CS 4803 / 7643
Deep Learning, Fall 2020

Reinforcement Learning: Module 1/3

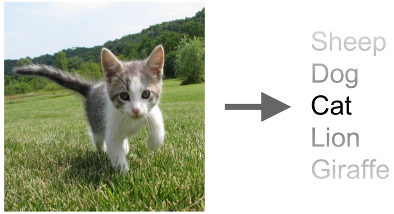
Presented by Nirbhay Modhe

Georgia Tech

Reinforcement Learning Introduction

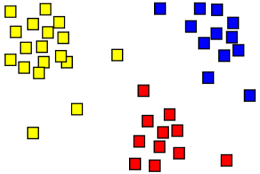
Supervised Learning

- Train Input: $\{X, Y\}$
- Learning output: $f : X \rightarrow Y$, $P(y|x)$
- e.g. classification



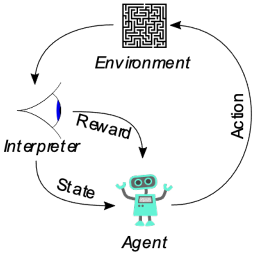
Unsupervised Learning

- Input: $\{X\}$
- Learning output: $P(x)$
- Example: Clustering, density estimation, etc.



Reinforcement Learning

- Evaluative feedback in the form of **reward**
- No supervision on the right action



RL: Sequential decision making in an environment with evaluative feedback.

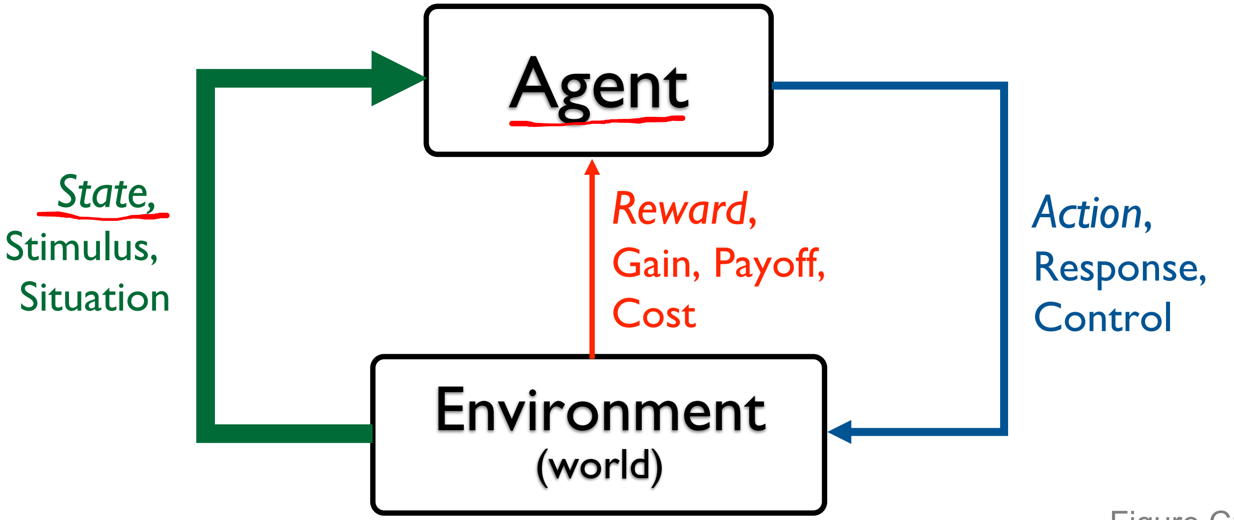


Figure Credit: Rich Sutton

- **Environment** may be unknown, non-linear, stochastic and complex.
- **Agent** learns a **policy** to map states of the environments to actions.
  - Seeking to maximize cumulative reward in the long run.

RL: Sequential decision making in an environment with evaluative feedback.

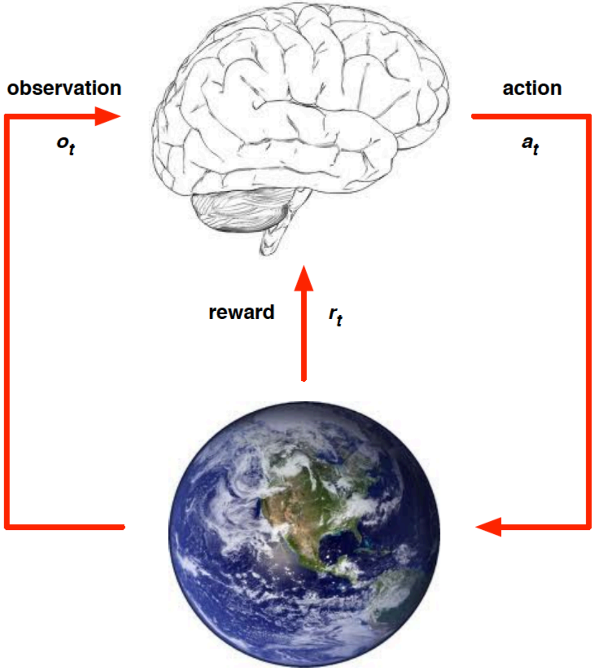
Evaluative Feedback

- Pick an action, receive a reward (positive or negative)
- No supervision for what the "correct" action is or would have been, unlike supervised learning

Sequential Decisions

- Plan and execute actions over a sequence of states
- Reward may be delayed, requiring optimization of future rewards (long-term planning).

RL: Environment Interaction API



- At each time step t, the agent:
  - Receives observation o_t
  - Executes action a_t
- At each time step t, the environment:
  - Receives action a_t
  - Emits observation o_{t+1}
  - Emits scalar reward r_{t+1}

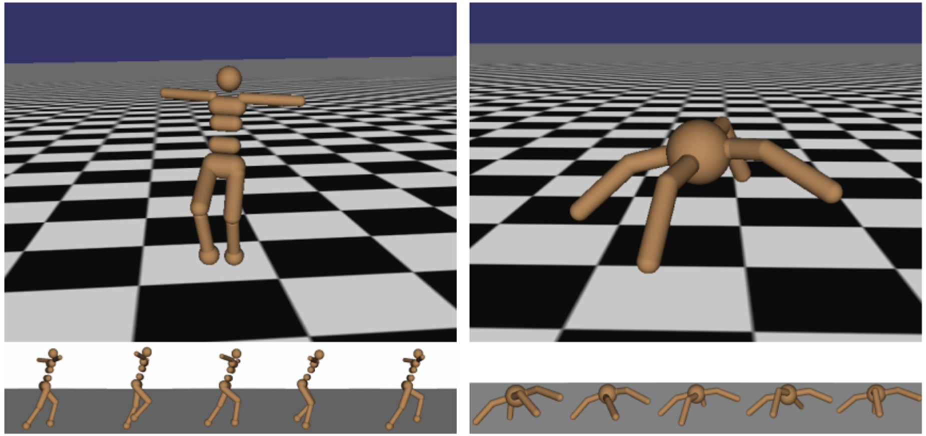
Slide credit: David Silver

Signature Challenges in Reinforcement Learning

- Evaluative feedback: Need trial and error to find the right action
- Delayed feedback: Actions may not lead to immediate reward
- Non-stationarity: Data distribution of visited states changes when the policy changes
- Fleeting nature of time and online data

Slide adapted from: Richard Sutton

Robot Locomotion



Figures copyright John Schulman et al., 2016. Reproduced with permission.

- **Objective**: Make the robot move forward
- **State**: Angle and position of the joints
- **Action**: Torques applied on joints
- **Reward**: +1 at each time step upright and moving forward

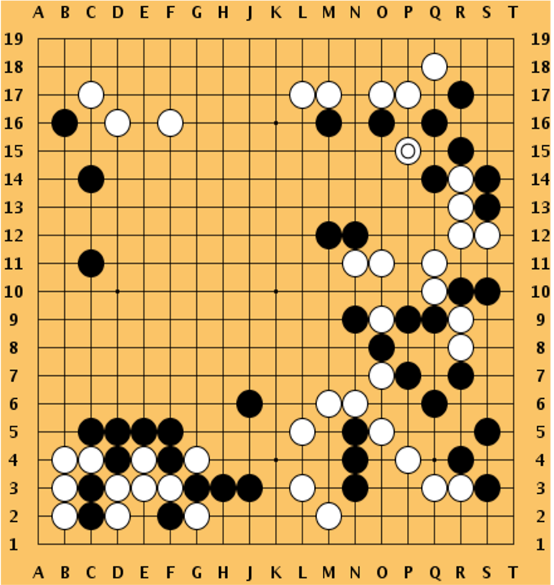
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Atari Games

- **Objective**: Complete the game with the highest score
- **State**: Raw pixel inputs of the game state
- **Action**: Game controls e.g. Left, Right, Up, Down
- **Reward**: Score increase/decrease at each time step

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Go



- **Objective**: Defeat opponent
- **State**: Board pieces
- **Action**: Where to put next piece down
- **Reward**: +1 if win at the end of game, 0 otherwise

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Markov Decision Processes

- **MDPs**: Theoretical framework underlying RL

- **MDPs**: Theoretical framework underlying RL
- An MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$
  - $\mathcal{S}$: Set of possible states
  - $\mathcal{A}$: Set of possible actions
  - $\mathcal{R}(s, a, s')$: Distribution of reward
  - $\mathbb{T}(s, a, s')$: Transition probability distribution, also written as p(s'|s,a)
  - γ : Discount factor

Markov Decision Processes (MDPs)

- **MDPs**: Theoretical framework underlying RL
- An MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$
  - $\mathcal{S}$: Set of possible states
  - $\mathcal{A}$: Set of possible actions
  - $\mathcal{R}(s, a, s')$: Distribution of reward
  - $\mathbb{T}(s, a, s')$: Transition probability distribution, also written as p(s'ls,a)
  - γ : Discount factor
- **Interaction trajectory**: $\ldots s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, r_{t+2}, s_{t+2}, \ldots$

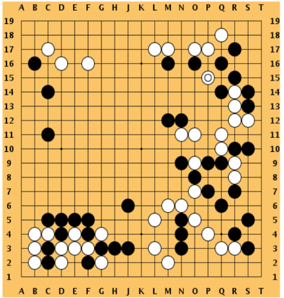
- **MDPs**: Theoretical framework underlying RL
- An MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$
  - $\mathcal{S}$: Set of possible states
  - $\mathcal{A}$: Set of possible actions
  - $\mathcal{R}(s, a, s')$: Distribution of reward
  - $\mathbb{T}(s, a, s')$: Transition probability distribution, also written as p(s'|s,a)
  - γ : Discount factor
- **Interaction trajectory**: $\ldots s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, r_{t+2}, s_{t+2}, \ldots$
- **Markov property**: Current state completely characterizes state of the environment
- **Assumption**: Most recent observation is a sufficient statistic of history

$$p\left(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, \ldots S_0 = s_0\right) = p\left(S_{t+1} = s' | S_t = s_t, A_t = a_t\right)$$

Markov Decision Processes (MDPs)

Fully observed MDP

- Agent receives the true state s_t at time t
- Example: Chess, Go



Partially observed MDP

- Agent perceives its own partial observation o_t of the state s_t at time t, using past states e.g. with an RNN
- Example: Poker, First-person games (e.g. Doom)

Source: https://github.com/mwydmuch/ViZDoom

MDP Variations

Georgia Tech

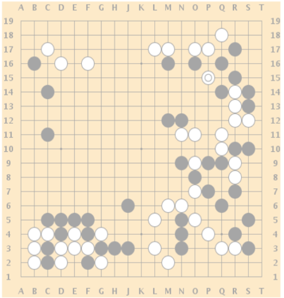
Fully observed MDP

- Agent receives the true state s_t at time t
- Example: Chess, Go

Partially observed MDP

- Agent perceives its own partial observation o_t of the state s_t at time t, using past

We will assume **fully observed MDPs** for this lecture



Source: https://github.com/mwydmuch/ViZDoom

MDP Variations

- In **Reinforcement Learning**, we assume an underlying **MDP** with unknown:
  - Transition probability distribution $\mathbb{T}$
  - Reward distribution $\mathcal{R}$

MDP
$(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$

MDPs in the context of RL

Georgia Tech

- In **Reinforcement Learning**, we assume an underlying **MDP** with unknown:
  - Transition probability distribution $\mathbb{T}$
  - Reward distribution $\mathcal{R}$

MDP
$(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$

- Evaluative feedback comes into play, trial and error necessary

MDPs in the context of RL

- In **Reinforcement Learning**, we assume an underlying **MDP** with unknown:
  - Transition probability distribution $\mathbb{T}$
  - Reward distribution $\mathcal{R}$

MDP
$(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$

- Evaluative feedback comes into play, trial and error necessary

- For this lecture, assume that we know the true reward and transition distribution and look at algorithms for **solving MDPs** i.e. finding the best policy
  - Rewards known everywhere, no evaluative feedback
  - Know how the world works i.e. all transitions

MDPs in the context of RL

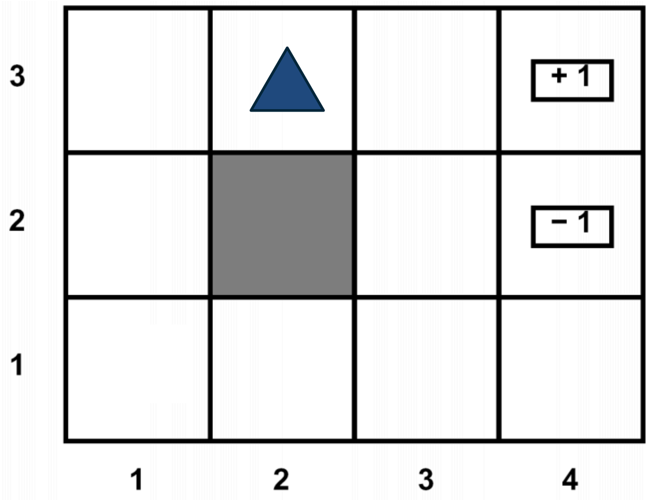


Figure credits: Pieter Abbeel

A Grid World MDP

- Agent lives in a 2D grid environment

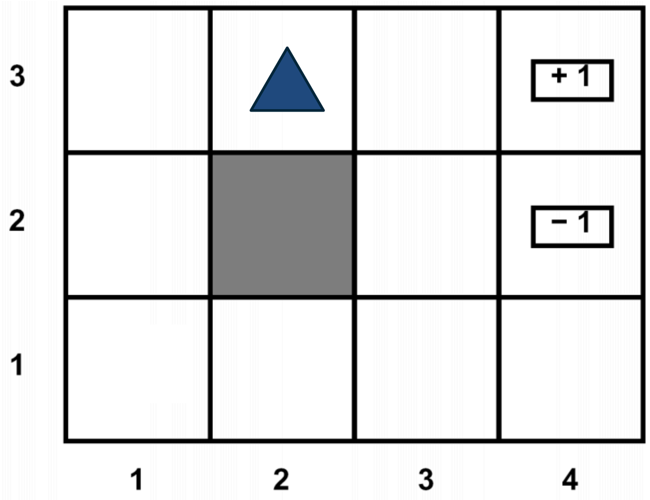


Figure credits: Pieter Abbeel

- Agent lives in a 2D grid environment

- State: Agent's 2D coordinates
- Actions: N, E, S, W
- Rewards: +1/-1 at absorbing states

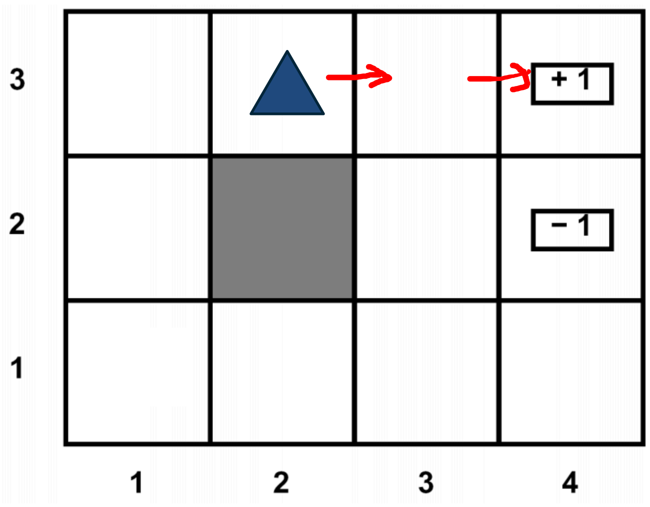
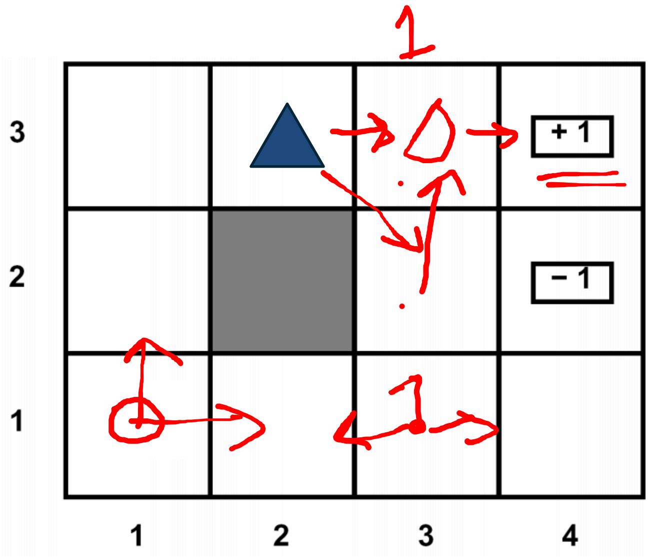


Figure credits: Pieter Abbeel

- Agent lives in a 2D grid environment

- State: Agent's 2D coordinates
- Actions: N, E, S, W
- Rewards: +1/-1 at absorbing states

- Walls block agent's path
- Actions to not always go as planned
  - 20% chance that agent drifts one cell left or right of direction of motion (except when blocked by wall).



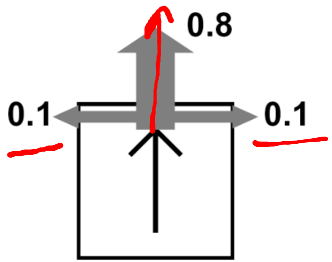


Figure credits: Pieter Abbeel

A Grid World MDP

- Solving MDPs by finding the **best/optimal policy**

- Solving MDPs by finding the **best/optimal policy**
- Formally, a **policy** is a mapping from states to actions

e.g.

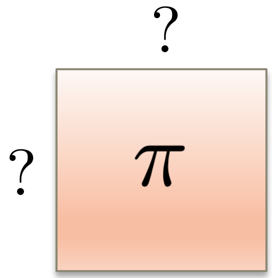
State	Action
A →	2
B →	1

Solving MDPs: Optimal policy

$n = |\mathcal{S}|$

$m = |\mathcal{A}|$

- Solving MDPs by finding the **best/optimal policy**
- Formally, a **policy** is a mapping from states to actions
  - Deterministic $\pi(s) = a$

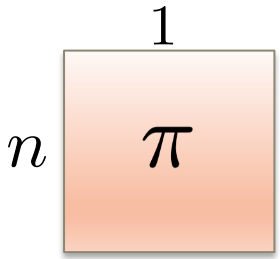


Solving MDPs: Optimal policy

$n = |\mathcal{S}|$

$m = |\mathcal{A}|$

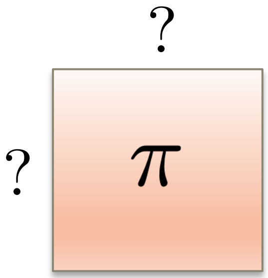
- Solving MDPs by finding the **best/optimal policy**
- Formally, a **policy** is a mapping from states to actions
  - Deterministic $\pi(s) = a$



Solving MDPs: Optimal policy

$$n = |\mathcal{S}|$$
$$m = |\mathcal{A}|$$

- Solving MDPs by finding the **best/optimal policy**
- Formally, a **policy** is a mapping from states to actions
  - Deterministic $\pi(s) = a$
  - Stochastic $\pi(a|s) = \mathbb{P}(A_t = a | S_t = s)$



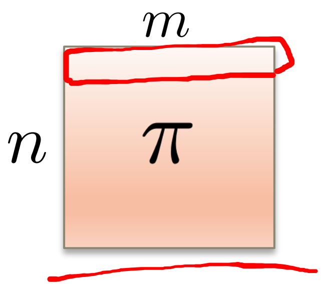
Solving MDPs: Optimal policy

Solving MDPs: Optimal policy

$n = |\mathcal{S}|$

$m = |\mathcal{A}|$

- Solving MDPs by finding the **best/optimal policy**
- Formally, a **policy** is a mapping from states to actions
  - Deterministic $\pi(s) = a$
  - Stochastic $\pi(a|s) = \mathbb{P}(A_t = a | S_t = s)$



- Solving MDPs by finding the **best/optimal policy**

- Formally, a **policy** is a mapping from states to actions
  - Deterministic $\pi(s) = a$
  - Stochastic $\pi(a|s) = \mathbb{P}(A_t = a|S_t = s)$

- What is a good policy?
  - Maximize **current reward**? Sum of all **future rewards**?
  - **Discounted sum of future rewards**!
    - Discount factor: γ

Solving MDPs: Optimal policy

Today, we saw

- **MDPs**: Theoretical framework underlying RL, solving MDPs

- **Policy**: How an agents acts at states (to be continued in next lecture)

Next Lecture:

- **Value function (Utility)**: How good is a particular state or state-action pair?

- **Algorithms** for solving MDPs (Value Iteration)

- Departure from known rewards and transitions: **Reinforcement Learning**