S&DS 265 / 565

**Introductory Machine Learning**

# Reinforcement Learning

November 7

Yale

# Reminders

- Assignment 4 due today
- Assignment 5 posted today (topic models, neural nets and RL)
- Final exam, Dec 16 at 2pm
  https://registrar.yale.edu/general-information/final-exams

# For today and next class

- Overview of RL
- Q-learning
- Illustration on taxi problem
- RL concepts
- Deep reinforcement learning
- Neuroscience connection

---

Main reference: “Reinforcement Learning: An Introduction” (Second Edition), Richard S. Sutton and Andrew G. Barto

# Reinforcement learning

- An agent interacts with an environment
- The actions the agent takes change the state of the environment
- The agent receives rewards for each action, and seeks to maximize the total cumulative reward

*Reinforcement learning is a framework for sequential decision making to achieve a long-term goal.*

# Reinforcement learning: Motivating examples

- Learning to walk or ride a bike
- A robot vacuum cleaning up the house
- Playing chess, backgammon, Atari games, etc.
- Drug discovery, personalized health, energy management

---

Warning! Thinking about RL can change how you think about the world! :-)

# Last year



RESEARCH ARTICLE

## Human-level play in the game of *Diplomacy* by combining language models with strategic reasoning

META FUNDAMENTAL AI RESEARCH DIPLOMACY TEAM (FAIR)†, ANTON BAKHTIN, NOAM BROWN, EMILY DINAN, GABRIELE FARINA, COLIN FLAHERTY, DANIEL FRIED, ANDREW GOFF, JONATHAN GRAY, [...], AND MARKUS ZIJLSTRA +17 authors Authors Info & Affiliations

*SCIENCE* • 22 Nov 2022 • First Release • DOI: 10.1126/science.ade9097

60,933


### Abstract

Despite much progress in training AI systems to imitate human language, building agents that use language to communicate intentionally with humans in interactive environments remains a major challenge. We introduce Cicero, the first AI agent to achieve human-level performance in *Diplomacy*, a strategy game involving both cooperation and competition that emphasizes natural language negotiation and tactical coordination between seven players. Cicero integrates a language model with planning and reinforcement learning algorithms by inferring players' beliefs and intentions from its conversations and generating dialogue in pursuit of its plans. Across 40 games of an anonymous online *Diplomacy* league, Cicero achieved more than double the average score of the human players and ranked in the top 10% of participants who played more than one game.


### CURRENT ISSUE


**Cortical wiring by synapse type–specific control of local protein synthesis**

BY CLÉMENCE BERNARD, DAVID EXPOSITO-ALONSO, ET AL.

**Unadjuvanted intranasal spike vaccine elicits protective mucosal immunity against sarbecoviruses**

BY TIANYANG MAO, BENJAMIN ISRAELOW, ET AL.

**A multivalent nucleoside-modified mRNA vaccine against all known influenza virus subtypes**

https://www.science.org/doi/10.1126/science.ade9097

# Last year

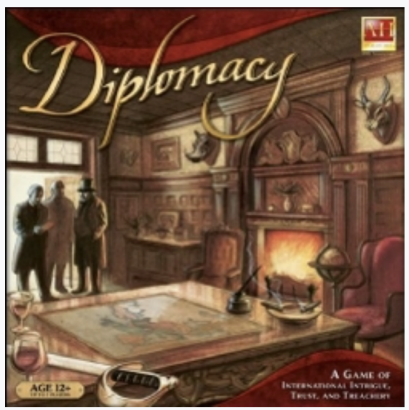
Article Talk Read Edit View history Search Wikipedia

## *Diplomacy* (game)

From Wikipedia, the free encyclopedia

***Diplomacy*** is an American strategic board game created by Allan B. Calhamer in 1954 and released commercially in the United States in 1959.[1] Its main distinctions from most board wargames are its negotiation phases (players spend much of their time forming and betraying alliances with other players and forming beneficial strategies)[2] and the absence of dice and other game elements that produce random effects. Set in Europe in the years leading to the Great War, *Diplomacy* is played by two to seven players,[3] each controlling the armed forces of a major European power (or, with fewer players, multiple powers). Each player aims to move their few starting units and defeat those of others to win possession of a majority of strategic cities and provinces marked as "supply centers" on the map; these supply centers allow players who control them to produce more units. Following each round of player negotiations, each player can issue attack and support orders, which are then executed during the movement phase. A player takes control of a province when the number of provinces that are given orders to support the attacking province exceeds the number of provinces given orders to support the defending province.

***Diplomacy***




| | |
|---|---|
| **Designers** | Allan B. Calhamer |
| **Publishers** | Wizards of the Coast |
| **Publication** | 1959; 63 years ago |

https://www.science.org/doi/10.1126/science.ade9097

# More recently

RESEARCH

## AI achieves silver-medal standard solving International Mathematical Olympiad problems

25 JULY 2024

AlphaProof and AlphaGeometry teams

Share

https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/

# More recently


https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/

# More recently

## AlphaProof: a formal approach to reasoning

AlphaProof is a system that trains itself to prove mathematical statements in the formal language Lean. It couples a pre-trained language model with the AlphaZero reinforcement learning algorithm, which previously taught itself how to master the games of chess, shogi and Go.

---

https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/

# Reinforcement learning: Formalization

- The environment is in state $s$ at a given time
- The agent takes action $a$
- The environment transitions to state $s' = \text{next}(s, a)$
- The agent receives reward $r = \text{reward}(s, a)$

# Reinforcement learning: Formalization

- The environment is in state $s$ at a given time
- The agent takes action $a$
- The environment transitions to state $s' = \text{next}(s, a)$
- The agent receives reward $r = \text{reward}(s, a)$

This is said to be a *Markov decision process*. It's "Markov" because the next state only depends on the current state and the action selected. It's a "decision process" because the agent is making choices of actions in a sequential manner.

# Characteristics

- RL is inherently sequential
- In between supervised and unsupervised learning
- Agent can't act too greedily; needs to be strategic

The aim of RL is to learn to make optimal decisions from experience

# Principles of RL

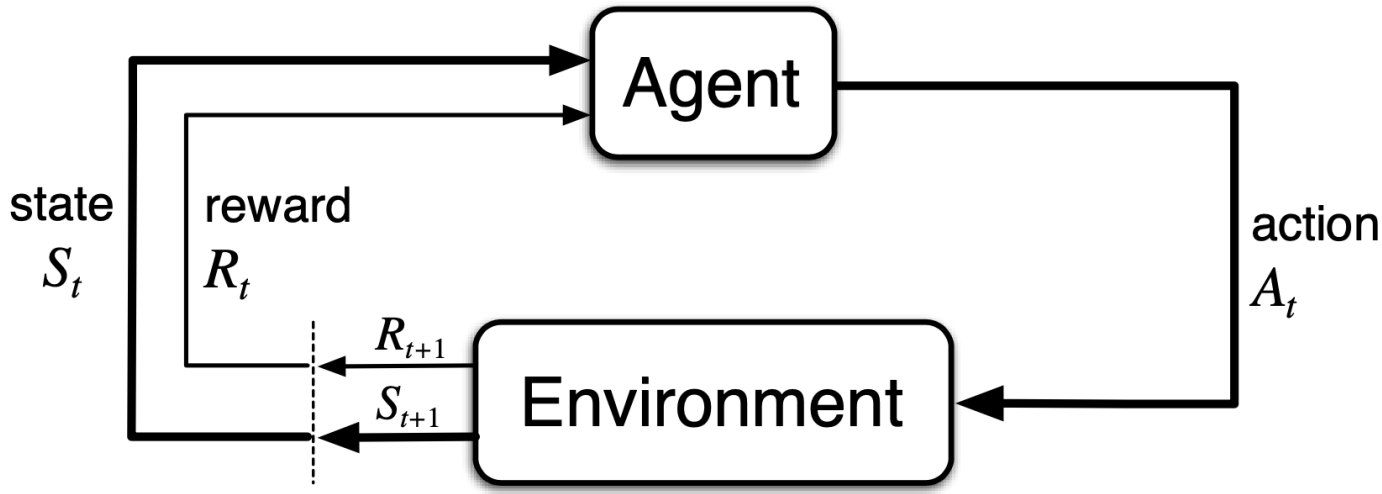
Some key RL concepts and principles:

Policy, reward signal, value function, model, Bellman equation

---

"Reinforcement Learning: An Introduction" (Second Edition), Richard S. Sutton and Andrew G. Barto

# Principles of RL




Rewards and state transitions are probabilistic, in general

---

"Reinforcement Learning: An Introduction" (Second Edition), Richard S. Sutton and Andrew G. Barto

# Principles of RL

*Policy*: A mapping from states to actions. An algorithm/rule to make decisions at each time step, designed to maximize the long term reward.

---

"Reinforcement Learning: An Introduction" (Second Edition), Richard S. Sutton and Andrew G. Barto

# Principles of RL

*Reward signal*: The sequence of rewards received at each time step. An abstraction of “pleasure” (positive reward) and “pain” (negative reward) in animal behavior.

---

“Reinforcement Learning: An Introduction” (Second Edition), Richard S. Sutton and Andrew G. Barto

# Principles of RL

*Value function*: A mapping from states to total reward. The total reward the agent can expect to accumulate in the future, starting from that state.

Rewards are short term. Values are predictions of future rewards.

---

"Reinforcement Learning: An Introduction" (Second Edition), Richard S. Sutton and Andrew G. Barto

# Principles of RL

*Model*: Used for planning to mimic the behavior of the environment, to predict rewards and next states.

A *model-free* approach directly estimates a value function, without modeling the environment.

Analogous to distinction between generative and discriminative classification models

---

# Taxi problem

We'll introduce the important Q-learning algorithm with the toy "Taxi problem"

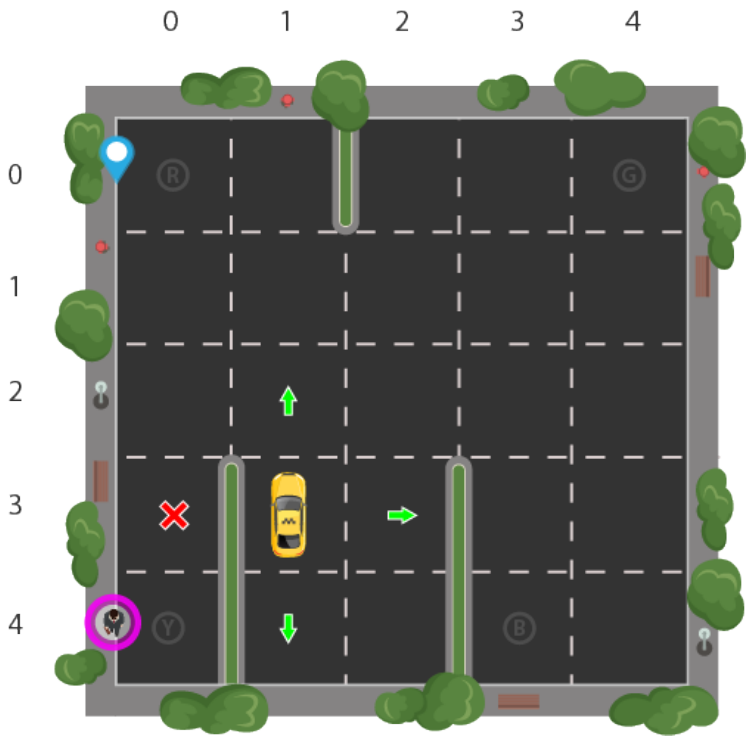
The code uses OpenAI gym

---

Our presentation follows the tutorial
https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/

# Taxi problem

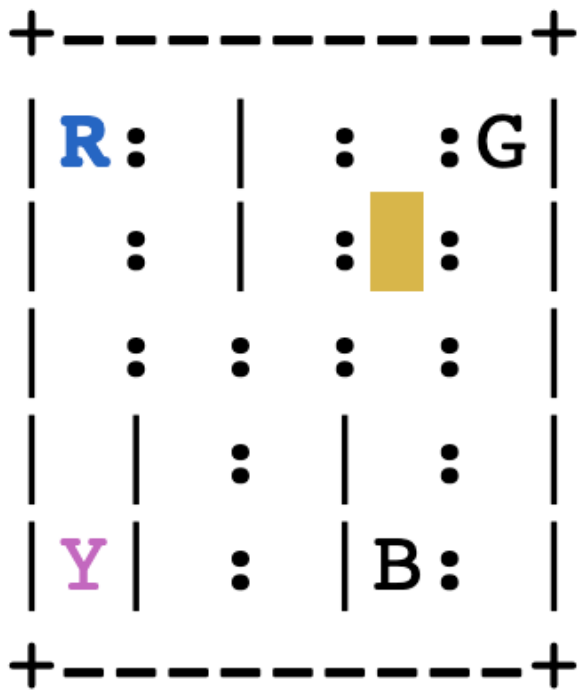
A taxicab drives around the environment, picking up and delivering a passenger at four locations



# Taxi problem

A taxicab drives around the environment, picking up and delivering a passenger at four locations

”Ascii art” rendition:



# Taxi problem: Description

- Four designated locations: R(ed), G(reen), Y(ellow), and B(lue)
- Taxi starts off at random square and passenger is at random location
- Taxi drives to passenger's location, picks up the passenger, drives to passenger's destination, drops off passenger
- Once the passenger is dropped off, the episode ends.

# Taxi problem: State space

- 25 taxi positions
- 5 possible locations of passenger: At waiting location or in taxi
- 4 possible destination locations
- Total number of states: $25 \times 5 \times 4 = 500$

# Taxi problem: State space

Passenger location coded as integers:

- 0: R(ed)
- 1: G(reen)
- 2: Y(ellow)
- 3: B(lue)
- 4: in taxi

# Taxi problem: State space

Destinations coded as:

- 0: R(ed)
- 1: G(reen)
- 2: Y(ellow)
- 3: B(lue)

# Taxi problem: State space

Agent actions coded as:

- 0: move south
- 1: move north
- 2: move east
- 3: move west
- 4: pickup passenger
- 5: drop off passenger

# Taxi problem: State space

Rewards:

- Default reward per step: -1
- Reward for delivering passenger: +20
- Illegal “pickup” or “drop-off”: -10

# Taxi problem: State space

State space represented as a tuple:

state = (taxi row, taxi column, passenger location, destination)

# Q-learning

- Maintains a “quality” variable $Q(s, a)$ for taking action $a$ in state $s$

# Q-learning

- Maintains a “quality” variable $Q(s, a)$ for taking action $a$ in state $s$
- A measure of the cumulative rewards obtained by the algorithm when it takes action $a$ in state $s$

# Q-learning

- Maintains a “quality” variable $Q(s, a)$ for taking action $a$ in state $s$
- A measure of the cumulative rewards obtained by the algorithm when it takes action $a$ in state $s$
- Quality should not be assessed purely based on the reward the action has in the current time step

# Q-learning

- Maintains a “quality” variable $Q(s, a)$ for taking action $a$ in state $s$
- A measure of the cumulative rewards obtained by the algorithm when it takes action $a$ in state $s$
- Quality should not be assessed purely based on the reward the action has in the current time step
- Need to take into account the future rewards

# Q-learning update

**Update:**

$$Q(s,a) \longleftarrow$$

$$Q(s,a) + \alpha\Big(\text{reward}(s,a) + \gamma \max_{a'} Q(\text{next}(s,a),a') - Q(s,a)\Big)$$

# Q-learning update

**Update:**

$$Q(s,a) \longleftarrow$$
$$Q(s,a) + \alpha\Big(\text{reward}(s,a) + \gamma \max_{a'} Q(\text{next}(s,a),a') - Q(s,a)\Big)$$

- When action $a$ is taken in state $s$, reward reward$(s,a)$ is given
- Then, the algorithm moves to a new state next$(s,a)$

# Q-learning update

**Update:**

$$Q(s,a) \longleftarrow$$
$$Q(s,a) + \alpha\Big(\text{reward}(s,a) + \gamma \max_{a'} Q(\text{next}(s,a),a') - Q(s,a)\Big)$$

- For example, if the taxi is location $(2,2)$ and takes the "West" action ($a = 3$), then there is a reward of -1, and the taxi moves to the new location $(2,1)$
- If cab is empty, it remains empty, and if it contains the passenger, the passenger remains.

# Q-learning update

**Update:**

$$Q(s,a) \longleftarrow$$
$$Q(s,a) + \alpha\Big(\text{reward}(s,a) + \gamma \max_{a'} Q(\text{next}(s,a),a') - Q(s,a)\Big)$$

- Cumulative future reward of this action is $\max_{a'} Q(\text{next}(s,a),a')$
- Future rewards discounted by factor $\gamma < 1$
- Trades off short-term against long-term rewards
- A gradient *ascent* algorithm, with step size $\alpha$

Let’s go to the notebook!

# Bellman equation

The optimal value function is the largest expected discounted long term reward starting from that state.

---

https://en.wikipedia.org/wiki/Richard_E._Bellman

# Bellman equation

### Curse of dimensionality [ edit ]

*Main article: Curse of dimensionality*

The *curse of dimensionality* is an expression coined by Bellman to describe the problem caused by the exponential increase in volume associated with adding extra dimensions to a (mathematical) space. One implication of the curse of dimensionality is that some methods for numerical solution of the Bellman equation require vastly more computer time when there are more state variables in the value function. For example, 100 evenly spaced sample points suffice to sample a unit interval with no more than 0.01 distance between points; an equivalent sampling of a 10-dimensional unit hypercube with a lattice with a spacing of 0.01 between adjacent points would require $10^{20}$ sample points: thus, in some sense, the 10-dimensional hypercube can be said to be a factor of $10^{18}$ "larger" than the unit interval. (Adapted from an example by R. E. Bellman, see below.) [15]

---

https://en.wikipedia.org/wiki/Richard_E._Bellman

# Bellman equation: Deterministic case

The optimality condition for the value function $v_*$ is

$$v_*(s) = \max_a \Big\{ \text{reward}(s, a) + \gamma v_*(\text{next}(s, a)) \Big\}$$

# Bellman equation: Deterministic case

The optimality condition for the Q-function is

$$Q_*(s,a) = \text{reward}(s,a) + \gamma \max_{a'} Q_*(\text{next}(s,a), a')$$

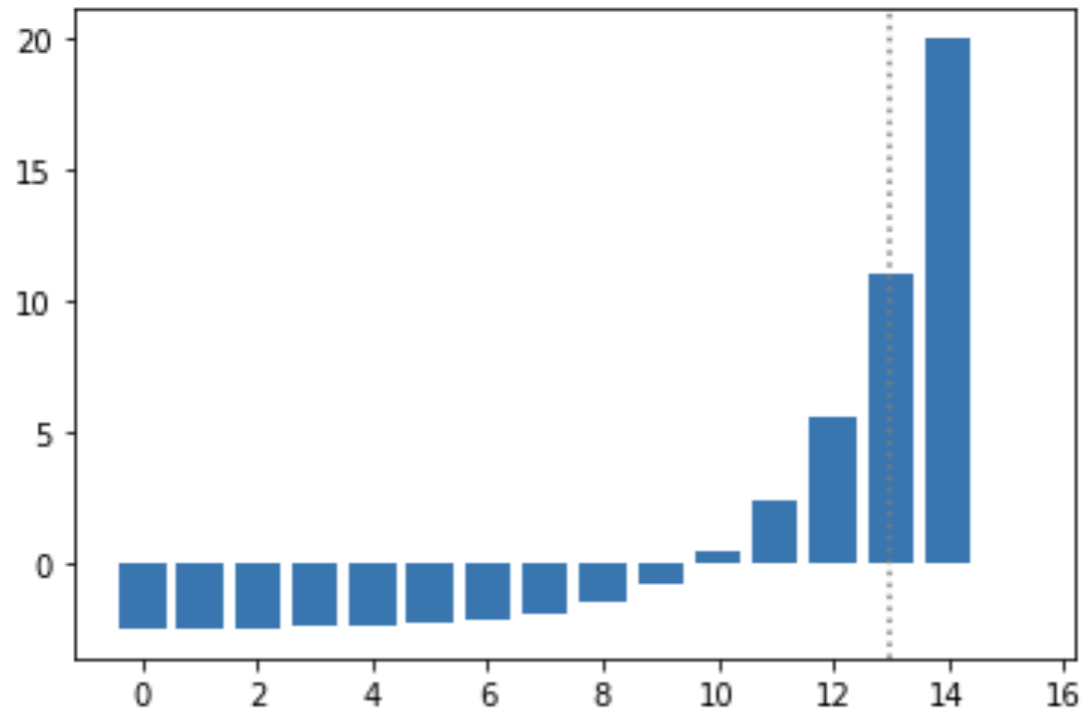
and then $v_*(s) = \max_{a'} Q_*(s,a')$

# Q-learning update

Note how this makes sense in terms of the update rule:

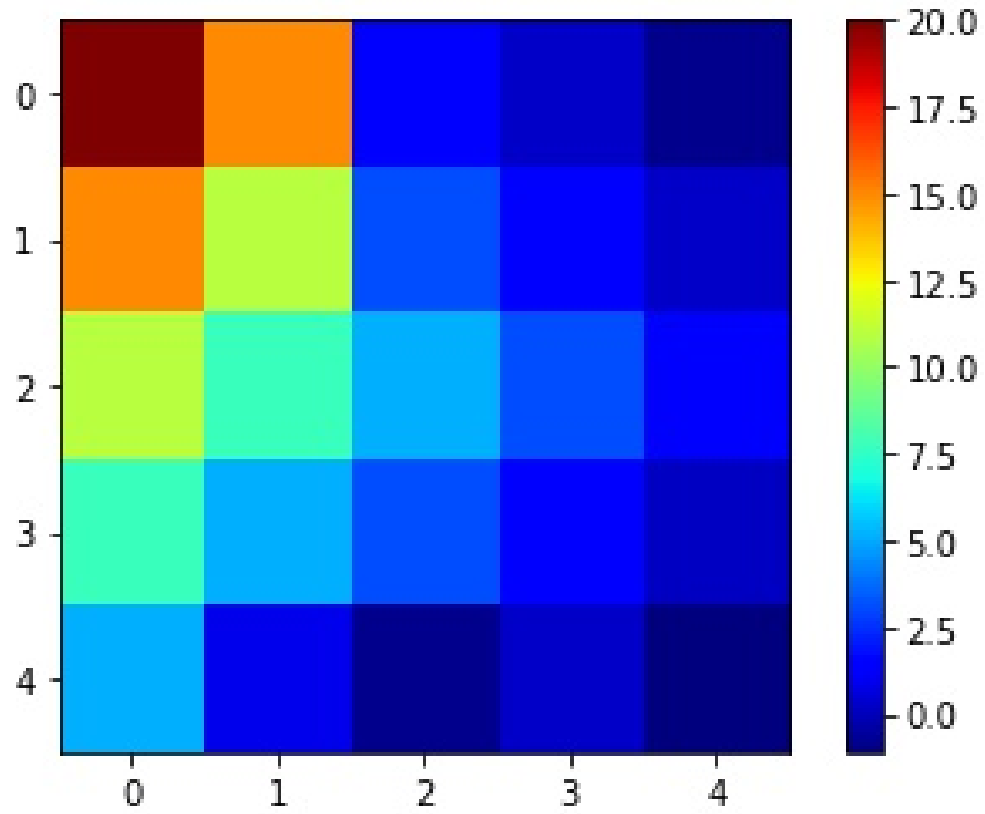
$$Q(s,a) \longleftarrow Q(s,a) + \alpha\Big(\text{reward}(s,a) + \gamma \max_{a'} Q(\text{next}(s,a), a') - Q(s,a)\Big)$$

```
+---------+
|R: | : :G|
| : | : :▄|
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (East)
```



# Question

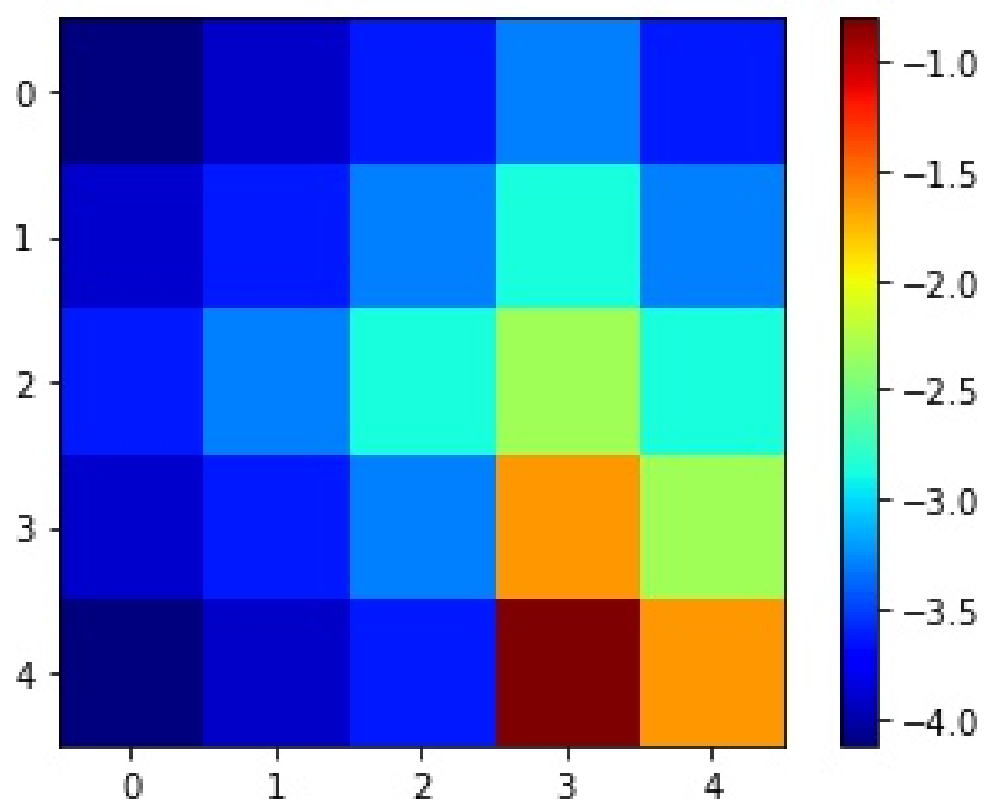
For a fixed passenger location and destination, the value function $v(\texttt{row}, \texttt{col})$ assigns a value to each of the $25 = 5 \times 5$ grid points.




Is the passenger waiting, or in the taxi?

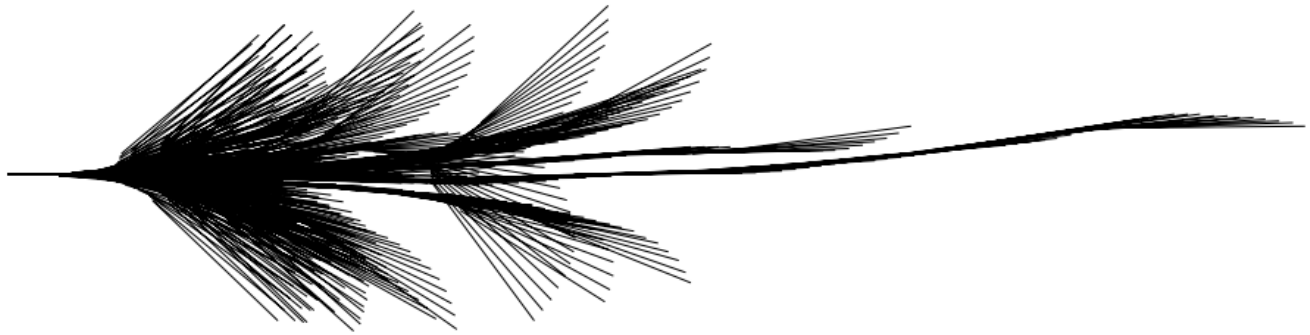
# Question

For a fixed passenger location and destination, the value function $v(\texttt{row}, \texttt{col})$ assigns a value to each of the $25 = 5 \times 5$ grid points.




Is the passenger waiting, or in the taxi?

# Learning (and unlearning) to ride a bike



"Learning to Drive a Bicycle Using Reinforcement Learning and Shaping," Jette Randløv and Preben Alstrøm, ICML 1998
Smarter Every Day: Learning to ride a backwards bike: `https://www.youtube.com/watch?v=MFzDaBzBlL0`

# Learning (and unlearning) to ride a bike

"Learning to Drive a Bicycle Using Reinforcement Learning and Shaping," Jette Randløv and Preben Alstrøm, ICML 1998
Smarter Every Day: Learning to ride a backwards bike: `https://www.youtube.com/watch?v=MFzDaBzBlL0`

# Summary

- Reinforcement learning is a framework for sequential decision making to achieve a long-term goal
- The agent receives rewards for each action, and seeks to maximize the total cumulative reward
- The value of a state is the total reward the agent can expect to accumulate in the future, starting from that state
- Q-learning is an iterative algorithm that maximizes the “quality” of each state-action pair
- The Bellman equations are optimality conditions that characterize the value function