CS 480

Introduction to Artificial Intelligence

April 21, 2022

Announcements / Reminders

- Final Exam: April 28th!
  - Ignore Registrar date for CS 480
  - Online section: please contact Mr. Charles Scott (scott@iit.edu) to make arrangements if necessary
- End of semester course evaluation: open. Thank you!

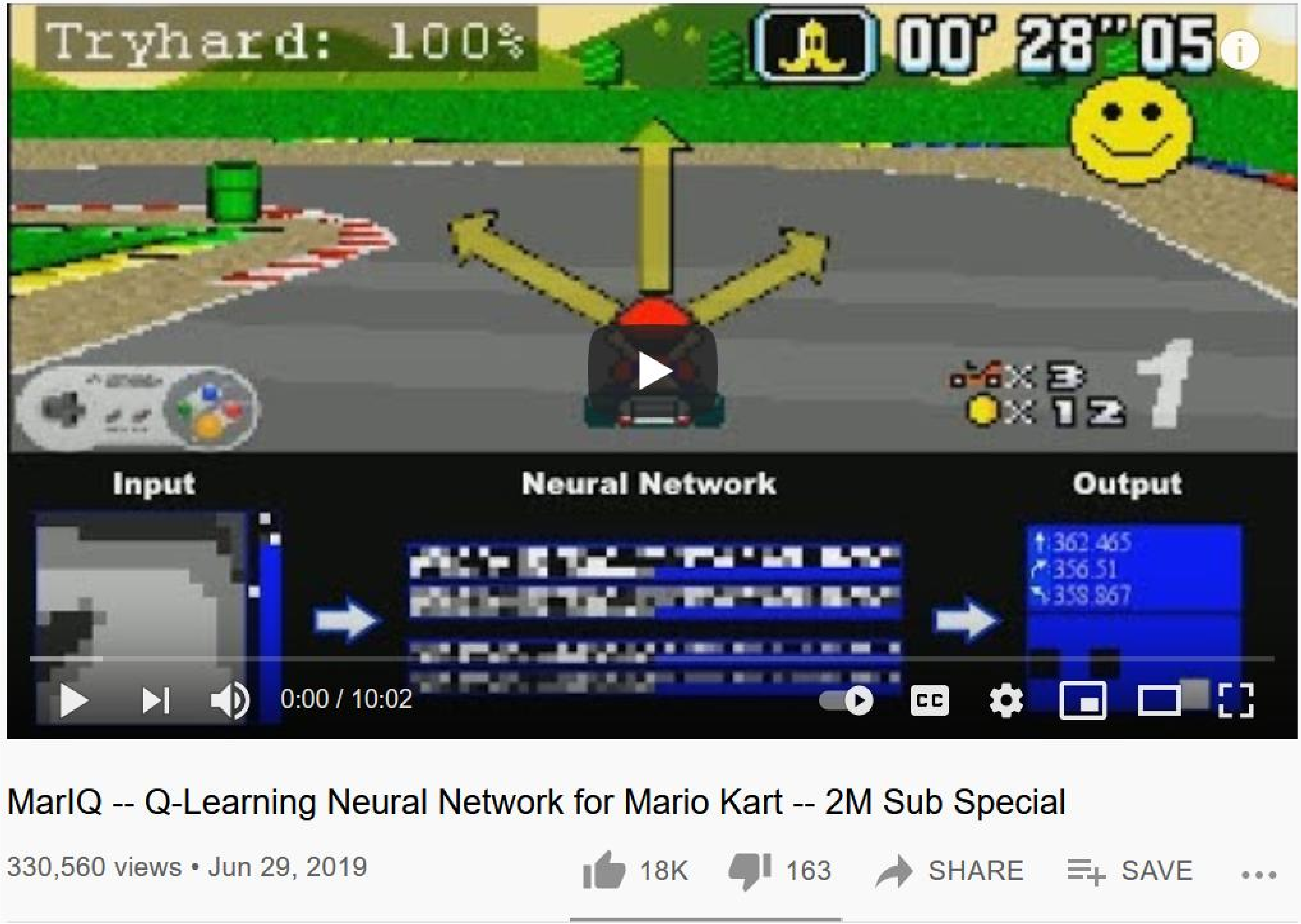
- Programming Assignment #02: due TOMORROW (04/22)
  - ADD COMMENTS!!!!!
- Written Assignment #04: due on Wednesday (04/27)
- Grading TA assignment:

  https://docs.google.com/spreadsheets/d/1Cav_GBTGC7fLGzxuBCAUmEuJYPeF-HMLCYvwPbq8Fus/edit?usp=sharing

Plan for Today

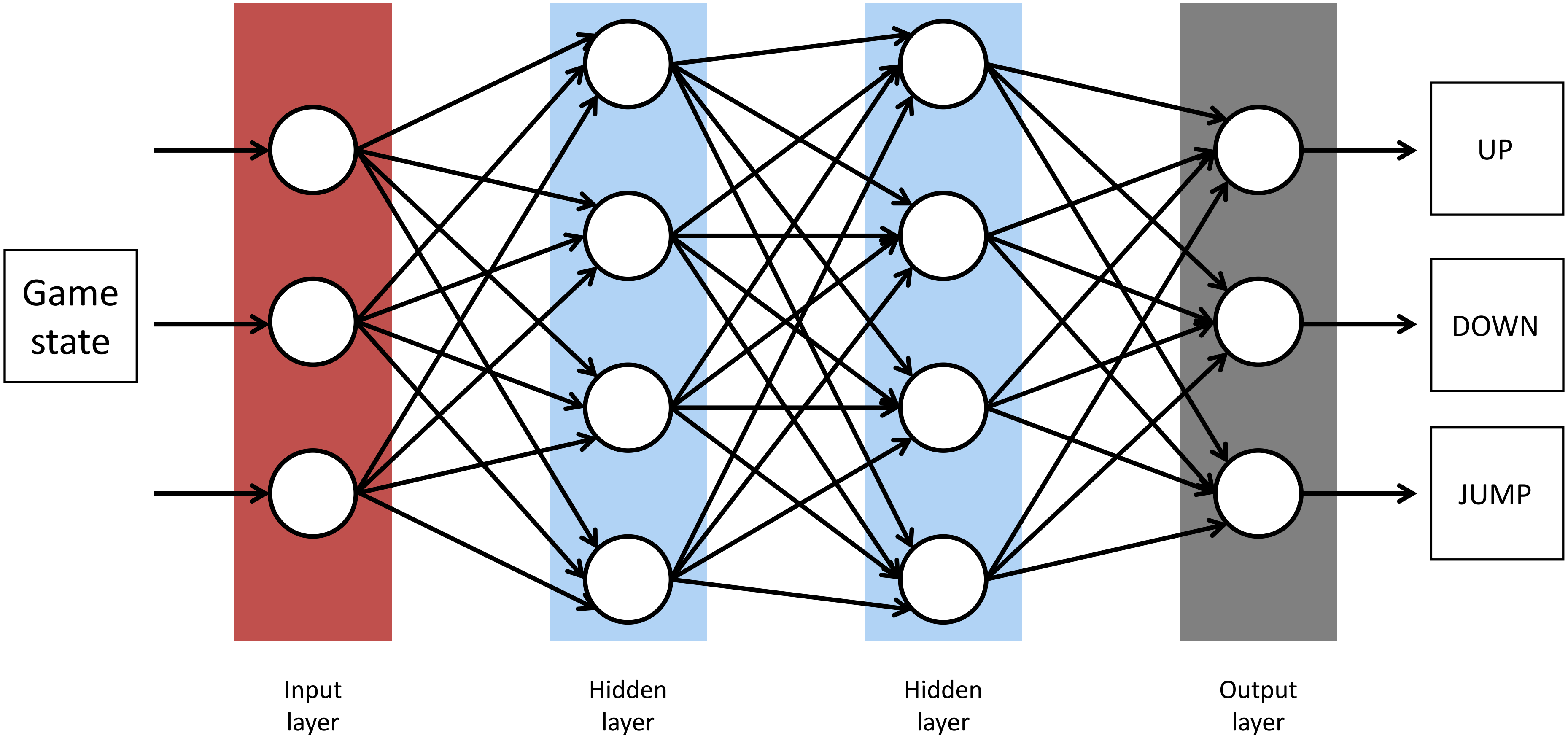
- **Casual Introduction to Machine Learning**
  - **Deep Learning**
  - **Reinforcement Learning**

Reinforcement Learning in Action



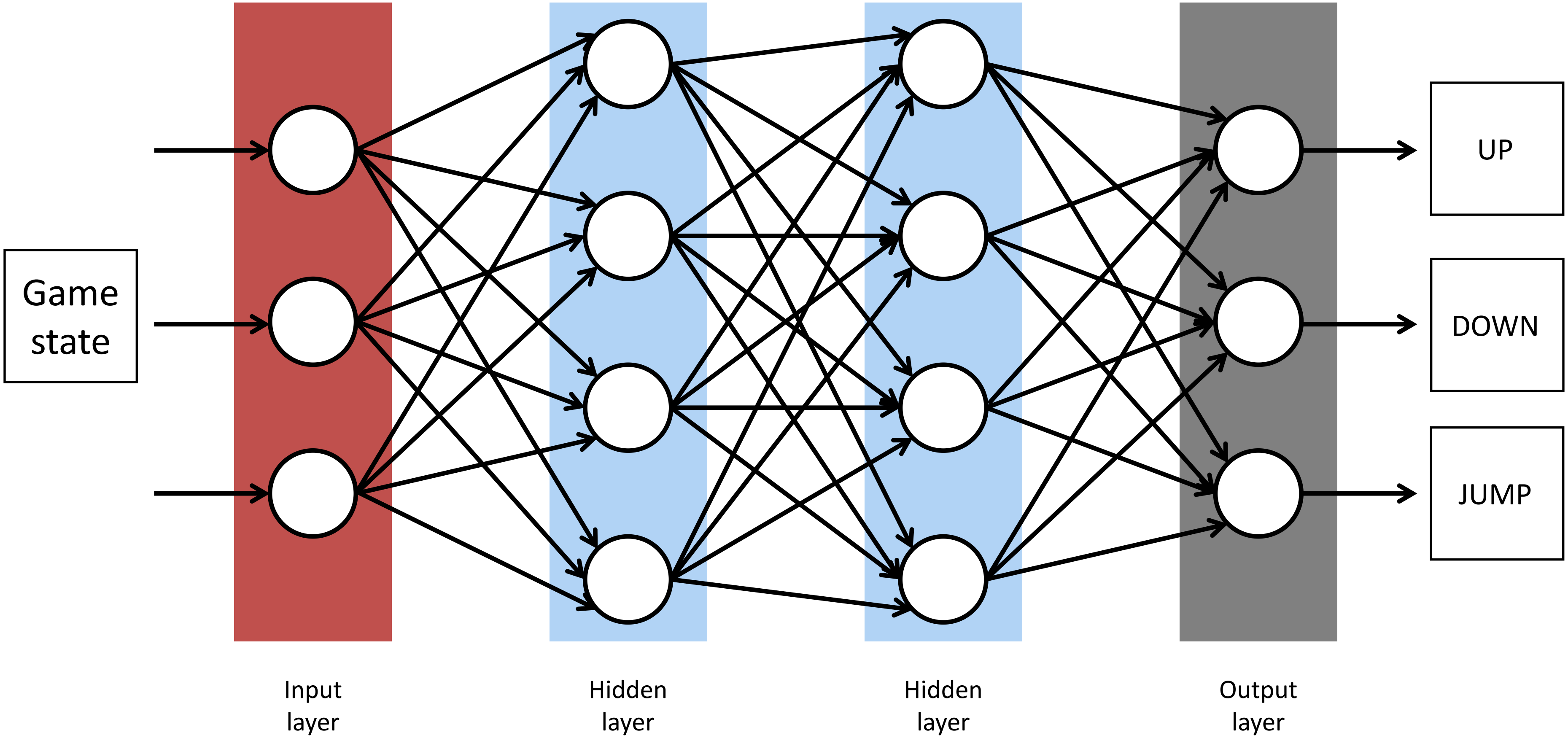
Source: https://www.youtube.com/watch?v=Tnu4O_xEmVk

ANN for Simple Game Playing



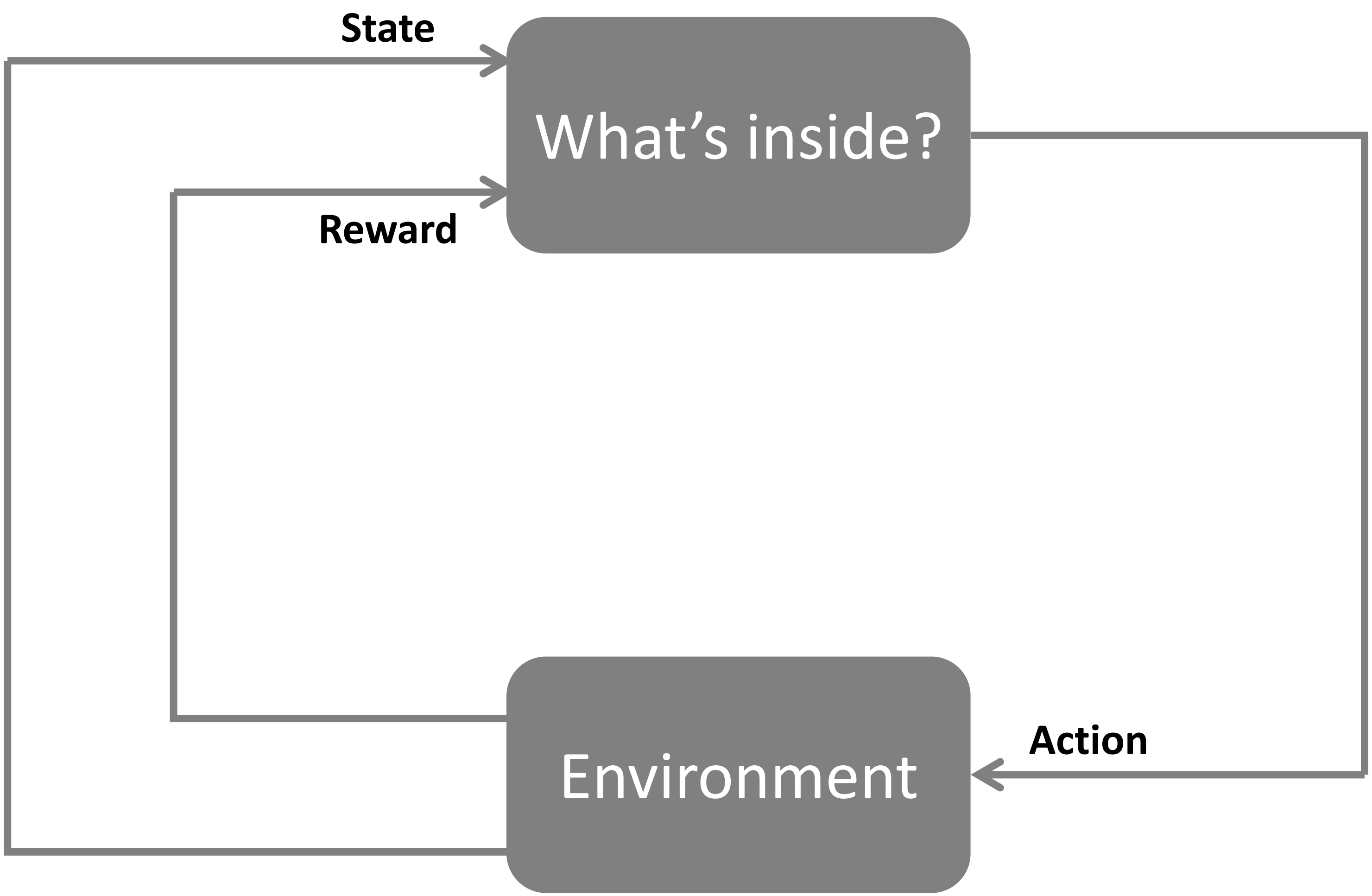
ANN for Simple Game Playing

Current game is an input. Decisions (UP/DOWN/JUMP) are rewarded/punished.

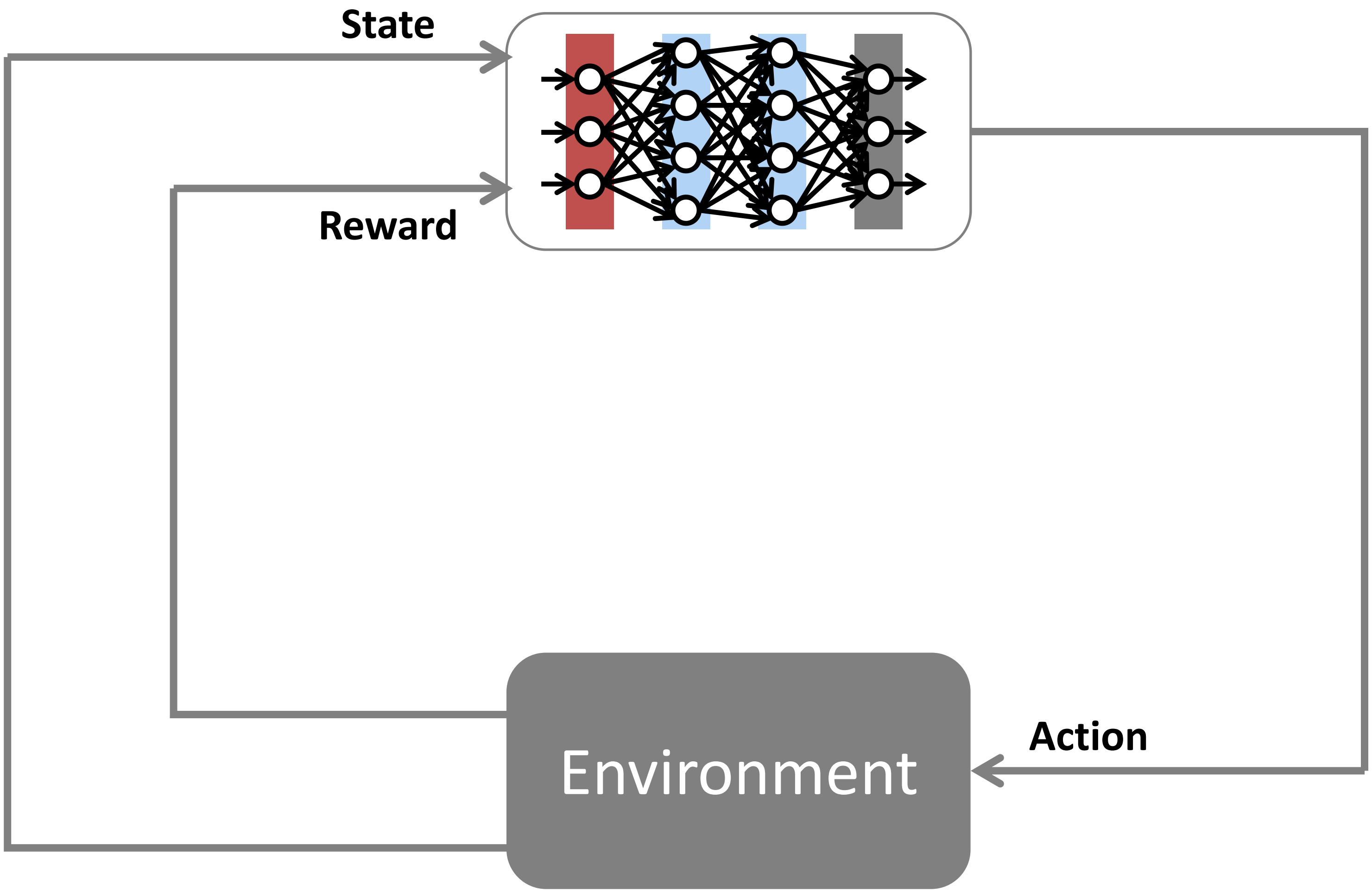


Correct all the weights using Reinforcement Learning.

RL: Agents and Environments

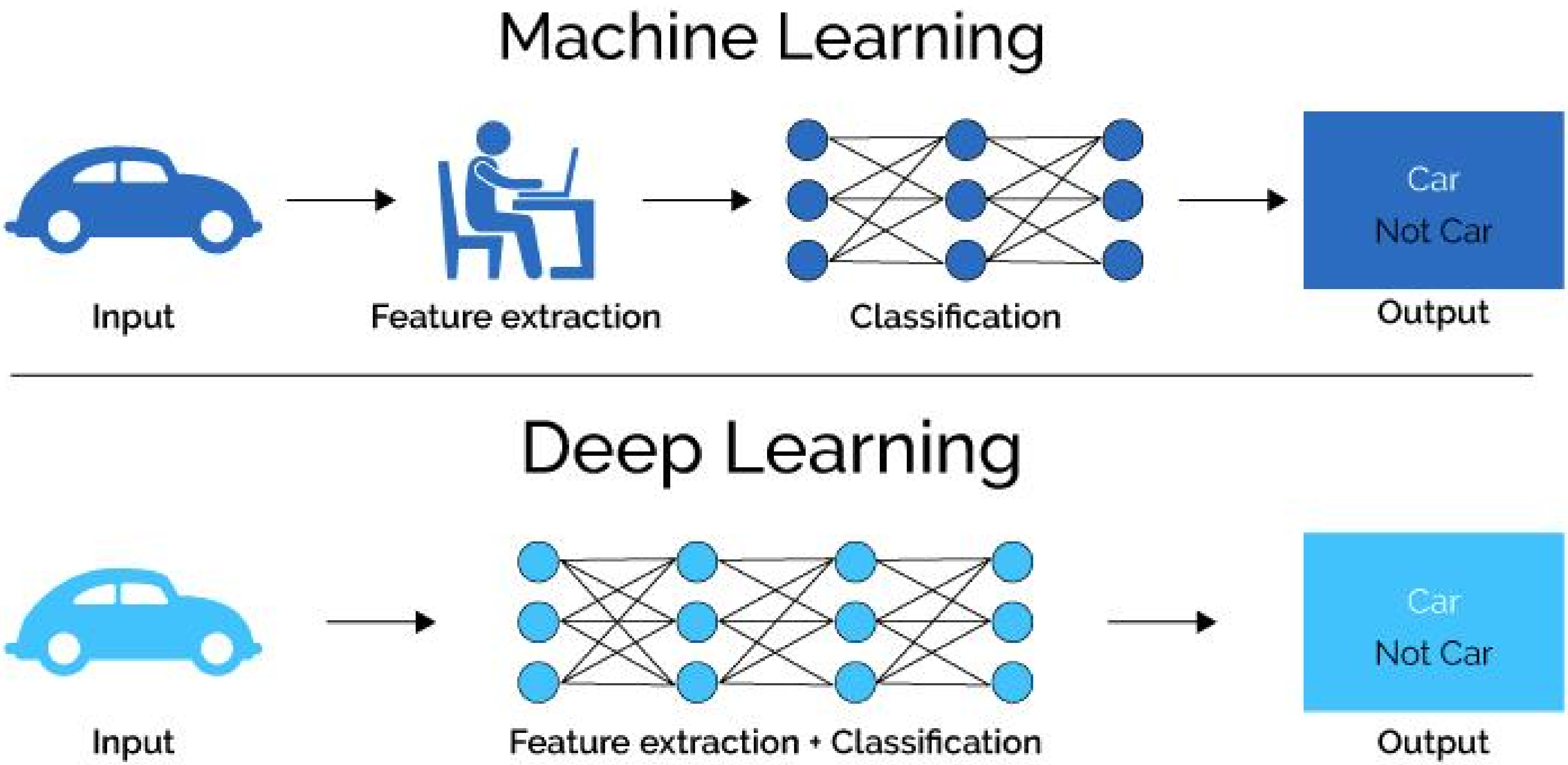


RL: Agents and Environments



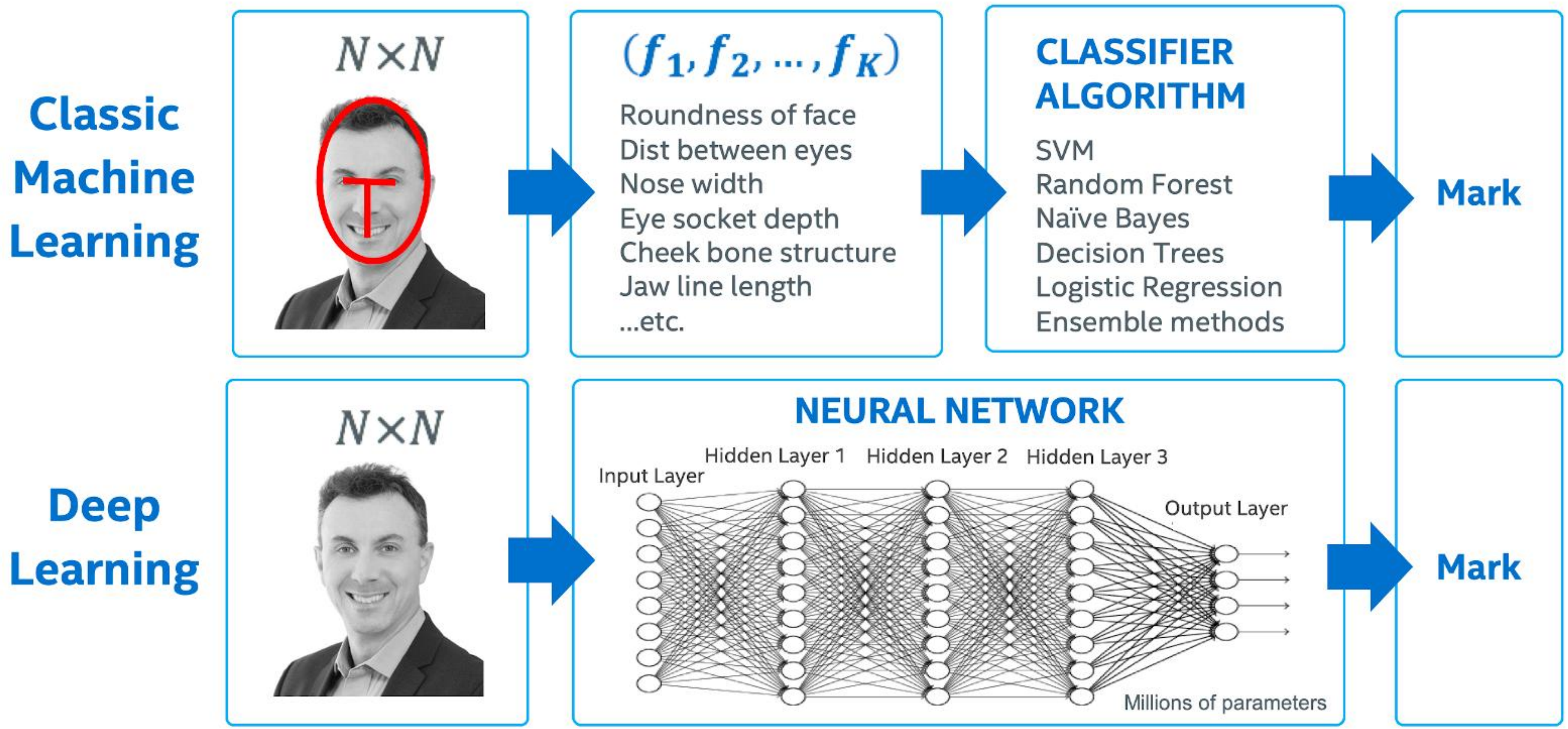
Deep Learning

Machine Learning vs. Deep Learning



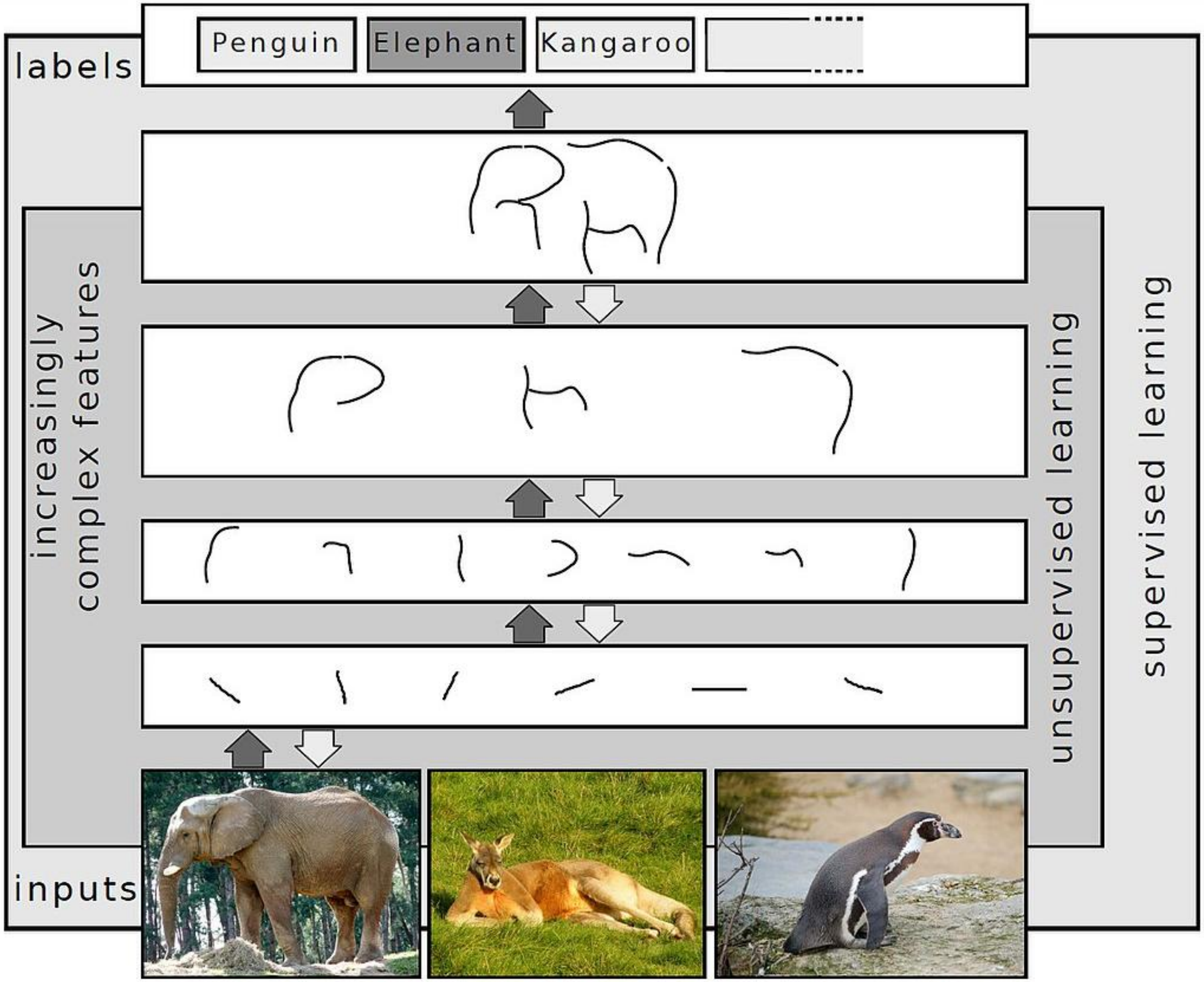
Source: https://www.quora.com/What-is-the-difference-between-deep-learning-and-usual-machine-learning

Machine Learning vs. Deep Learning



Source: https://www.intel.com/content/www/us/en/artificial-intelligence/posts/difference-between-ai-machine-learning-deep-learning.html

Deep Learning: Feature Extraction



Source: https://en.wikipedia.org/wiki/Deep_learning

Exercise: Object Recognition

https://braneshop.com.au/object-detection-in-the-browser.html

(you can try it on your smartphone)

Exercise: Image Colorizer

https://deepai.org/machine-learning-model/colorizer

Exercise: Deep Learning

https://www.handwriting-generator.com/

Main Machine Learning Categories

Supervised learning	Unsupervised learning	Reinforcement learning
Supervised learning is one of the most common techniques in machine learning. It is based on **known relationship(s) and patterns within data** (for example: relationship between inputs and outputs). Frequently used types: **regression**, and **classification**.	**Unsupervised learning** involves finding underlying patterns within data. Typically used in **clustering** data points (similar customers, etc.)	Reinforcement learning is inspired by behavioral psychology. It is **based on a rewarding / punishing an algorithm**. Rewards and punishments are based on algorithm's action within its environment.

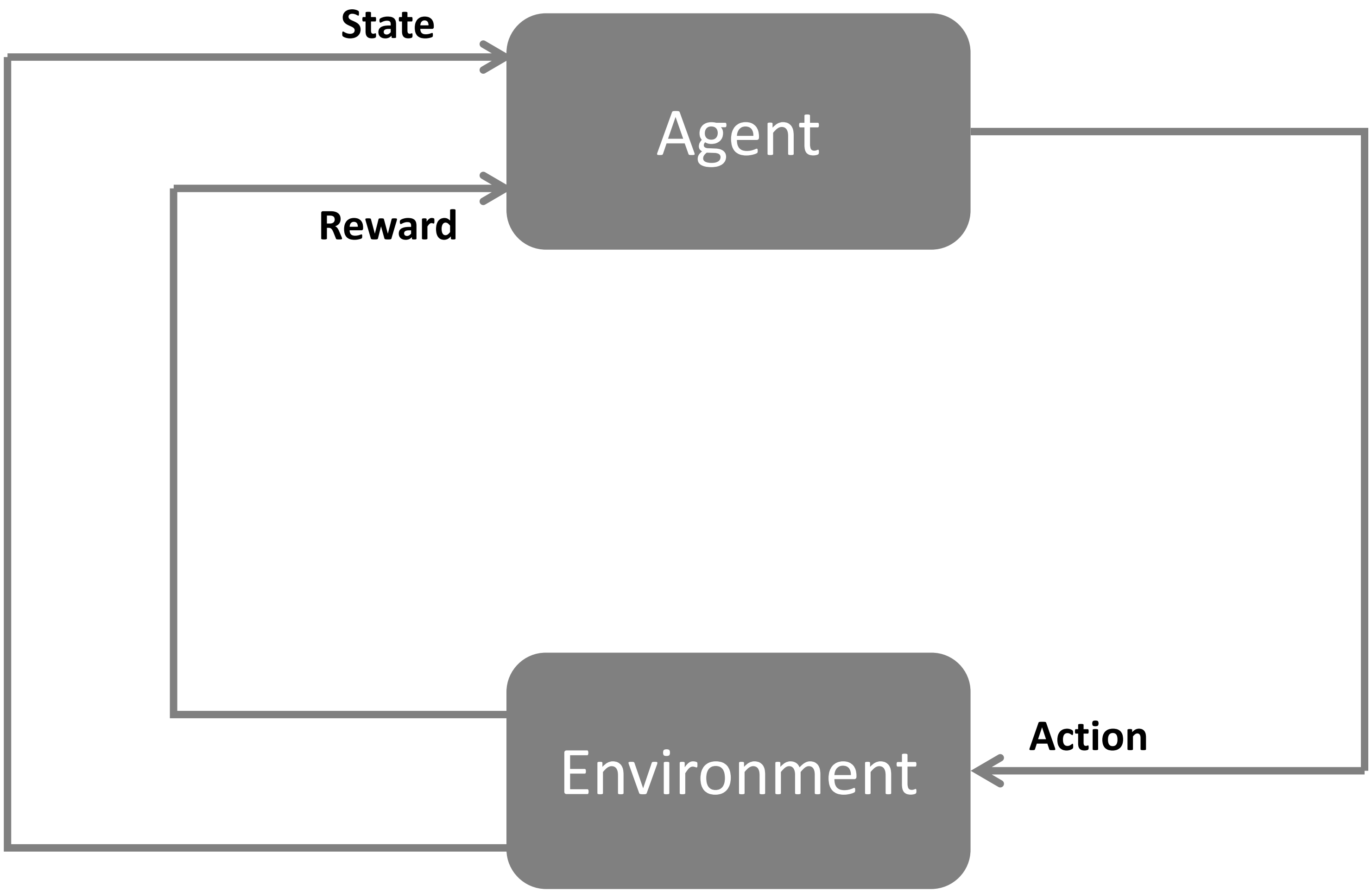
What is Reinforcement Learning?

Idea:

Reinforcement learning is inspired by behavioral psychology. It is **based on a rewarding / punishing an algorithm**.

Rewards and punishments are based on algorithm's action within its environment.

RL: Agents and Environments



Reinforcement Learning in Action

Source: https://www.youtube.com/watch?v=kopoLzvh5jY

K-Armed Bandit Problem

K-Armed Bandit Problem

The K-armed bandit problem is a problem in which a fixed limited set of resources must be allocated between competing (alternative) choices in a way that maximizes their expected gain.

Each choice's properties are only partially known at the time of allocation, and may become better understood as time passes or by allocating resources to the choice.

K-Armed Bandit Problem

In the problem, each machine provides a random reward from a probability distribution specific to that machine, that is not known a-priori.

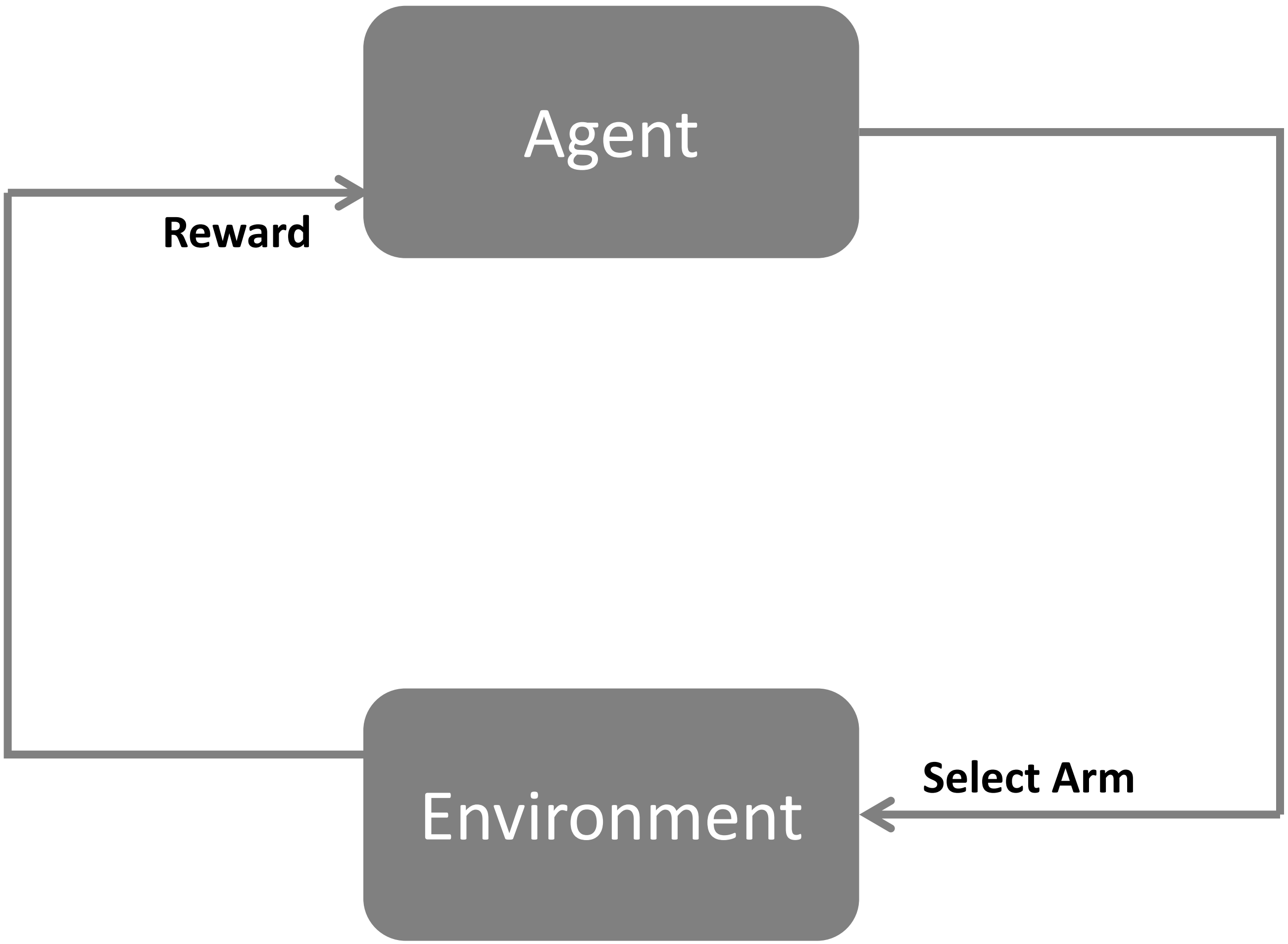
The objective of the gambler is to maximize the sum of rewards earned through a sequence of lever pulls.

K-Armed Bandit Problem

Bandit/Arm 1	Bandit/Arm 2	Bandit/Arm 3
33 %	52 %	78 %
current success (win) rate	**current** success (win) rate	**current** success (win) rate

Which bandit shall we play next?

K-Armed Bandit



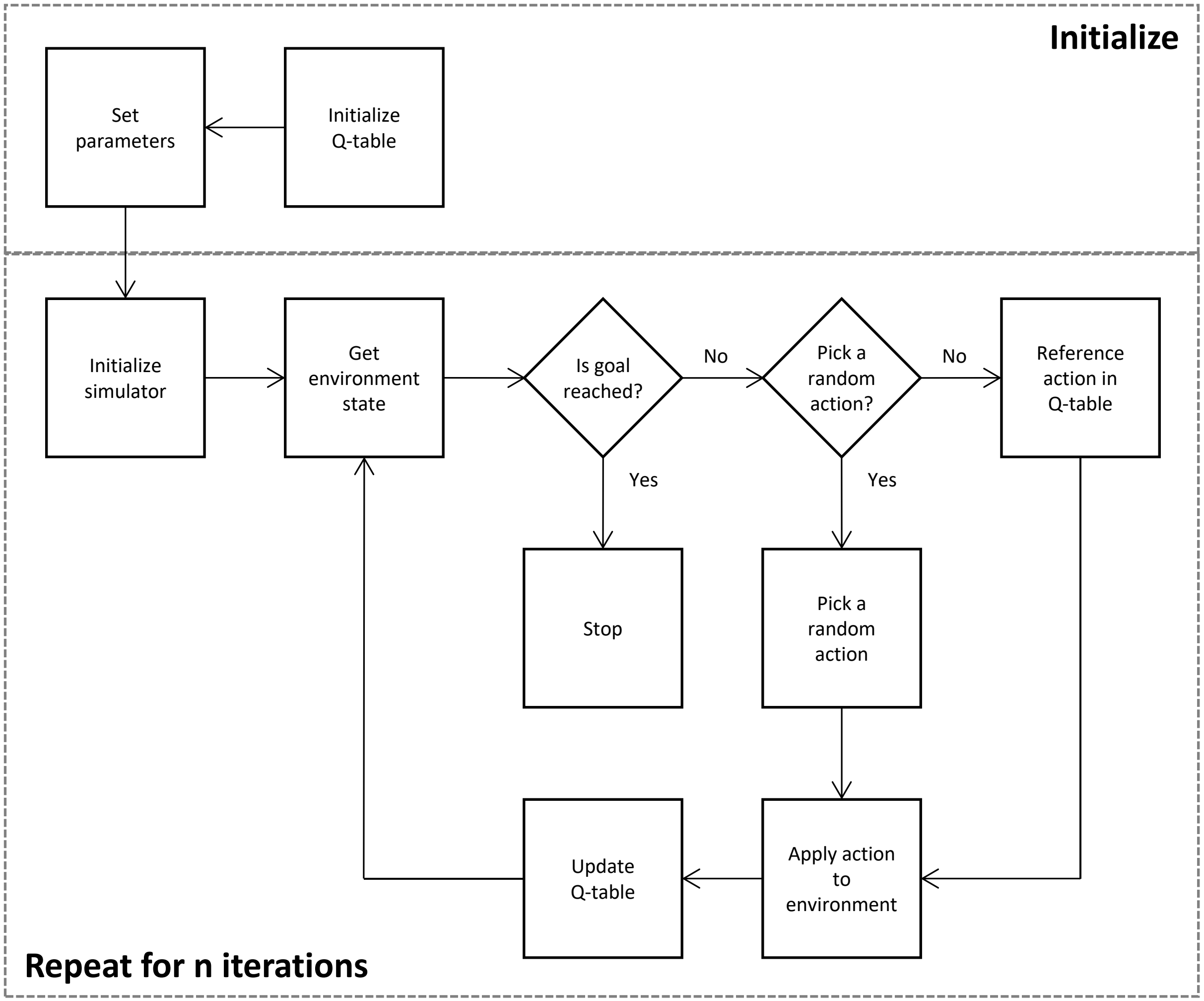
Exploration vs. Exploitation

The crucial tradeoff the gambler faces at each trial is between "exploitation" of the machine that has the highest expected payoff and "exploration" to get more information about the expected payoffs of the other machines.

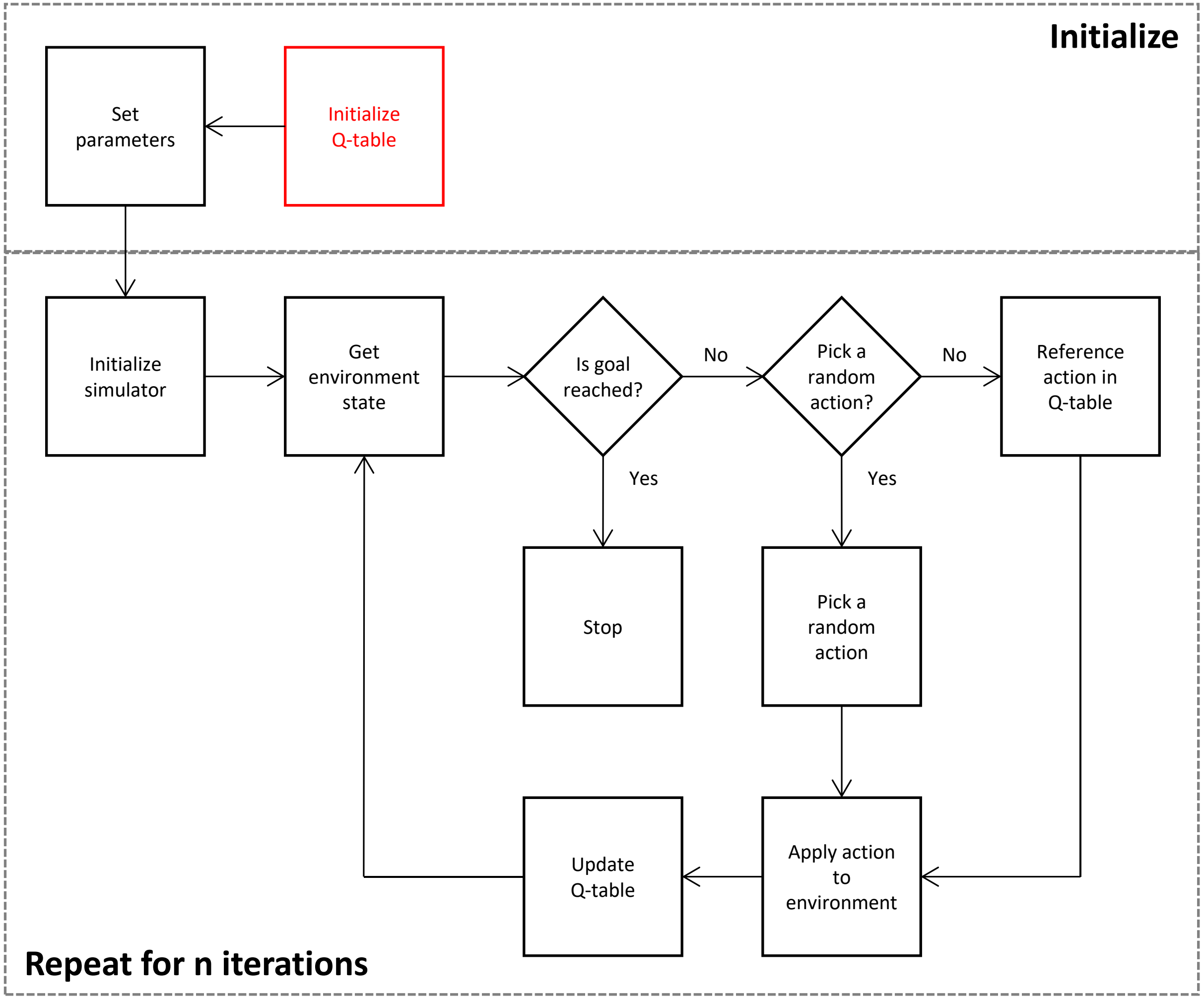
ε-greedy Algorithm

```
generate random number p ∈ [0,1]

if (p < ε)          // explore

    select random arm

else                // exploit

    select current best arm

end
```

Q-Learning Algorithm



Q-Learning Algorithm

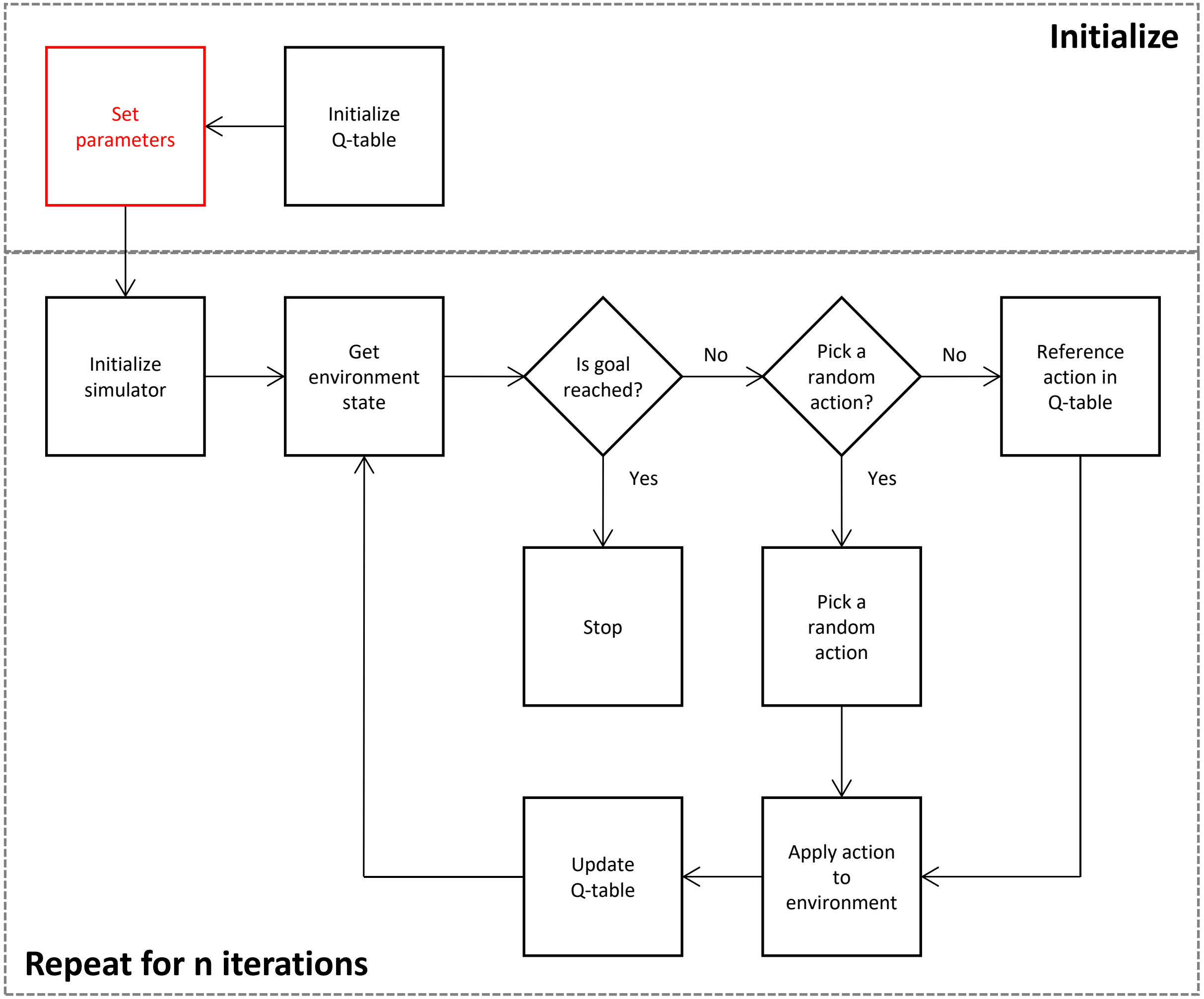


Initialize Q-table:
Set up and initialize (**all values set to 0**) a table where:

- rows represent **possible states**
- columns represent **actions**

Note that additional states can be added to the table when encountered.

Q-Learning Algorithm

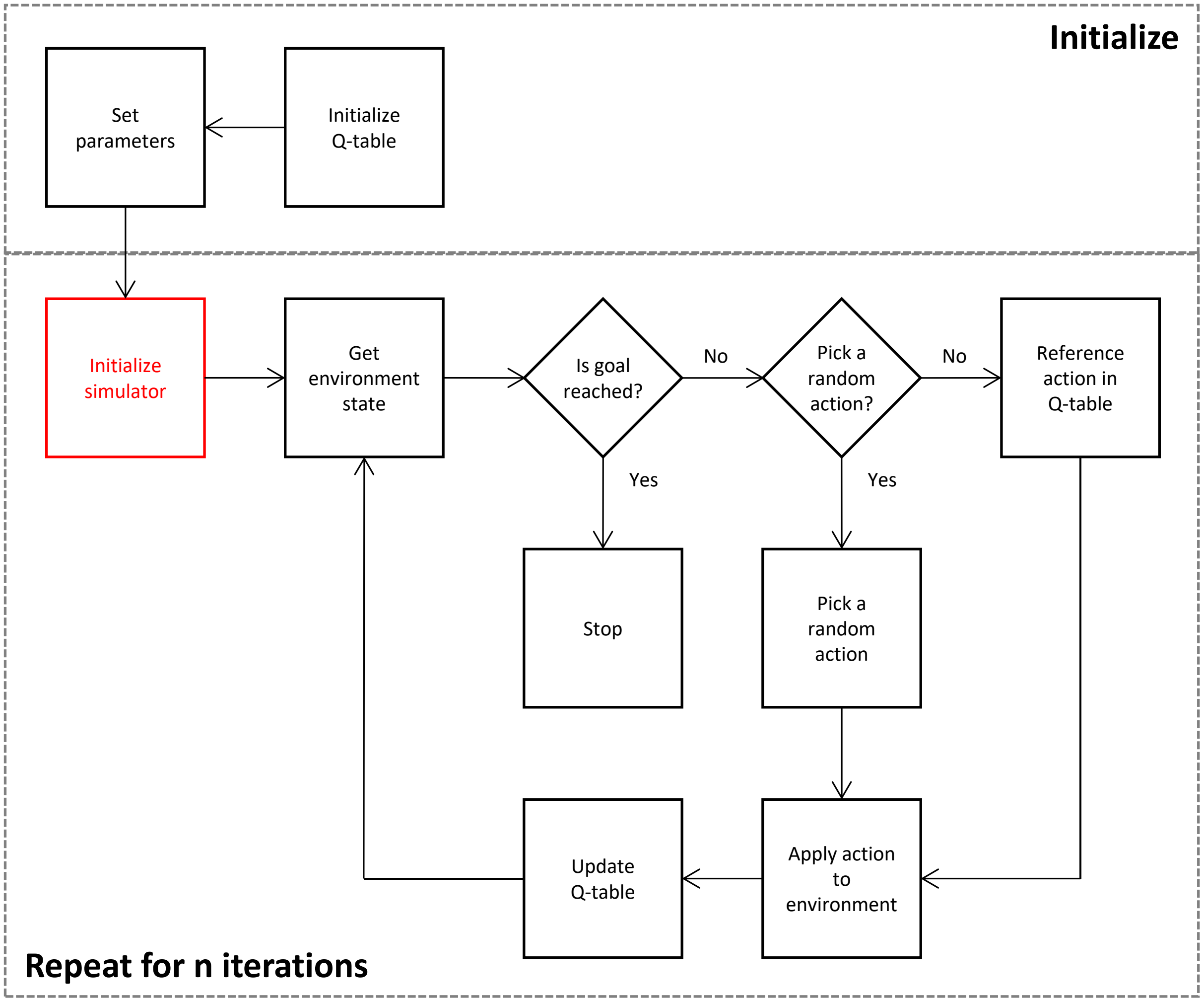


Set parameters:
Set and initialize **hyperparameters** for the Q-learning process.

Hyperparemeters include:

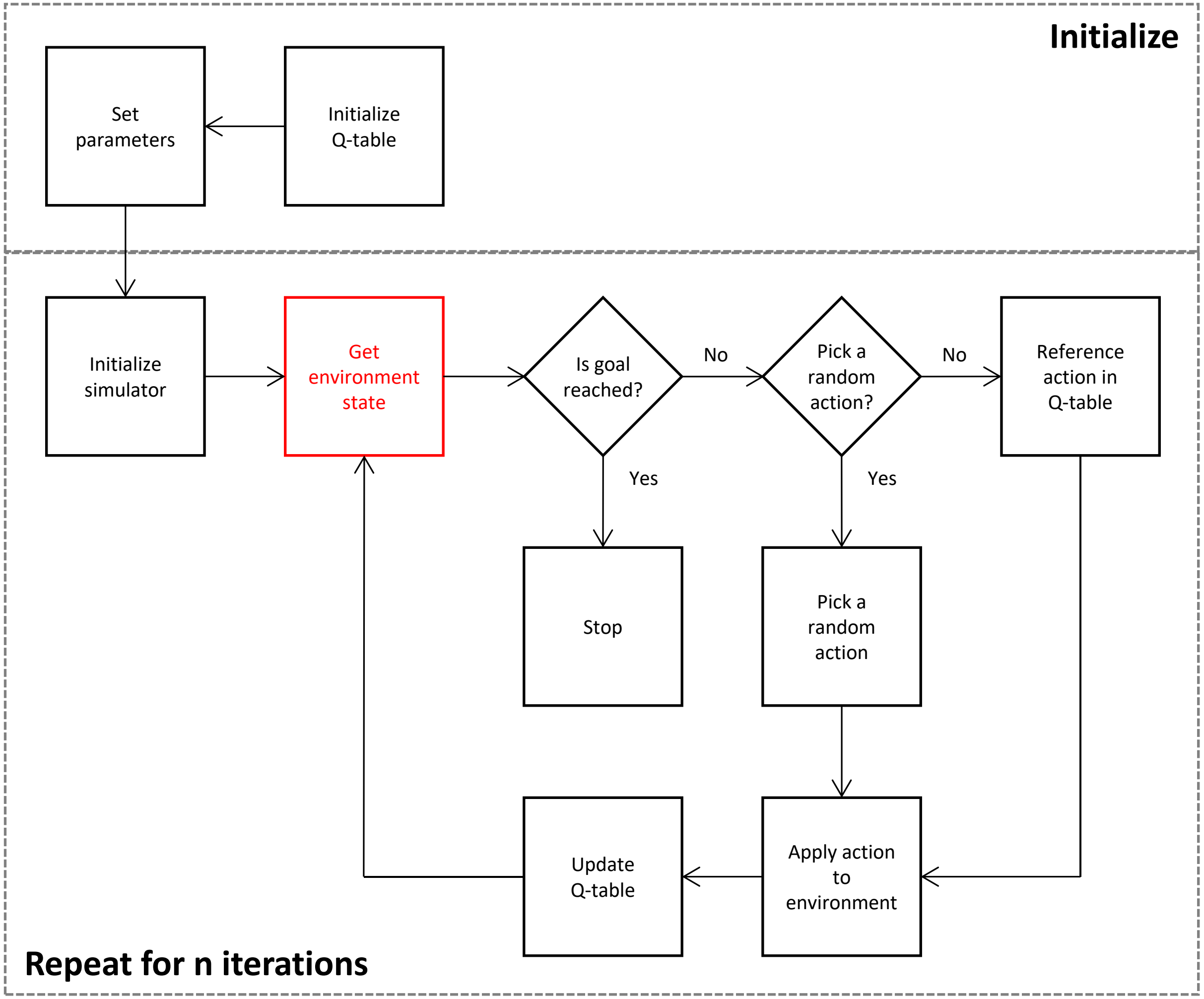
- **chance of choosing a random action**: a threshold for choosing a random action over an action from the Q-table
- **learning rate**: a parameter that describes how quickly the algorithm should learn from rewards in different states
  - high: faster learning with erratic Q-table changes
  - low: gradual learning with possibly more iterations
- **discount factor**: a parameter that describes how valuable are future rewards. It tells the algorithm whether it should seek “immediate gratification” (small) or “long-term reward” (large)

Q-Learning Algorithm



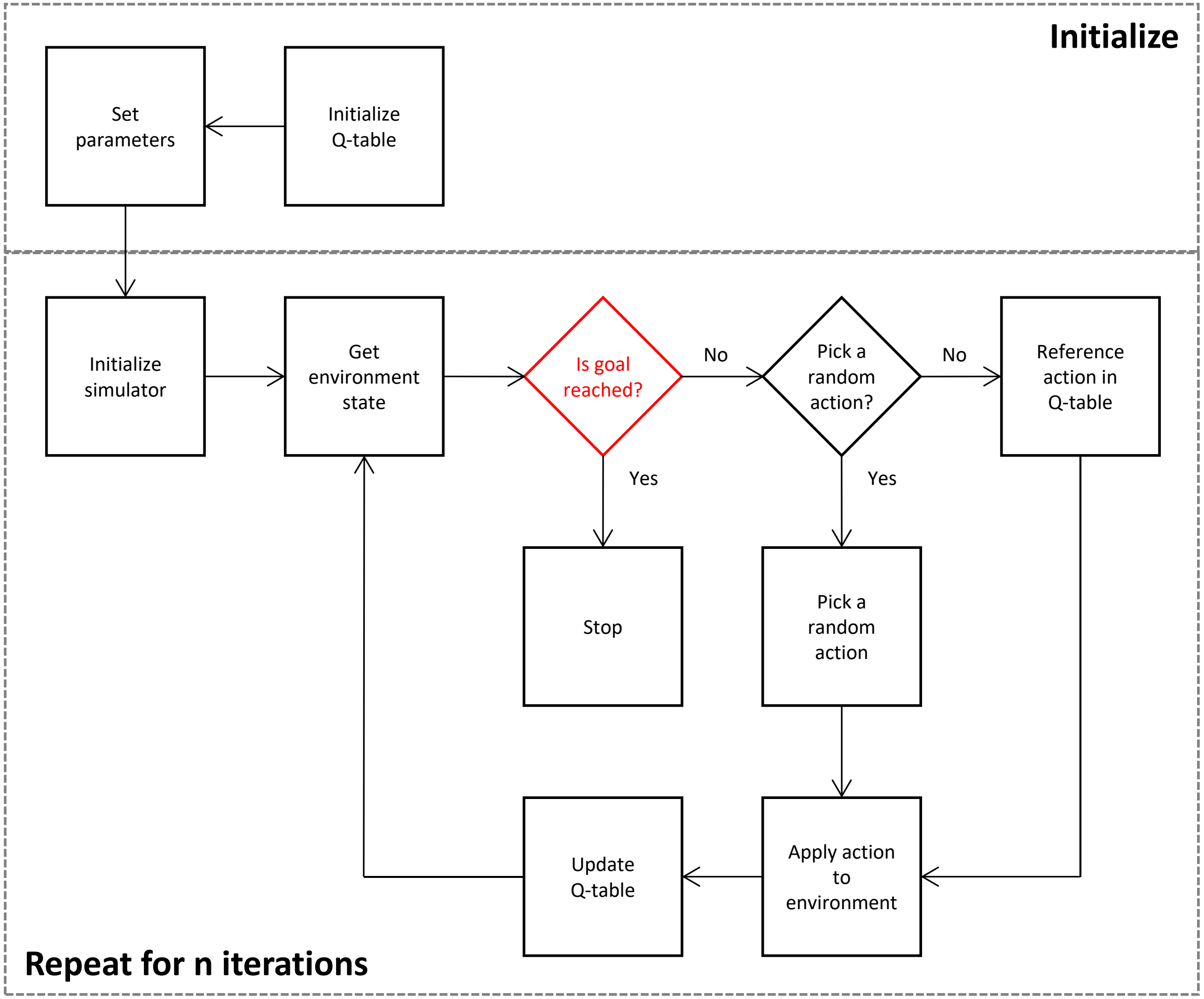
Initialize simulator:
Reset the simulated environment to its initial state and place the agent in a neutral state.

Q-Learning Algorithm



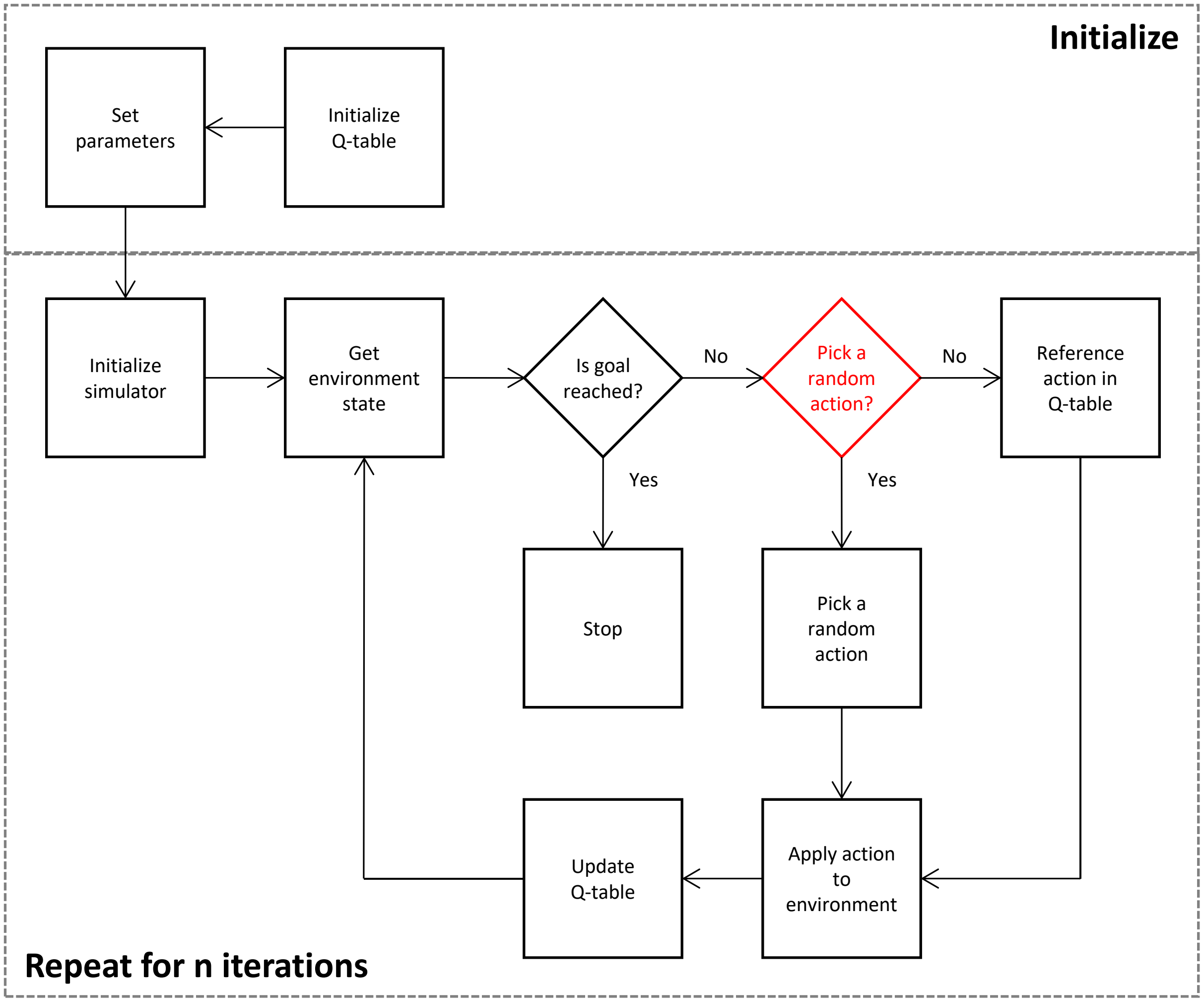
Get environment state:
Report the current state of the environment. Typically a vector of values representing all relevant variables.

Q-Learning Algorithm



Is goal reached?:
Verify if the goal of the simulation has been achieved. It could be decided with the agent arriving in expected final state or by some simulation parameter.

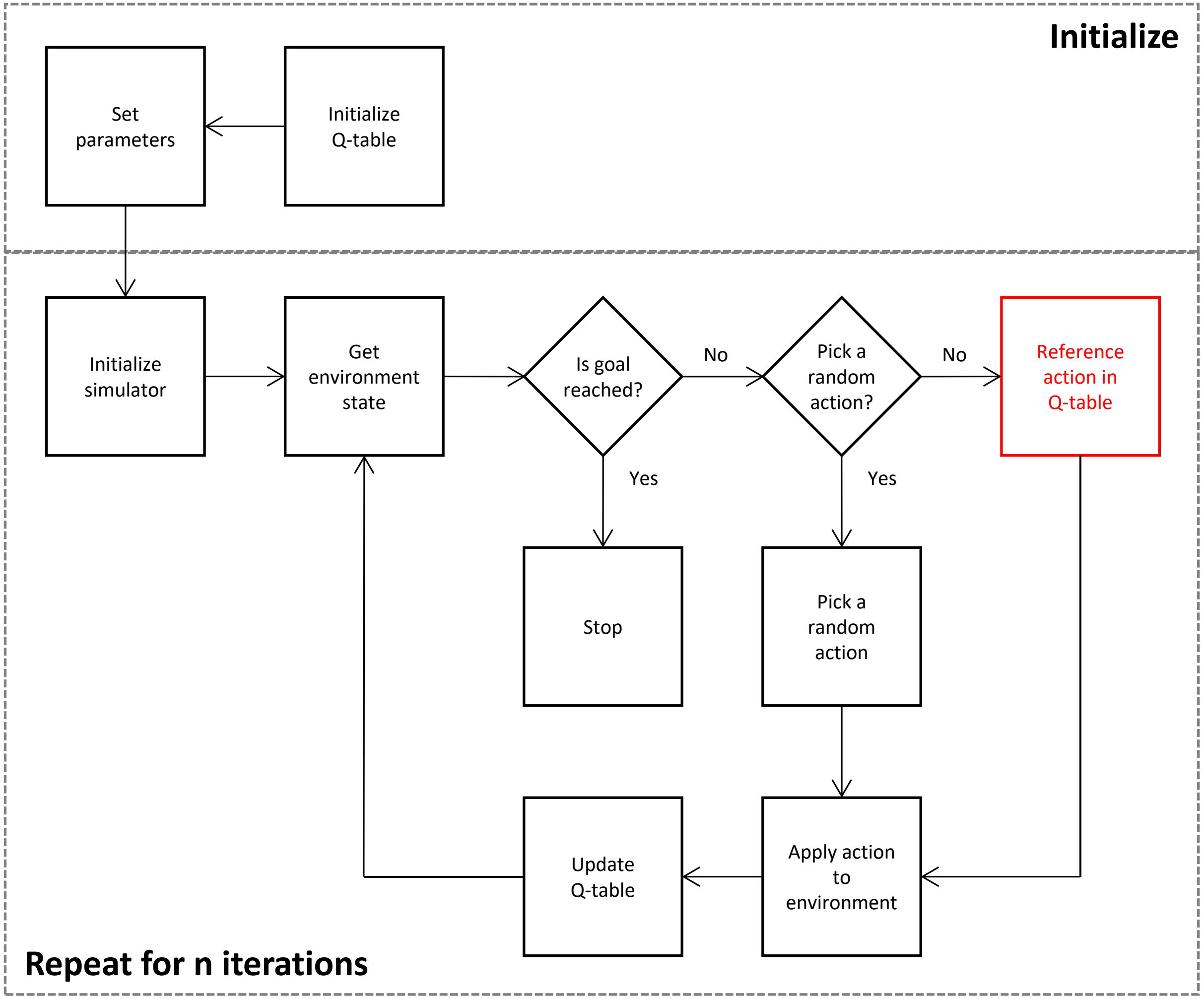
Q-Learning Algorithm



Pick a random action?:
Decide whether next action should be picked at random or not (it will be selected based on Q-table data then).

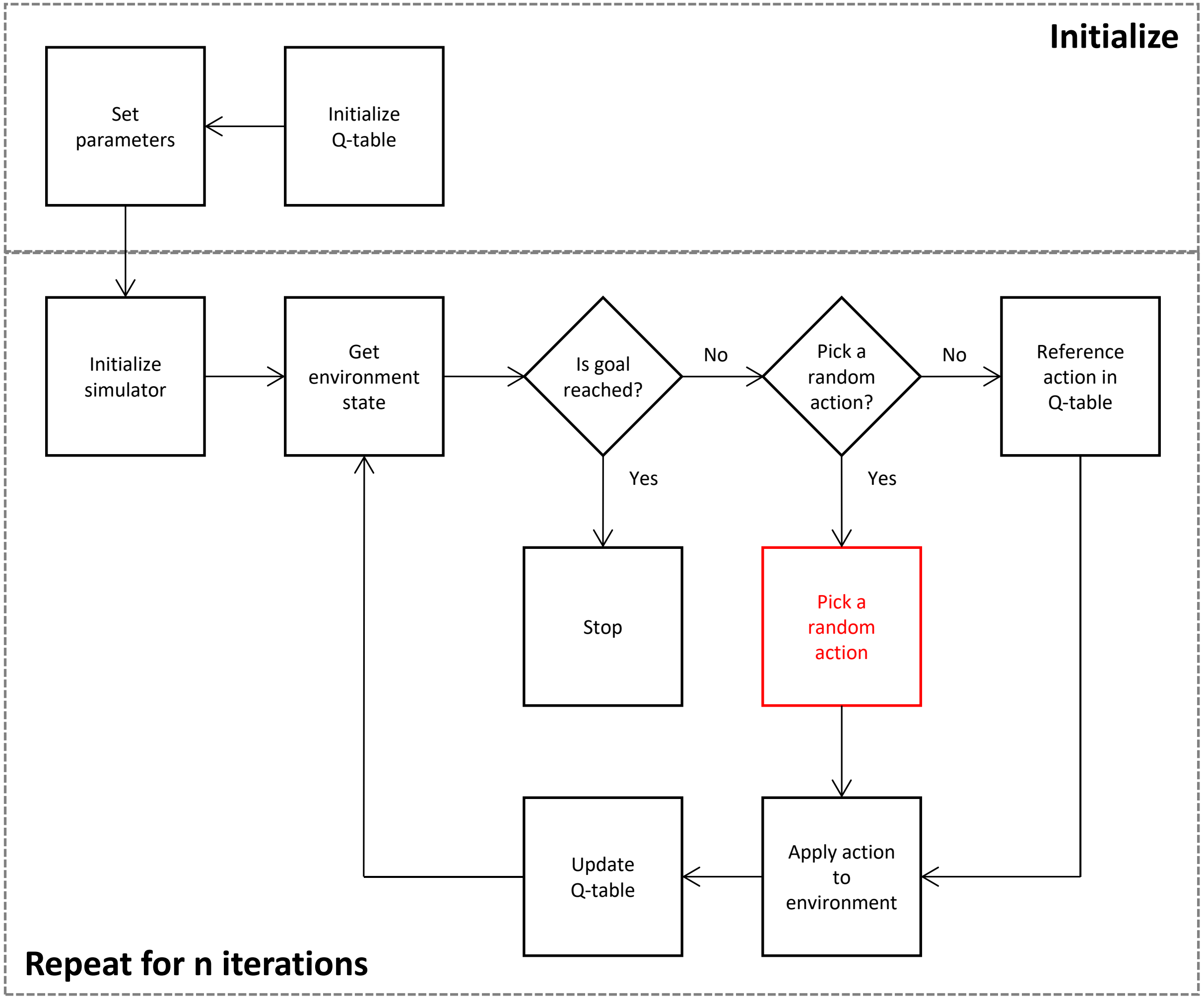
Use the **chance of choosing a random action hyperparameter** to decide.

Q-Learning Algorithm



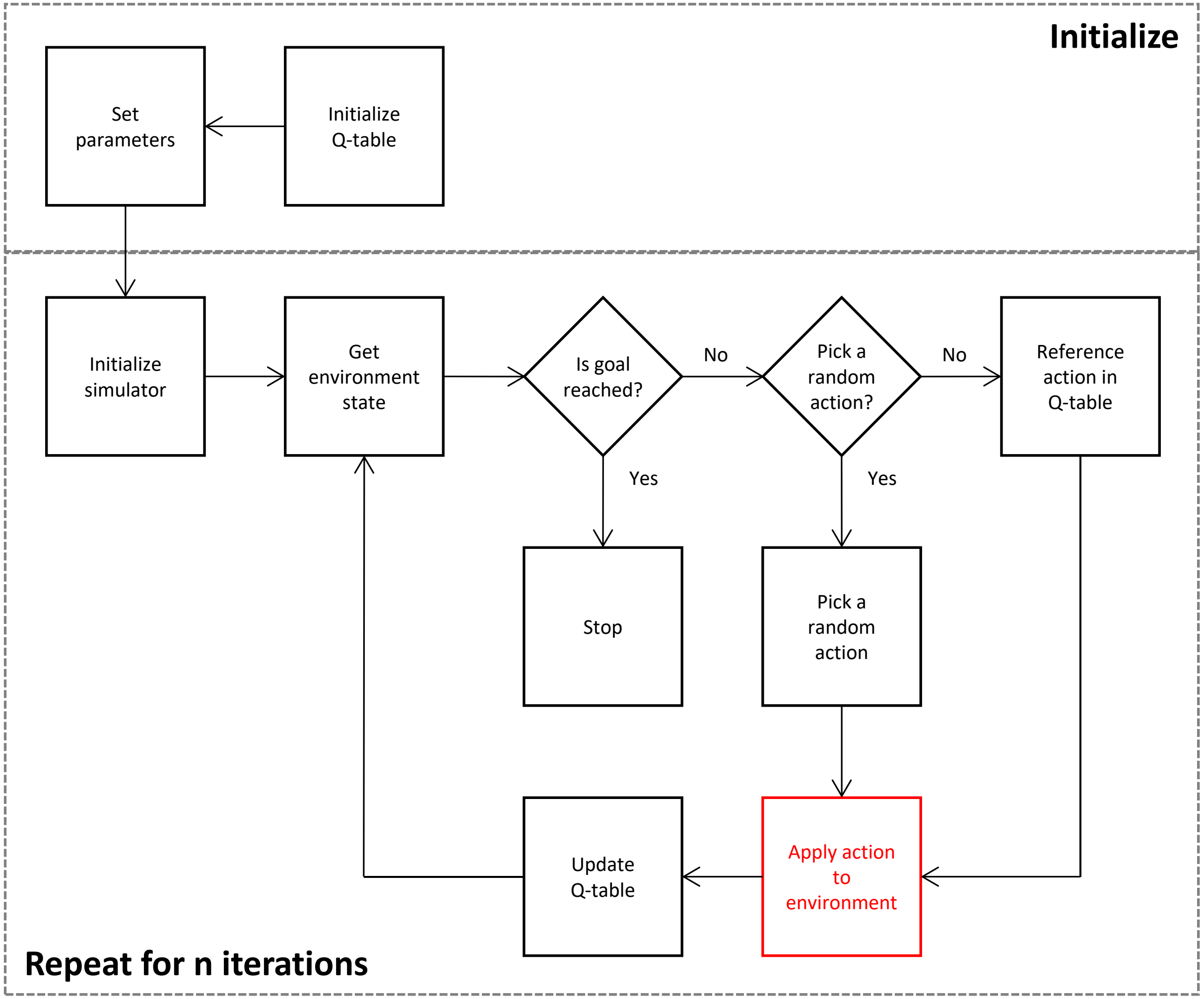
Reference action in Q-table:
Next action decision will be based on data from the Q-table given the current state of the environment.

Q-Learning Algorithm



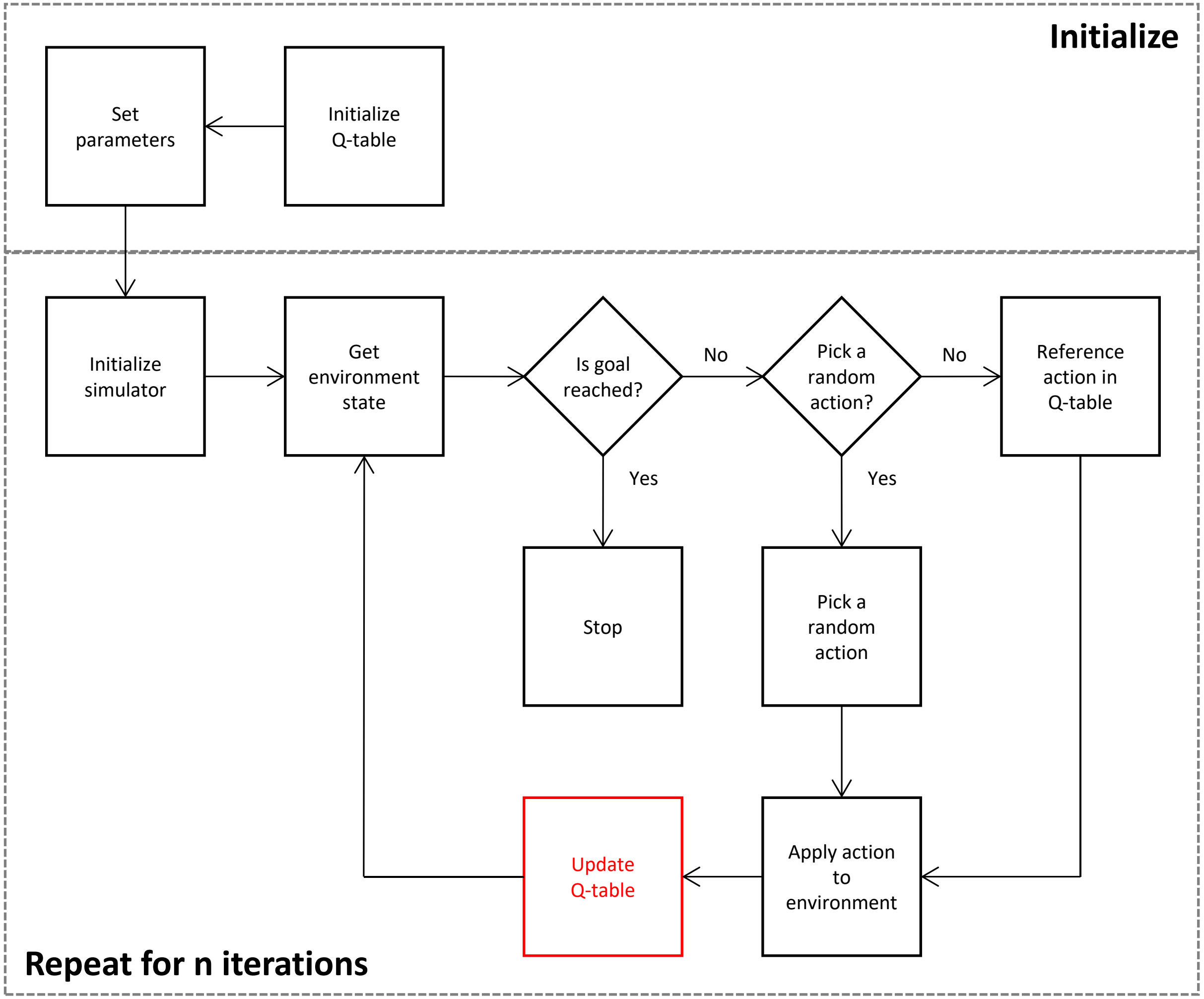
Pick a random action:
Pick any of the available actions at random. Helpful with exploration of the environment.

Q-Learning Algorithm



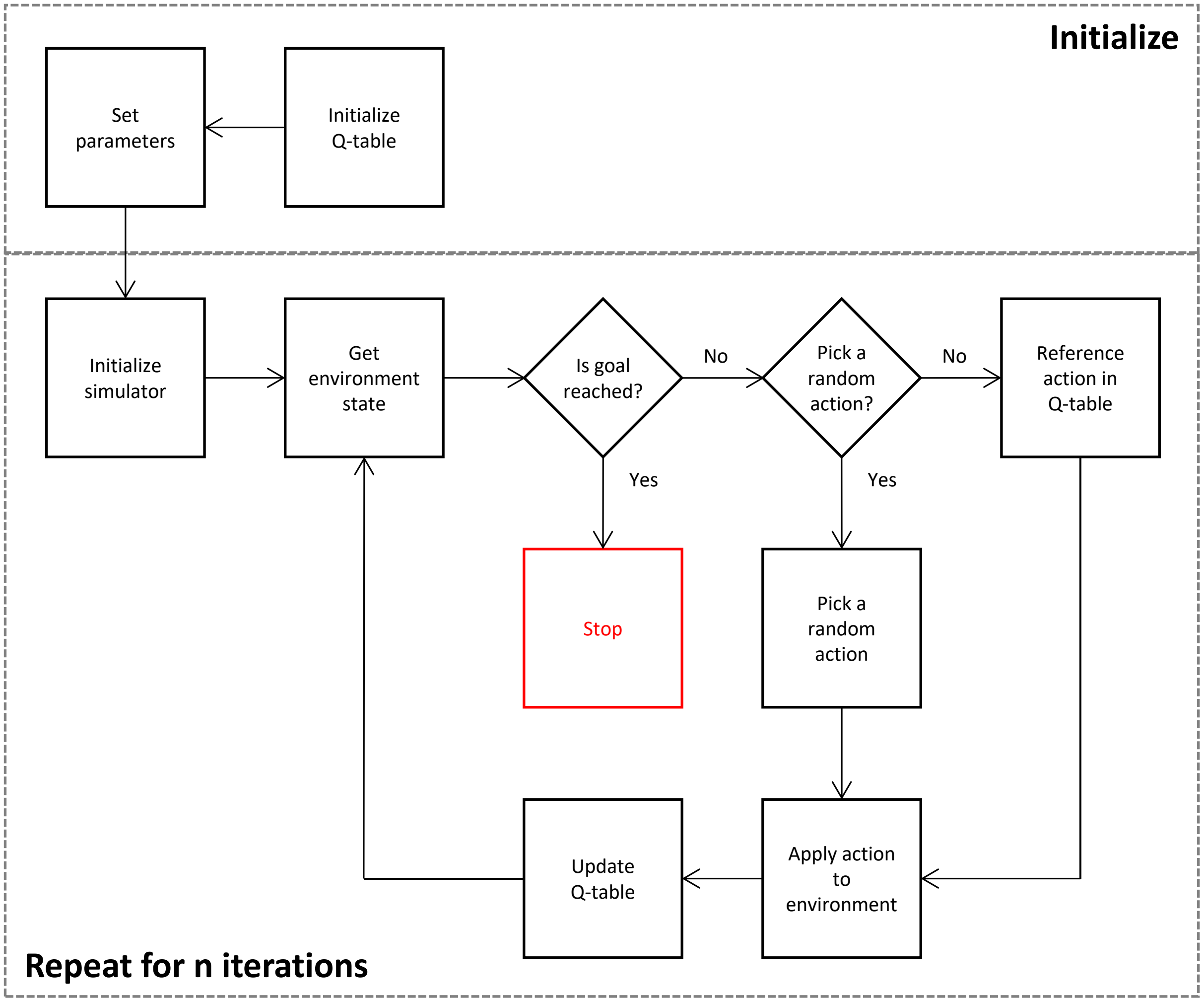
Apply action to environment:
Apply the action to the environment to change it. Each action will have its own reward.

Q-Learning Algorithm



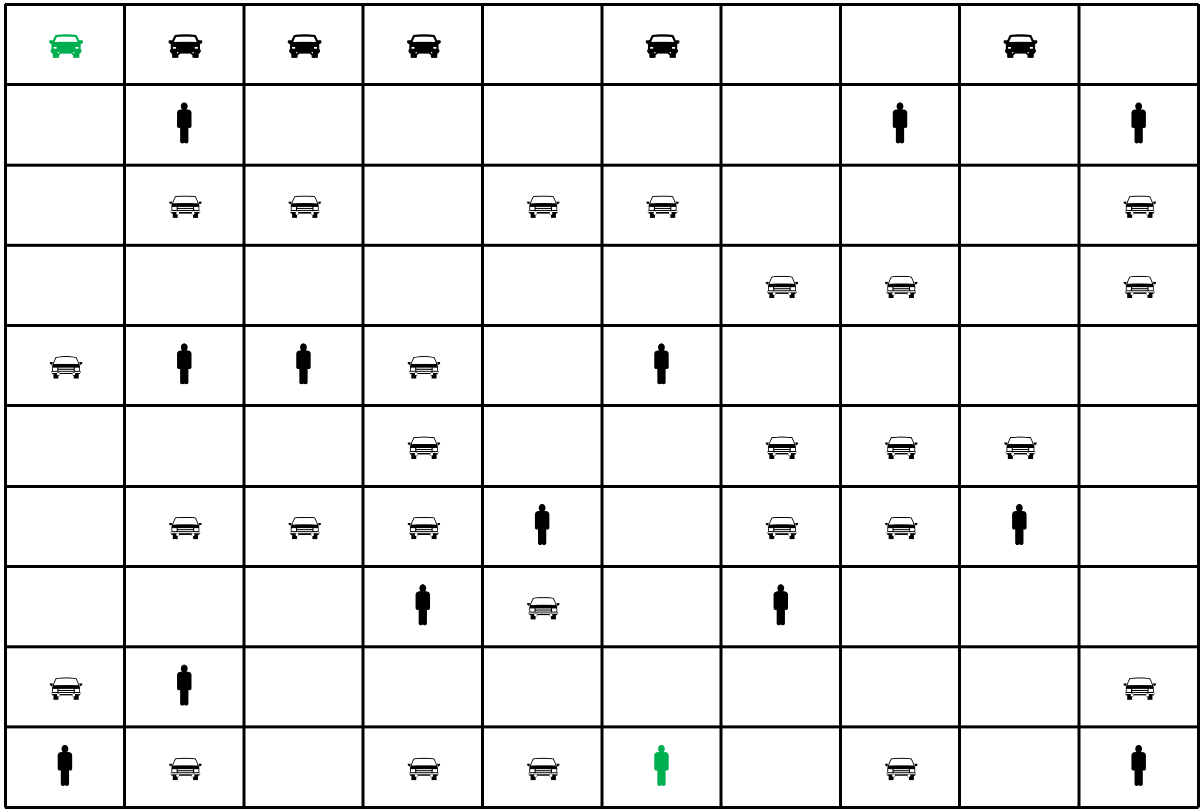
Update Q-table:
Update the Q-table given the reward resulting from recently applied action (feedback from the environment).

Q-Learning Algorithm



Stop:
Stop the learning process

Q-Learning Algorithm



Q-table		Actions			
		↑	↓	→	←
States	1	0	0	0	0
	2	0	0	0	0
	…	…	…	…	…
	n	0	0	0	0

Rewards:
Move into car: -100
Move into pedestrian: -1000
Move into empty space: 100
Move into goal: 500

Action:

Reward:

Q-table value:

Learning rate

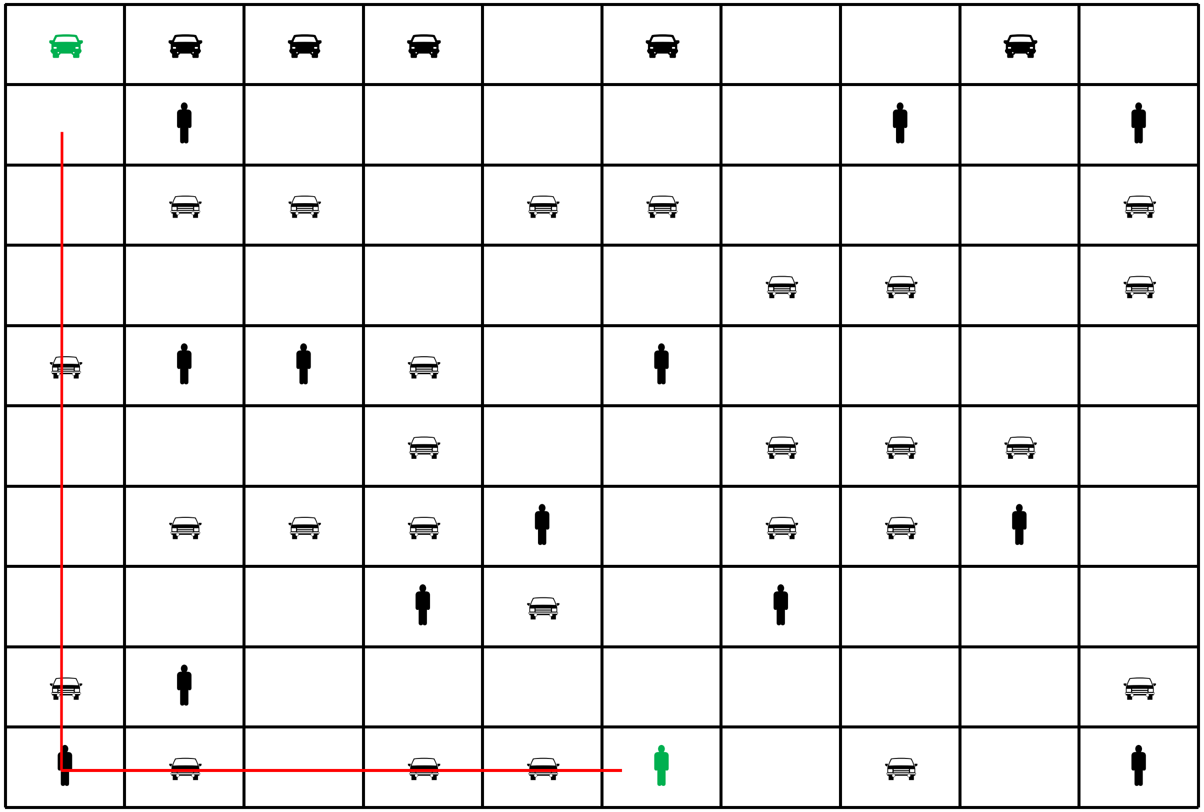
Discount

$$\mathrm{Q(state, action)} = (1 - \mathrm{alpha}) * \mathrm{Q(state, action)} + \mathrm{alpha} * (\mathrm{reward} + \mathrm{gamma} * \mathrm{Q(next\ state, all\ actions)})$$

Current value

Maximum value of all actions on next state

Q-Learning Algorithm



Q-table		Actions			
		↑	↓	→	←
States	1	0	0	0	0
	2	0	0	0	0
	…	…	…	…	…
	n	0	0	0	0

Rewards:

Move into car: -100

Move into pedestrian: -1000

Move into empty space: 100

Move into goal: 500

Action:

Reward:

Q-table value:

Learning rate

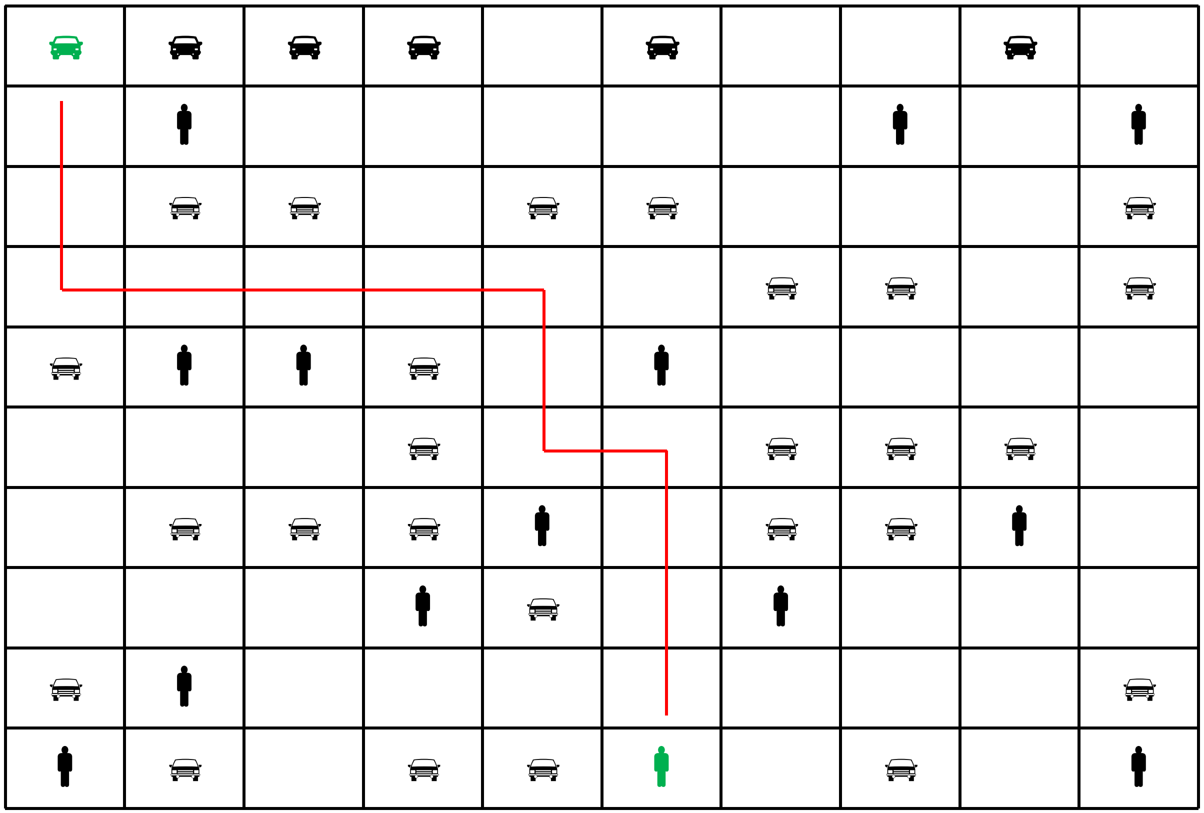
Discount

$$Q(\text{state, action}) = (1 - \text{alpha}) * Q(\text{state, action}) + \text{alpha} * (\text{reward} + \text{gamma} * Q(\text{next state, all actions}))$$

Current value

Maximum value of all actions on next state

Q-Learning Algorithm



Q-table		Actions			
		↑	↓	→	←
States	1	0	0	0	0
	2	0	0	0	0
	…	…	…	…	…
	n	0	0	0	0

Rewards:

Move into car: -100

Move into pedestrian: -1000

Move into empty space: 100

Move into goal: 500

Action:

Reward:

Q-table value:

Learning rate

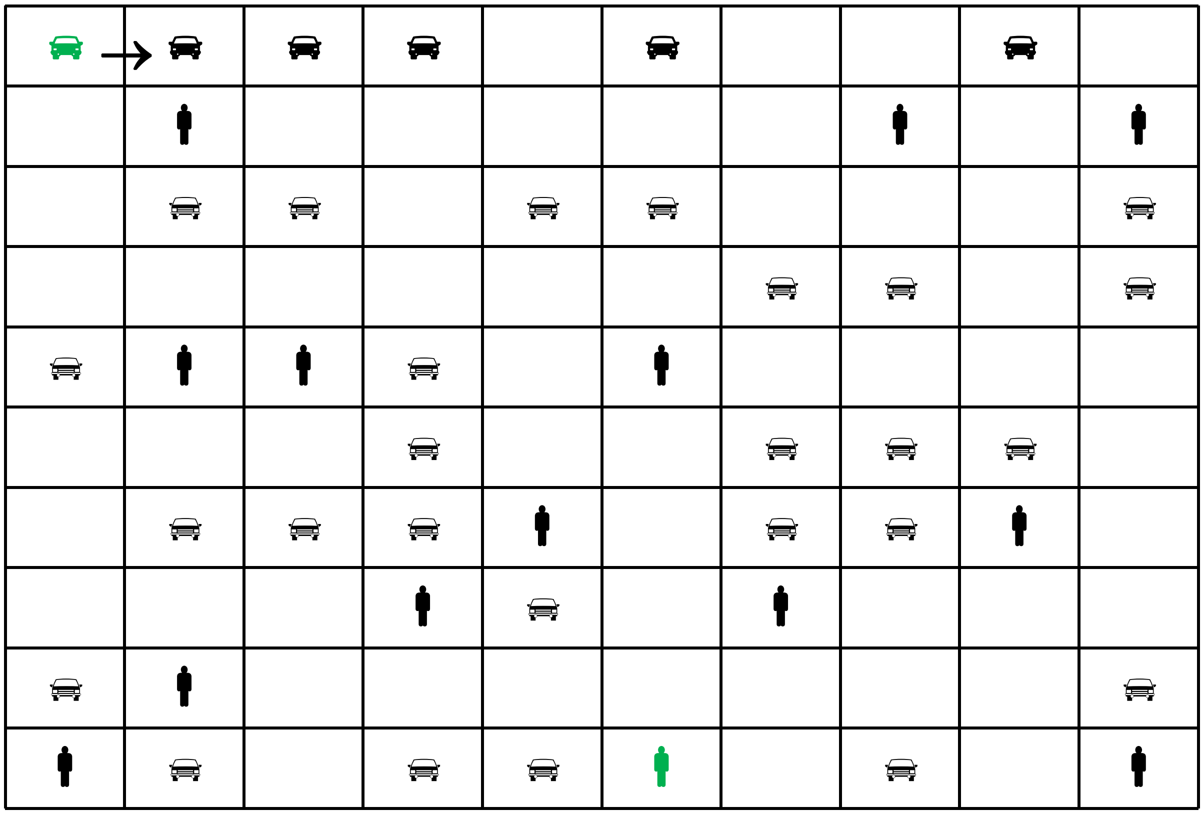
Discount

$$\text{Q(state, action)} = (1 - \text{alpha}) * \text{Q(state, action)} + \text{alpha} * (\text{reward} + \text{gamma} * \text{Q(next state, all actions)})$$

Current value

Maximum value of all actions on next state

Q-Learning Algorithm



Q-table		Actions			
		↑	↓	→	←
States	1	0	0	0	0
	2	0	0	0	0
	…	…	…	…	…
	n	0	0	0	0

Rewards:
Move into car: -100
Move into pedestrian: -1000
Move into empty space: 100
Move into goal: 500

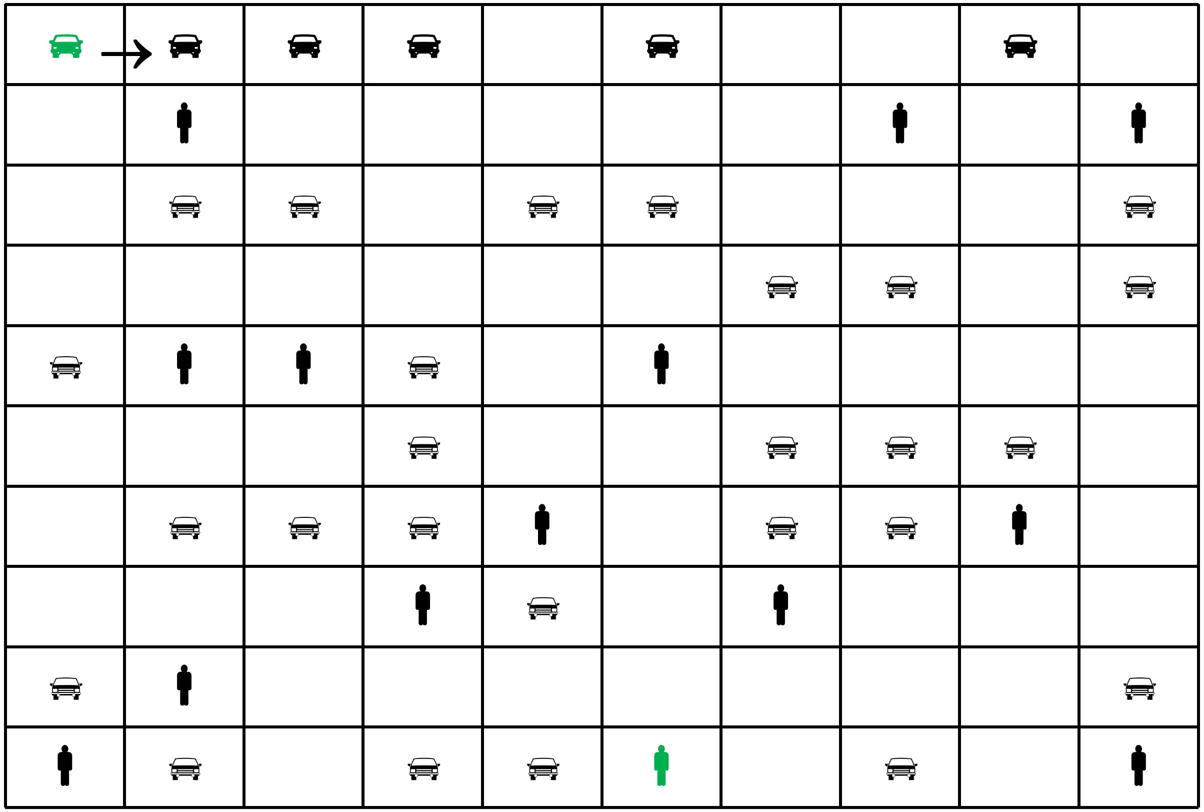
Action: →

Reward: -100

Q-table value:

$$Q(1, \text{east}) = (1 - 0.1) * 0 + 0.1 * (-100 + 0.6 * \max \text{ of } Q(2, \text{all actions}))$$

Q-Learning Algorithm



Q-table		Actions			
		↑	↓	→	←
States	1	0	0	-10	0
	2	0	0	0	0
	…	…	…	…	…
	n	0	0	0	0

Rewards:
Move into car: -100
Move into pedestrian: -1000
Move into empty space: 100
Move into goal: 500

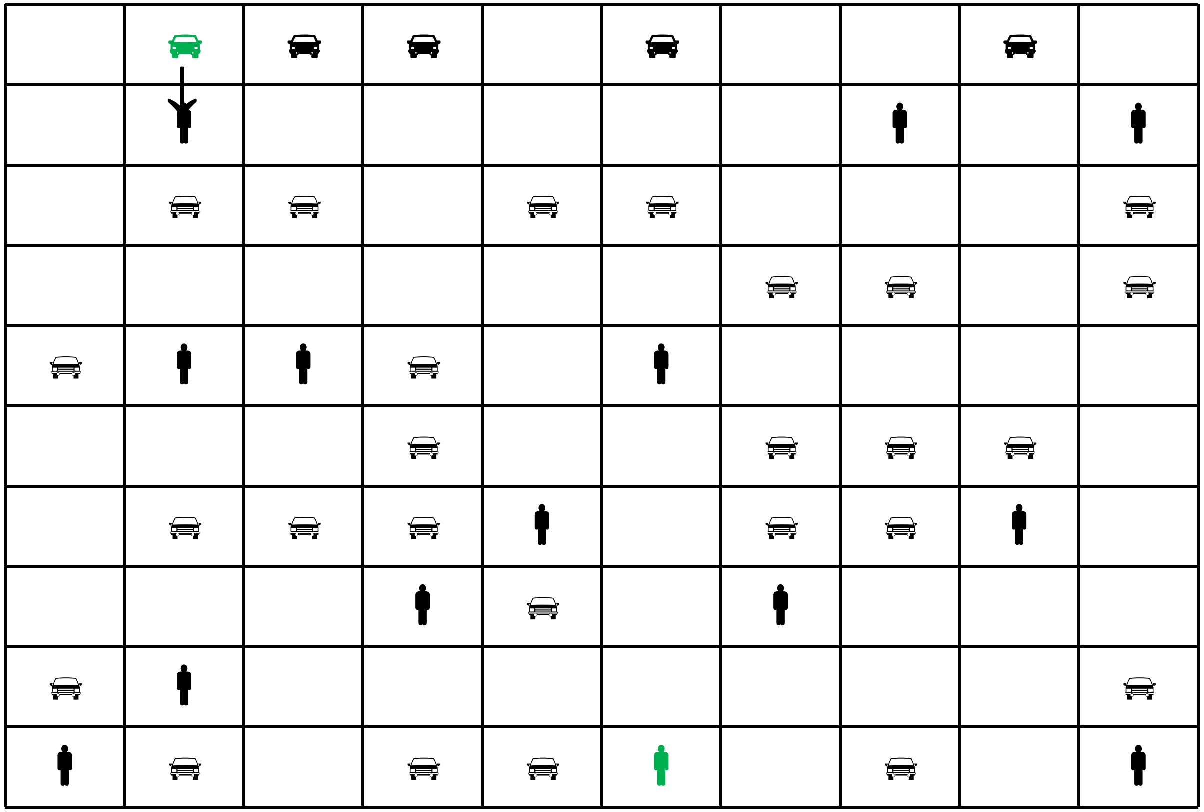
Action: →

Reward: -100

Q-table value:

$$Q(1, \text{east}) = (1 - 0.1) * 0 + 0.1 * (-100 + 0.6 * 0) = \; -10$$

Q-Learning Algorithm



Q-table		Actions			
		↑	↓	→	←
States	1	0	0	-10	0
	2	0	0	0	0
	…	…	…	…	…
	n	0	0	0	0

Rewards:
Move into car: -100
Move into pedestrian: -1000
Move into empty space: 100
Move into goal: 500

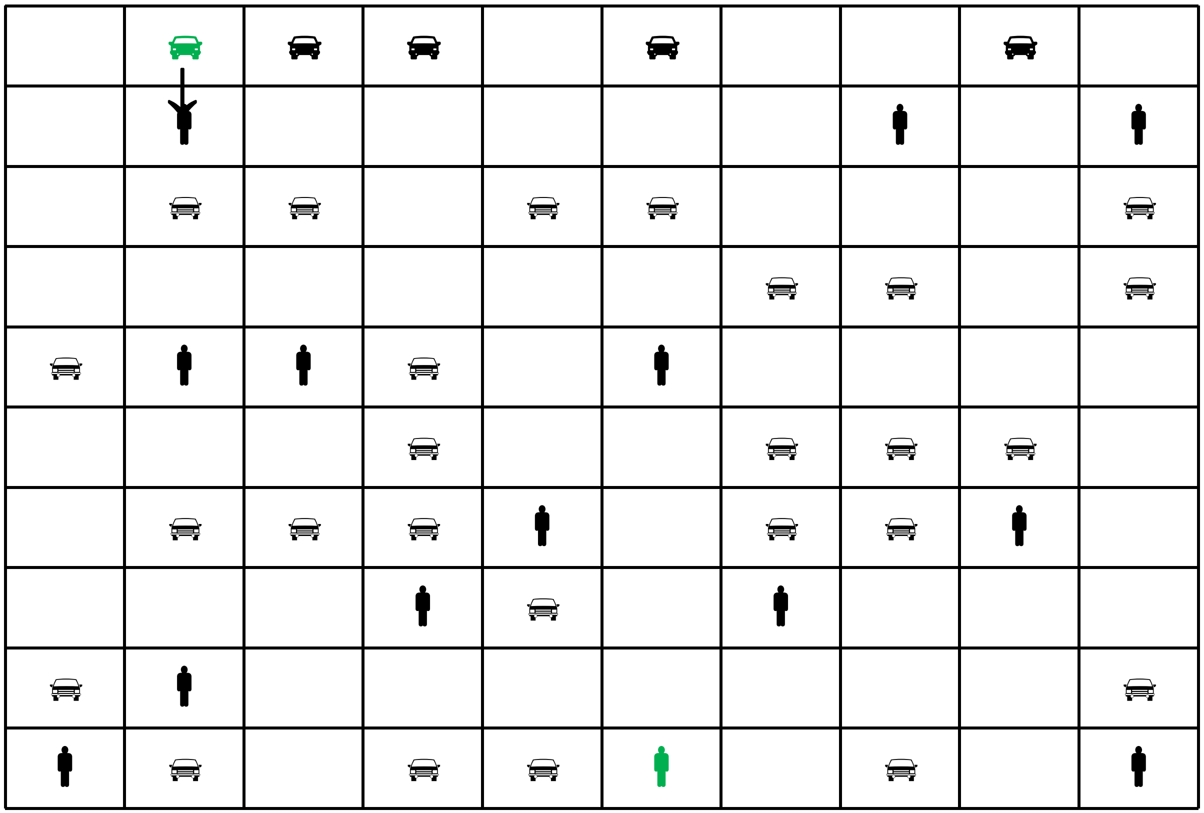
Action: →

Reward: -1000

Q-table value:

$$Q(2, \text{south}) = (1 - 0.1) * 0 + 0.1 * (-1000 + 0.6 * \text{max of } Q(3, \text{all actions}))$$

Q-Learning Algorithm



Q-table

States		Actions			
		↑	↓	→	←
	1	0	0	-10	0
	2	0	-100	0	0
	...	...	...	...	...
	n	0	0	0	0

Rewards:
Move into car: -100
Move into pedestrian: -1000
Move into empty space: 100
Move into goal: 500

Action: →

Reward: -1000

Q-table value:

$$Q(2, south) = (1 - 0.1) * 0 + 0.1 * (-1000 + 0.6 * 0) = -100$$

Deep Reinforcement Learning