CS 4803 / 7643: Deep Learning

Topic:

- Reinforcement Learning (RL)
  - Overview
  - Markov Decision Processes

Viraj Prabhu

Georgia Tech

Topics we'll cover

- Overview of RL
  - RL vs other forms of learning
  - RL "API"
  - Applications
- Framework: Markov Decision Processes (MDP's)
  - Definitions and notations
  - Policies and Value Functions
  - Solving MDP's
    - Value Iteration
    - Policy Iteration
- Reinforcement learning
  - Value-based RL (Q-learning, Deep-Q Learning)
  - Policy-based RL (Policy gradients)

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

Topics we'll cover

- Overview of RL
  - RL vs other forms of learning
  - RL "API"
  - Applications
- Framework: Markov Decision Processes (MDP's)
  - Definitions and notations
  - Policies and Value Functions
  - Solving MDP's
    - Value Iteration
    - Policy Iteration
- Reinforcement learning
  - Value-based RL (Q-learning, Deep-Q Learning)
  - Policy-based RL (Policy gradients)

This lecture:
– Focus on MDP's
– No learning (deep or otherwise)

Topics we'll cover

- Overview of RL
  - RL vs other forms of learning
  - RL "API"
  - Applications
- Framework: Markov Decision Processes (MDP's)
  - Definitions and notations
  - Policies and Value Functions
  - Solving MDP's
    - Value Iteration
    - Policy Iteration
- Reinforcement learning
  - Value-based RL (Q-learning, Deep-Q Learning)
  - Policy-based RL (Policy gradients)

This lecture:
– Focus on MDP's
– No learning (deep or otherwise)

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Cat

Classification

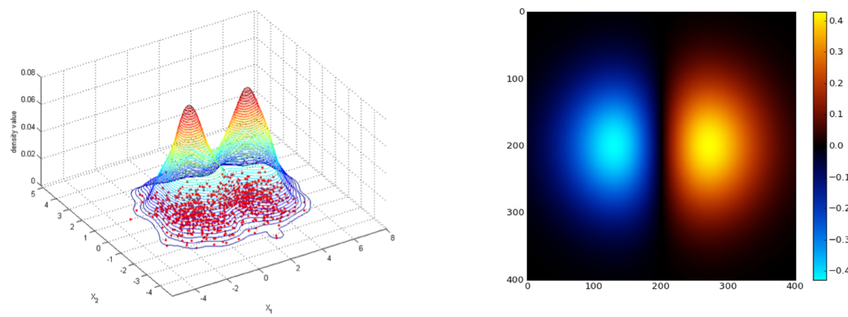
Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

1-d density estimation



2-d density estimation

Types of Learning

- Supervised learning
  - Learning from a “teacher”
  - Training data includes desired outputs
- Unsupervised learning
  - Discover structure in data
  - Training data does not include desired outputs
- Reinforcement learning
  - Learning to act under evaluative feedback (rewards)

What is Reinforcement Learning?

- Learning to make good sequences of decisions

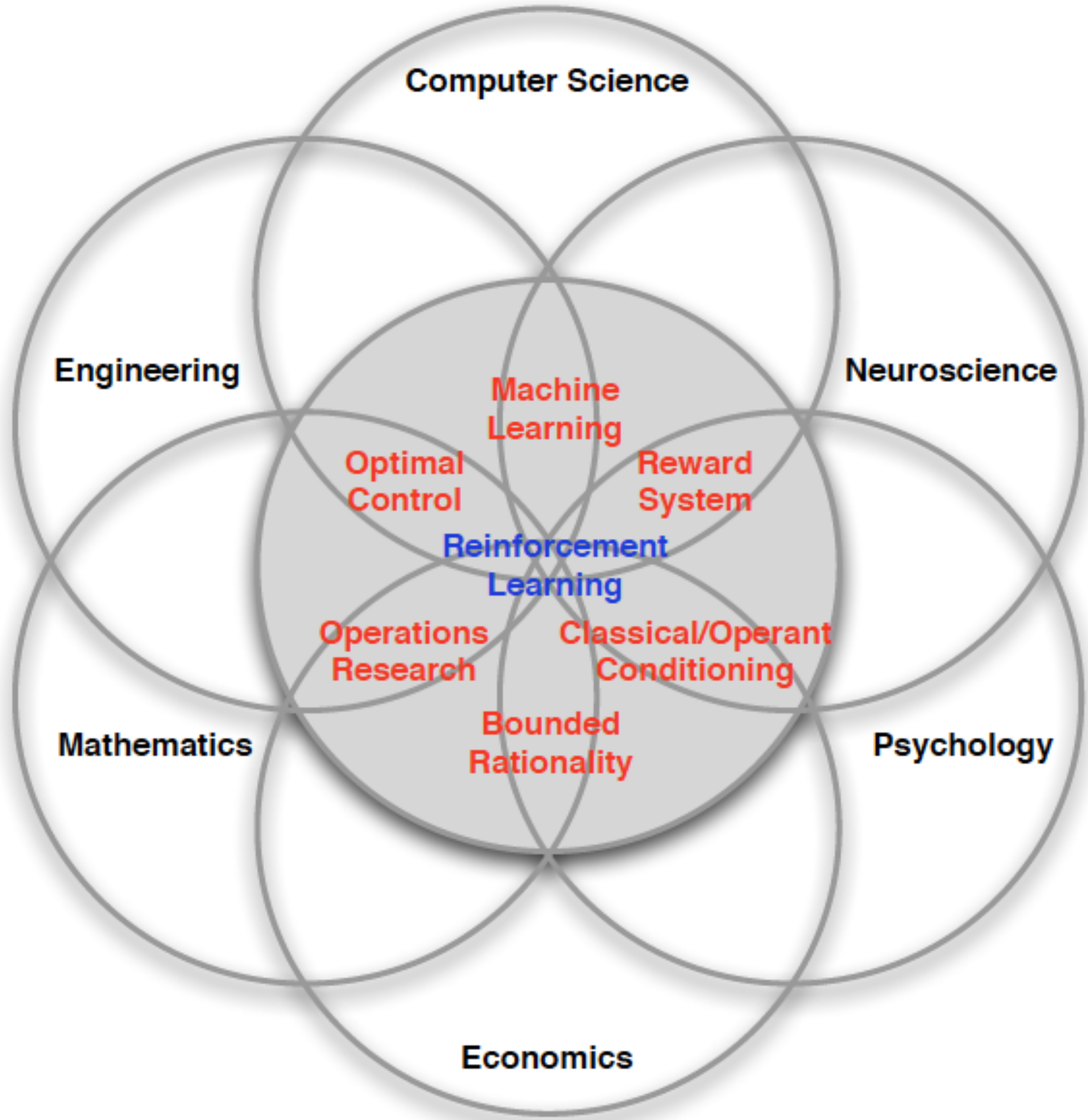
What is Reinforcement Learning?

- Learning to make good sequences of decisions
- Agent-oriented learning—learning by interacting with an environment to achieve a goal
  - more realistic and ambitious than other kinds of machine learning

What is Reinforcement Learning?

- Learning to make good sequences of decisions
- Agent-oriented learning—learning by interacting with an environment to achieve a goal
    - more realistic and ambitious than other kinds of machine learning
- Learning by trial and error, with only delayed evaluative feedback (reward)
    - the kind of machine learning most like natural learning
    - learning that can tell for itself when it is right or wrong

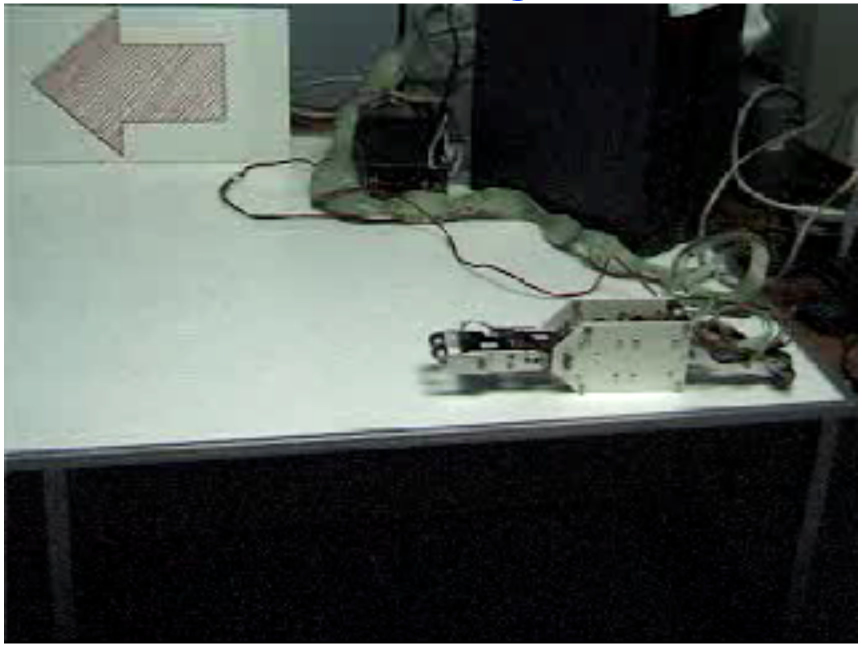
Slide Credit: Rich Sutton, Emma Brunskill



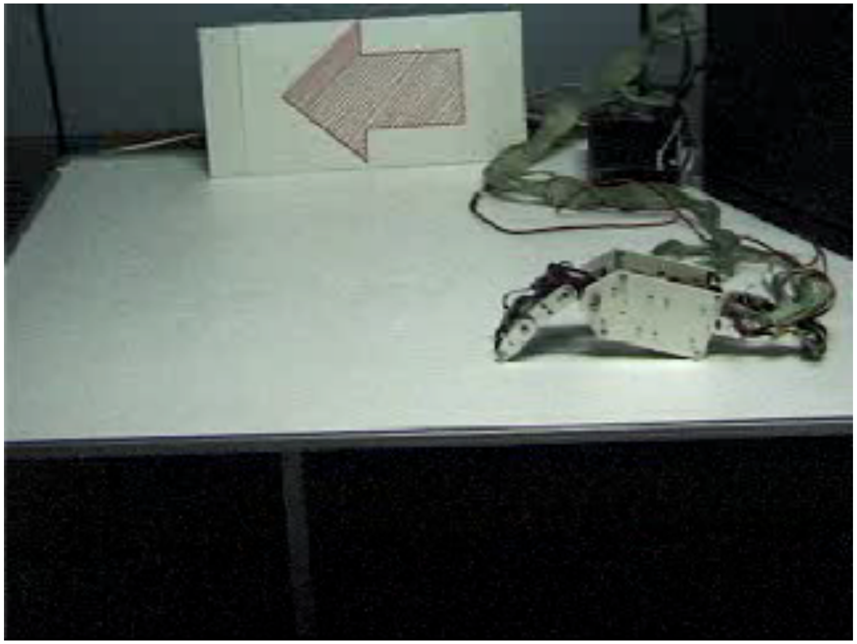
Computer Science
Engineering
Neuroscience
Machine Learning
Optimal Control
Reward System
Reinforcement Learning
Operations Research
Classical/Operant Conditioning
Bounded Rationality
Mathematics
Psychology
Economics

Silver 2015

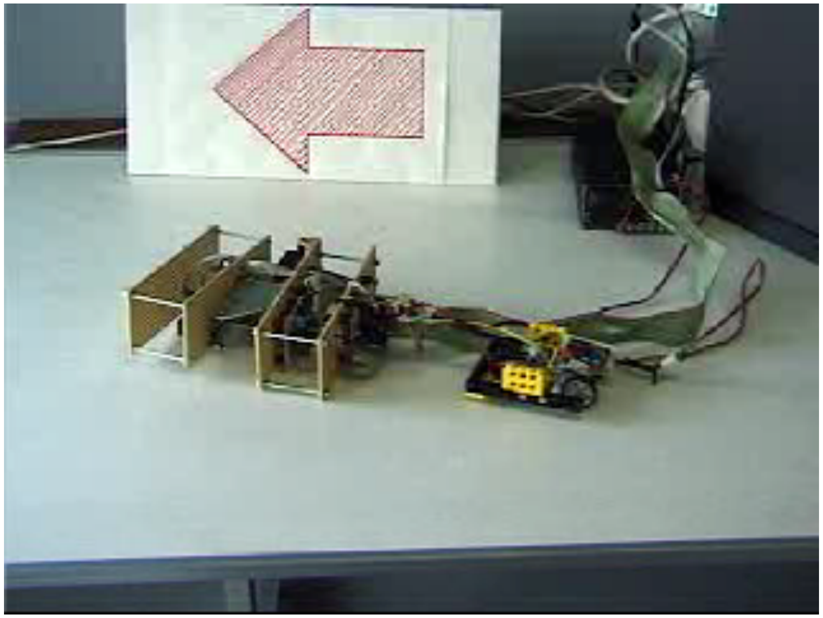
Example: Hajime Kimura's RL Robots



Before

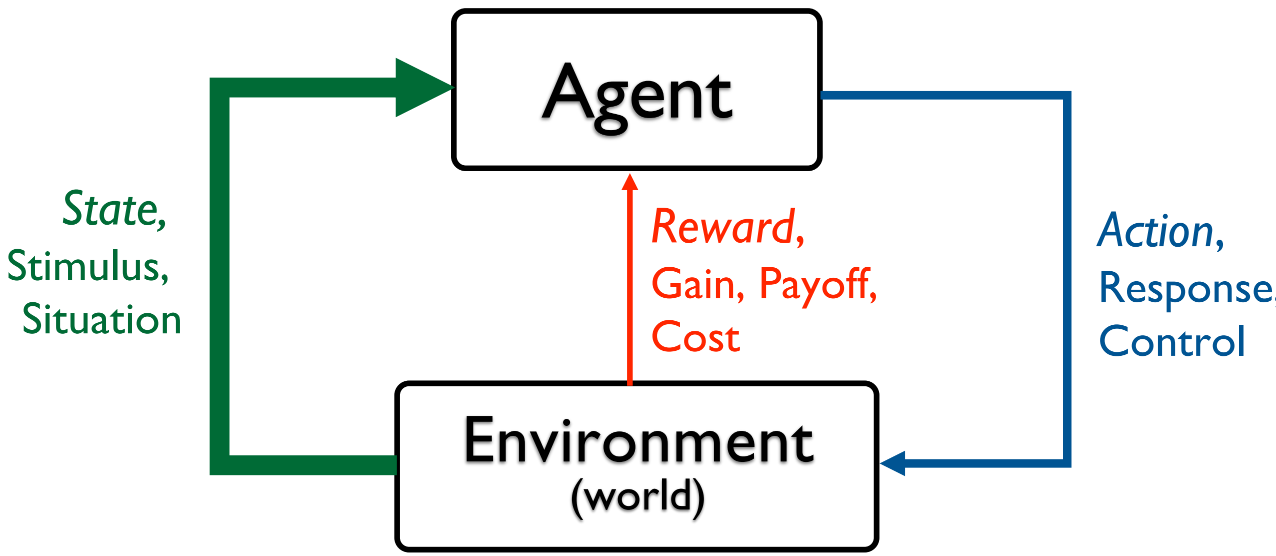


After



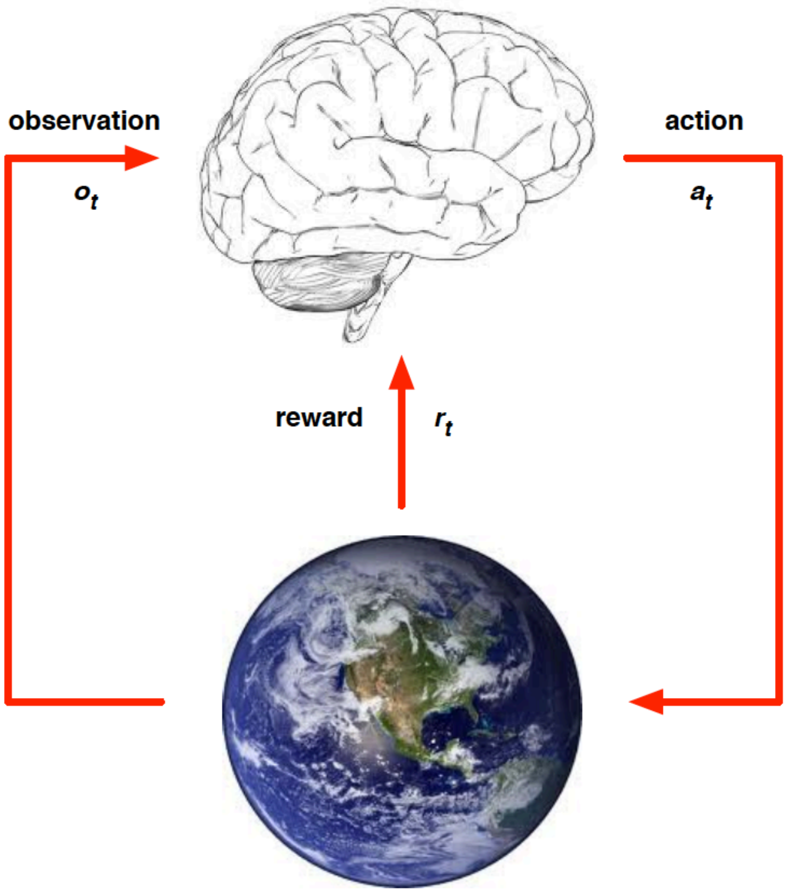
New Robot, Same algorithm

RL API



- Environment may be unknown, nonlinear, stochastic and complex
- Agent learns a policy mapping states to actions
  - Seeking to maximize its cumulative reward in the long run

RL API



- 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}

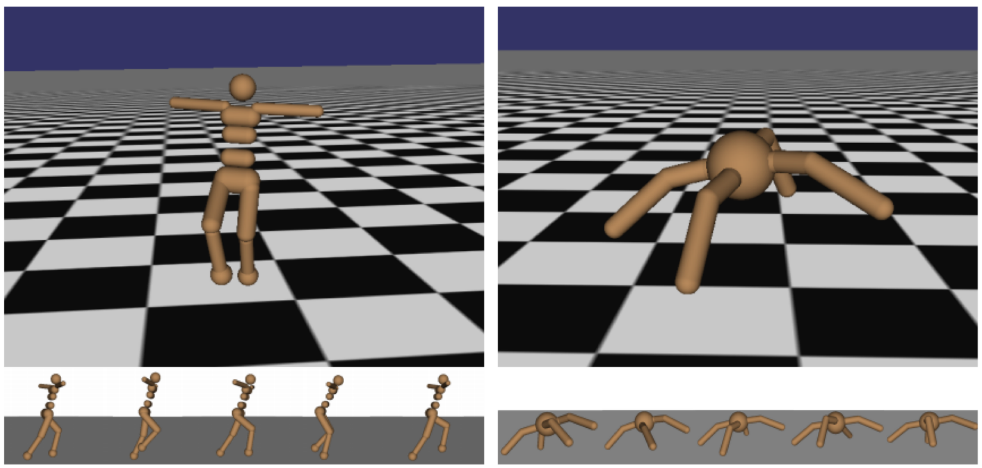
Slide Credit: David Silver

Signature challenges of RL

- Evaluative feedback (reward)
- Sequentiality, delayed consequences
- Need for trial and error, to explore as well as exploit
- Non-stationarity
- The fleeting nature of time and online data

Slide Credit: Rich Sutton

Robot Locomotion



Objective: Make the robot move forward

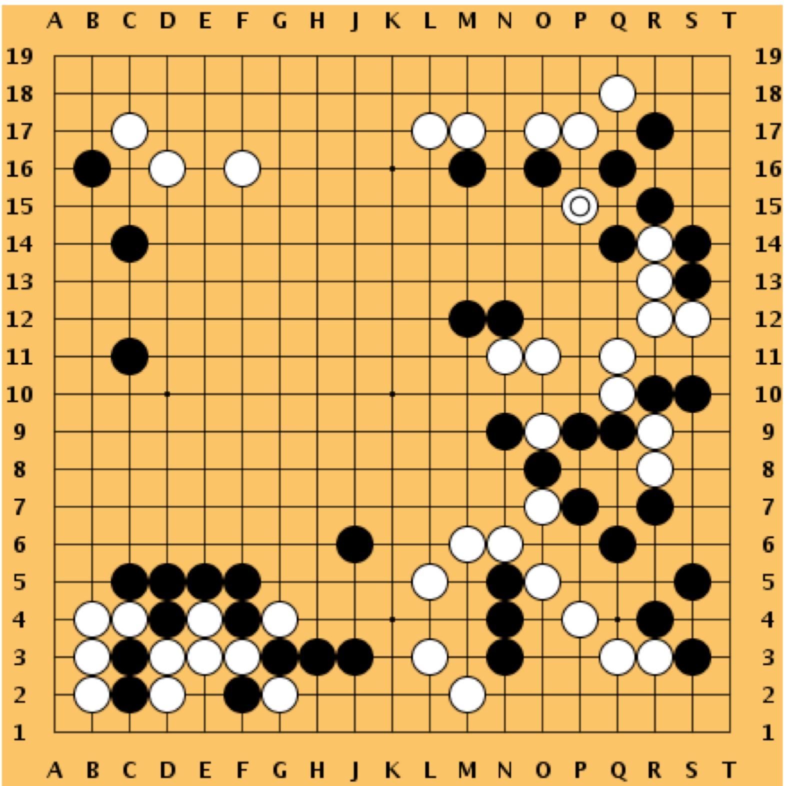
State: Angle and position of the joints
Action: Torques applied on joints
Reward: 1 at each time step upright + forward movement

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

Go



Objective: Win the game!

State: Position of all pieces
Action: Where to put the next piece down
Reward: 1 if win at the end of the game, 0 otherwise

Topics we'll cover

- Overview of RL
  - RL vs other forms of learning
  - RL "API"
  - Applications
- Framework: Markov Decision Processes (MDP's)
  - Definitions and notations
  - Policies and Value Functions
  - Solving MDP's
    - Value Iteration
    - Policy Iteration
- Reinforcement learning
  - Value-based RL (Q-learning, Deep-Q Learning)
  - Policy-based RL (Policy gradients)

This lecture:
– Focus on MDP's
– No learning (deep or otherwise)

Markov Decision Process (MDP)

Markov Decision Process (MDP)

- RL operates within a framework called a Markov Decision Process
- MDP's: General formulation for decision making under uncertainty

Markov Decision Process (MDP)

- RL operates within a framework called a Markov Decision Process
- MDP's: General formulation for decision making under uncertainty

Defined by: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$

Markov Decision Process (MDP)

- RL operates within a framework called a Markov Decision Process
- MDP's: General formulation for decision making under uncertainty

Defined by: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$

$\mathcal{S}$: set of possible states [start state = $s_{0,}$ optional terminal / absorbing state]

Markov Decision Process (MDP)

- RL operates within a framework called a Markov Decision Process
- MDP’s: General formulation for decision making under uncertainty

Defined by: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$

$\mathcal{S}$: set of possible states [start state = s_0, optional terminal / absorbing state]
$\mathcal{A}$: set of possible actions

Markov Decision Process (MDP)

- RL operates within a framework called a Markov Decision Process
- MDP's: General formulation for decision making under uncertainty

Defined by: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$

$\mathcal{S}$: set of possible states [start state = $s_{0,}$ optional terminal / absorbing state]
$\mathcal{A}$: set of possible actions
$\mathcal{R}(s, a, s')$: distribution of reward given (state, action, next state) tuple

Markov Decision Process (MDP)

- RL operates within a framework called a Markov Decision Process
- MDP's: General formulation for decision making under uncertainty

Defined by: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$

$\mathcal{S}$: set of possible states [start state = s_0, optional terminal / absorbing state]
$\mathcal{A}$: set of possible actions
$\mathcal{R}(s, a, s')$: distribution of reward given (state, action, next state) tuple
$\mathbb{T}(s, a, s')$: transition probability distribution, also written as $p(s'|s, a)$

Markov Decision Process (MDP)

- RL operates within a framework called a Markov Decision Process
- MDP's: General formulation for decision making under uncertainty

Defined by: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$

$\mathcal{S}$: set of possible states [start state = s_0, optional terminal / absorbing state]
$\mathcal{A}$: set of possible actions
$\mathcal{R}(s, a, s')$: distribution of reward given (state, action, next state) tuple
$\mathbb{T}(s, a, s')$: transition probability distribution, also written as $p(s'|s, a)$
γ : discount factor

Markov Decision Process (MDP)

- RL operates within a framework called a Markov Decision Process
- MDP's: General formulation for decision making under uncertainty

Defined by: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$

$\mathcal{S}$: set of possible states [start state = s_0, optional terminal / absorbing state]
$\mathcal{A}$: set of possible actions
$\mathcal{R}(s, a, s')$: distribution of reward given (state, action, next state) tuple
$\mathbb{T}(s, a, s')$: transition probability distribution, also written as $p(s'|s, a)$
γ : discount factor

- Life is trajectory: $\ldots \underline{s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}}, r_{t+2}, s_{t+2}, \ldots$

Markov Decision Process (MDP)

- RL operates within a framework called a Markov Decision Process
- MDP's: General formulation for decision making under uncertainty

Defined by: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$

$\mathcal{S}$: set of possible states [start state = s_0, optional terminal / absorbing state]
$\mathcal{A}$: set of possible actions
$\mathcal{R}(s, a, s')$: distribution of reward given (state, action, next state) tuple
$\mathbb{T}(s, a, s')$: transition probability distribution, also written as $p(s'|s, a)$
γ : discount factor

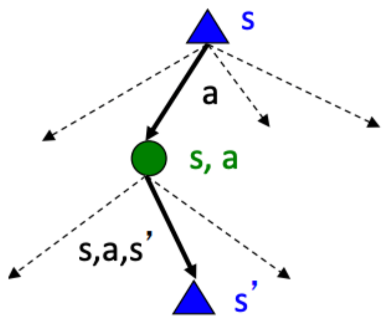
- Life is trajectory: $\ldots \underline{s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}}, r_{t+2}, s_{t+2}, \ldots$

- **Markov property**: Current state completely characterizes state of the world
- **Assumption**: Most recent observation is sufficient statistic of history

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

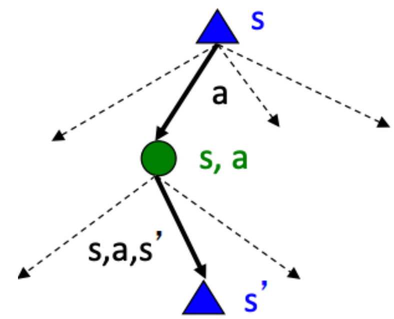
Markov Decision Process (MDP)

- MDP state projects a search tree



Markov Decision Process (MDP)

- MDP state projects a search tree



- **Observability**:
  - **Full:** In a fully observable MDP, $o_t = s_t$
    - Example: Chess
  - **Partial:** In a partially observable MDP, agent *constructs* its own state, using history, of beliefs of world state, or an RNN, …
    - Example: Poker

Markov Decision Process (MDP)

- In RL, we don't have access to $\mathbb{T}$ or $\mathcal{R}$ (i.e. the environment)
    - Need to *actually try* actions and states out to learn
    - Sometimes, need to model the environment

Markov Decision Process (MDP)

- In RL, we don't have access to $\mathbb{T}$ or $\mathcal{R}$ (i.e. the environment)
  - Need to *actually try* actions and states out to learn
  - Sometimes, need to model the environment

- For today, let's assume we *do* have access to how the world works

Markov Decision Process (MDP)

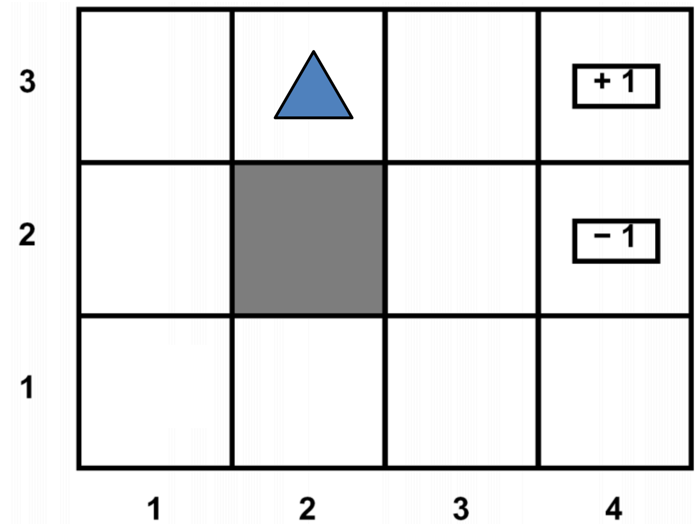
- In RL, we don't have access to $\mathbb{T}$ or $\mathcal{R}$ (i.e. the environment)
  - Need to *actually try* actions and states out to learn
  - Sometimes, need to model the environment

- For today, let's assume we *do* have access to how the world works

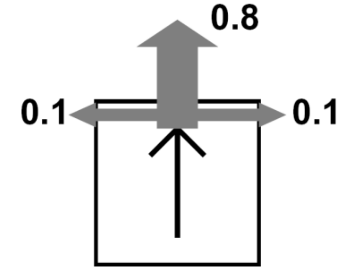
- And that our goal is to find an optimal behavior strategy for an agent

Canonical Example: Grid World

- Agent lives in a grid
- Walls block the agent's path
- Actions do not always go as planned
  - 80% of the time, action North takes the agent North (if there is no wall)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall, the agent stays put



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



Solving MDP's

Solving MDP's

- Policy
  - How should an agent behave?

Solving MDP's

- Policy
  - How should an agent behave?

- Value function (Utility)
  - How good is each state and/or state-action pair?

Policy

- A policy is how the agent acts

e.g.

State	Action
A	2
B	1

Policy

- A policy is how the agent acts

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

e.g.

State	Action
A $\longrightarrow$	2
B $\longrightarrow$	1

The optimal policy **π***

What's a good policy?

The optimal policy **π***

What's a good policy?

Maximizes current reward? Sum of all future reward?

The optimal policy **π***

What's a good policy?

Maximizes current reward? Sum of all future reward?

Discounted future rewards!

The optimal policy **π***

What's a good policy?

Maximizes current reward? Sum of all future reward?

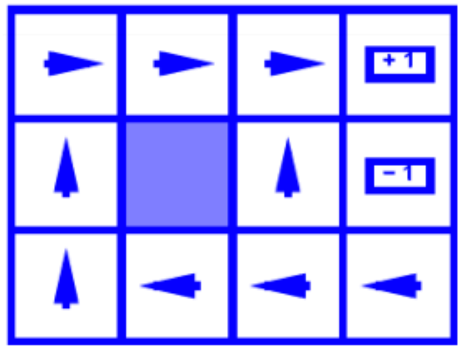
Discounted future rewards!

Formally: $$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi\right]$$

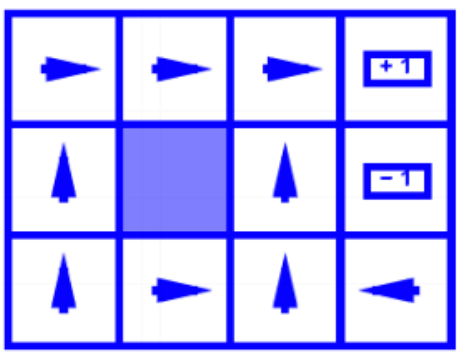
(Typically for a fixed horizon T)

with $$\boldsymbol{s}_0 \sim p\left(\boldsymbol{s}_0\right), a_t \sim \pi\left(\cdot|\boldsymbol{s}_t\right), \boldsymbol{s}_{t+1} \sim p\left(\cdot|\boldsymbol{s}_t, a_t\right)$$

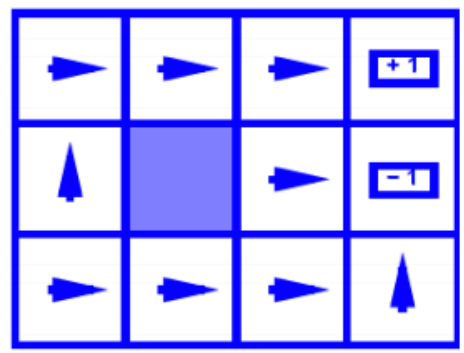
The optimal policy **π***



R(s) = -0.03



R(s) = -
0.4



R(s) = -
2.0

Discounting

- Prefer rewards now to rewards later
- Helps with convergence
- Alternate interpretation: Contending with possibility of “death”

Discounting

- Prefer rewards now to rewards later
- Helps with convergence
- Alternate interpretation: Contending with possibility of “death”

10				**1**

- Given an MDP:
  - Actions: East, West, Exit (at first and last position)
  - Deterministic transitions
- What is the optimal policy for:
  - $\gamma = 1$
  - $\gamma = 0.1$

Value Function

Value Function

- A value function is a prediction of future reward

Value Function

- A value function is a prediction of future reward
- State Value Function or simply **Value Function**
  - How good is a state?
  - Am I screwed? Am I winning this game?

Value Function

- A value function is a prediction of future reward

- State Value Function or simply **Value Function**
  - How good is a state?
  - Am I screwed? Am I winning this game?

- Action-Value Function or **Q-function**
  - How good is a state action-pair?
  - Should I do this now?

Value Function

Following policy π that produces sample trajectories s_0, a_0, r_0, s_1, a_1, …

Value Function

Following policy π that produces sample trajectories s_0, a_0, r_0, s_1, a_1, …

How good is a state?

The **value function** at state s, is the expected cumulative reward from state s (and following the policy thereafter):

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, \pi\right]$$

Value Function

Following policy π that produces sample trajectories $s_0, a_0, r_0, s_1, a_1, \ldots$

How good is a state?
The **value function** at state s, is the expected cumulative reward from state s (and following the policy thereafter):

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, \pi\right]$$

How good is a state-action pair?
The **Q-value function** at state s and action a, is the expected cumulative reward from taking action a in state s (and following the policy thereafter):

$$Q^{\pi}(s, a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

Optimal Quantities

Given *optimal* policy π^* that produces sample trajectories s_0, a_0, r_0, s_1, a_1, …

Optimal Quantities

Given *optimal* policy π^* that produces sample trajectories s_0, a_0, r_0, s_1, a_1, …

How good is a state?
The **optimal value function** at state s, and acting optimally thereafter

$$V^*(s) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, \pi^*\right]$$

Optimal Quantities

Given *optimal* policy π^* that produces sample trajectories s_0, a_0, r_0, s_1, a_1, …

How good is a state?
The **optimal value function** at state s, and acting optimally thereafter

$$V^*(s) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, \pi^*\right]$$

How good is a state-action pair?
The **optimal Q-value function** at state s and action a, is the expected cumulative reward from taking action a in state s and acting optimally thereafter

$$Q^*(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi^*\right]$$

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

Recursive definition of value

Recursive definition of value

- Extracting optimal value / policy from Q-values:

$$V^*(s) = \max_a Q^*(s, a) \qquad \pi^*(s) = \arg\max_a Q^*(s, a)$$

Recursive definition of value

- Extracting optimal value / policy from Q-values:

$$V^*(s) = \max_a Q^*(s,a) \qquad \pi^*(s) = \arg\max_a Q^*(s,a)$$

- Bellman Equations:

$$V^*(s) = \max_a \sum_{s'} p\left(s'|s,a\right)\left[r(s,a) + \gamma V^*\left(s'\right)\right]$$

Recursive definition of value

- Extracting optimal value / policy from Q-values:

$$V^*(s) = \max_a Q^*(s,a) \qquad \pi^*(s) = \arg\max_a Q^*(s,a)$$

- Bellman Equations:

$$V^*(s) = \max_a \sum_{s'} p(s'|s,a)\left[r(s,a) + \gamma V^*(s')\right]$$

$$Q^*(s,a) = \sum_{s'} p(s'|s,a)\left[r(s,a) + \gamma V^*(s')\right]$$

Recursive definition of value

- Extracting optimal value / policy from Q-values:

$$V^*(s) = \max_a Q^*(s,a) \qquad \pi^*(s) = \arg\max_a Q^*(s,a)$$

- Bellman Equations:

$$V^*(s) = \max_a \sum_{s'} p\left(s'|s,a\right)\left[r(s,a) + \gamma V^*\left(s'\right)\right]$$

$$Q^*(s,a) = \sum_{s'} p\left(s'|s,a\right)\left[r(s,a) + \gamma V^*\left(s'\right)\right]$$

- Characterize optimal values in a way we'll use over and over

Slide credit: Byron Boots, CS 7641

Value Iteration (VI)

- Bellman equations characterize optimal values, VI is a fixed-point DP solution method to *compute* it

Value Iteration (VI)

- Bellman equations characterize optimal values, VI is a fixed-point DP solution method to *compute* it

- Algorithm
  – Initialize values of all states $V_0(s) = 0$
  – Update: $V^{i+1}(s) \leftarrow \max_a \sum_{s'} p\left(s' | s, a\right) \left[r(s, a) + \gamma V^{i}\left(s'\right)\right]$
  – Repeat until convergence (to V^*)

Value Iteration (VI)

- Bellman equations characterize optimal values, VI is a fixed-point DP solution method to *compute* it

- Algorithm
  - Initialize values of all states $V_0(s) = 0$
  - Update: $V^{i+1}(s) \leftarrow \max_a \sum_{s'} p(s'|s,a) \left[r(s,a) + \gamma V^i(s') \right]$
  - Repeat until convergence (to V^*)

- Complexity per iteration (DP): $O(|S|^2|A|)$

Slide credit: Byron Boots, CS 7641

Value Iteration (VI)

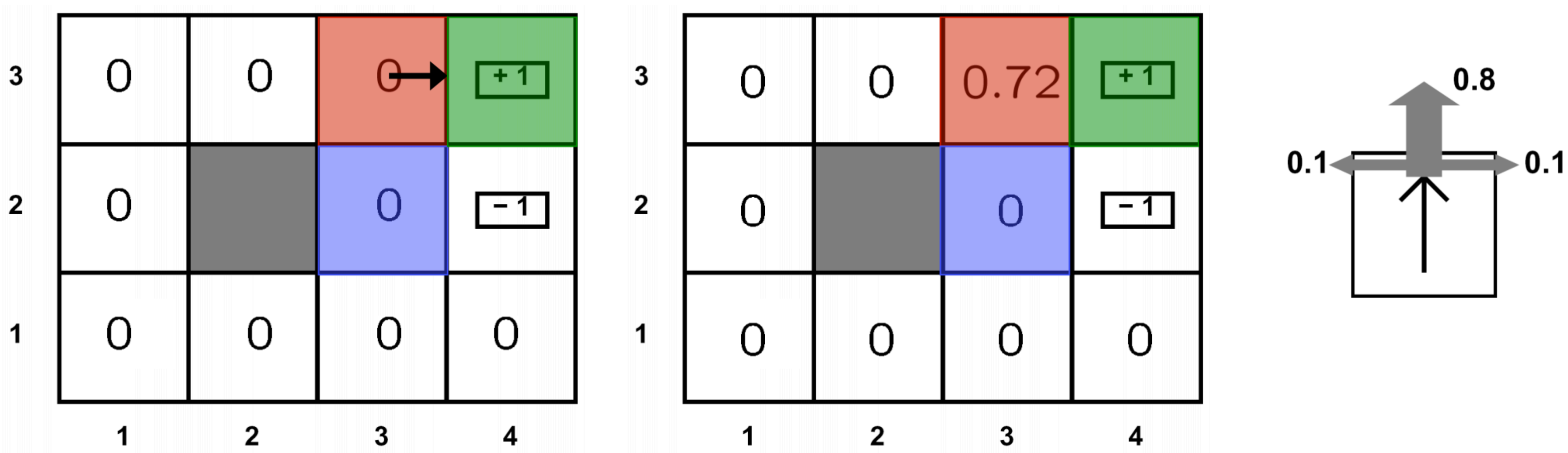
- Bellman equations characterize optimal values, VI is a fixed-point DP solution method to *compute* it

- Algorithm
  - Initialize values of all states $V_0(s) = 0$
  - Update: $V^{i+1}(s) \leftarrow \max_a \sum_{s'} p(s'|s,a)\left[r(s,a) + \gamma V^i(s')\right]$
  - Repeat until convergence (to V^*)

- Complexity per iteration (DP): $O(|S|^2|A|)$

- Convergence
  - Guaranteed for $\gamma < 1$
  - Sketch: Approximations get refined towards optimal values
  - In practice, policy may converge before values do

Value Iteration (VI)



$$V^{i+1}(s) \leftarrow \max_a \sum_{s'} p\left(s'|s,a\right)\left[r(s,a) + \gamma V^i\left(s'\right)\right]$$

$$V^2(\langle 3,3\rangle) = \sum_{s'} P\left(s'|\,\text{right}, \langle 3,3\rangle\right)\left[r(\langle 3,3\rangle) + \gamma V^1\left(s'\right)\right]$$

$$= 0.9[0.8 \cdot 1 + 0.1 \cdot 0 + 0.1 \cdot 0]$$

Demo

- https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

Next class

- Solving MDP's
  - Policy Iteration
- Reinforcement learning
  - Value-based RL
    - Q-learning
    - Deep Q Learning

Slide Credit: David Silver