

DS-GA 3001 007 | Lecture 1

Reinforcement Learning

Jeremy Curuksu, PhD

NYU Center for Data Science

jeremy.cur@nyu.edu

February 1, 2023

Week 1

1. Course Information
2. Introduction to Reinforcement Learning

Course Information

DS-GA 3001 RL Instructional Team

Instructor:

- ▶ Dr. Jeremy Curuksu, jeremy.cur@nyu.edu

Section Leader:

- ▶ Anudeep Tubati, at5373@nyu.edu

Graders:

- ▶ Shreya Sinha, ss14468@nyu.edu
- ▶ Anudeep Tubati, at5373@nyu.edu

DS-GA 3001 RL Schedule

DS-GA 3001 RL Lecture:

- ▶ Thursdays from 4:55pm-6:35pm EST
- ▶ Location: GCASL, 238 Thompson St, Room 361

DS-GA 3001 RL Lab:

- ▶ Wednesdays from 8:10pm-9:00pm EST
- ▶ Location: Bobst Library, Room LL138

DS-GA 3001 RL Curriculum

Reinforcement Learning:

- ▶ 1. Introduction to Reinforcement Learning
- ▶ 2. Multi-armed Bandits
- ▶ 3. Dynamic Programming on Markov Decision Process
- ▶ 4. Model free RL Prediction and Control
- ▶ 5. Value Function Approximation (Deep RL)
- ▶ 6. Policy Function Approximation (Actor-Critic)
- ▶ 7. Planning from a Model of the Environment
- ▶ 8. Examples of Industrial Applications
- ▶ 9-10. Advanced Topics and Development Platforms

DS-GA 3001 RL Resources

- ▶ Lecture and lab practice code + lecture slides
- ▶ **Reinforcement Learning: An introduction** (2018) by R. Sutton and A. Barto
- ▶ Python RL libraries used in this course have free, recent and online documentations. The library we will use most is **OpenAI Gym** (maintained as **Gymnasium** since 2022)

Advice to Succeed in this Course

- ▶ **Attend both lectures and labs.** Lectures and labs complement each other to set you up for success
- ▶ **Read book from Sutton & Barto** at least chapt 3-10 and 13
- ▶ **Read the Gym documentation**, at least the introduction
- ▶ **Before implementing a RL solution**, define the agent's goal, states, actions, and reward functions
- ▶ **Implement, Document, and Customize RL code.** Modern RL libraries are very recent and not always reliable
- ▶ **Ask questions!**

Introduction to Reinforcement Learning

Introduction to Reinforcement Learning

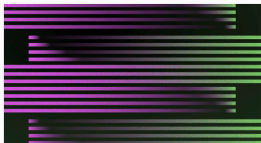
Today topics:

- ▶ What is Reinforcement Learning?
- ▶ Key components of Reinforcement Learning
- ▶ Introduction to the Gym Python library

What is Reinforcement Learning?

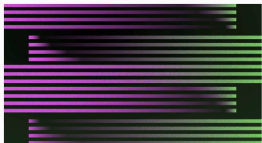
What is Reinforcement Learning?

Example: Please create a song about mathematics



What is Reinforcement Learning?

Example: Please create a song about mathematics

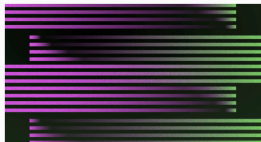


“ Math is the language of the universe,
It helps us understand, and make things diverse,
From the smallest atom to the vast expanse,
Math is the foundation, give it a chance! ”



What is Reinforcement Learning?

Example: Please create a song about mathematics



“ Math is the language of the universe,
It helps us understand, and make things diverse,
From the smallest atom to the vast expanse,
Math is the foundation, give it a chance! ”



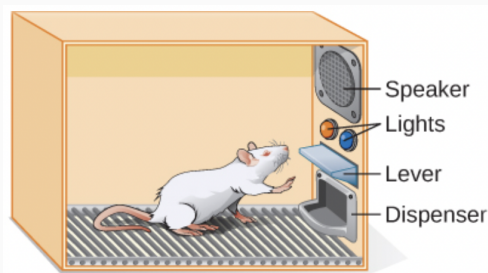
“ Sure, what's the new playlist name? ”

What is Reinforcement Learning?

Behavior is primarily shaped by reinforcement rather than free-will:

"Skinner saw human action as dependent on consequences of previous actions, a theory he called the principle of reinforcement: If the consequences to an action are bad, there is a high chance the action will not be repeated; if the consequences are good, the probability of the action being repeated becomes stronger"

On B. Skinner, 1904-1990 (in *Psychology*, Schacter, 2011)



What is Reinforcement Learning?

- ▶ **RL is the science of learning to make decisions from interactions with an environment**
 - ▶ **Industrial revolution and Machine Age (1750-1940)**
Automation of repeated physical solutions
 - ▶ **Digital revolution and Information Age (1960-Now)**
Automation of repeated mental solutions
 - ▶ **Artificial Intelligence revolution (Now -?)**
Allow machines to find solutions themselves
- ▶ **Why learn by reinforcement?**
 - ▶ Find previously unknown solutions
 - ▶ Find solutions online for unforeseen circumstances

Reinforcement Learning History

In short...

- ▶ **1850-1911:** Trial-and-error learning in psychology: The *Law of Effect* (Thorndike)
- ▶ **1927:** Theory of conditioned reflexes (Pavlov)
- ▶ **1938:** Radical behaviorism and principle of reinforcement (Skinner)
- ▶ **1948:** Trial-and-error learning in a computer system (Turing)
- ▶ **1952-1954:** Maze-running robot (Shannon), Reinforcement calculator (Minsky)
- ▶ **1957:** Dynamic programming to solve optimal control in MDP (Bellman)
- ▶ **1959:** Checkers-playing program (Samuel)
- ▶ **1960s-70s:** Learning automata, K-armed Bandits, Genetic algorithms
- ▶ **1972:** Model of classical conditioning based on temporal-difference (Klopf)
- ▶ **1973:** RL system ("with a critic") to play Blackjack
- ▶ **1983:** Actor-critic architecture to solve pole-balancing problem (Sutton, Barto)
- ▶ **1989:** Q-learning algorithm (Watkins)
- ▶ **1992:** Backgammon playing program (Tesauro's TD-Gammon)
- ▶ **2013:** Deep Q-Network reaches superhuman ability at Atari 2600
- ▶ **2016:** AlphaGo reaches superhuman ability at Go (Silver)

Reinforcement Learning Examples

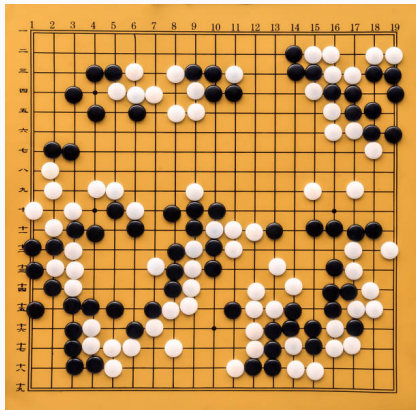
Problem requiring to make decisions with a goal in mind:

- ▶ Drive a car
- ▶ Fly a helicopter
- ▶ Manage a financial portfolio
- ▶ Play video or board game
- ▶ Control a power station
- ▶ Make a robot walk

Reinforcement Learning Examples

Play Go (to win)

- **State:**
Configuration of
the playing board
- **Action:**
Any valid move
- **Reward:**
Win = +1
Lose = -1
Else = 0



Reinforcement Learning Examples

Drive (safely to a destination)

► State:

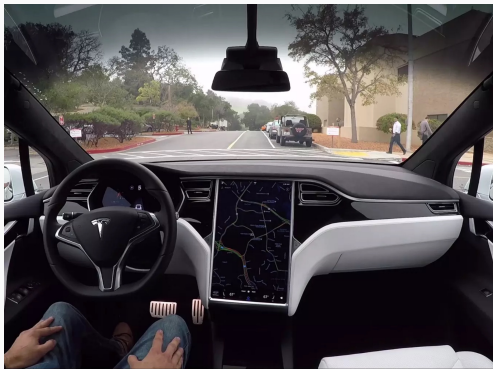
Camera pixels
Traffic
Weather

► Actions:

Steering wheel
Accelerator/Break

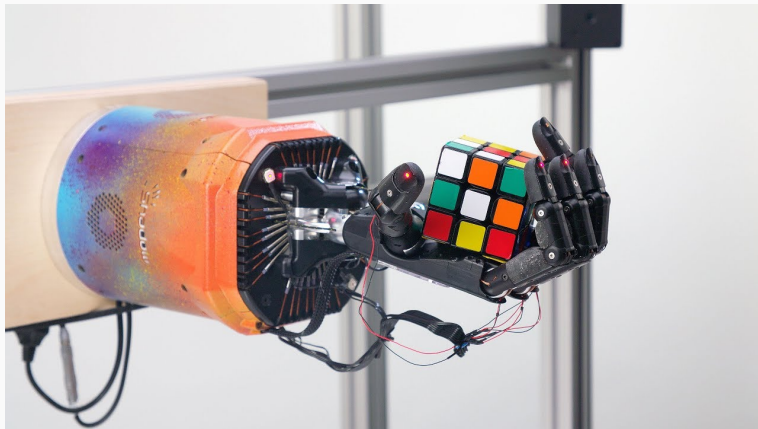
► Reward:

Destination = +1
Honking = -1
Collision = -100



Reinforcement Learning Examples

Example of robot operating on its own



How to formalize Reinforcement Learning?

- ▶ To find unknown or online solutions, intelligent beings learn by interacting with their environment:
 - ▶ Actively gather experience
 - ▶ Learn long-term consequences of actions
 - ▶ Predict an uncertain future
- ▶ Supervised Learning formalism is limited:

$$h_{\theta}(x^{(n)}) \mapsto y^{(n)} \qquad \theta^{(k+1)} = \theta^k - \alpha \frac{\partial J}{\partial \theta}$$

$$J(\theta) = \frac{1}{2N} \sum_{n=1}^N \left(h_{\theta}(x^{(n)}) - y^{(n)} \right)^2$$

How to formalize Reinforcement Learning?

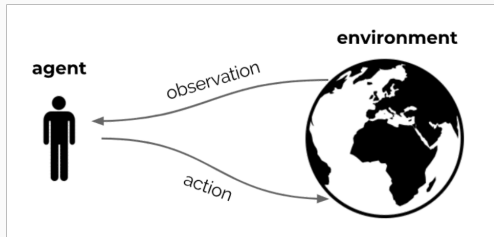
- ▶ To find unknown or online solutions, intelligent beings learn by interacting with their environment:
 - ▶ Actively gather experience
 - ▶ Learn long-term consequences of actions
 - ▶ Predict an uncertain future
- ▶ Supervised Learning formalism is limited:

$$h_{\theta}(x^{(n)}) \mapsto y^{(n)} \qquad \theta^{(k+1)} = \theta^k - \alpha \frac{\partial J}{\partial \theta}$$

$$J(\theta) = \frac{1}{2N} \sum_{n=1}^N \left(h_{\theta}(x^{(n)}) - y^{(n)} \right)^2$$

What if $y^{(n)}$ is not known?

Formalizing Reinforcement Learning



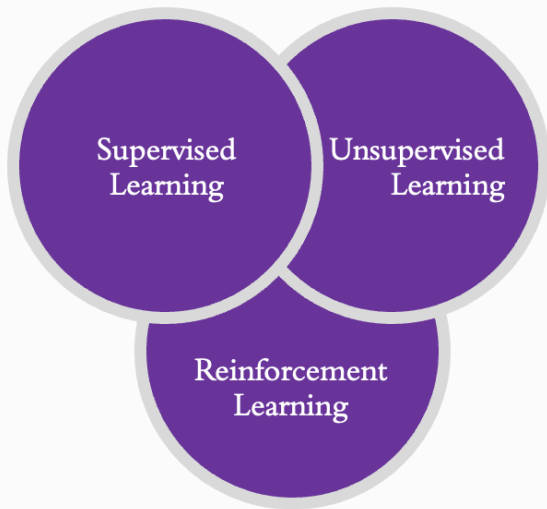
$$\pi_{\theta}(s_t) \mapsto p(a_t)$$

$$J(\theta) = ?$$

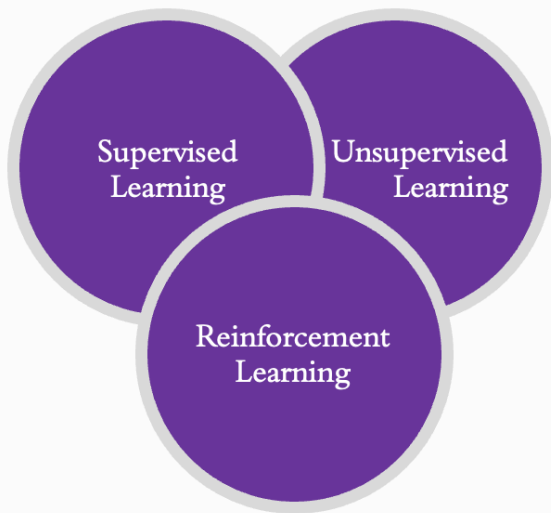
Science of learning to make decisions from interactions

- ▶ Sample experience by interacting with the environment
- ▶ No supervision, only reward signals
- ▶ Feedback can be delayed, time matters
- ▶ Goal-directed. Need learn how to act in sequence over time to reach goals

Formalizing Reinforcement Learning



Formalizing Reinforcement Learning



Key Components of Reinforcement Learning

Components of Reinforcement Learning



At each step, the agent:

- Receives observation o_t
- Executes action a_t
- Receives reward r_{t+1}

The environment:

- Receives action a_t
- Send reward r_{t+1}
- Send observation o_{t+1}

Components of Reinforcement Learning

A Reinforcement Learning solution has 3 components:

1. The Environment

2. The Agent:

- ▶ States and Observations
- ▶ Actions and Policies
- ▶ Value Function (for states and/or actions)
- ▶ Model of the environment dynamics

3. A Reward signal

The Environment

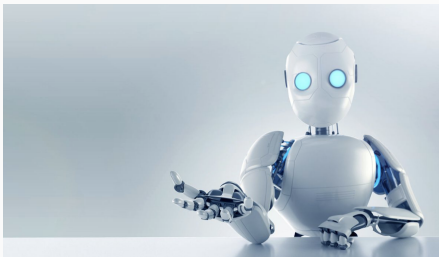
= What is **not** defined as the agent



- ▶ Well-defined in simulation environment, but infinite in reality
- ▶ The environment has its own internal state which is not usually visible to the agent. It may contain lot of irrelevant information
- ▶ Can be defined relative to the agent: the environment is what the agent *perceives* from it = observations and rewards

The Agent

= An entity equipped with sensors, effectors, and goals



- ▶ The agent performs actions to reach a goal
- ▶ It may learn *policies* mapping specific states to specific actions
- ▶ It may learn value functions for states or actions
- ▶ It may learn a model of the environment dynamics

States and Observations

The agent state s_t captures information available to the agent at step t about its environment

- ▶ An observation a.k.a. sensation is the (raw) input of the agent's sensors such as measurements, images, tactile signals
- ▶ A state can be a raw or processed observation, or a structure built up over time from sequences of observations
- ▶ The agent state is often not the same as the environment state. It depends on what the agent observes on the environment
- ▶ In the simplest case, there is only one state (next lecture)

Defining the Agent State

The state can depend on previous states and actions

- ▶ The agent may see the full environment state or only get partial observations
- ▶ A history is a sequence of observations, actions, and rewards:

$$\text{history}_t = (o_0, a_0, r_1, o_1, \dots, \dots, o_{t-1}, a_{t-1}, r_t, o_t)$$

- ▶ The agent state can be defined as a function of the history:

$$s_{t+1} = f(s_t, a_t, r_{t+1}, o_{t+1})$$

where f is a *state update function*

- ▶ The agent's action depends on its state

Defining the Agent State

Example of Maze Environment



- The agent may observe the full environment

Defining the Agent State

Example of Maze Environment



- The agent may only partially observe the environment

Defining the Agent State

Example of Maze Environment



- The agent may only partially observe the environment

Defining the Agent State

Example of Maze Environment



- How would you construct the agent state to distinguish between these two locations?

Actions and Policies

The goal of the agent is to select actions to maximise expected accumulated reward across an entire policy

- ▶ **Actions may have long term consequences** on future accessible states and reward: reward may be *delayed*
- ▶ **Examples of Actions:**
 - ▶ Suggest a song or a movie
 - ▶ Translate & rotate articulation joints of a robot
 - ▶ Manage a financial portfolio (may take months to mature)
 - ▶ Block opponent moves (help win game many moves later)
- ▶ **A policy is a function that maps states to actions:**
 - ▶ Deterministic policy: $\pi : s \mapsto a$
 - ▶ Stochastic policy: $\pi : s \mapsto p(a_1|s), p(a_2|s), \dots, p(a_k|s)$

Reward and Return

The rewards received evaluate the actions taken

- ▶ A reward r_t at time t is a scalar feedback signal which defines the agent's goal = maximize the accumulated rewards:

$$G_t = r_{t+1} + r_{t+2} + \dots r_{\text{end}}$$

- ▶ G_t is called the return
- ▶ **Examples:**
 - ▶ $r_T = +1$ or $r_T = 0$ (win or lose), $r_t = 0$ if $t \neq T$
 - ▶ $r_t = -n \times \text{price}$ or $r_t = n \times \text{price}$ (buy or sell n stocks)
- ▶ **Definition of the Reward Hypothesis:** "All goals and purposes can be encoded mathematically as maximizing the sum of accumulated rewards"

State Value $v_{\pi}(s)$

The value of a state is the expected return for that state

- Value of a state s for a given policy π :

$$v_{\pi}(s) = \mathbb{E}_{\pi}(G_t | s)$$

$v_{\pi}(s)$ indicates how good it is to be in s when following π

- G_t can be discounted to trade off short- vs. long-term rewards:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots$$

- G_t can be defined recursively, and so too can $v_{\pi}(s)$:

$$G_t = r_{t+1} + \gamma G_{t+1}$$

$$v_{\pi}(s) = \mathbb{E}_{\pi}(r_{t+1} + \gamma G_{t+1} | s)$$

$$v_{\pi}(s) = \mathbb{E}_{\pi}(r_{t+1} + \gamma v_{\pi}(s_{t+1}) | s) \quad (\text{Bellman equation})$$

State-Action Value $q_{\pi}(s, a)$

A value can also be defined for each action in a state

- ▶ Value of an action a in a state s for a given policy π :

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}(G_t | s, a)$$

$q_{\pi}(s, a)$ indicates how good it is to choose a in s under π

- ▶ **Again $q_{\pi}(s, a)$ can be defined recursively:**

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}(r_{t+1} + \gamma q_{\pi}(s_{t+1}, a_{t+1}) | s, a)$$

- ▶ **Bellman equations hold for optimal (=highest possible) values:**

$$q_*(s, a) = \mathbb{E}(r_{t+1} + \gamma \max_{a_k} q_*(s_{t+1}, a_k) | s, a)$$

They are used to create RL algorithms (example: Q-learning)

Model of Environment Dynamics

A model can be used to simulate an environment

- **Reward function for Bandits model** (single state model):

$$p(r | a) = p(r_{t+1} = r | a_t = a) \Leftrightarrow r(a) = \mathbb{E}(r, a)$$

- **General transition function to predict next state and reward:**

$$p(s', r | s, a) = p(s_{t+1} = s', r_{t+1} = r | s_t = s, a_t = a)$$

A RL model does not infer policies (does not "act")

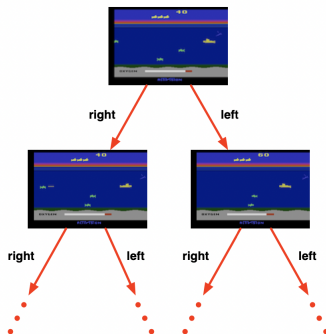
- **A model can be a fully observable Markov Decision Process:**

$$p(s, r | s_t, a_t) = p(s, r | H_t, a_t)$$

Or a *partially observable* Markov Decision Process (POMDP)

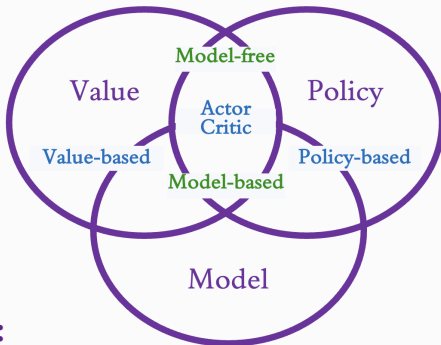
Model of Environment Dynamics

Planning from a model:



- ▶ If the rules of the game are perfectly known, couldn't an agent plan ahead and find an optimal policy?
- ▶ An agent could "plan" (= think): *If I take action a from state s , what would the next state be? What would the score be?*
- ▶ Querying a game emulator is an example of (generative) model

RL Taxonomy



Categories of RL agents:

- ▶ **Model-based:** Use a model to learn policy and/or value function (lecture 3, 7)
- ▶ **Model-free:** Learn policy and/or value function without model (lecture 4-5)
- ▶ **Value-based:** Learn value function, not policy function (lectures 4-5)
- ▶ **Policy-based:** Learn policy function, not value function (lecture 6)
- ▶ **Actor-Critic:** Learn policy and value functions (lecture 6)

RL Challenges and Opportunities

1. **Control vs. Prediction.** How to search for optimal policies and estimate their values simultaneously?
2. **Exploration vs. Exploitation.** An agent must explore to find information, yet must exploit known information to get reward
3. **Generalization to large state-action spaces.** For example the game Go has over 10^{170} possible positions
4. **Combining Learning with Planning:** In addition to interact with the environment, an agent can learn and plan from a model
5. **Simulated vs. real experience.** *Trial-and-error* impractical
6. **Other forms of learning:** From rewards? From demonstrations? Imitations? Supervisions? What does Psychology teach us?
7. **Convergence vs. tracking and adaptability**

Introduction to Python's Gym library

Practice: RL in Gym environments

What is Gym?

- ▶ An open source toolkit for testing RL algorithms
- ▶ Provides you with baseline RL environments
- ▶ Has a standard API to access these environments
- ▶ Up to you to create RL agents to interact with these environments
- ▶ Widely used, safe simulations

Practice: RL in Gym environments

Example of Gym environments: Atari 2600 video games



- ▶ Rules of games unknown by agent
- ▶ Agent can learn directly from interactive game-play
- ▶ Agent picks actions, sees pixels, receives scores

Practice: RL in Gym environments

Implementing a Gym environment

1.

Instantiate environment

```
env =  
gym.make("name")
```

- ▶ Assign it to a variable
- ▶ Access it anytime later

2.

Initialize environment

```
s =  
env.reset()
```

- ▶ Used to begin an episode
- ▶ Return initial state

3.

Select actions

```
a =  
yourAgent(state)
```

- ▶ Apply your policy
- ▶ Map state to action(s)

4.

Step through environment

```
s, r, done, info =  
env.step(action)
```

- ▶ Used at every step
- ▶ Test if episode should end

Practice

What comes next...

DS-GA 3001 RL Syllabus:

1. Introduction to Reinforcement Learning
2. Multi-armed Bandit
3. Markov Decision Process and Dynamic Programming
4. Model free RL Prediction and Control
5. Function Approximation
6. Policy Optimization
7. Planning from a RL Model
8. Examples of Industrial Applications
9. Advanced RL Topics
10. RL Development Platforms

Thank you!