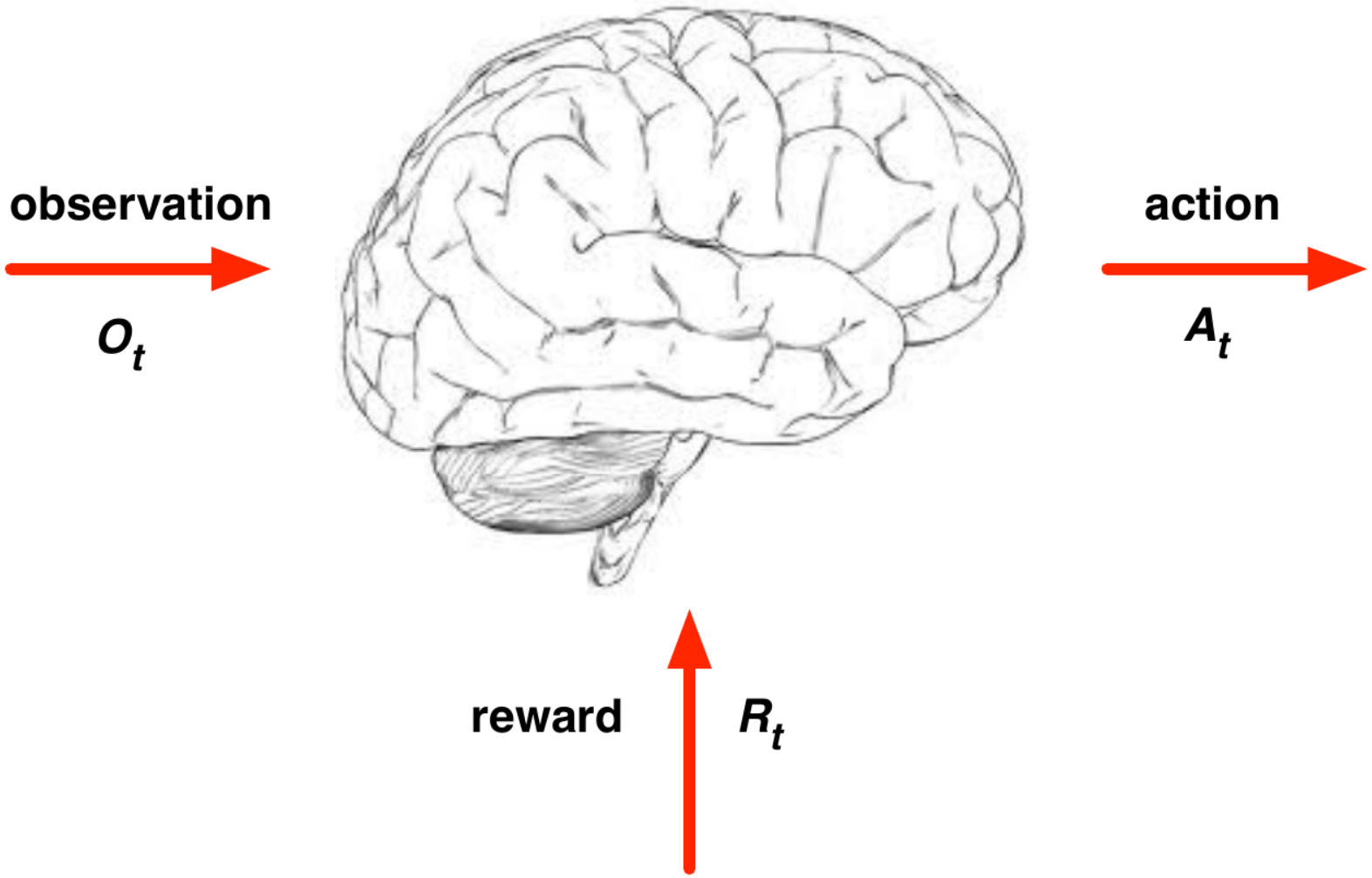
- Goal: select actions to maximize total future reward
- Actions may have long term consequences
- Reward may be delayed
- It may be better to sacrifice immediate reward to gain long-term reward
- Examples:
  - Robot control: correction step avoids falling many timesteps later
  - Games: Blocking opponent moves might increase winning chances many moves later

Short Break

Talk to you neighbors about:

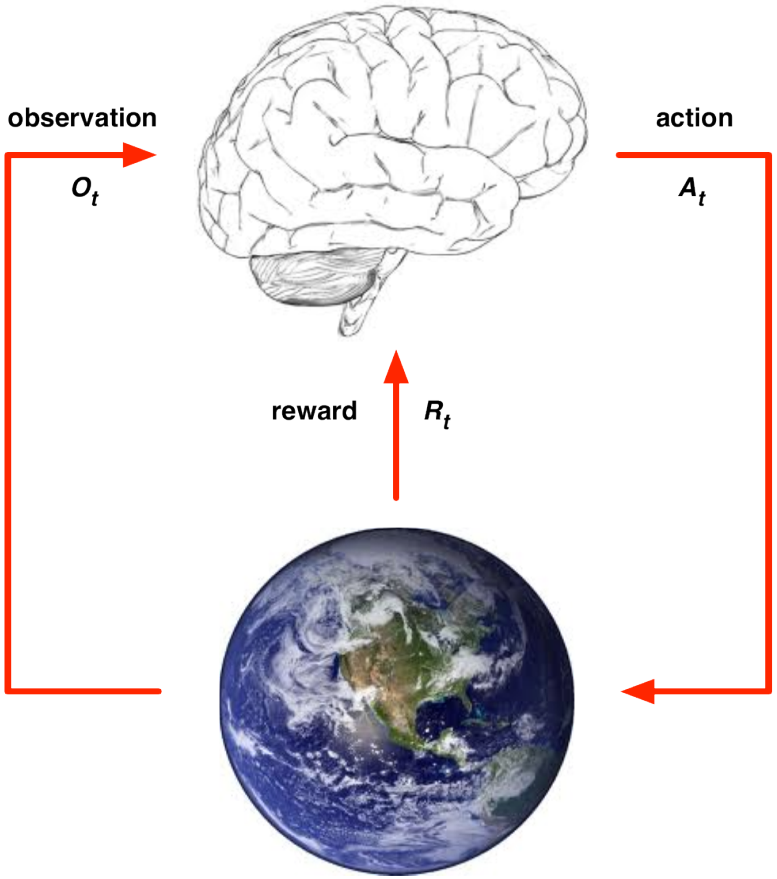
- where do they come from
- what did they do during the summer break
- ...

Agent and Environment



[Slide adapted from David Silver]

Agent and Environment



- At each step t the agent:
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at env. step

[Slide adapted from David Silver]

History and State

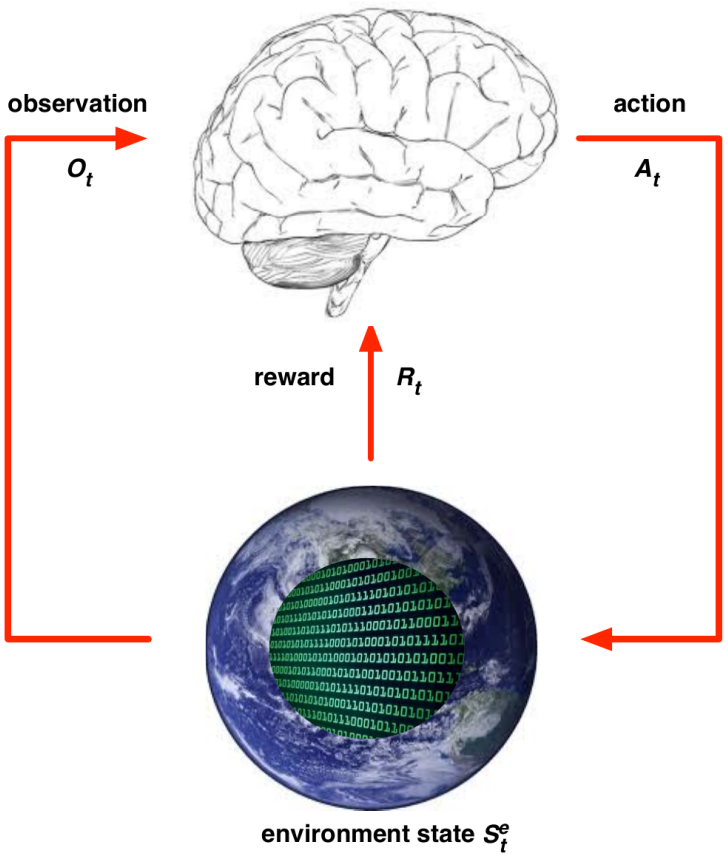
- The history is the sequence of observations, actions, rewards

$$H_t = O_1, R_1, A_1, ..., A_{t-1}, O_t, R_t$$

- i.e. all observable variables up to time t
- i.e. the sensorimotor stream of a robot or embodied agent
- What happens next depends on the history:
  - The agent selects actions
  - The environment selects observations/rewards
- State is the information used to determine what happens next
- Formally, the state is a function of the history:

$$S_t = f(H_t)$$

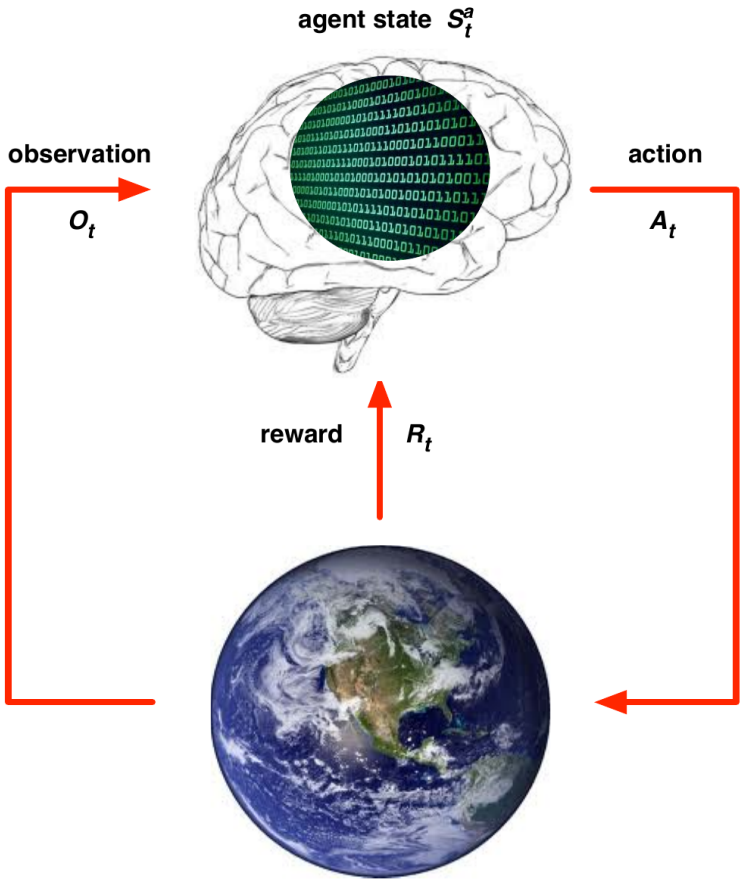
Environment/World State



- The environment state S_t^e is the environment's private representation
- i.e. whatever data the environment uses to pick the next observation/reward
- The environment state is not usually visible to the agent
- Even if S_t^e is visible, it may contain irrelevant information

[Slide adapted from David Silver]

Agent State



- The agent state S_t^a is the agent's internal representation
- i.e. whatever information the agent uses to pick the next action
- i.e. it is the information used by reinforcement learning algorithms
- It can be any function of the history:

$$S_t^a = f(H_t)$$

[Slide adapted from David Silver]

Stochasticity and Notation

We are considering the most general case where all processes are stochastic.

- S_t is random variable and s_t is its realization (actually observed state)
- So $P(S_{t+1} \mid S_t = s_t, A_t = a_t)$ refers to the probability distribution over states when the system as in state s_t and action a_t was performed

Information State

An information state (a.k.a. Markov state) contains all useful information from the history.

Definition

A state S_t is Markov if and only if

$$P(S_{t+1} \mid S_t) = P(S_{t+1} \mid S_1, ..., S_t)$$

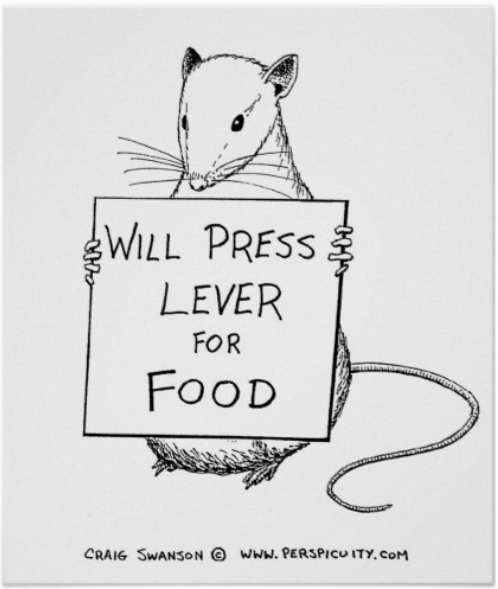
- The future is independent of the past given the present

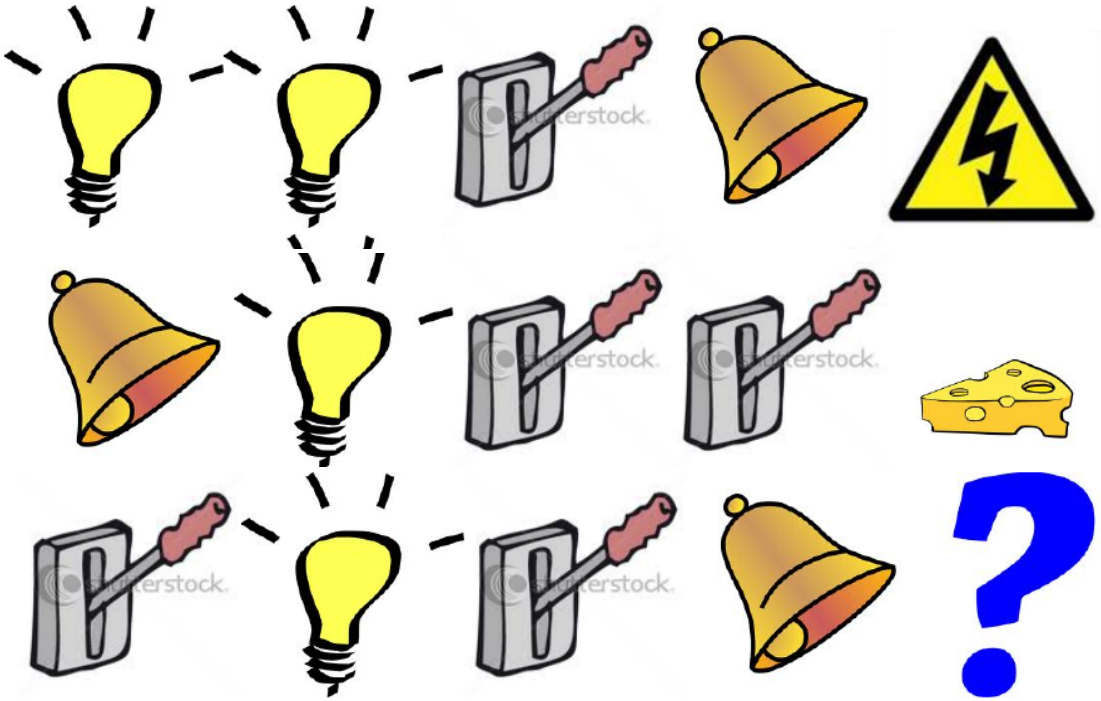
$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty}$$

- Once the state is known, the history may be thrown away
- The environment state S_t^e is Markov
- The history H_t is Markov

[Slide adapted from David Silver]

Rat Example

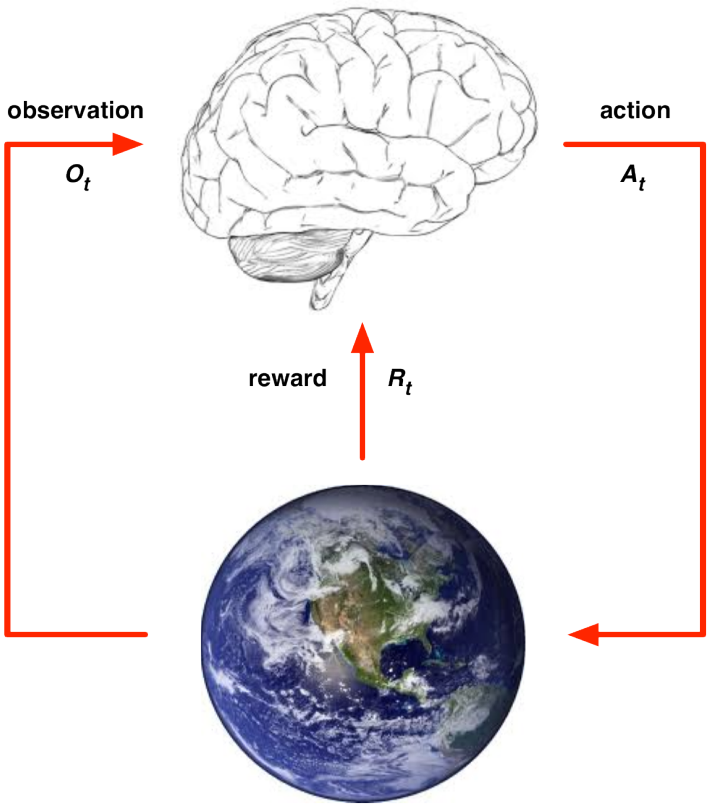




- What if agent state = last 3 items in sequence?
- What if agent state = counts for lights, bells and levers?
- What if agent state = complete sequence?

[Slide adapted from David Silver]

Fully Observeable Environments



Full observability: agent directly observes environment state

$$O_t = S_t^a = S_t^e$$

- Agent state = environment state = information state
- Formally, this is a Markov decision process (MDP)

[Slide adapted from David Silver]

Partially Observable Environments

- Partial observability: agent indirectly observes environment:
  - A robot with camera vision isn't told its absolute location
  - A trading agent only observes current prices
  - A poker playing agent only observes public cards
- Now agent state $\neq$ environment state
- Formally this is a partially observable Markov decision process (POMDP)
- Agent must construct its own state representation S_t^a, e.g.
  - Complete history: $S_t^a = H_t$
  - Beliefs of environment state: $S_t^a = (P(S_t^e = s^1), ..., P(S_t^e = s^n))$
  - Recurrent neural network: $s_t^a = \sigma(s_{t-1}^a W_s + o_t W_o)$

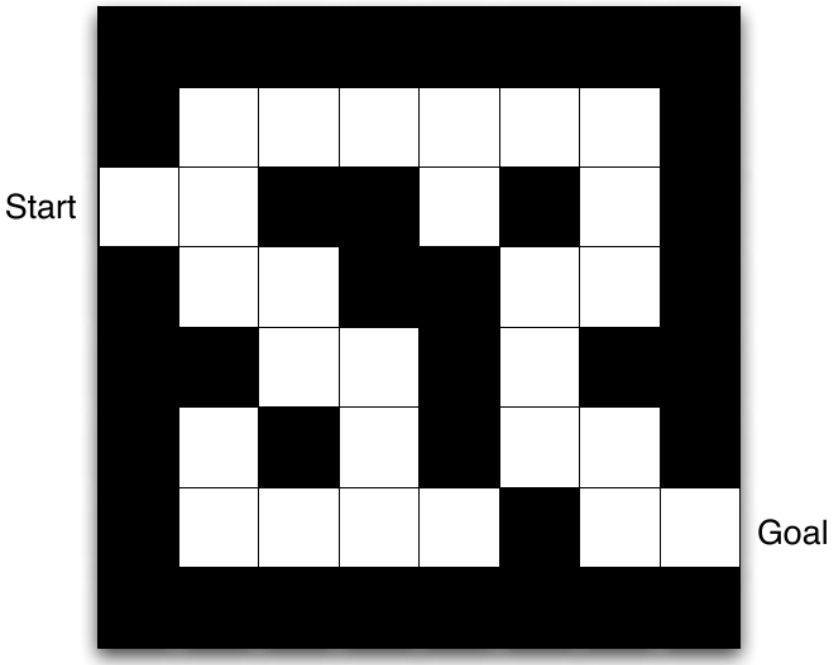
[Slide adapted from David Silver]

Components of an RL Agent

Components that an RL Agent can have:

- Policy: determines the actions that the agent takes
- Value function: captures how good each state and/or action is
- Model: mimics the environment (the agent's internal representation of the world)

Maze Example



Environment

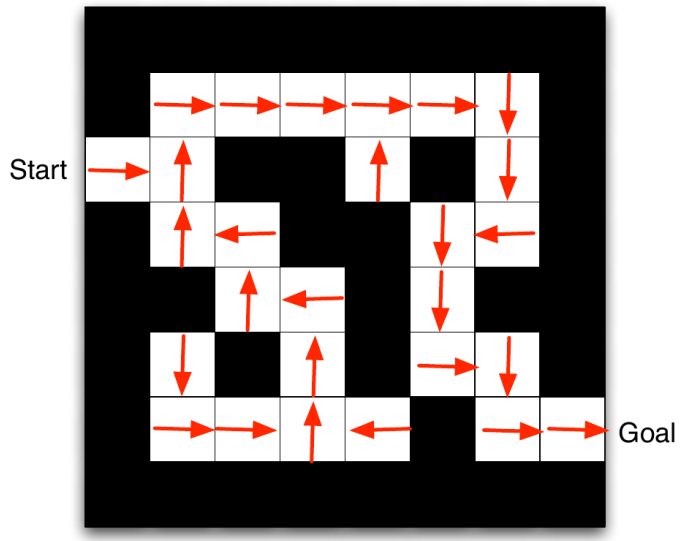
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location
- Terminates at Goal state

[Image: S&B RL Intro]

Policy

- The policy maps from state to action, e.g.
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a \mid s) = P(A_t = a \mid S_t = s)$

Example: optimal policy



Arrows represent policy $\pi(s)$

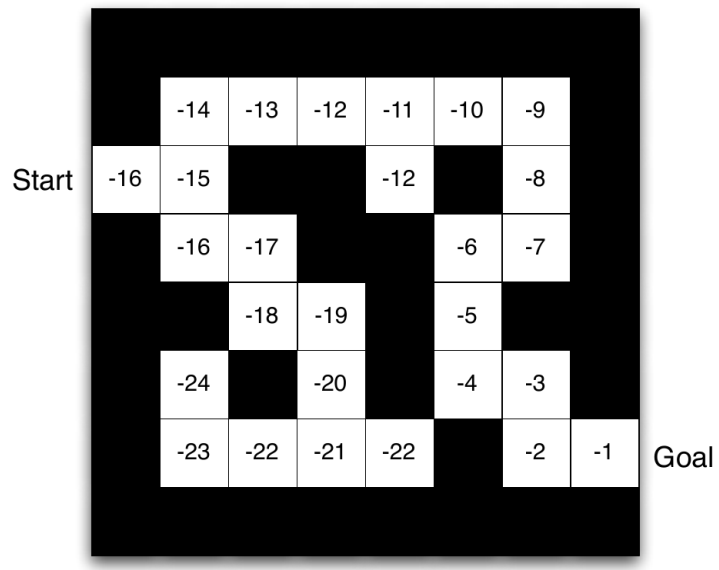
Value Function

- Value function captures how much reward will be collected
- represents goodness/badness of states
- used to select which action to take
- the *return* G_t is the total discounted reward from time-step t:

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

- Value Function models the expected discounted future reward = return:

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$



Numbers represent value $v_\pi(s)$ for $\gamma = 1$

Model

- The model predicts the dynamics of the environment
- Two models: for transitions and rewards
- Transition model $\mathcal{P}$ predicts the next state

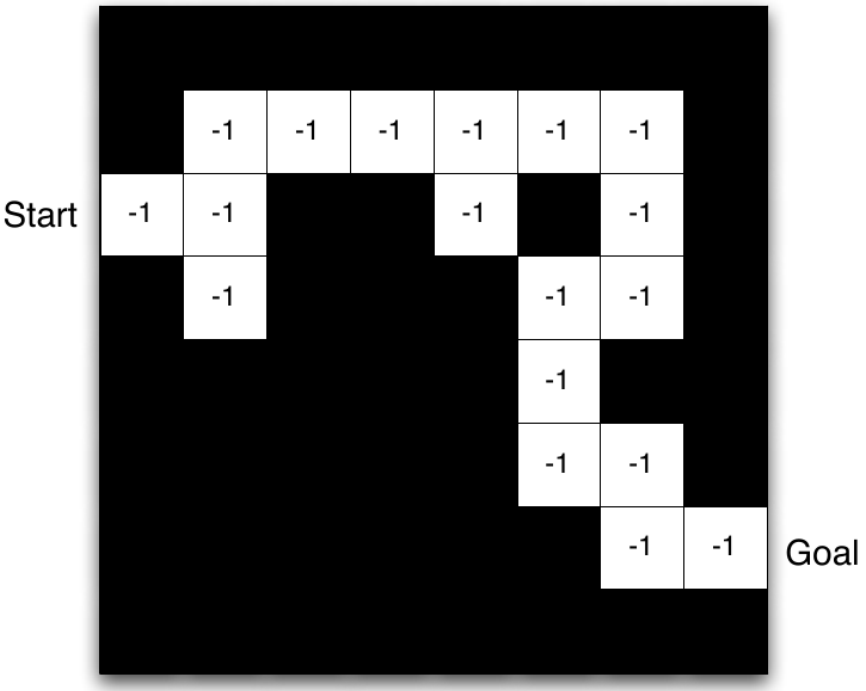
$$\mathcal{P}^{a}{}_{ss'} = P(S_{t+1} = s' \mid S_t = s, A_t = a)$$

- Reward model $\mathcal{R}$ predicts the next (immediate) reward

$$\mathcal{R}^{a}{}_{s} = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

- Can be used for planning without actually performing actions

Maze Example: Model



- Agent may have an internal model of the environment
- Dynamics: how actions change the state
- Rewards: how much reward from each state
- The model may be imperfect (most likely is)

- Grid layout represents transition model $\mathcal{P}^a_{ss'}$,
- Numbers represent immediate reward $\mathcal{R}^a_s$ from each state s (here the same for all a)

[Image: S&B RL Intro]

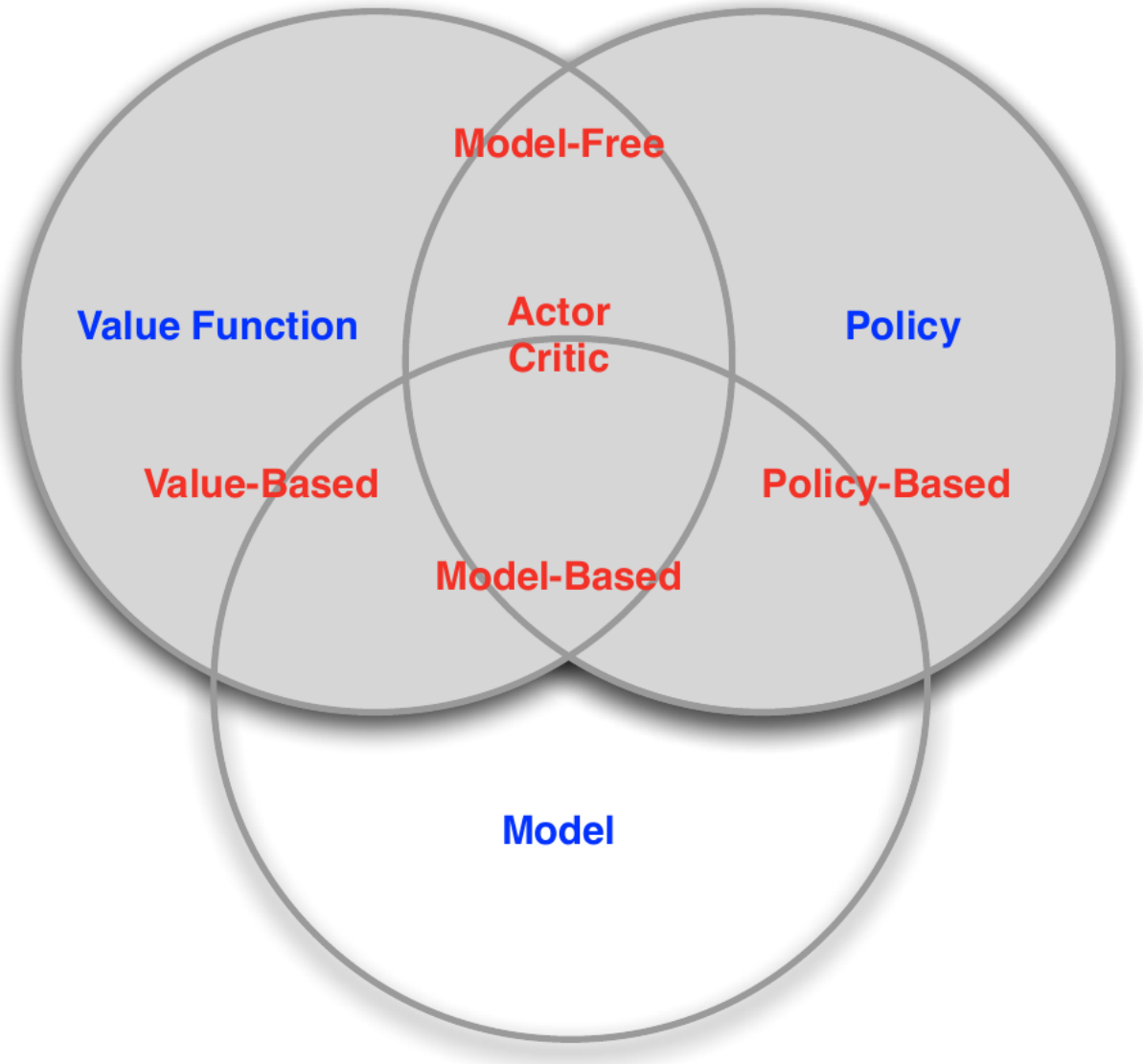
Categorization of RL agents

- Value Based
  - No Policy (Implicit)
  - Value Function
- Policy Based
  - Policy
  - No Value Function
- Actor Critic
  - Policy
  - Value Function

- Model Free
  - Policy and/or Value Function
  - No Model
- Model Based
  - Policy and/or Value Function
  - Model

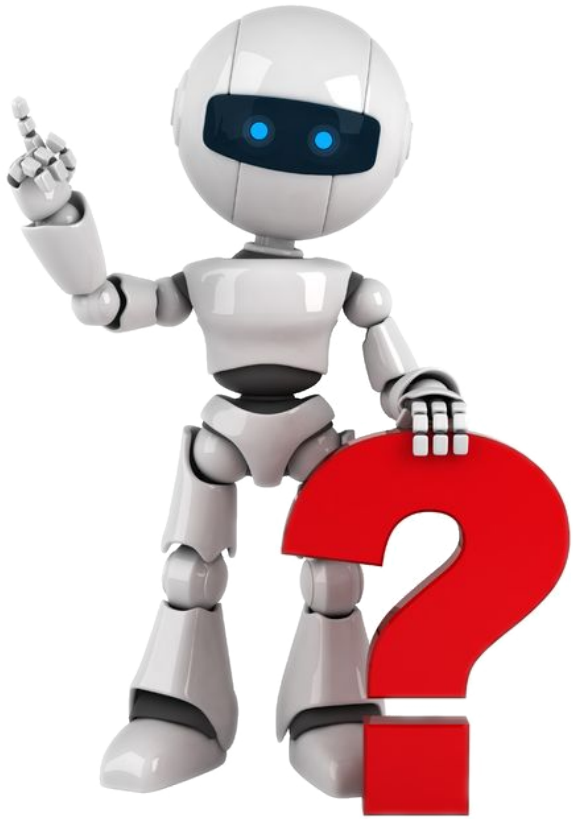
[Slide adapted from David Silver]

RL Agent Taxonomy



[Image: David Silver]

Questions?



[Image: globalrobots.com]

Next time

- Learning and Planning
- MDP's and Bellmann Equation
- Dynamic Programming and Value Iteration