Reinforcement Learning

Profa. Dra. Esther Luna Colombini
esther@ic.unicamp.br

Prof. Dr. Alexandre Simoes
alexandre.simoes@unesp.br

LaRoCS – Laboratory of Robotics and Cognitive Systems

UNICAMP

Advanced Institute for Artificial Intelligence

Summary

- Course logistics
- What is this course about?
- Reinforcement Learning: an overview
- Applications Reinforcement Learning
- Summary

Course Logistics

Course logistics

- **Credits:** 2
- **Class hours:** Wednesday @14h
- **Website:** https://classroom.google.com and http://www.ic.unicamp.br/~esther/teaching/2020s2/mo436

- **Course Goals:** by the end of the course, the students must understand the main concepts related to Reinforcement Learning, define the key features of RL that distinguishes it from AI and other machine learning paradigms. They should also be able to build models using Reinforcement Learning as a basis.

- **Prerequisite:**
  - Proficiency in Python
    - a good tutorial: https://cs231n.github.io/python-numpy-tutorial/
  - Calculus, Linear Algebra
    - https://www.3blue1brown.com
  - Basic Probability and Statistics
  - Foundations of Machine Learning
  - Atari games ☺
    - http://www.virtualatari.org/

- **Syllabus**
  - Introduction to RL
  - Markov Decision Processes
  - Dynamic Programming
  - Model-Free Prediction and Control
  - Value-based Methods
  - Policy Gradient Methods
  - Exploration and Exploitation
  - RL Topics: Imitation Learning
  - RL Topics: Meta-RL
  - RL applications

- **Evaluation**
  - MF = 0.10R + 0.30P1 + 0.60P2
    - Where
      - R is a set of varied tasks that will have grades distributed proportionally. Tasks include readings, reviews of recommended articles and online tests.
      - P1 and P2 are practical projects with weights 30% and 60%, respectively
        - Groups must have 4 students
      - The reports for the project must present the adopted solution, discussing the results achieved in scientific article format, in the model proposed by the professor
      - The code and the reports must be delivered via Google Classroom
      - Project P2 must be presented by the group as indicated by the professor

- **Evaluation**
  - Deadlines
    - Project 1 (P1): 29/11/2020
    - Project 2 (P2): 03/01/2021
    - Project 2 presentation: 06-13/01/2021
  - For graduate students, the grade range will be:
    - A: >=8.5
    - B: >=7.0 and <8.5
    - C: >=5 e <7.0
    - D: <5
  - No exams

- **Material and submissions**
  - The course material will be available our Google Classroom
  - Practical work and projects carried out during the course must be submitted through Google Classroom in the area corresponding to the course
  - We will use Python as the course reference programming language.

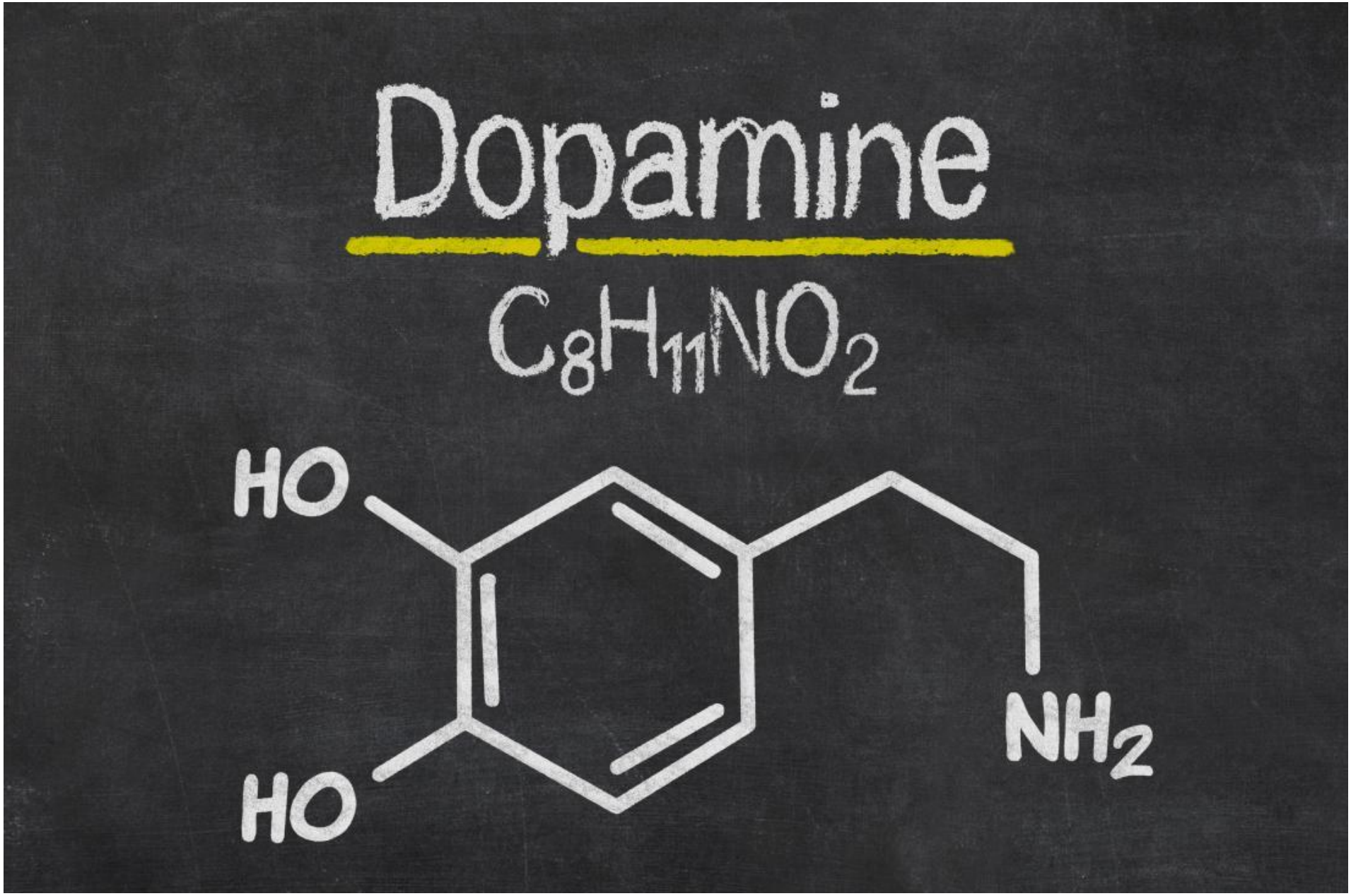
Tools

- General tools: Anaconda, Docker
- RL environments:
  - OpenAI Gym: is a toolkit for developing and comparing reinforcement learning algorithms. It is an open source interface to reinforcement learning tasks (https://gym.openai.com/)
  - Other envs: https://medium.com/@mauriciofadelargerich/reinforcement-learning-environments-cff767bc241f
- Numerical computation libraries: TensorFlow, PyTorch
- Simulators: Mujoco, Atari, Robotics, V-Rep
- RL frameworks/libraries (https://blog.dataiku.com/on-choosing-a-deep-reinforcement-learning-library):
  - OpenAi SpinningUp (https://spinningup.openai.com/en/latest/index.html)
  - OpenAI Baselines is a set of high-quality implementations of reinforcement learning algorithms (https://github.com/openai/baselines)
  - RLKIT: Reinforcement learning framework and algorithms implemented in PyTorch (https://github.com/vitchyr/rlkit)
  - Garage: garage is a toolkit for developing and evaluating reinforcement learning algorithms, and an accompanying library of state-of-the-art implementations built using that toolkit (https://github.com/rlworkgroup/garage)

Bibliography

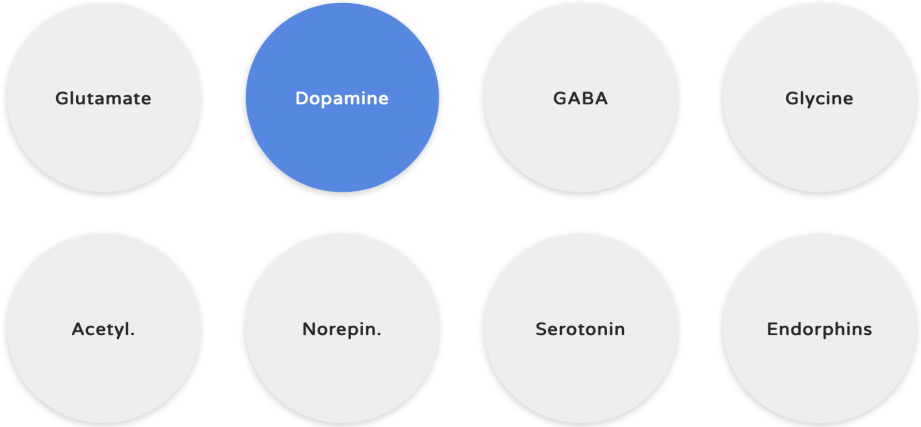
- Reinforcement Learning: An Introduction, Sutton and Barto, 2nd Edition. This is available for free in http://incompleteideas.net/book/RLbook2018.pdf
- Reinforcement Learning: State-of-the-Art, Marco Wiering and Martijn van Otterlo, Eds.
- Artificial Intelligence: A Modern Approach, Stuart J. Russell and Peter Norvig.
- Deep Learning, Ian Goodfellow, Yoshua Bengio, and Aaron Courville.
- David Silver's course on Reinforcement Learning.

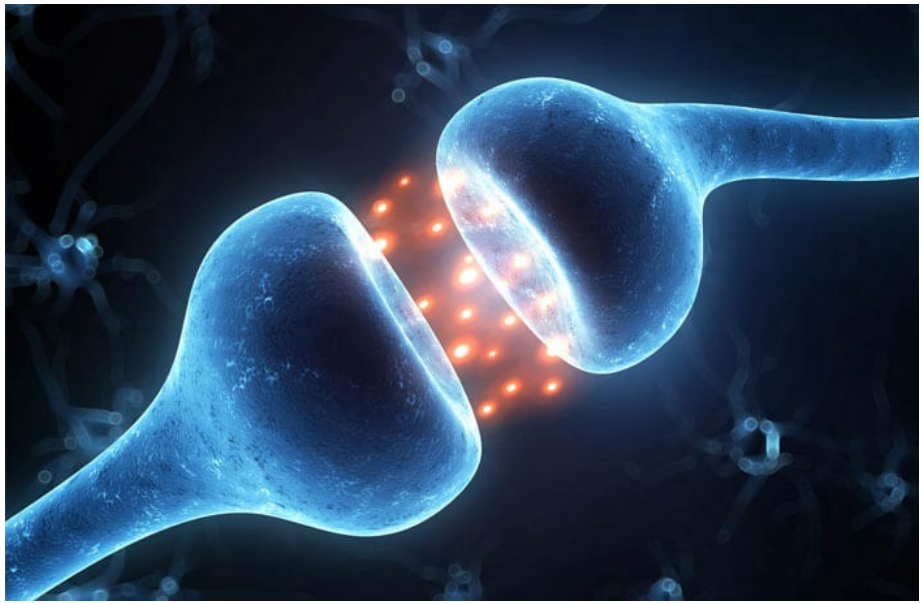
What is this course about?

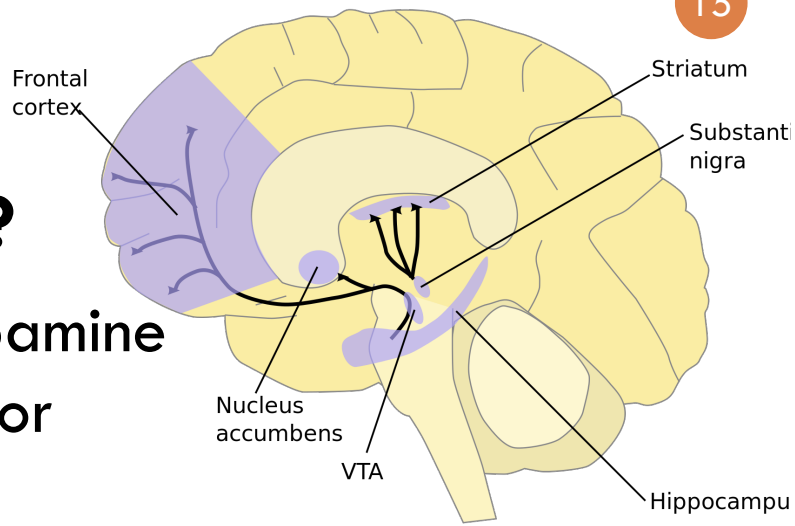

Dopamine
$C_8H_{11}NO_2$
HO
HO
NH_2

Dopamine

- **Dopamine** is one of the brain's eight neurotransmitters. A neurotransmitter is a chemical that carries information back and forth between neurons.
- It is known as the **feel-good neurotransmitter.**







Why do we care about Dopamine?

- The phasic activity of the midbrain dopamine neurons provides a global mechanism for synaptic modification.
- These synaptic modifications, in turn, provide the support for a specific class of **reinforcement learning mechanisms** that are now believed to **underlie much of human** and animal **behavior**.
- In other words, dopamine allows us to **see a reward and take action.** We then take an action to **receive rewards—** that is, through the use of regulating movement, learning, attention and emotional responses.

Source: https://towardsdatascience.com/dopamine-and-reinforcement-learning-algorithms-d5deb5ac6479

Dopamine and Learning

- Dopamine works as a **reward prediction error signal**
- It calculates the **difference** between the reward that was *expected* and the reward that was *actually received*

- **Why is Reward Prediction Error Important?**
  - Learning can be defined as the **process of improving these predictions of the future**
  - These prediction errors are one of the most basic teaching signals that can be used to **improve prediction accuracy for future rewards.**
  - The ultimate goal of learning is to make accurate predictions, thus eliminating the prediction error.

Source: https://towardsdatascience.com/dopamine-and-reinforcement-learning-algorithms-d5deb5ac6479

- **Dopamine and Reinforcement Learning**
  - The **prediction error hypothesis** (Glimcher, 2011) is interesting because **Reinforcement Learning algorithms** use **temporal difference** learning which makes heavy use of a signal that encodes prediction error.
- **In Summary:**
  - Dopamine neurons may provide neurons in the brain with detailed information about the **value of the future.**
  - This information is used to plan and execute **profitable behaviors** and **decisions** well **in advance of actual reward occurrence** and to learn about even earlier reliable predictors of reward.

(Glimcher, 2011) https://www.pnas.org/content/108/Supplement_3/15647

Dopamine

- **Remember:** This important neurochemical boosts mood, motivation, and attention, and helps regulate movement, learning, and emotional responses.
- **Low-levels** of dopamine are related to addiction, **ADHD** and neurogenerative diseases like PD
- **More dopamine?**
  - Eat foods rich in tyrosine including cheese, meats, fish, dairy, soy, seeds, nuts, beans, lentils, among others.
  - Up magnesium intake with foods such as seeds, nuts, soy, beans, whole grains, among others.
  - Avoid processed foods, high-fats, sugar, caffeine.
  - Proper sleep hygiene is mandatory, as it fuels dopamine production.
  - Exercise daily.
  - Avoid stress, apply techniques such as meditation, visualization, breathing exercises.

The basics behind it

- Pavlov (Pavlov, 1927) observed, in his famous experiment on the salivating dog, that if one rings a bell and follows that bell with food, dogs become **conditioned** to salivate after the bell is rung.
- Bush and Mosteller (1951) proposed that the probability of Pavlov's dog expressing the salivary response on sequential trials could be computed through an iterative equation
  - $A_{next\_trial} = A_{last\_trial} + \alpha(R_{current\_trial} - A_{last\_trial})$
    - It computes an average of previous rewards across previous trials. In this average, the most recent rewards have the greatest impact.

(Pavlov, 1927) Conditioned Reflexes: An Investigation of the Physiological Activity of the Cerebral Cortex (Dover, New York).
(Bush RR, Mosteller F, 1951) A mathematical model for simple learning. Psychol Rev **58**:313–323.

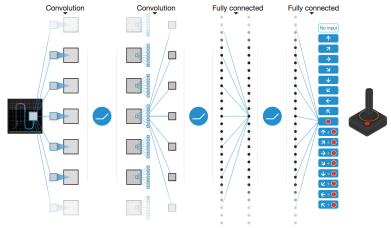
The basics behind it

- Sutton and Barto
  - The story of reinforcement learning described up to this point is a story largely from psychology and mostly focused on associative learning.
  - Remember, in Bush and Mosteller models, reward prediction error is the difference between a weighted average of past rewards and the reward that has just been experienced.
    - When those are the same, there is no error, and the system does not learn.
  - To address these issues, Sutton and Barto (1998) developed what has come to be known as temporal difference (TD) learning.
  - Sutton and Barto (1998) argue that Bush and Mosteller approach were trying to solve incorrectly the problem.
  - Bush and Mosteller's equation learns the values of previous events.
  - Sutton and Barto argued that the goal of a learning system should instead be to predict the value of future events, but based on previous experience.
  - Therefore, these two ideas are closely related; however, TD learning was designed with a clear goal in mind: predict the value of the future.

(Sutton RS, Barto AG, 1998) Reinforcement Learning: An Introduction (MIT Press, Cambridge, MA).

Reinforcement Learning: an overview

RL Timeline



Pavlovian Conditioning

1927

Bush and Mosteller

1951

MENACE tic-tac-toe

1961

Rich Sutton, TD(λ)

1986

Baxter, TDLeaf(λ)

1999

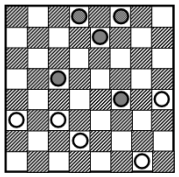
Silver, DQN

2015

1949

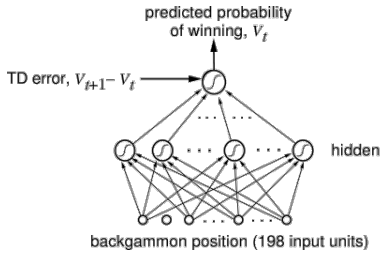
Hebb learning rule

1959



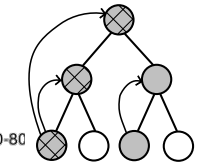
Arthur Samuel, Checkers (Future TDLeaf(λ))

1989



Gerald Tesauro, TD-Gammon

2009



Veness, TreeStrap

RL Timeline

- Model-free methods
  - 1961 MENACE tic-tac-toe (Donald Michie)
  - 1986 TD(λ) (Rich Sutton)
  - 1989 TD-Gammon (Gerald Tesauro)
  - 2015 Deep Q Learning for Atari Games
  - 2016 A3C (Mnih et al.)
  - 2017 OpenAI Evolution Strategies (Salimans et al.)
- Methods relying on a world model
  - 1959 Checkers (Arthur Samuel)
  - 1999 TD-leaf (Baxter et al.)
  - 2009 TreeStrap (Veness et al.)
  - 2016 Alpha Go (Silver et al.)

ML Approaches

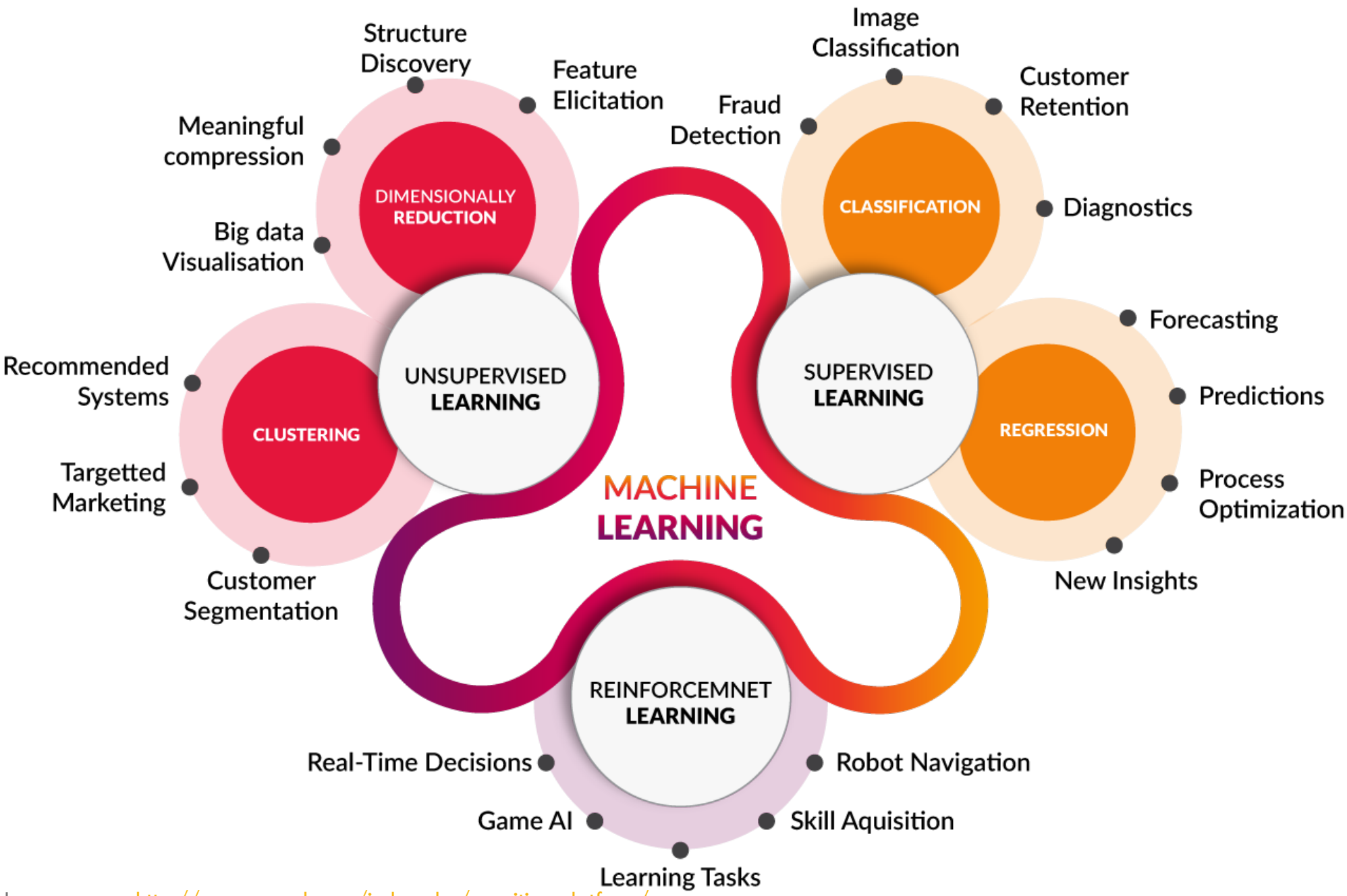


Image source: http://www.cognub.com/index.php/cognitive-platform/

RL and its many faces

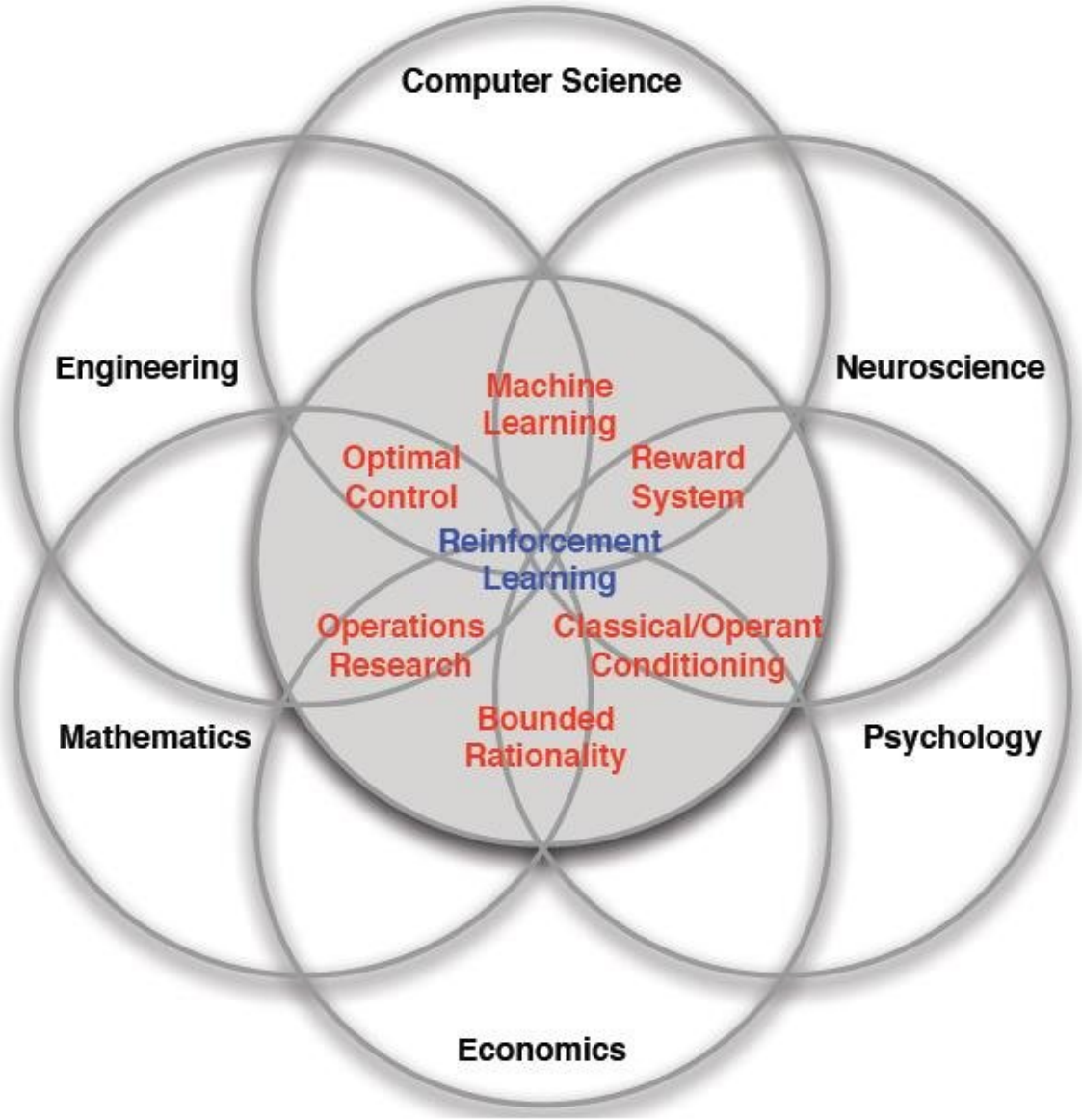


Image source: https://medium.com/hackernoon/david-silver-rl-course-lecture-1-notes-2e650270d626

- The fundamental challenge:
  - Make **good sequences of decisions** under **uncertainty**
  - Through **learning**

Reinforcement Learning Assumptions

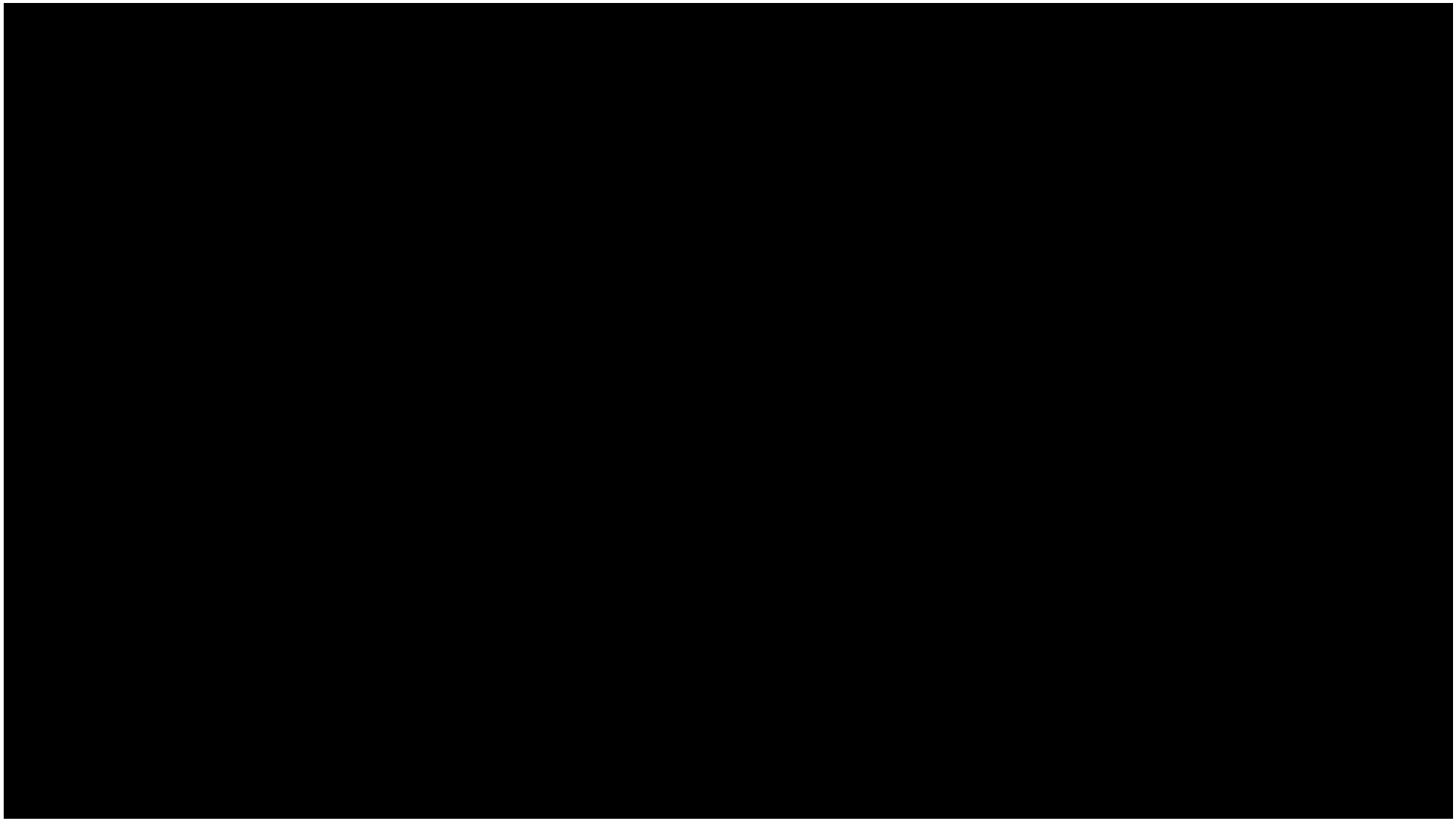
- Experiential learning
  - The world is the best model of itself
- Specify what to do, not how to do
  - This is done through the reward function
- Usually find the best end solutions
  - Based on current experiences, there are no assumptions of the programmer
- In short:
  - Less human time is needed to find a good solution
  - It is not necessary to define heuristics, techniques to solve the problem…but you need reward shaping!
  - Just set the learning system and let the system learn! ☺ Not so fast…

RL Examples

Chelsea Finn, Sergey Levine, Pieter Abbeel

RL Examples

https://www.youtube.com/watch?v=VCdxqn0fcnE

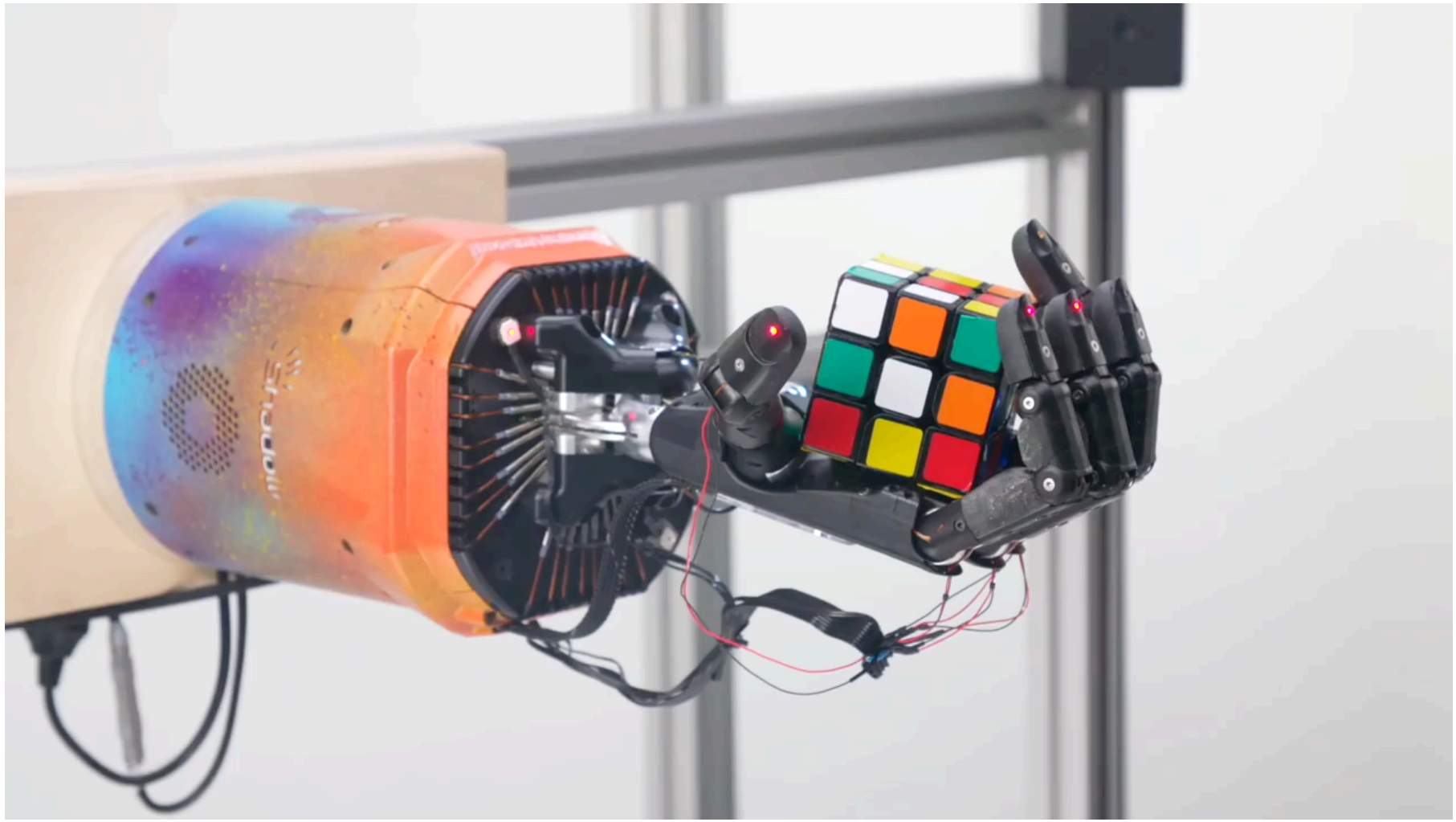


RL Examples

https://openai.com/blog/emergent-tool-use/

RL Examples

https://www.youtube.com/watch?v=jm-ihc7CASY

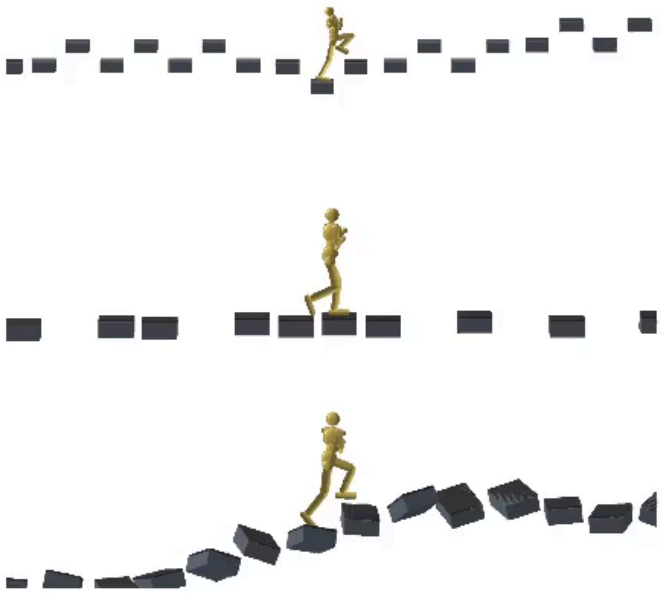


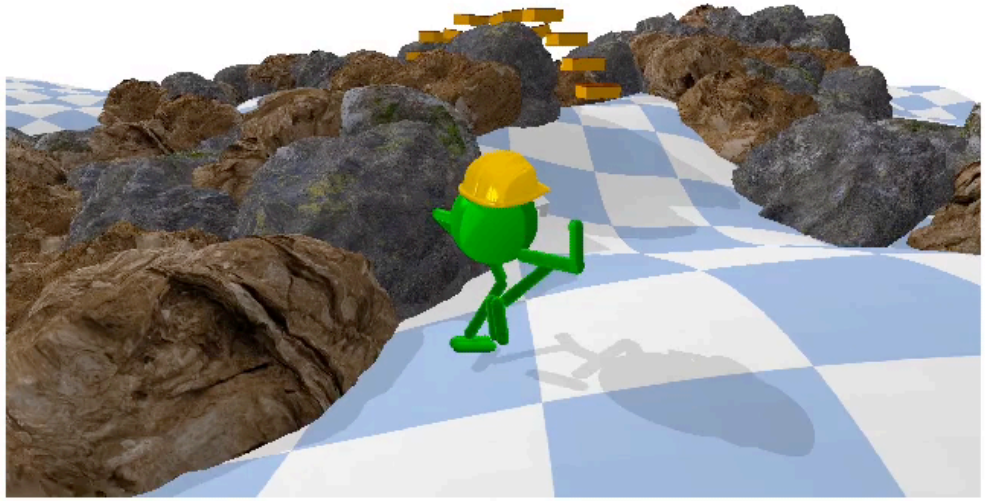
RL Examples

DQN Atari Playing

ALLSTEPS: Curriculum-driven Learning of Stepping Stone skills

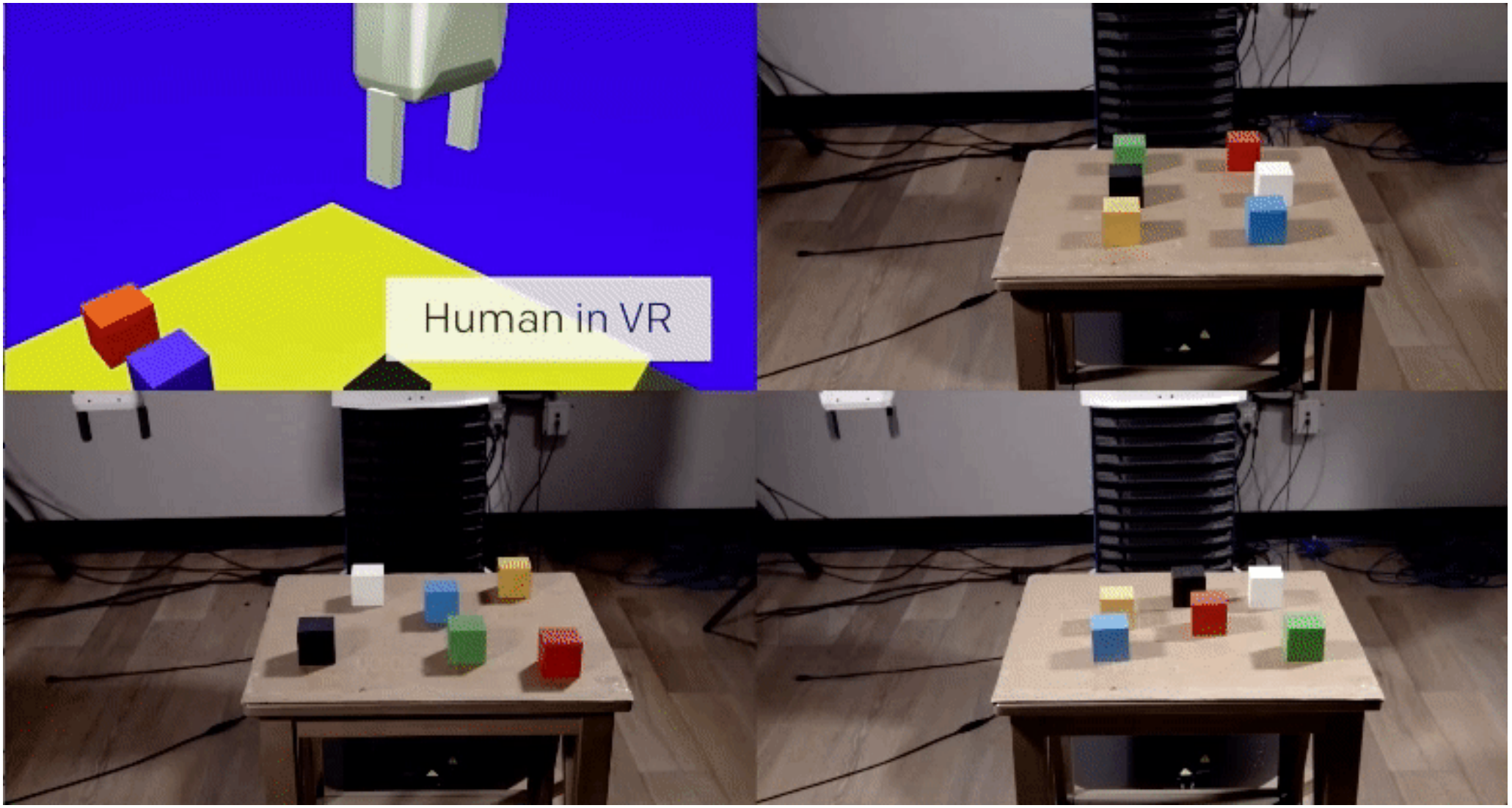
ALLSTEPS: Curriculum-driven Learning of Stepping Stone Skills





RL Examples

Imitation Learning: https://openai.com/blog/robots-that-learn/



RL Involves

Make **good sequences of decisions** under **uncertainty**
Through **learning**

- Optimization
- Delayed consequences
- Exploration
- Generalization

- **Optimization**
  - Goal is to find an optimal way to make decisions
    - Yielding best outcomes or at least very good outcomes
  - Explicit notion of **utility** of decisions
  - Example: finding the best way to kick a ball to score a goal

- **Delayed consequences**
  - Decisions now can impact things much later...
    - Saving for retirement
    - Hiding not to be found
  - Introduces two challenges
    - **When planning:** decisions involve reasoning about not just immediate benefit of a decision but also its longer term ramifications
    - **When learning:** temporal credit assignment is hard (what caused later high or low rewards?)

- **Exploration**
  - Learning about the world by making decisions
    - Agent as scientist
    - Learn to ride a bike by trying (and failing)
    - Learning to walk (while falling)
  - Censored data
    - Only get a reward (label) for decision made
    - Don't know what would have happened if we had taken another path
  - Decisions impact what we learn about
    - If we choose to go make another course at University, we will have different later experiences...

- **Generalization**
  - Policy is mapping from past experience to action
    - Why not just pre-program a policy?

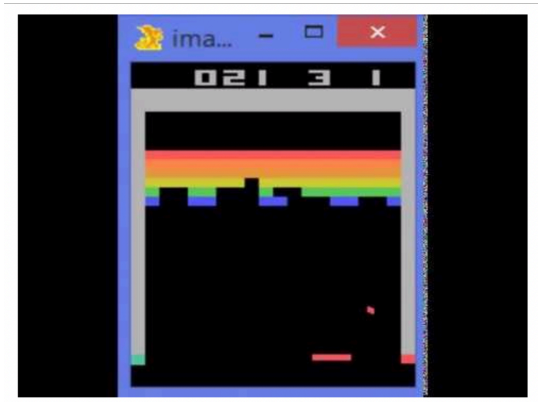


Figure: DeepMind Nature, 2015

- How many possible images are there?
  - $(256^{100\times200})^{3}$

RL x Other AI

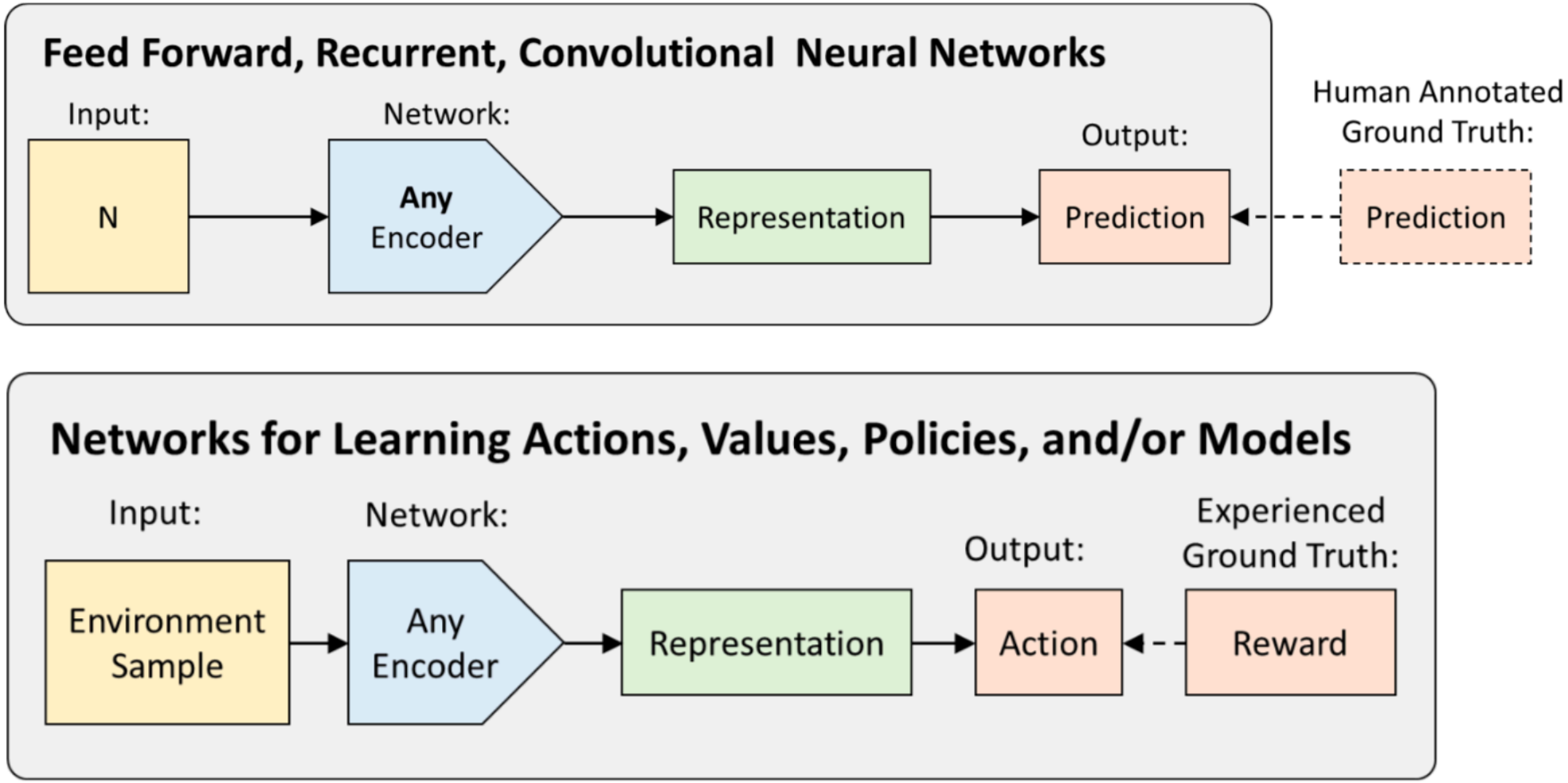
- What makes reinforcement learning different from other machine learning paradigms?
  - There is no supervisor, only a **reward signal**
  - Feedback is **delayed,** not instantaneous
  - **Time** really matters (sequential)
  - Agent's **actions affect** the subsequent data it receives

Reinforcement Learning x Other AI

Has a model

	AI Planning	SL	UL	RL	IL
Optimization	X	X	X	X	X
Learns from experience		X	X	X	X
Generalization	X	X	X	X	X
Delayed Consequences	X			X	X
Exploration				X	

- **SL** = Supervised learning – learn from experience, but data is labeled
- **UL** = Unsupervised learning – learn from experience, but data is not labeled
- **RL** = Reinforcement Learning;
- **IL** = Imitation Learning
- Imitation learning assumes input demonstrations of good policies
- IL reduces RL to SL. IL + RL is promising area



https://deeplearning.mit.edu

RL: The Reward Hypothesis

- 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

- **Reward Hypothesis**
  - All **goals** can be described by the maximization of expected cumulative reward

- Is this easy to define?
  - How can I define a reward function for **walking well?**

Reward Examples

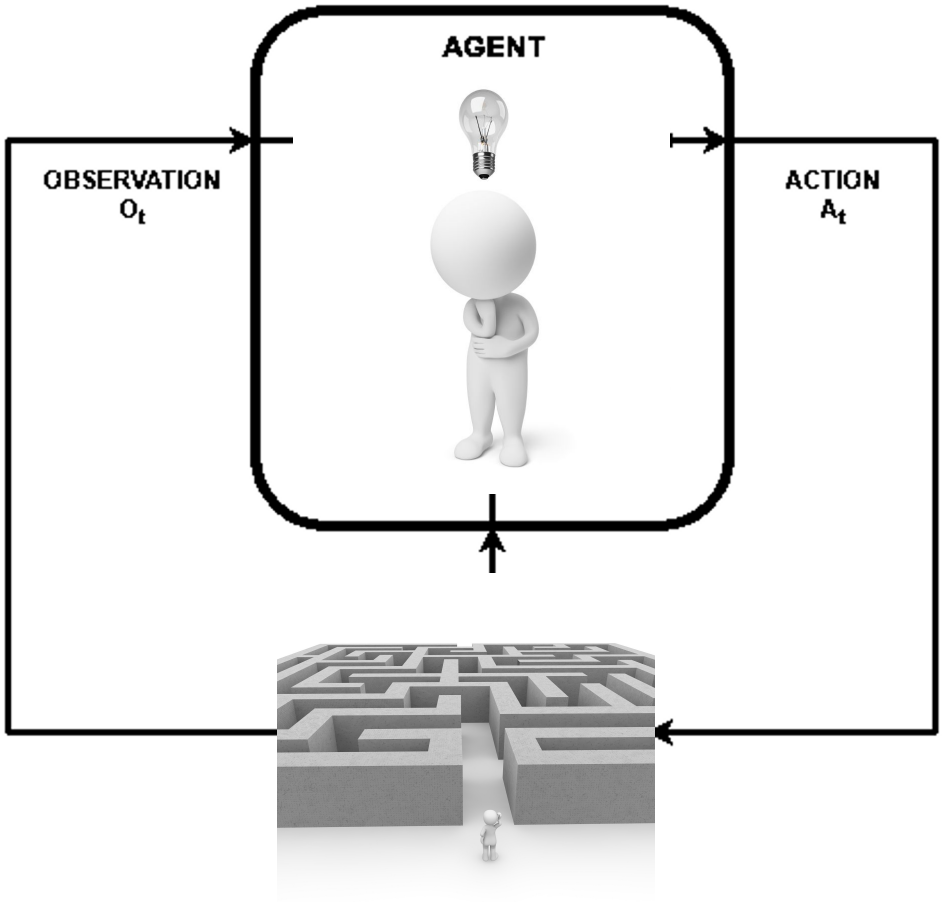
- Fly stunt manoeuvres in a helicopter
  - +ve reward for following desired trajectory
  - −ve reward for crashing
- Defeat the world champion at Backgammon
  - +/−ve reward for winning/losing a game
- Manage an investment portfolio
  - +ve reward for each $ in bank
- Control a power station
  - +ve reward for producing power
  - −ve reward for exceeding safety thresholds
- Make a humanoid robot walk
  - +ve reward for forward motion
  - −ve reward for falling over
- Play many different Atari games better than humans
  - +/−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 more long-term reward
  - Examples:
    - A financial investment (may take months to mature)
    - Refuelling a helicopter (might prevent a crash in several hours)
    - Blocking opponent moves (might help winning chances many moves from now)

Reinforcement Learning Elements

- 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**



Reinforcement Learning Elements

- The **history** is the sequence of observations, actions, rewards
  - $H_t = O_1, R_1, A_1, \ldots, A_{t-1}, O_t, R_t$
  - 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, state is a function of the history:
    - $S_t = f(H_t)$

Reinforcement Learning Elements

- The **environment state** S_t^e is the environment's private representation
  - This is true state of the world used to determine how world generates next observation and reward
  - i.e. whatever data the environment uses to pick the next observation/reward
  - Often hidden or unknown to agent
  - Even if S_t^e is visible, it may contain irrelevant information
- The **agent state** S_t^q is the agent's internal representation
  - 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 history:
    - $S_t^q = f(H_t)$
  - Could include meta information like state of algorithm (how many computations executed, etc) or decision process (how many decisions left until an episode ends)

Reinforcement Learning Elements

- An **information state** (a.k.a. Markov state) contains all useful information from the history.
  - A state S_t is **Markov** if and only if
    - $P[S_{t+1} \mid S_t] = P[S_{t+1} \mid S_1,\ldots,S_t]$
  - 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
  - i.e. The state is a sufficient statistic of the future
  - The environment state S_t^e is Markov
  - The history H_t is Markov

- Are our systems Markov?
  - **Hypertension control**: let state be current blood pressure, and action be whether to take medication or not. Is this system Markov?
  - **Website shopping:** state is current product viewed by customer, and action is what other product to recommend. Is this system Markov?
  - **Playing CS:** state is the current view of the scene by the agent, an action in what move to make. Is this Markov? Can we make it Markov?

- Why is Markov Assumption Popular?
  - Can always be satisfied
    - Setting state as history always Markov: $S_t = H_t$
  - In practice often assume most recent observation is sufficient statistic of history:
    - $S_t = O_t$
  - State representation has big implications for:
    - Computational complexity
    - Data required
    - Resulting performance

Fully observable environment

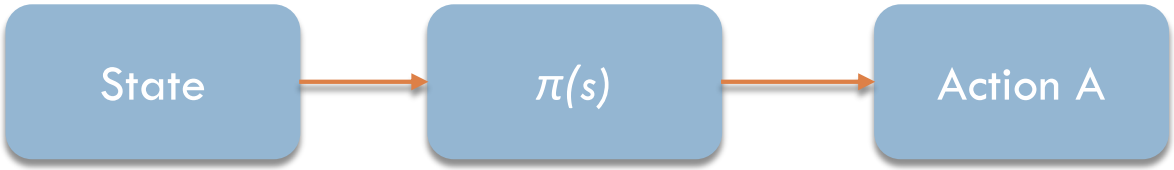
- When an agent can determine the state of the system at all times, it is called fully observable.
  - Full observability = agent directly observes environment state
    - $O_t = S_t^a = S_t^e$
  - Agent state = environment state = information state
  - Formally, this is a Markov decision process (MDP)
  - For example, in a chess game, the state of the system, that is, the position of all the players on the chess board, is available the whole time so the player can make an optimal decision.

- **Partially observable environment**
  - When an agent cannot determine the state of the system at all times, it is called partially observable.
  - Partial observability = agent indirectly observes environment
    - A robot with camera vision isn't told its absolute location
    - A trading agent only observes current price
    - A poker playing agent only observes public cards
- Now $S_t^a \mathrel{!}= S_t^e$
- 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: $S_t^a = (P[S_t^a = s^1], \ldots, P[S_t^a = s^n])$
  - Recurrent neural network: $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_0)$

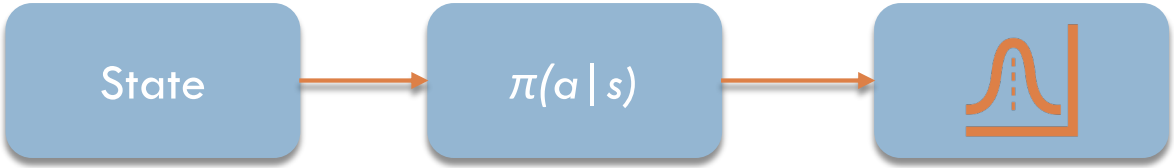
RL: Agent's components

- An RL agent may include one or more of these components:
  - **Policy**: agent's behaviour function
  - **Value function**: how good is each state and/or action
  - **Model**: agent's representation of the environment

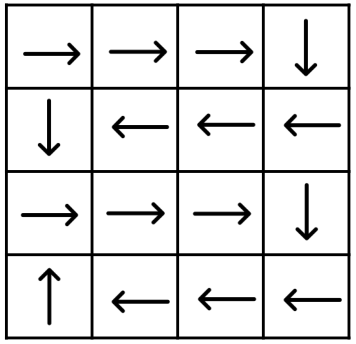
- **Policy**
  - A policy is the agent's behaviour
  - It determines how the agent chooses actions
  - It is a map from state to action $\pi : S \longrightarrow A$
  - It can be:
    - **Deterministic policy:** $a = \pi(s)$



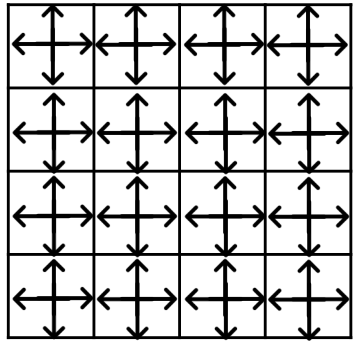
    - **Stochastic policy:** $\pi(a \mid s) = P[A_t = a \mid S_t = s]$



- **Policy**
  - What kind of policy is π_1? And π_2?



π_1



π_2

- **Value Function**
  - Is a prediction of future reward
  - Used to evaluate the goodness/badness of states
  - Therefore, used to select between actions, e.g.
    - $v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s]$

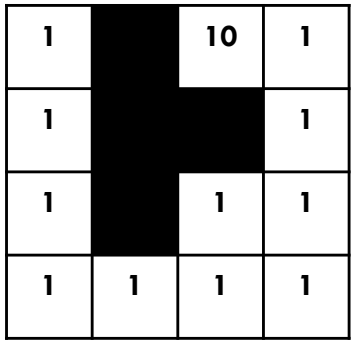
1	1	1	-5
-5	1	1.3	1.3
1	1	1.1	1.2
10	1	1	1

$v_{\pi 1}$

    - Numbers represent value $v_{\pi 1}(s)$ of each state s

- **Model**
  - A model predicts what the environment will do next
  - Agent's representation of how world changes given agent's action Transition/dynamics model predicts next agent state
  - *P* predicts the next state
    - $P_{SS'}^{a} = \mathbb{P}\left[S_{t+1} = s' \middle| S_t = s, A_t = a\right]$
  - *R* predicts the next (immediate) reward
    - $R_s^a = \mathbb{P}[R_{t+1} | S_t = s, A_t = a]$

- **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
    - Grid layout represents transition model $P^a_{SS'}$
    - Numbers represent immediate reward R^a_S from each state s



RL: Types of RL environment

- **Deterministic environment**
  - An environment is said to be deterministic when we know the outcome based on the current state.
    - In the maze example, if you move one step to the right at state (1,1), you are guarantee to get to state (1,2)
- **Stochastic environment**
  - An environment is said to be stochastic when we cannot determine the outcome based on the current state. There will be a greater level of uncertainty.
    - In the maze example, if you move one step to the right at state (1,1), you may end up in state (1,2) but you may slip and stay in state (1,1)

- **Discrete environment**
  - When there is only a finite state of actions available for moving from one state to another, it is called a discrete environment.
    - For example, in a chess game, we have only a finite set of moves.
- **Continuous environment**
  - When there is an infinite state of actions available for moving from one state to another, it is called a continuous environment.
    - For example, we have multiple routes available for traveling from the source to the destination.

RL: Types of RL environment

- **Episodic environment**
  - The agent's experience is divided into atomic episodes (each episode consists of the agent perceiving and then performing a single action) and the choice of action in each episode depends only on the episode itself.
  - The episodic environment is also called the **non-sequential** environment. In an episodic environment, an agent's current action will not affect a future action.
- **Non-episodic environment**
  - In a non-episodic environment, an agent's current action will affect a future action and is also called the **sequential** environment.
  - That is, the agent performs the independent tasks in the episodic environment, whereas in the non-episodic environment all agents' actions are related.
- Most environments (and agents) are sequential
- Many are both – a number of episodes containing a number of sequential steps to a conclusion

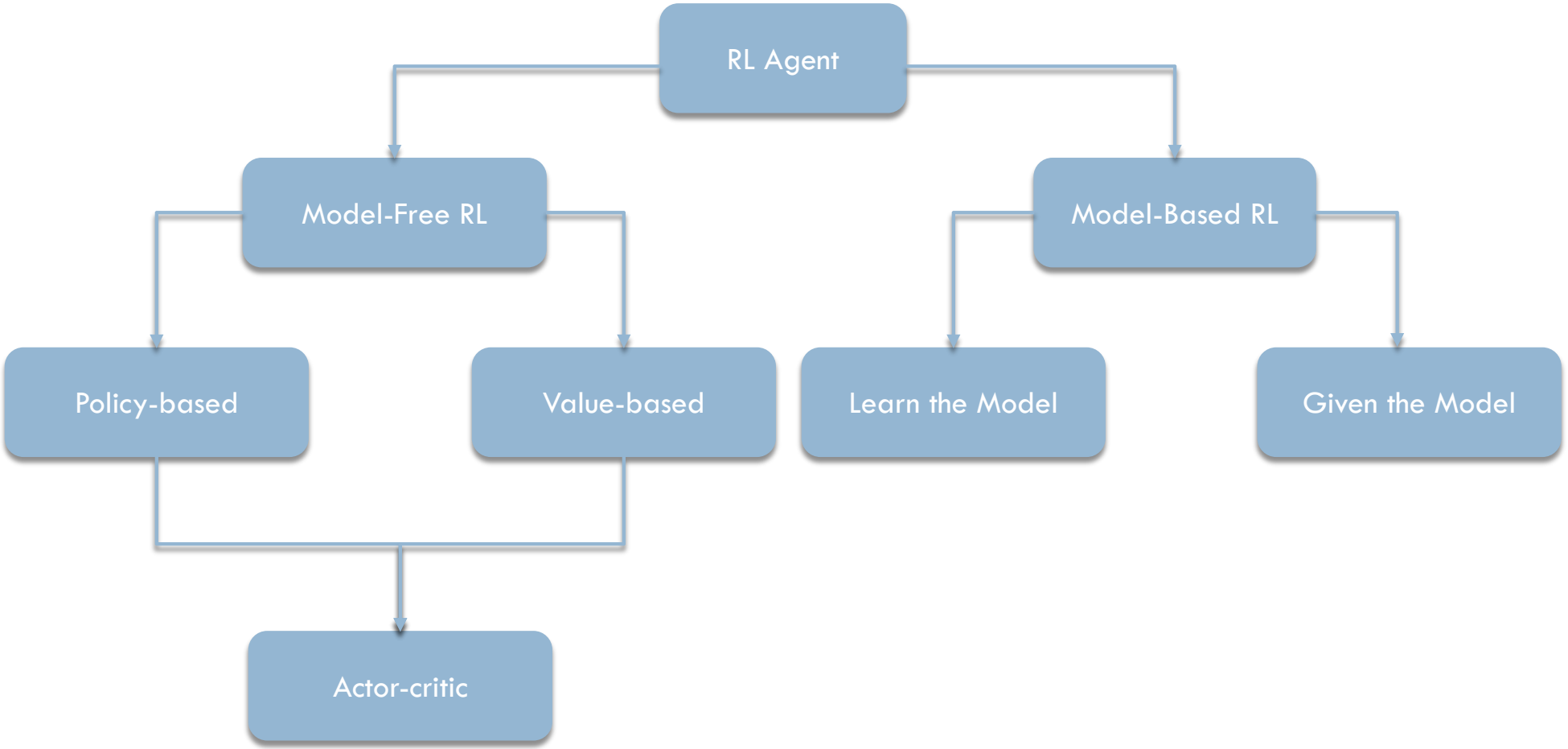
- **Single and multi-agent environment**
  - As the names suggest, a single-agent environment has only a single agent and the multi-agent environment has multiple agents.
  - Multi-agent environments are extensively used while performing complex tasks. There will be different agents acting in completely different environments. Agents in a different environment will communicate with each other.
  - A multi-agent environment will be mostly stochastic as it has a greater level of uncertainty.

- **Categorizing RL agents**
  - Value Based
    - No Policy (Implicit)
    - Value Function
  - Policy Based
    - Policy
    - No Value Function
  - Actor Critic
    - Policy
    - Value Function

- **Categorizing RL agents**
  - Model Free
    - Policy and/or Value Function
    - No Model
  - Model Based
    - Policy and/or Value Function
    - Model

RL: Types of Agents

Taxonomy



RL: Types of Agents

Taxonomy

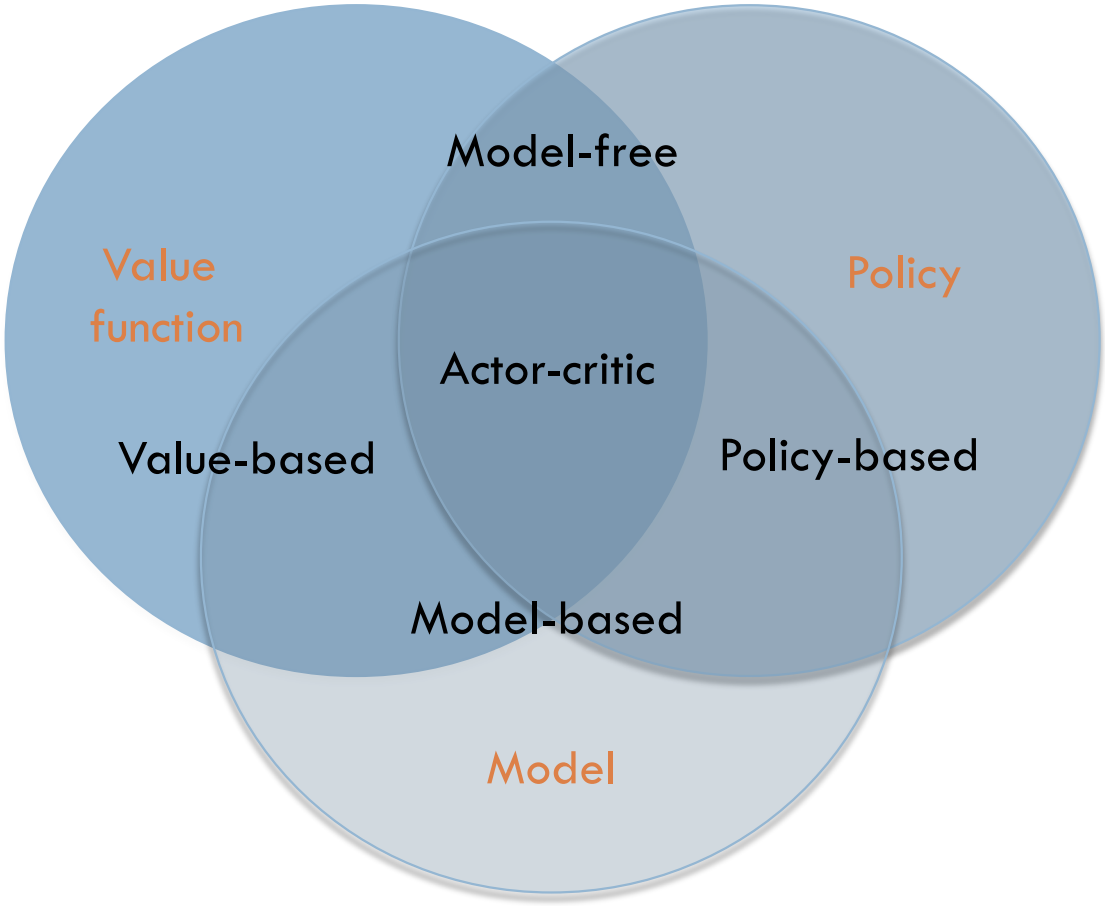
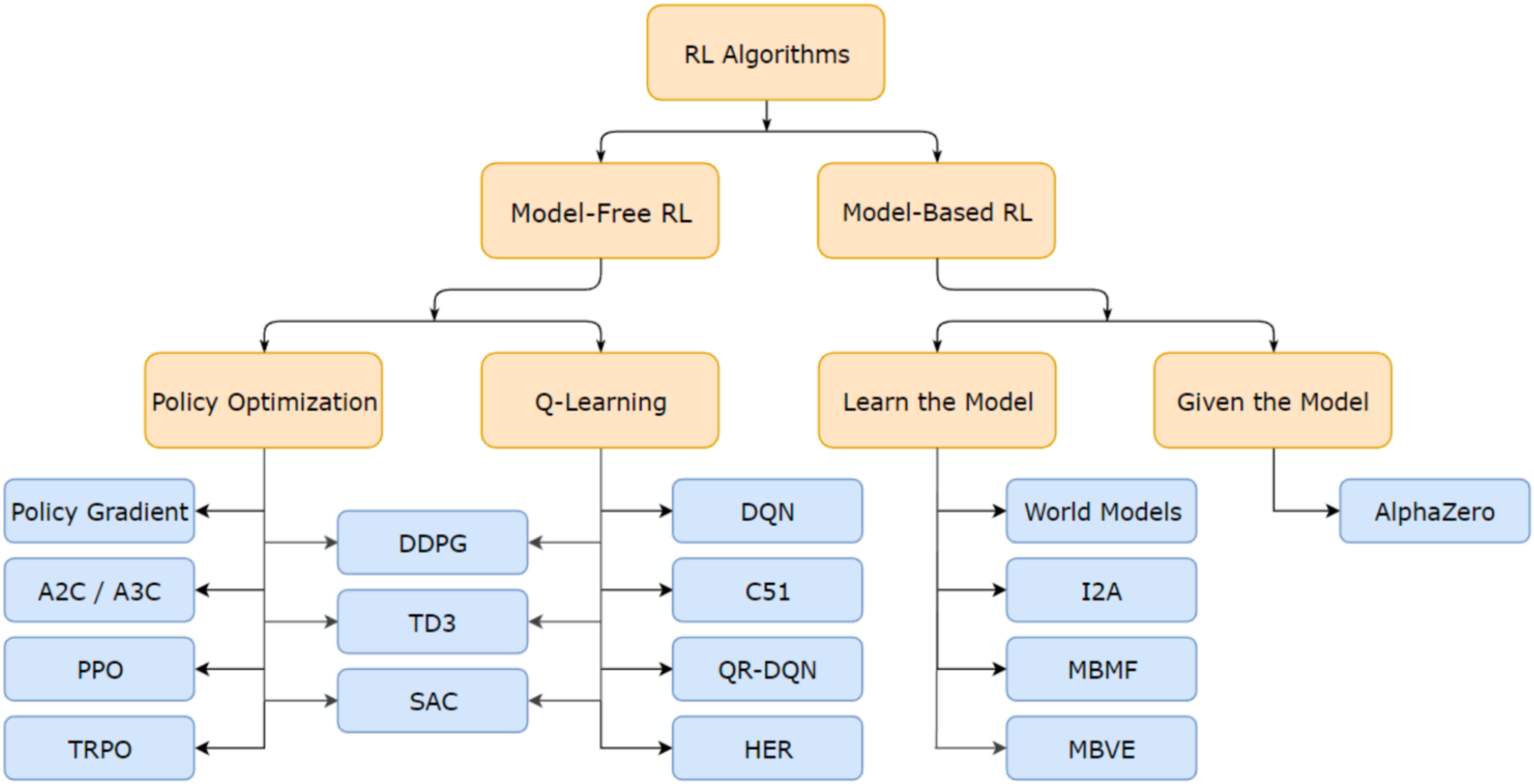


Figure: Figure adapted from David Silver's RL course

RL Algorithms

Taxonomy



Link: https://spinningup.openai.com

RL Algorithms

Taxonomy

- Learn the model of the world, then plan using the model
- Update model often
- Re-plan often

Value-based

- Learn the state or state-action value
- Act by choosing best action in state
- Exploration is a necessary add-on

Policy-based

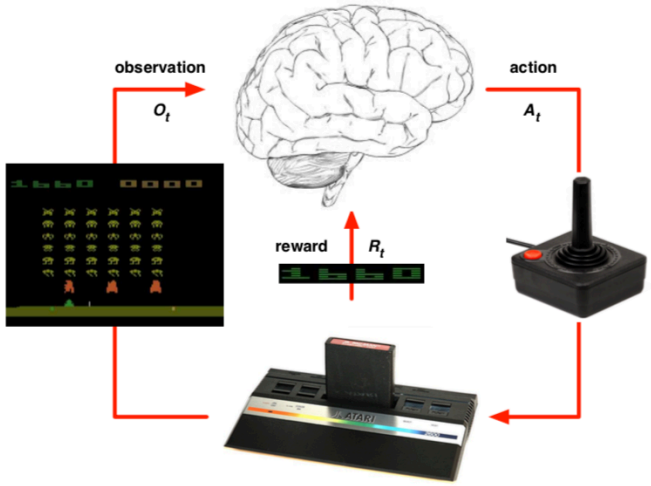
- Learn the stochastic policy function that maps state to action
- Act by sampling policy
- Exploration is baked in

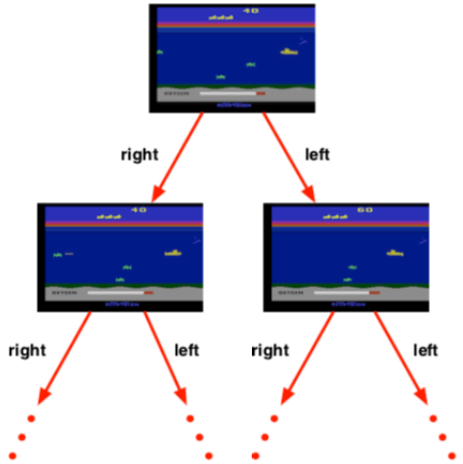
Learning and Planning

- Two fundamental problems in sequential decision making
  - Reinforcement Learning:
    - The environment is initially unknown
    - The agent interacts with the environment
    - The agent improves its policy
  - Planning:
    - A model of the environment is known
    - The agent performs computations with its model (without any external interaction)
    - The agent improves its policy
    - a.k.a. deliberation, reasoning, introspection, pondering, thought, search

Learning and Planning

- Atari Learning
  - Rules of the game are unknown
  - Learn directly from interactive game-play
  - Pick actions on joystick, see pixels and scores
- Atari Planning
  - Rules of the game are known
  - Can query emulator
    - perfect model inside agent's brain
  - If I take action a from state s:
    - what would the next state be?
    - what would the score be?
  - Plan ahead to find optimal policy
    - e.g. tree search





The Exploration and Exploitation Dilemma

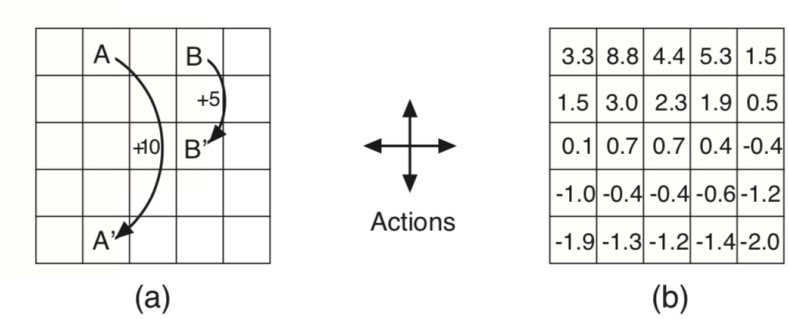
- Reinforcement learning is like trial-and-error learning
- The agent should discover a good policy
- From its experiences of the environment
- Without losing too much reward along the way
- **Exploration** finds more information about the environment
- **Exploitation** exploits known information to maximize reward
- It is usually important to explore as well as exploit

The Exploration and Exploitation Dilemma

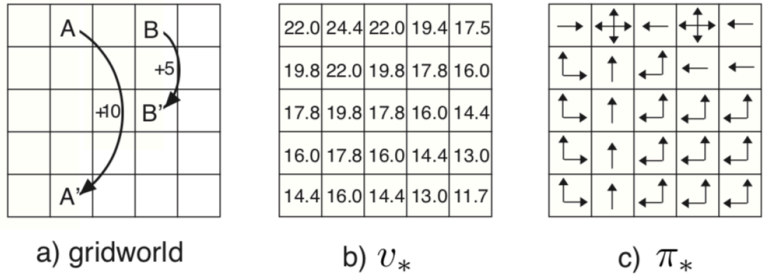
- Restaurant Selection
  - Exploitation Go to your favorite restaurant
  - Exploration Try a new restaurant
- Online Banner Advertisements
  - Exploitation Show the most successful advert
  - Exploration Show a different advert
- Oil Drilling
  - Exploitation Drill at the best known location
  - Exploration Drill at a new location
- Game Playing
  - Exploitation Play the move you believe is best
  - Exploration Play an experimental move

Prediction and Control

- **Prediction:** evaluate the future Given a policy
- **Control:** optimize the future Find the best policy
- What is the value function for the uniform random policy?

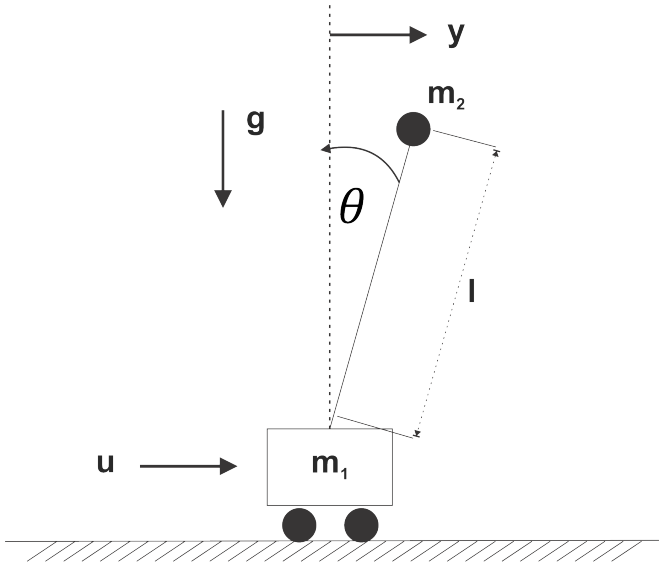


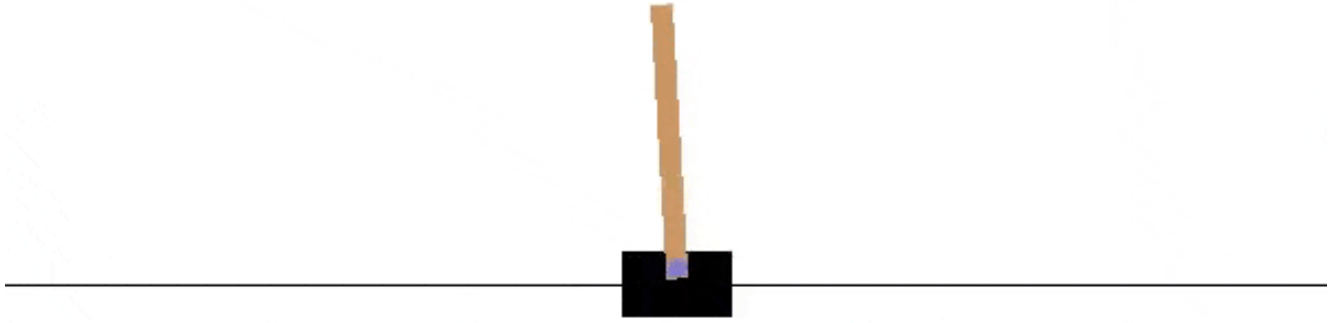
- What is the optimal value function over all possible policies? What is the optimal policy?



Examples of RL

Cart-pole balancing - traininng



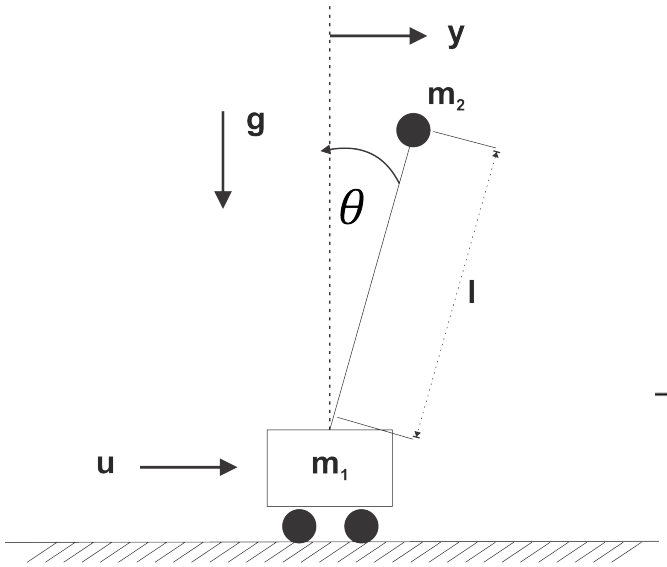


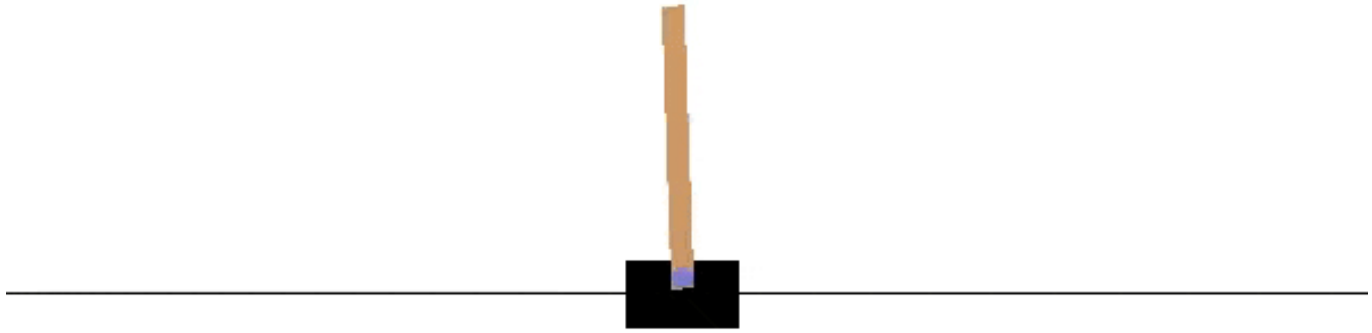
https://medium.com/@tuzzer/cart-pole-balancing-with-q-learning-b54c6068d947

- **Goal:** Balance the pole on top of a moving cart
- **State:** Pole angle, angular speed. Cart position, horizontal velocity
- **Actions:** horizontal force to the cart
- **Reward:** 1 at each time step if the pole is upright

Examples of RL

Cart-pole balancing – learned policy



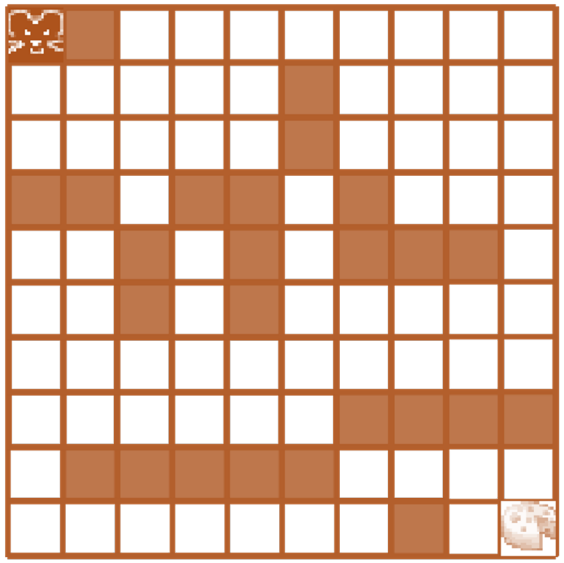


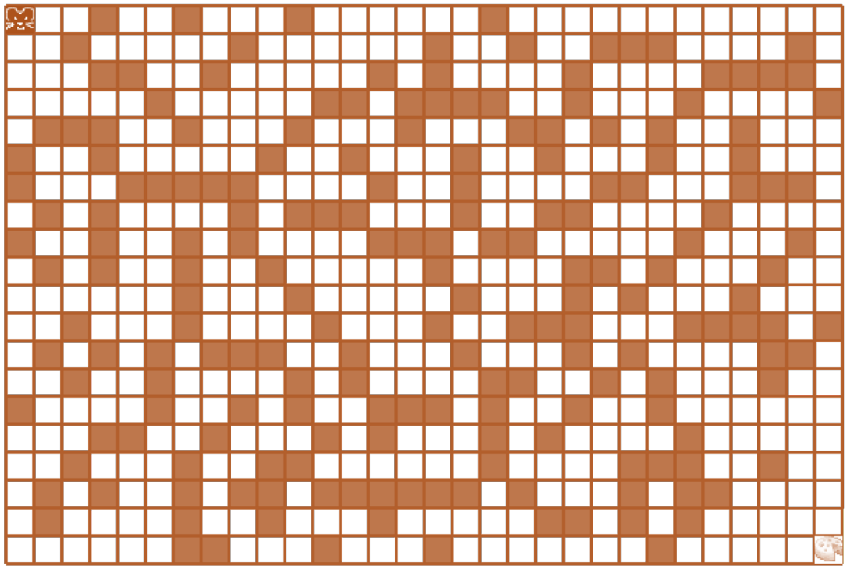
https://medium.com/@tuzzer/cart-pole-balancing-with-q-learning-b54c6068d947

- **Goal:** Balance the pole on top of a moving cart
- **State:** Pole angle, angular speed. Cart position, horizontal velocity
- **Actions:** horizontal force to the cart
- **Reward:** 1 at each time step if the pole is upright

Examples of RL

Maze-solving



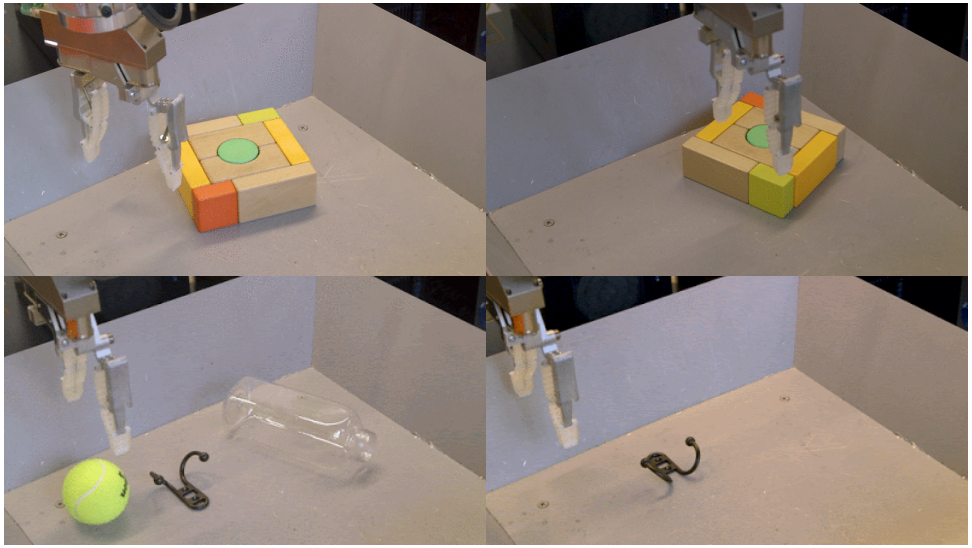


https://www.samyzaf.com/ML/rl/qmaze.html

- **Goal:** To get the cheese while avoiding collision
- **State:** Grid with cells that can be: occupied, free, target, visited
- **Actions:** left, up, right, down
- **Reward:**
  - 1 when the rat hits the cheese cell
  - -0.04 for each move from one cell to an adjacent cell
  - -0.8 for an attempt to move outside the maze boundaries
  - -0.75 when hit a blocked cell (dark-orange cell)
  - -0.25 points for any move to a cell which he has already visited
- **Stop criteria:** when the total reward hits -0.5 * maze.size

Examples of RL

Grasping Objects with Robotic Arm



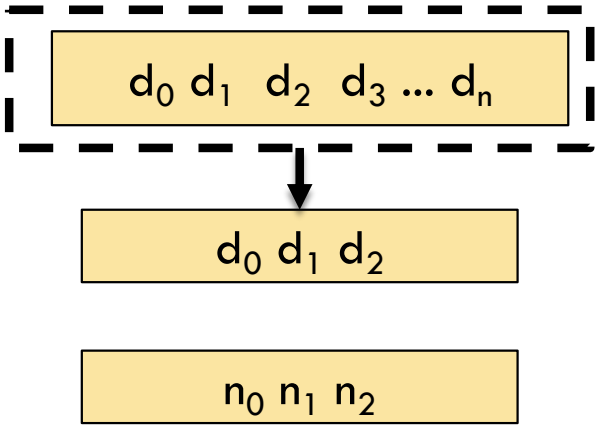
https://ai.googleblog.com/2018/06/scalable-deep-reinforcement-learning.html

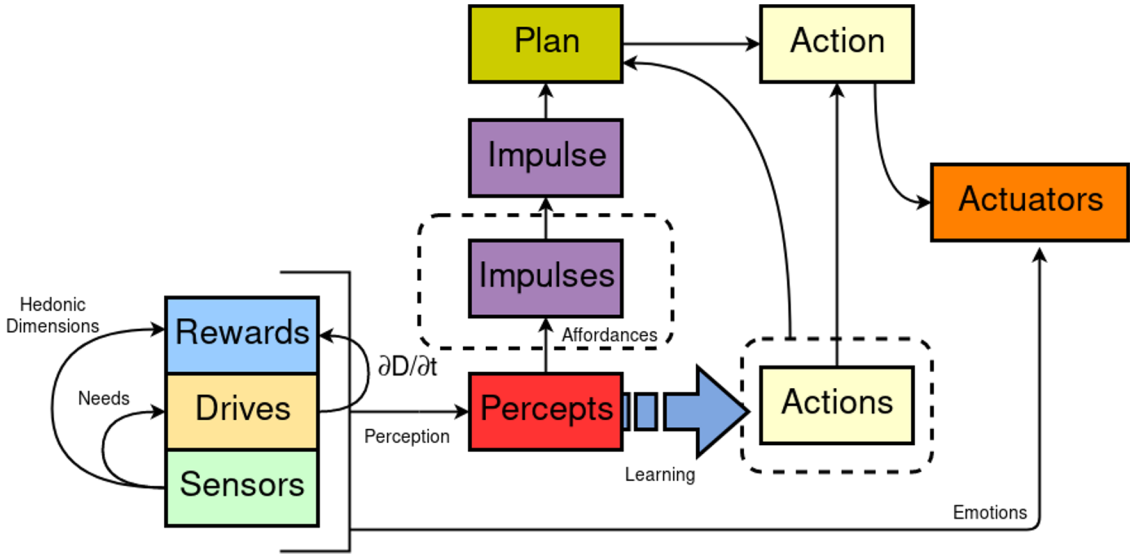
- **Goal:** Pick an object of different shapes
- **State:** Raw pixels from camera
- **Actions:** Move arm. Grasp
- **Reward:** Positive when pickup is successful

Examples of RL

Human Life

Homeostasis is a reference state for drives





- **Goal:** To satisfy one' needs
- **State:** Sight. Hearing. Taste. Smell. Touch. Level on unsatisfaction of needs (drives)
- **Actions:** Think. Move.
- **Reward:** Homeostasis of needs?

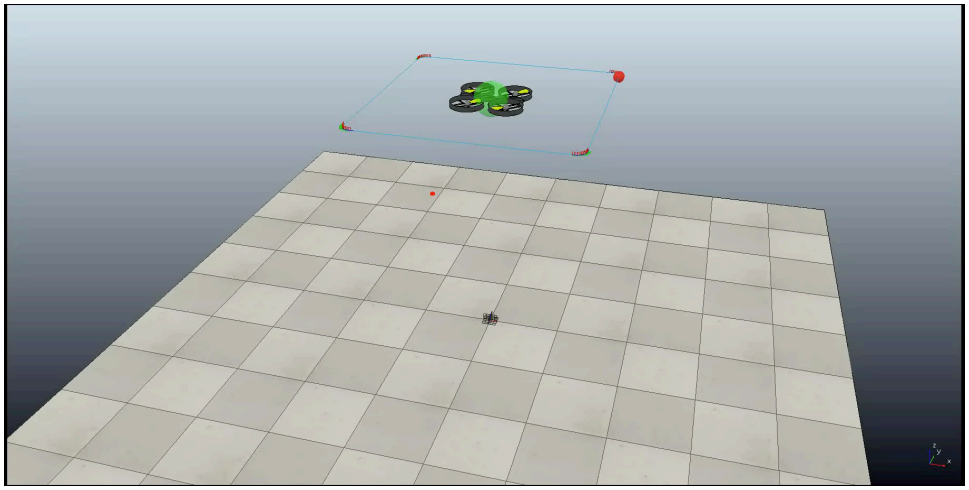
RL Applications

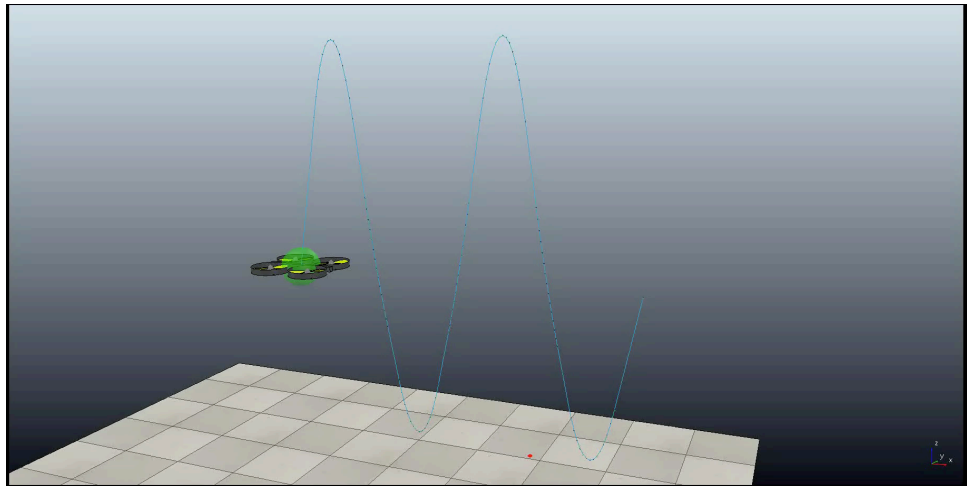
https://www.youtube.com/watch?v=W_gxLKSsSIE

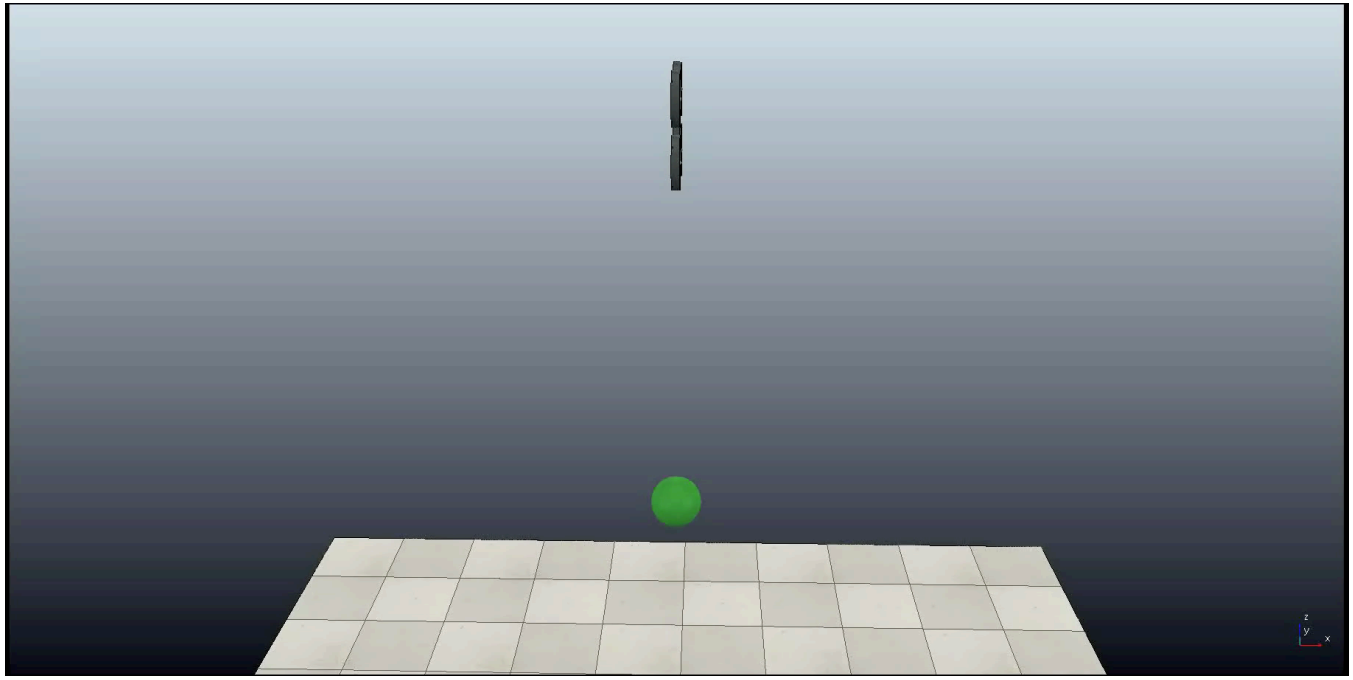
RL Applications

https://www.youtube.com/watch?v=2iNrJx6IDEo

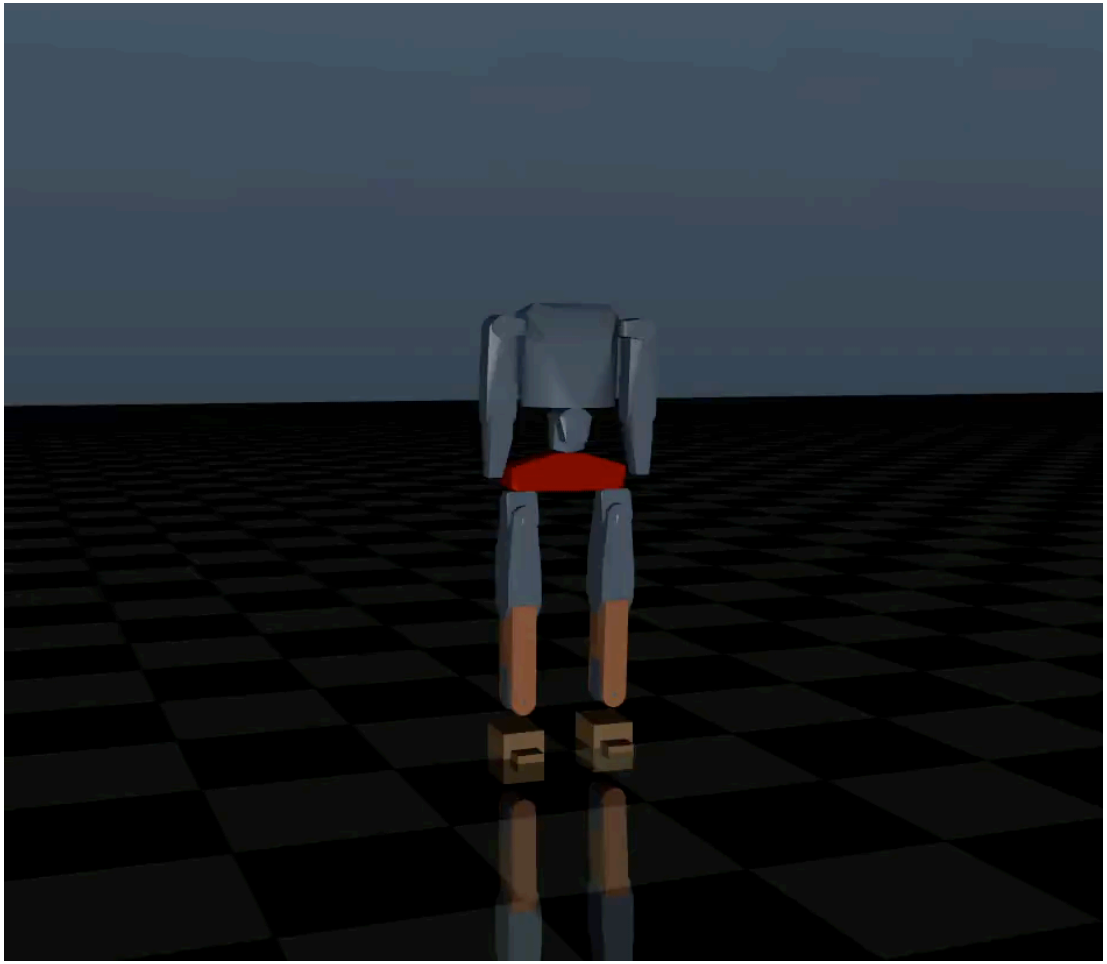
RL Applications





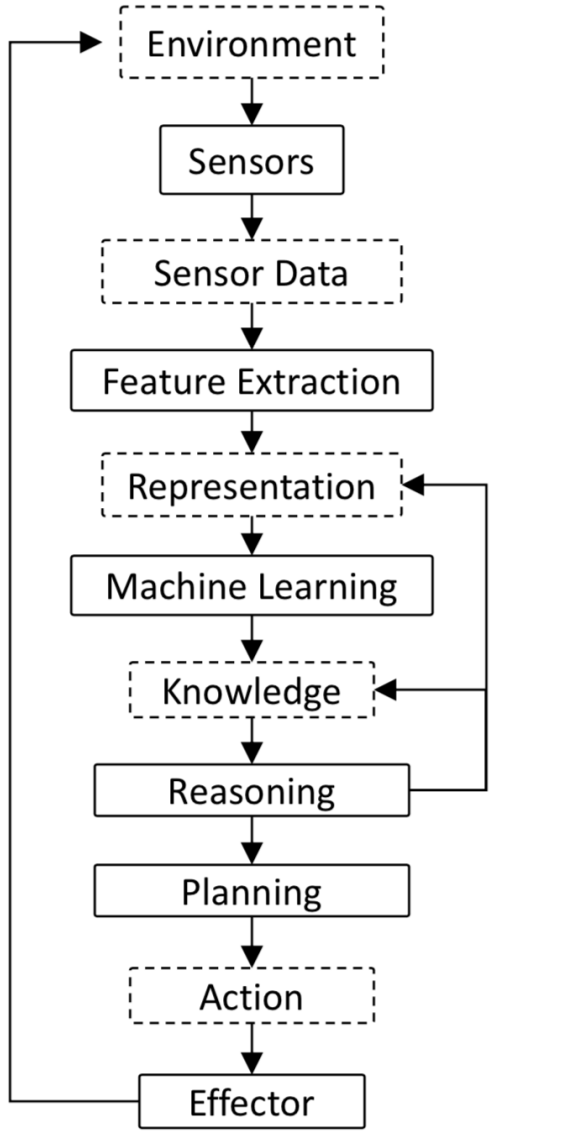


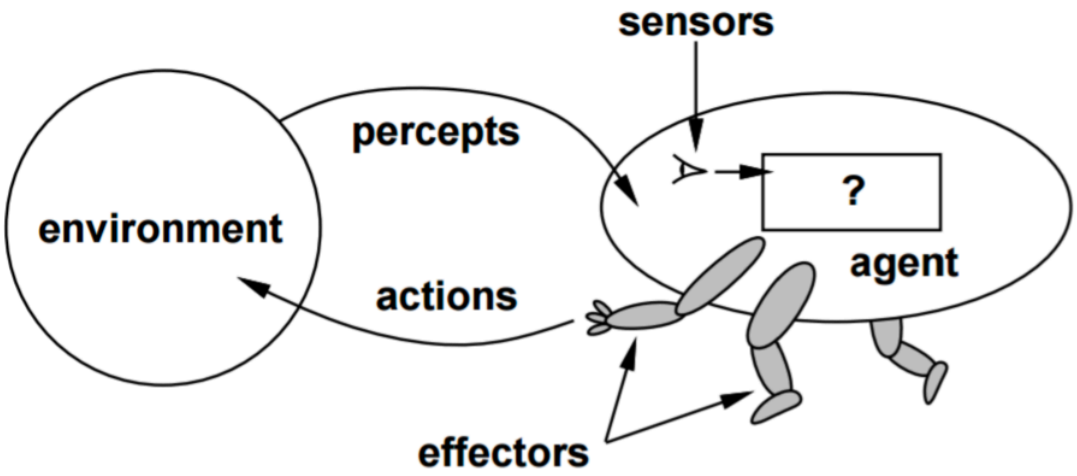
RL Applications: Model-based



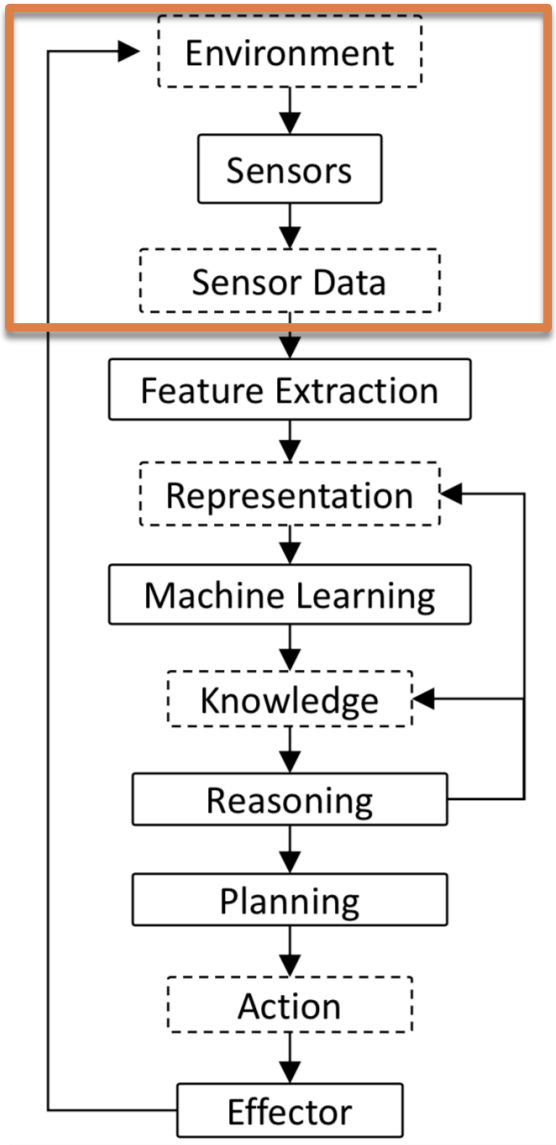
RL Applications

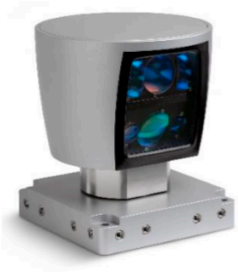
The promise of DRL





The promise of DRL





Lidar

Camera
(Visible, Infrared)

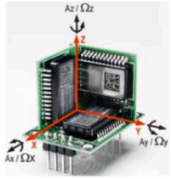
Radar

GPS

Stereo Camera

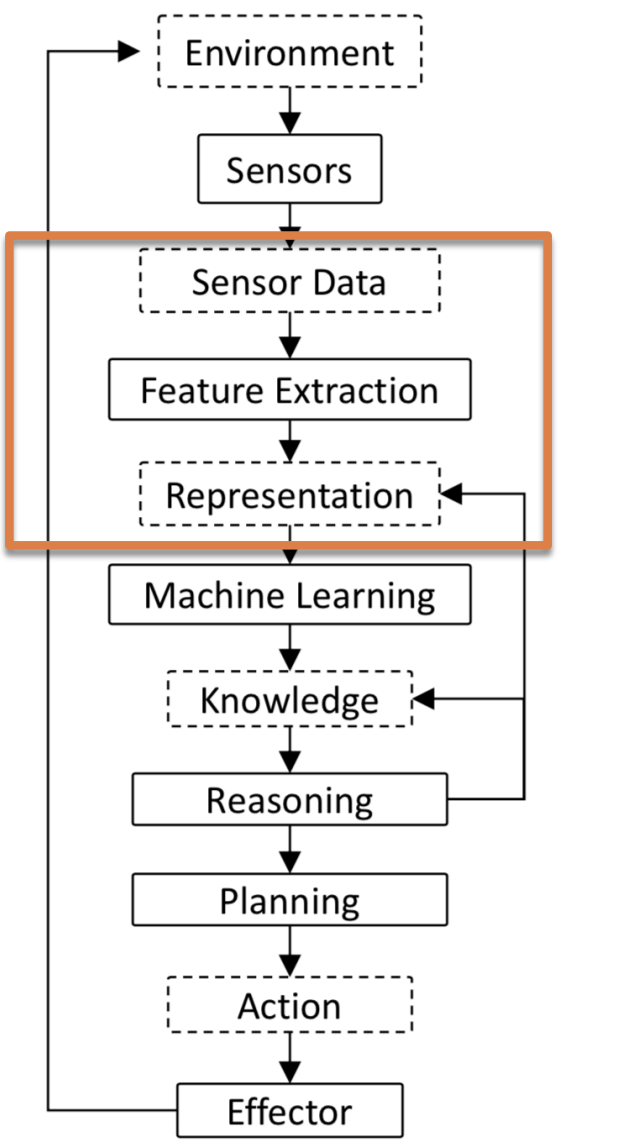
Microphone

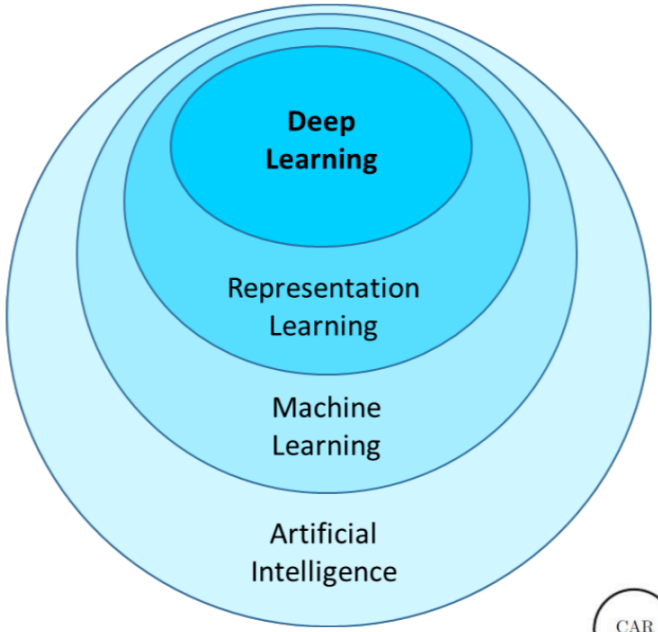
Networking
(Wired, Wireless)

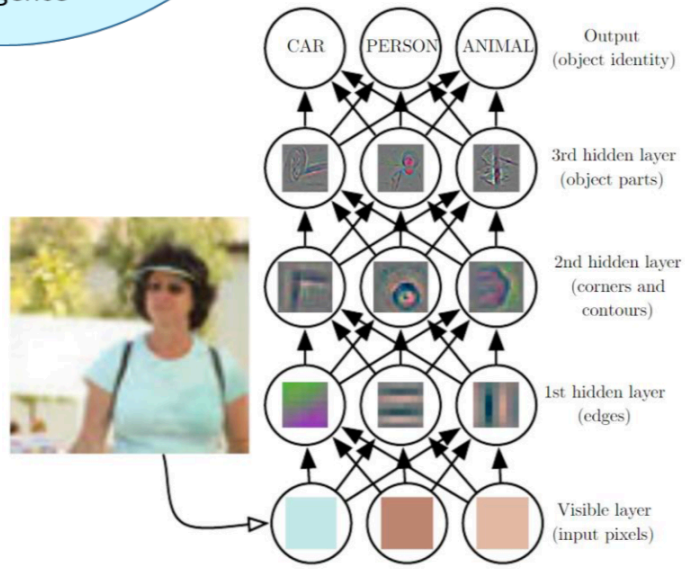


IMU

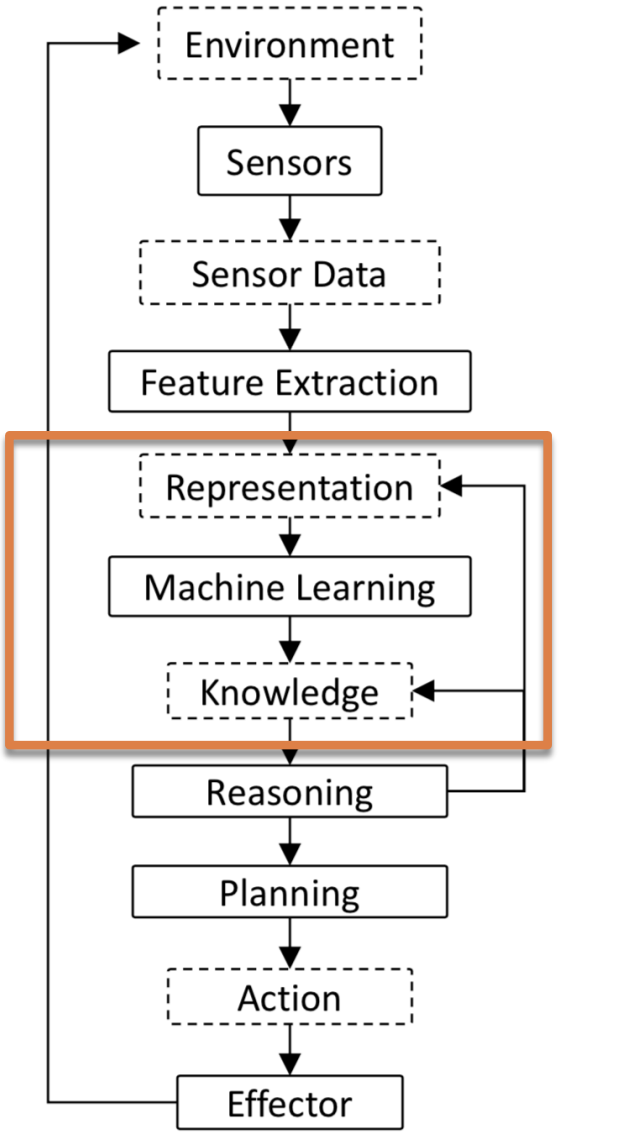
The promise of DRL

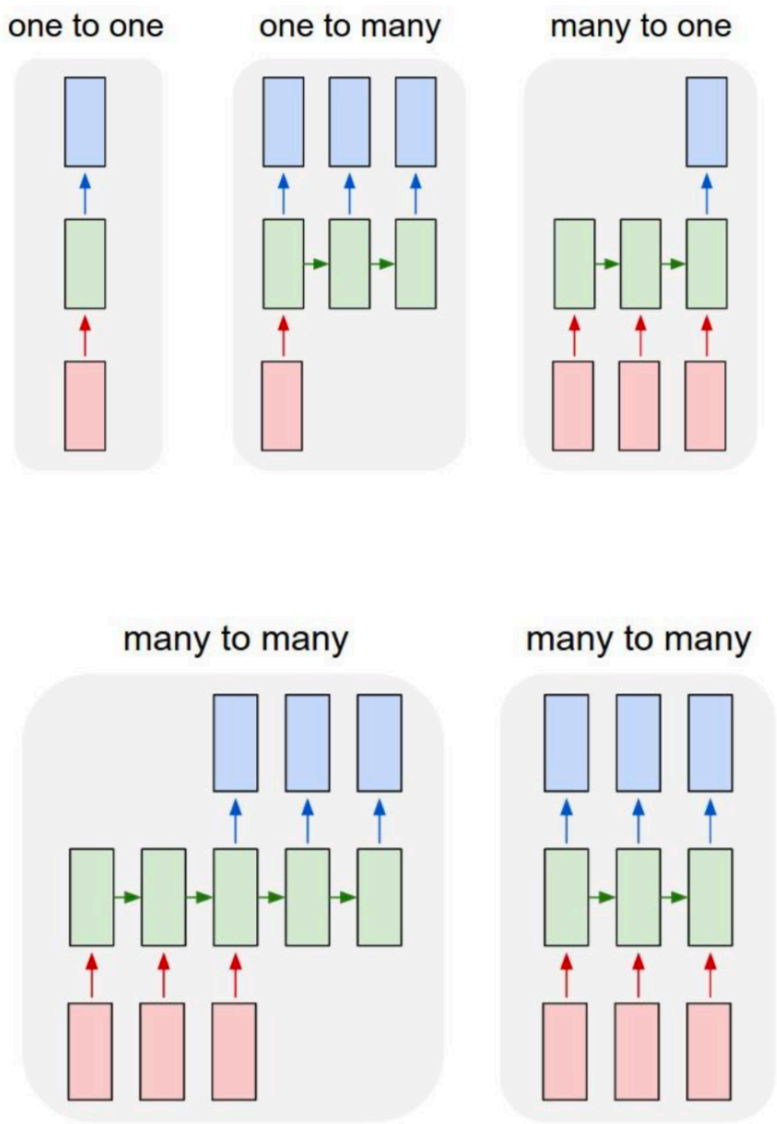






The promise of DRL





The promise of DRL

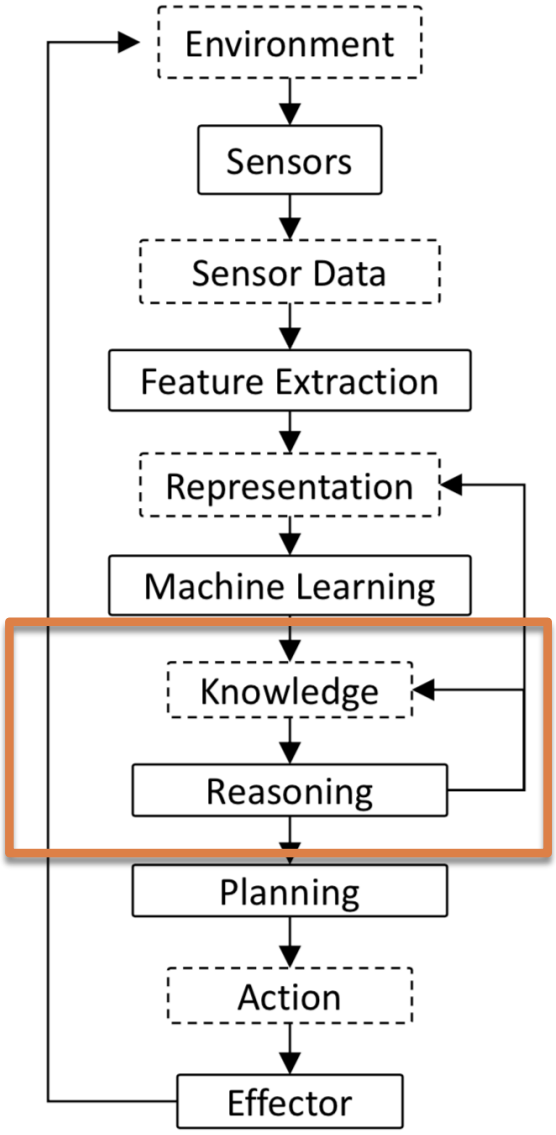
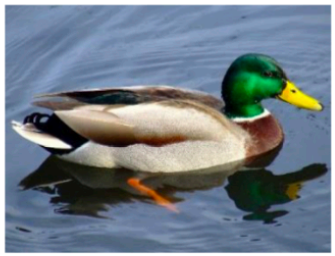
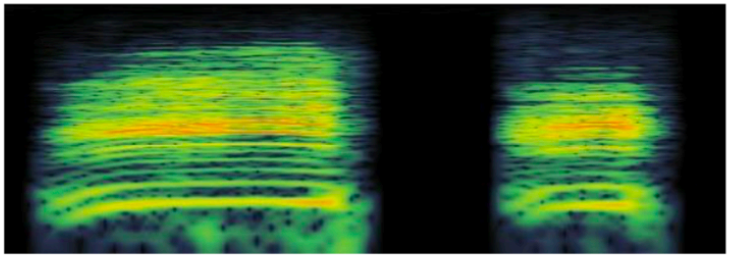


Image Recognition:
If it looks like a duck

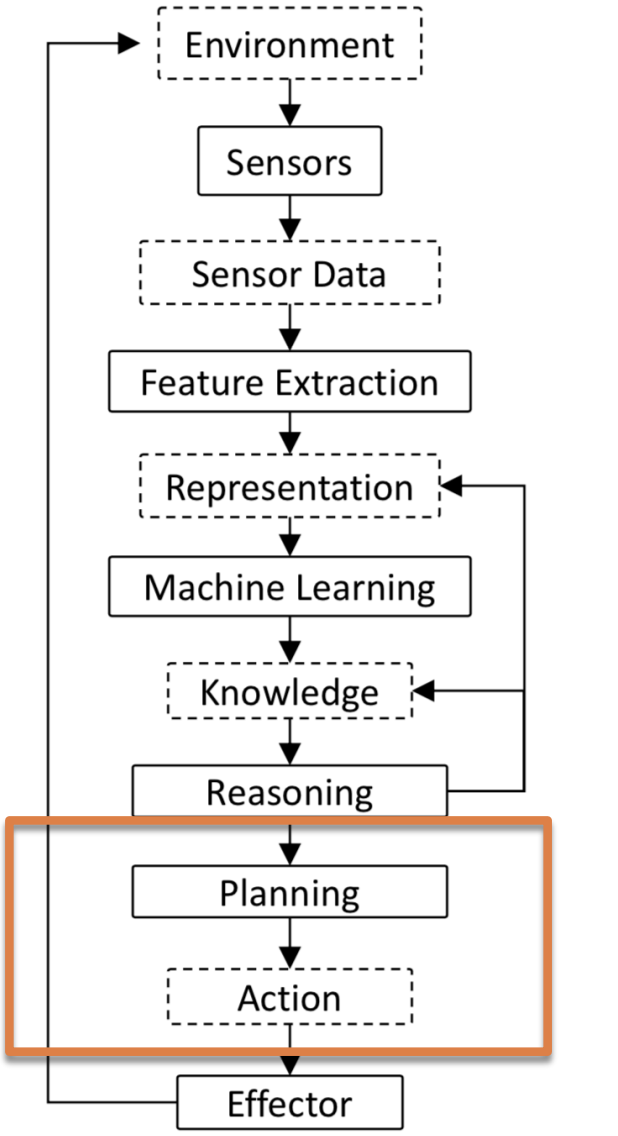


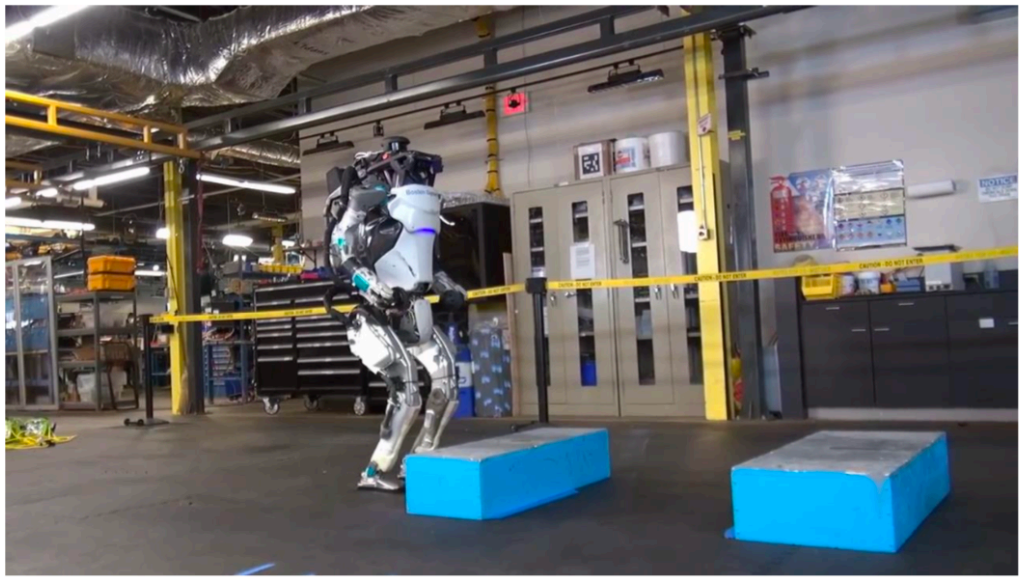
Audio Recognition:
Quacks like a duck

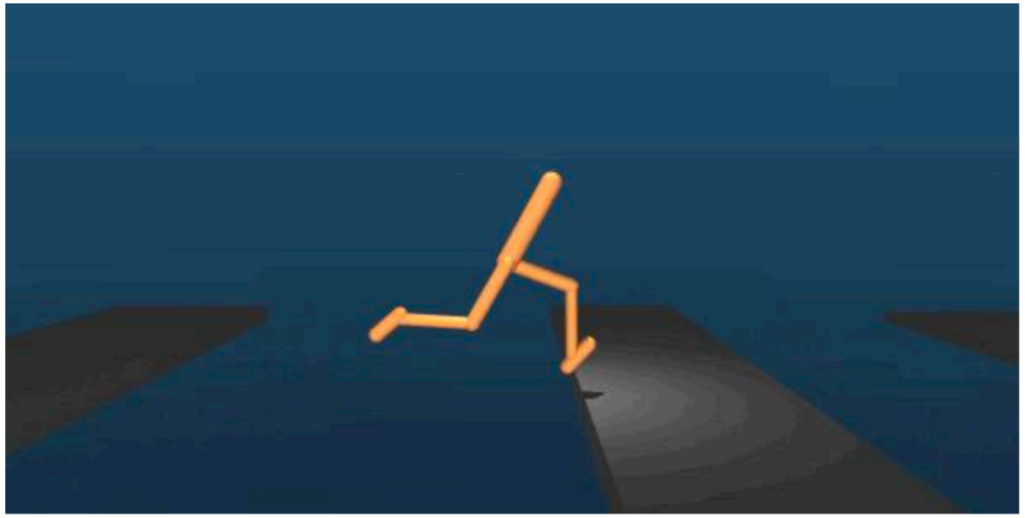


Activity Recognition:
Swims like a duck

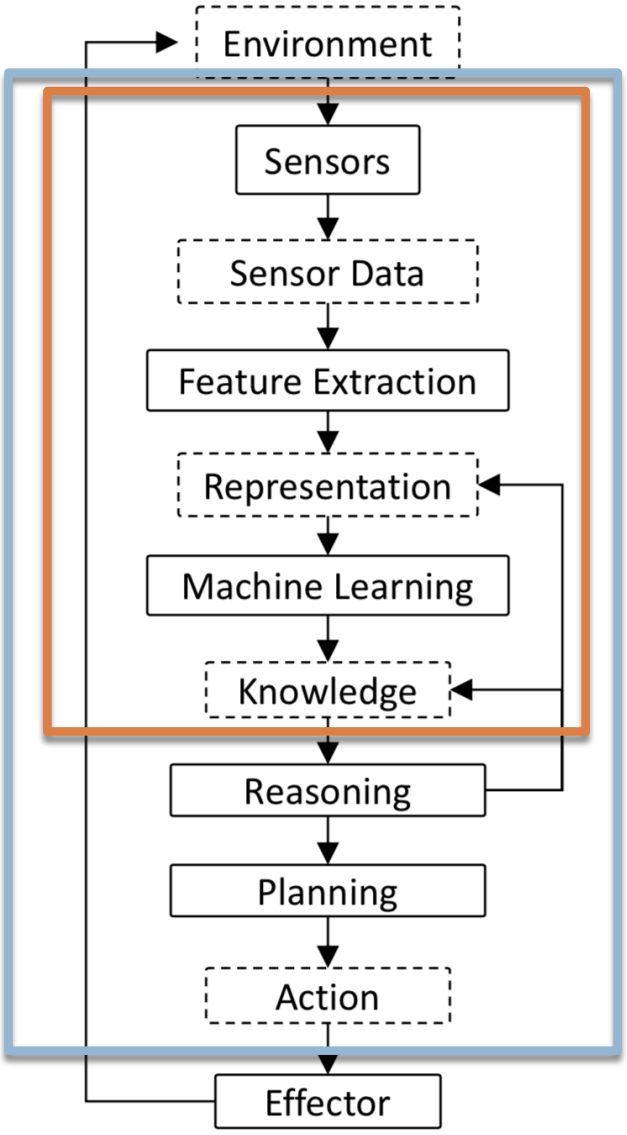
The promise of DRL







The promise of DRL



The promise of
Deep Learning

The promise of
Deep Reinforcement Learning

Considerations

- RL will solve many of your problems, however:
  - Needs a lot of training
- Choosing random actions can be dangerous and time consuming, but the system may not converge if they are not chosen
  - It can take a lot of time to learn
- Not all problems fit into MDP format
  - The algorithm finds the optimal solution (theoretically proved) in infinite iterations
  - That is, sometimes we have to settle for sub-optimal solutions

What next?

- Markov Decision Processes

- Reading:
  - BARTO, A., SUTTON, R. Reinforcement Learning: An Introduction. Second Edition.
    - High level introduction: Chapter 1

References

Lecture 1

- BARTO, A., SUTTON, R. Reinforcement Learning: An Introduction. Second Edition.
- David Silver, Introduction to Reinforcement Learning – Lecture Notes.
- Emma Brunskill , CS234 RL Winter 2020 Class, Stanford – Lecture Notes.
- Lex Fridman, MIT Deep Learning Course, MIT, 2019.
- RUSSEL, S. NORVIG, P. Artificial Intelligence: a modern approach. Prentice Hall, 2002.

This material is part of the Machine Learning Course

By Esther Colombini and Alexandre Simões