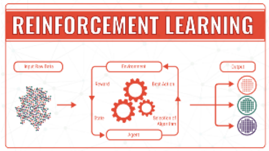
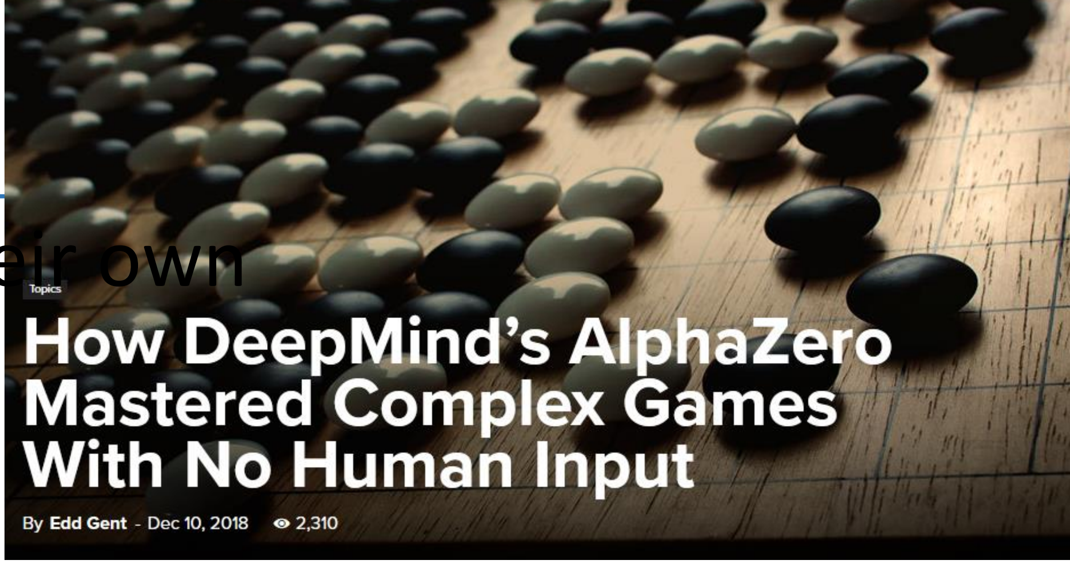
Reinforcement Learning



Agenda

- Artificial Intelligence
- Agents
- Reinforcement Learning
- OpenAI Gym

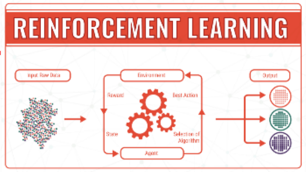
Playing Games with AI



- AI used to play computer games on their own
- Deepmind.
- AlphaGo program
  - defeated the South Korean Go world champion in 2016.
- There had been many successful attempts in the past to develop agents with the intent of playing Atari games like Breakout, Pong, and Space Invaders.
- These programs follow a paradigm of Machine Learning known as Reinforcement Learning.

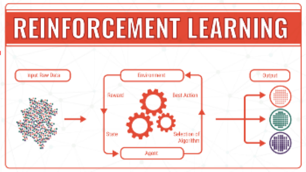
Artificial Intelligence, or AI

- Artificial Intelligence is the synthesis and analysis of computational agents that act intelligently.
- An agent is something that acts in an environment.
- An agent acts intelligently if:
  - Its actions are appropriate for its goals and circumstances
  - It is flexible to changing environments and goals
  - It learns from experience
  - It makes appropriate choices given perceptual and computational limitations



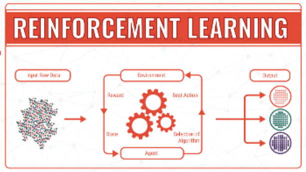
Agent

- An agent is something that acts in an environment - it does something
- We are interested in what an agent does;
  - how it **acts**. We judge an agent by its actions.



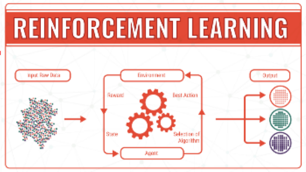
Computational agent

- A computational agent is an agent whose decisions about its actions can be explained in terms of computation.
- The decision can be broken down into primitive operation - can be implemented in a device.
- Computation can take many forms :
  - In humans - computation is carried out in "wetware";
  - in computers - carried out in "hardware."



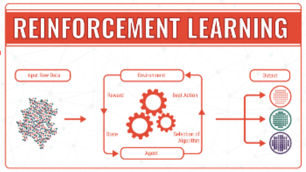
Goals of Artificial Intelligence

- **Scientific goal:** to understand the principles that make intelligent behavior possible in natural or artificial systems.
  - analyze natural and artificial agents
  - formulate and test hypotheses about what it takes to construct intelligent agents
  - design, build, and experiment with computational systems that perform tasks that require intelligence
- **Engineering goal:** design useful, intelligent artifacts.



Intelligence

This idea of intelligence being defined by external behavior was the motivation for a test for intelligence designed by Turing , which has become known as the Turing test.



Turing test

- Turing test consists of an imitation game
- where an interrogator can ask a witness, via a text interface, any question.
- If the interrogator cannot distinguish the witness from a human,
- the witness must be intelligent.
- An agent that is not really intelligent could not fake intelligence for arbitrary topics.

Interrogator: In the first line of your sonnet which reads "Shall I compare thee to a summer's day," would not "a spring day" do as well or better?

Witness: It wouldn't scan.

Interrogator: How about "a winter's day," That would scan all right.

Witness: Yes, but nobody wants to be compared to a winter's day.

Interrogator: Would you say Mr. Pickwick reminded you of Christmas?

Witness: In a way.

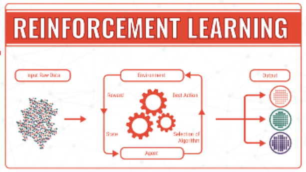
Interrogator: Yet Christmas is a winter's day, and I do not think Mr. Pickwick would mind the comparison.

Witness: I don't think you're serious. By a winter's day one means a typical winter's day, rather than a special one like Christmas.

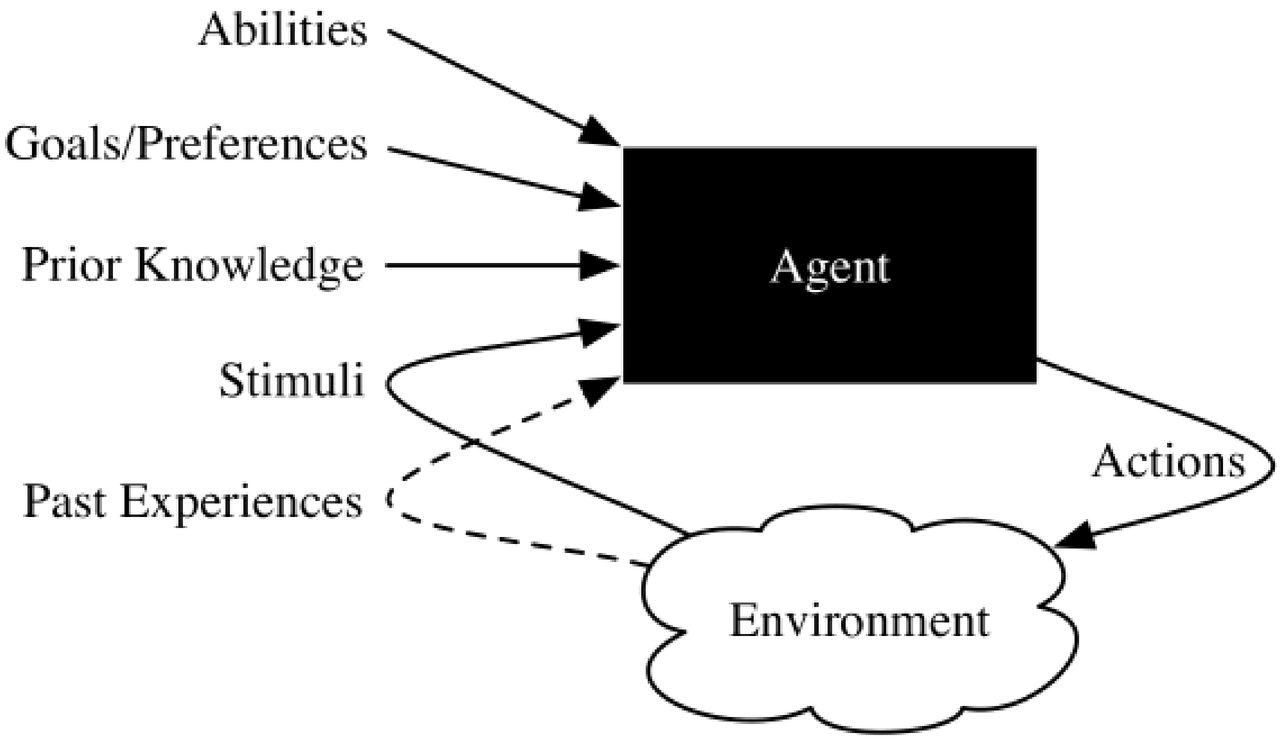
Figure 1.1: Part of Turing's possible dialog for the Turing test

Turing Test

- much debate about the usefulness of Turing test.
- Unfortunately, although it may provide a test for how to recognize intelligence, it does not provide a way to realize intelligence.

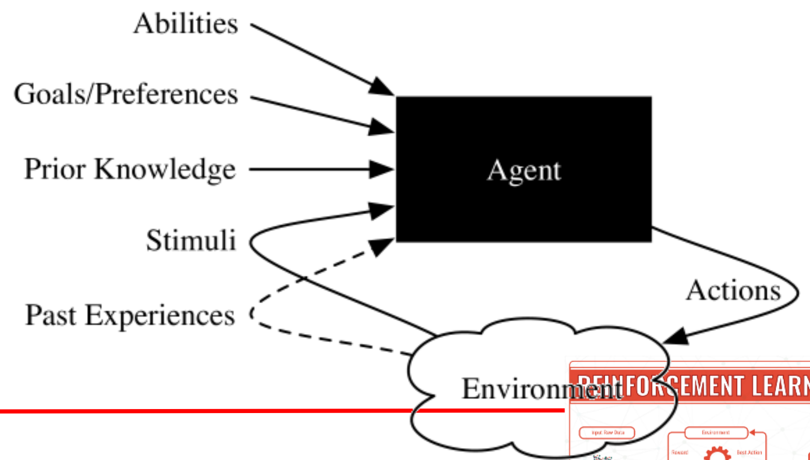


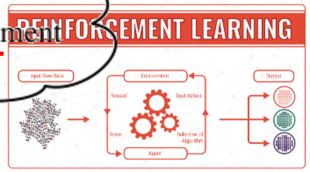
Agents acting in an environment



Inputs to the Agents

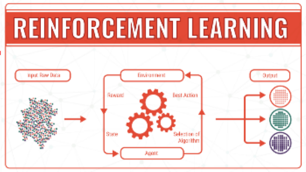
- Abilities — the set of possible actions it can perform
- Goals/Preferences — what it wants, its desires, its values,...
- Prior Knowledge — what it comes into being knowing, what it doesn’t get from experience,...
- History of stimuli
  - (current) stimuli — what it receives from environment now (observations, percepts)
  - past experiences — what it has received in the past





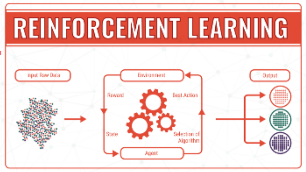
Agent

- An agent in a current state (S_t) takes an action (A_t) to which the environment reacts and responds, returning a new state(S_{t+1}) and reward (R_{t+1}) to the agent.
- Given the updated state and reward, the agent chooses the next action, and the loop repeats until an environment is solved or terminated.



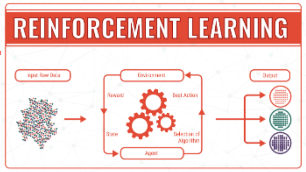
Examples of Agents

- Organisations - Microsoft, FIFA, Government of India,
- People - teacher, physician, stock trader, engineer, researcher,
- Computers/devices - thermostat, user interface, airplane controller, network controller, game, advising system,
- Animals, dog, mouse, bird, insect, worm, bacterium, bacteria...
- book(?), sentence(?), word(?), letter(?)



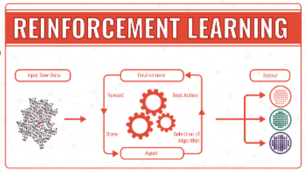
Example agent: autonomous car

- abilities: steer, accelerate, brake
- goals: safety, get to destination, timeliness,. . .
- prior knowledge: what signs mean, what to stop for
- stimuli: vision, laser, GPS. . .
- past experiences: streetmaps, how breaking, steering affects direction..



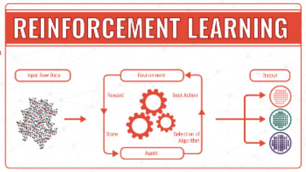
Example agent: robot

- abilities: movement, grippers, speech, facial expressions,. . .
- goals: deliver food, rescue people, score goals, explore,. . .
- prior knowledge: what is important feature, categories of objects, what a sensor tell us,. . .
- stimuli: vision, sonar, sound, speech recognition, gesture recognition,. . .
- past experiences: effect of steering, slipperiness, how people move,. .



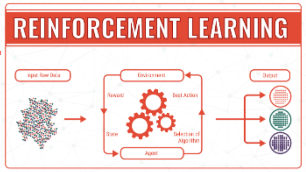
Example agent: teacher

- abilities: present new concept, drill, give test, explain concept,. . .
- goals: particular knowledge, skills, inquisitiveness, social skills,. . .
- prior knowledge: subject material, teaching strategies,. . .
- stimuli: test results, facial expressions, errors, focus,. . .
- past experiences: prior test results, effects of teaching strategies, . . .

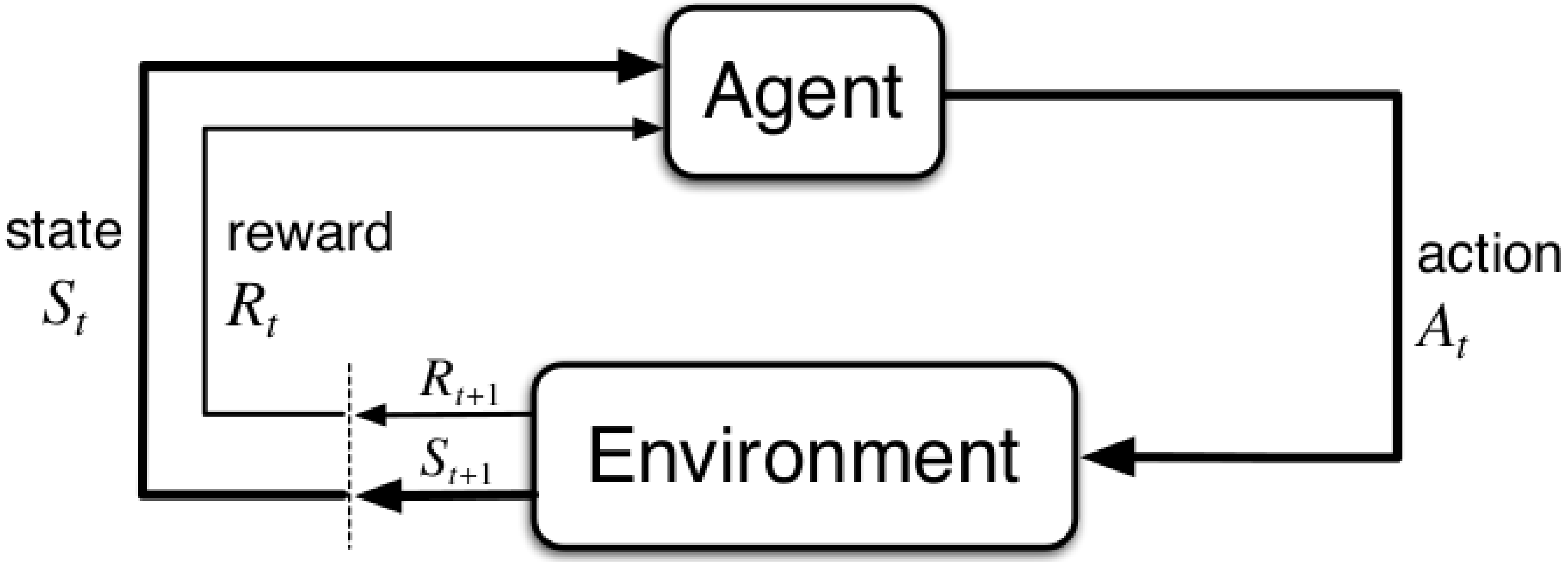


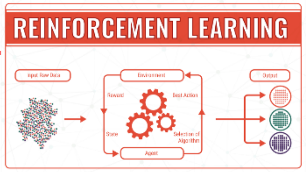
Reinforcement Learning (RL)

- is a computational approach where an agent interacts with an environment by taking actions in which it tries to maximize an accumulated reward.
- RL is currently excelling in many game environments,
- it is a novel way to solve problems that require optimal decisions and efficiency, and will surely play a part in machine intelligence to come.



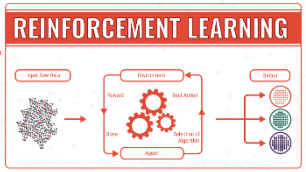
Reinforcement learning





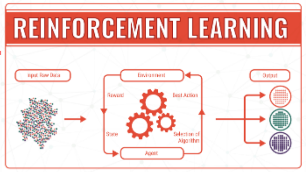
Reinforcement Learning Problem

- A reinforcement learning (RL) agent acts in an environment, observing its state and receiving rewards.
- From its perceptual and reward information, it must determine what to do.



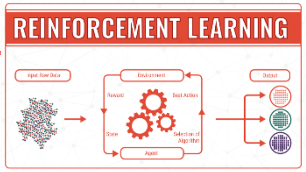
Reinforcement Learning Analogy

- Scenario of teaching a dog new tricks.
- Dog doesn't understand our language, so we can't tell him what to do.
- We follow a different strategy.
- We emulate a situation (or a cue), and the dog tries to respond in many different ways.
- If the dog's response is the desired one, we reward them with snacks.
- The next time the dog is exposed to the same situation, the dog executes a similar action with even more enthusiasm in expectation of more food.
- That's like learning "what to do" from positive experiences.
- Similarly, dogs will tend to learn what not to do when face with negative experiences.



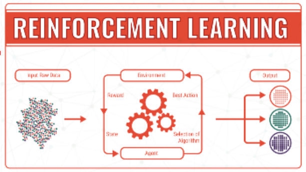
Reinforcement Learning Analogy

- Dog is an "agent" that is exposed to the **environment**.
- The environment - your house, you.
- The situations they encounter are analogous to a **state**.
  - An example of a state could be - dog standing and you use a specific word in a certain tone in the living room
- Agents react by performing an **action** to transition from one "state" to another "state,"
  - Example : Dog goes from standing to sitting
- After the transition, they may receive a **reward** or **penalty** in return.
  - Give them a treat! Or a "No" as a penalty.
- The **policy** is the strategy of choosing an action given a state in expectation of better outcomes.



Reinforcement Learning Steps

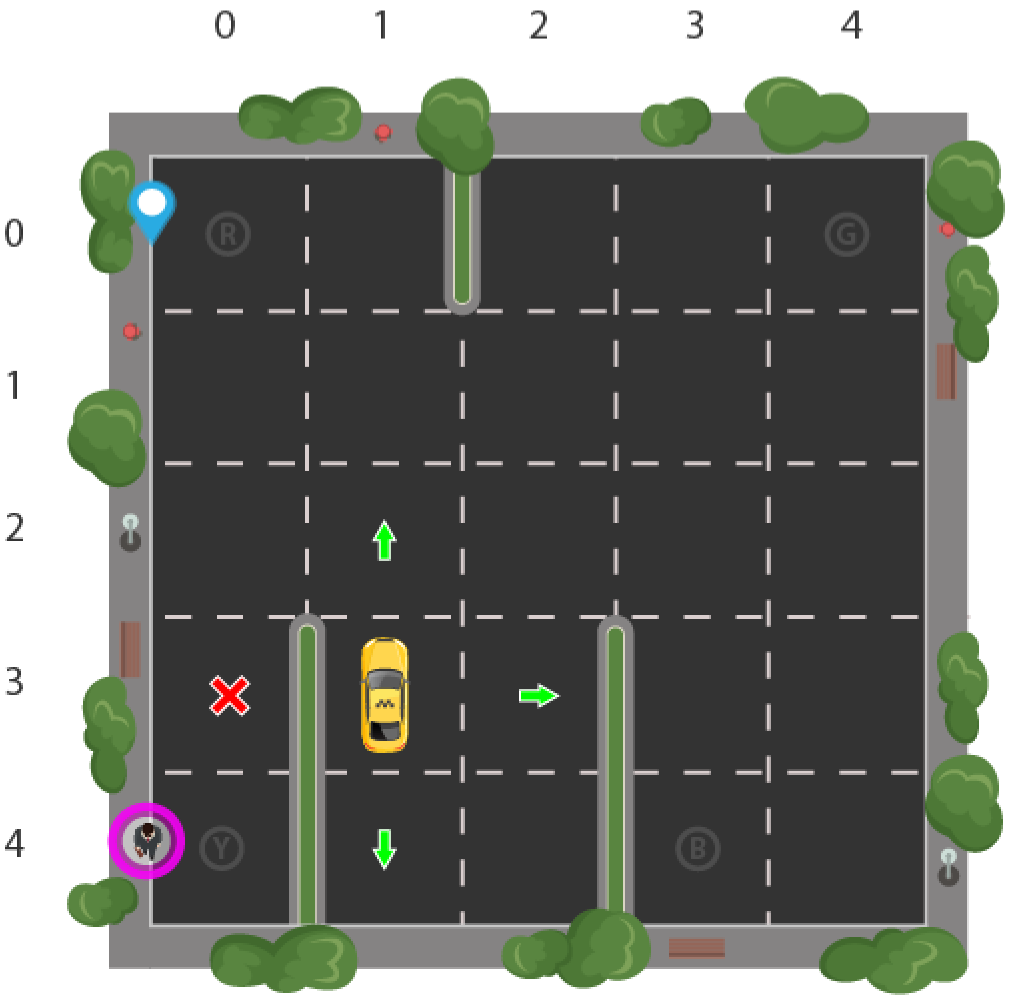
- Observation of the environment
- Deciding how to act using some strategy
- Acting accordingly
- Receiving a reward or penalty
- Learning from the experiences and refining our strategy
- Iterate until an optimal strategy is found



Self-Driving Cab

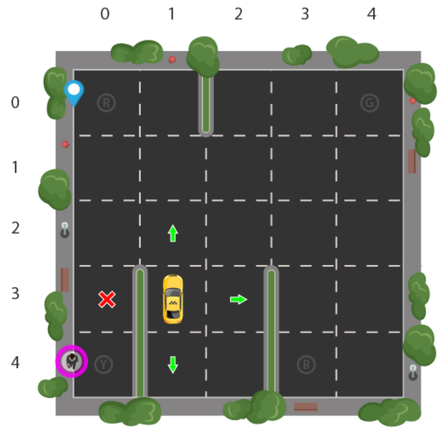
- Simulation of a self-driving cab.
- Major goal - demonstrate, in a simplified environment, how to use RL techniques to develop an efficient and safe approach for tackling this problem.
- Smartcab's job is to pick up the passenger at one location and drop them off in another.
- Things Smartcab has take care of:
  - Drop off the passenger to the right location.
  - Save passenger's time by taking minimum time possible to drop off
  - Take care of passenger's safety and traffic rules

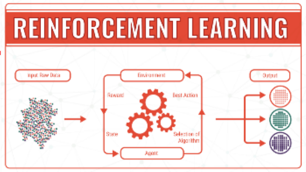
Self-Driving Cab



State Space

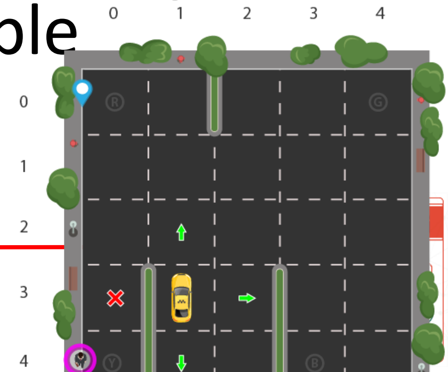
- training area for Smartcab
- teaching it to transport people in a parking lot to four different locations (R, G, Y, B):
- assume Smartcab is the only vehicle in the parking lot.
- Break up the parking lot into a 5x5 grid, which gives 25 possible taxi locations.
- These 25 locations are one part of the state space.
- Current location state





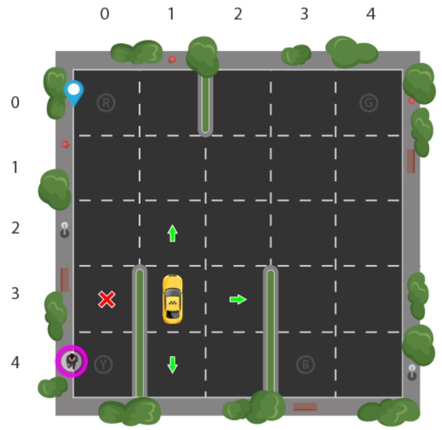
Rewards

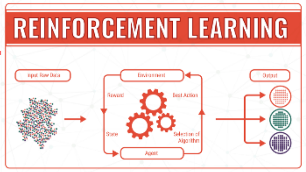
- Agent (the imaginary driver) is reward-motivated and is going to learn how to control the cab by trial experiences in the environment, need to decide the rewards and/or penalties and their magnitude accordingly.
- Agent should receive a high positive reward for a successful dropoff because this behavior is highly desired
- Agent should be penalized if it tries to drop off a passenger in wrong locations
- Agent should get a slight negative reward for not making it to the destination after every time-step.
- "Slight" negative – to prefer the agent to reach late instead of making wrong moves trying to reach to the destination as fast as possible



State Space

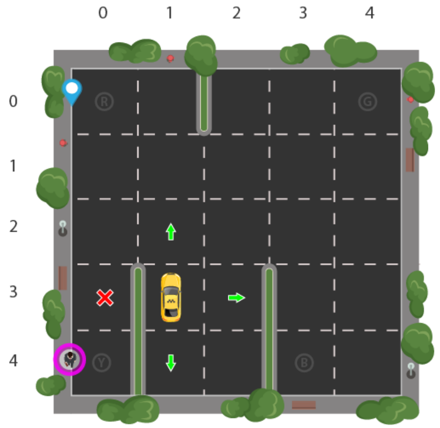
- there are four (4) locations to pick up and drop off a passenger:
  - R, G, Y, B or
  - [(0,0), (0,4), (4,0), (4,3)] in (row, col) coordinates.
- Now passenger is in location Y and wishes to go to location R.
- there's four (4) destinations and
- five (4 + 1) passenger locations.





Action Space

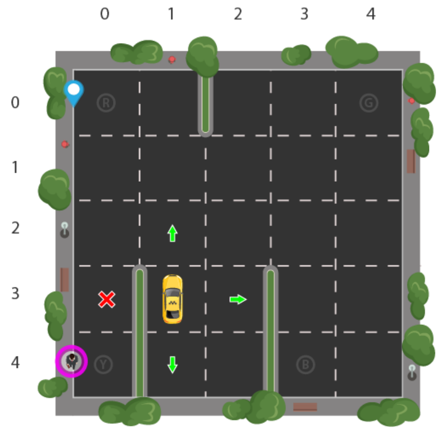
- Taxi environment has 500 total possible states.
- The agent encounters one of the 500 states and it takes an action.
- The action - e can be to move in a direction or decide to pickup/dropoff a passenger.
- six possible actions:
  - south
  - north
  - east
  - west
  - pickup
  - dropoff



- This is the action space: the set of all the actions that the agent can take in a given state.

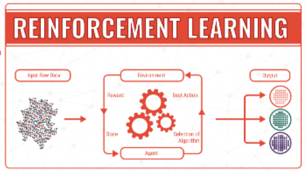
Action Space

- The taxi cannot perform certain actions in certain states due to walls.
- In environment's code, simply provide a -1 penalty for every wall hit and the taxi won't move anywhere.
- This will just rack up penalties causing the taxi to consider going around the wall.



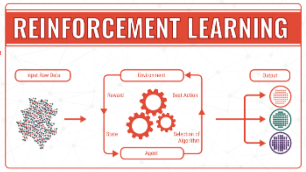
OpenAI’s Gym

- based upon these fundamentals
- Gym is a toolkit for developing and comparing reinforcement learning algorithms.
- It makes no assumptions about the structure of the agent, and is compatible with any numerical computation library, such as TensorFlow or Theano.
- The gym library is a collection of test problems — environments — that can be used to work out the reinforcement learning algorithms.
- These environments have a shared interface, allowing us to write general algorithms.



OpenAI

- OpenAI was founded in late 2015
- a non-profit with
- mission - "build safe artificial general intelligence (AGI) and ensure AGI's benefits are as widely and evenly distributed as possible."
- In addition to exploring many issues regarding AGI, one major contribution that OpenAI made to the machine learning world was developing both the Gym and Universe software platforms.

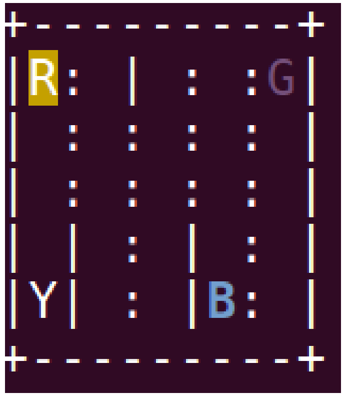


Gym

- Gym is a collection of environments/problems
  - designed for testing and
  - developing reinforcement learning algorithms—
  - it saves the user from having to create complicated environments.
- Gym is written in Python,
- there are multiple environments such as robot simulations or Atari games.
- There is also an online leaderboard for people to compare results and code.

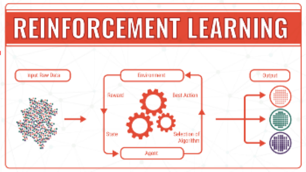
Problem Statement

- There are 4 locations (labelled by different letters) and your job is to pick up the passenger at one location and drop him off in another.
- You receive +20 points for a successful drop-off, and lose 1 point for every time-step it takes.
- There is also a 10 point penalty for illegal pick-up and drop-off actions.
- In this environment
  - the filled square represents the taxi,
  - the (“|”) represents a wall,
  - the blue letter represents the pick-up location, and
  - the purple letter is the drop-off location.
  - The taxi will turn green when it has a passenger aboard



Load the Environment

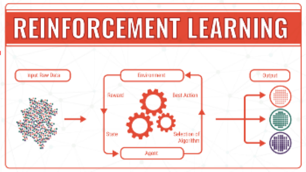
- `env = gym.make("Taxi-v2")`



Initialise the Environment

- env.reset()
- Resetting the environment will return an integer.
- This number will be the initial state.
- All possible states in this environment are represented by an integer ranging from 0 to 499.
- We can determine the total number of possible states using :

- env.observation_space.n

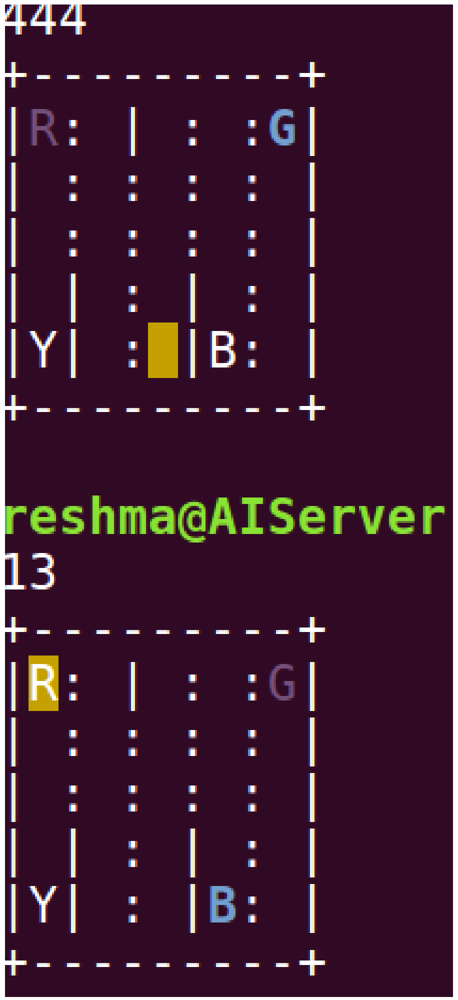


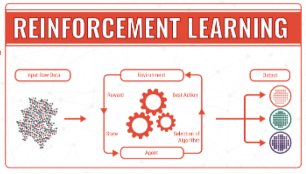
Visualize the Environment

- env.render()

Visualize the Environment

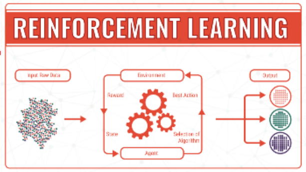
- the yellow square represents the taxi,
- the ("|") represents a wall,
- the blue letter represents the pick-up location,
- the purple letter is the drop-off location.
- The taxi will turn green when it has a passenger aboard.
- we see colors and shapes that represent the environment,
- the algorithm does not think like us and only understands
- a flattened state, in this case an integer.





Actions available to the agent

- env.action_space.n
- This shows us there are a total of six actions available.
- the six possible actions are:
  - down (0),
  - up (1),
  - right (2),
  - left (3),
  - pick-up (4), and
  - drop-off (5).



OpenAI’s Gym

- based upon these fundamentals
- Gym is a toolkit for developing and comparing reinforcement learning algorithms.
- It makes no assumptions about the structure of the agent, and is compatible with any numerical computation library, such as TensorFlow or Theano.
- The gym library is a collection of test problems — environments — that can be used to work out the reinforcement learning algorithms.
- These environments have a shared interface, allowing us to write general algorithms.