Reinforcement Learning

Yishay Mansour
Tel-Aviv University

Reinforcement Learning: Course Information

- Classes: Wednesday
  - Lecture 10-13
    - Yishay Mansour
  - Recitations:14-15/15-16
    - Lee Cohen
- Course Web site:

  rl-tau-2019.wikidot.com
- Resources:
- Markov Decision Processes
  - Puterman
- Reinforcement Learning
  - Sutton and Barto
- Neuro-dynamic Programming
  - Bertsekas and Tsitsiklis

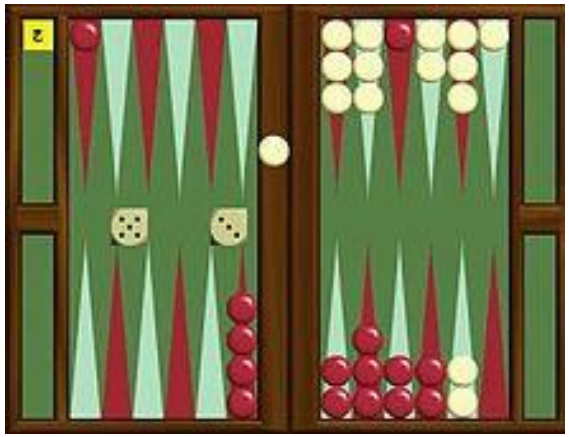
Reinforcement Learning: Course requirements

- Homework:
  - Every two week: Theory and Programming
- Project:
  - Near the end of the term
  - Deep RL/Atari
- Final Exam

- Grade:
  - 60% Final exam
    - Have to pass
  - 20% HW
  - 20% project

Playing Board Games

AlphaGo, DeepMind 2015-17



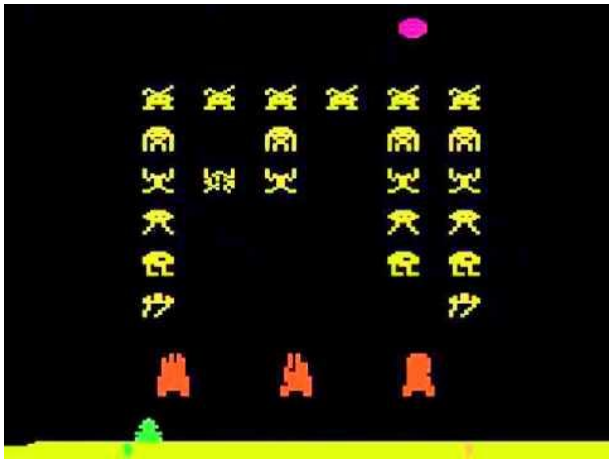
Gerald Tesauro,
TD-gammon 1992

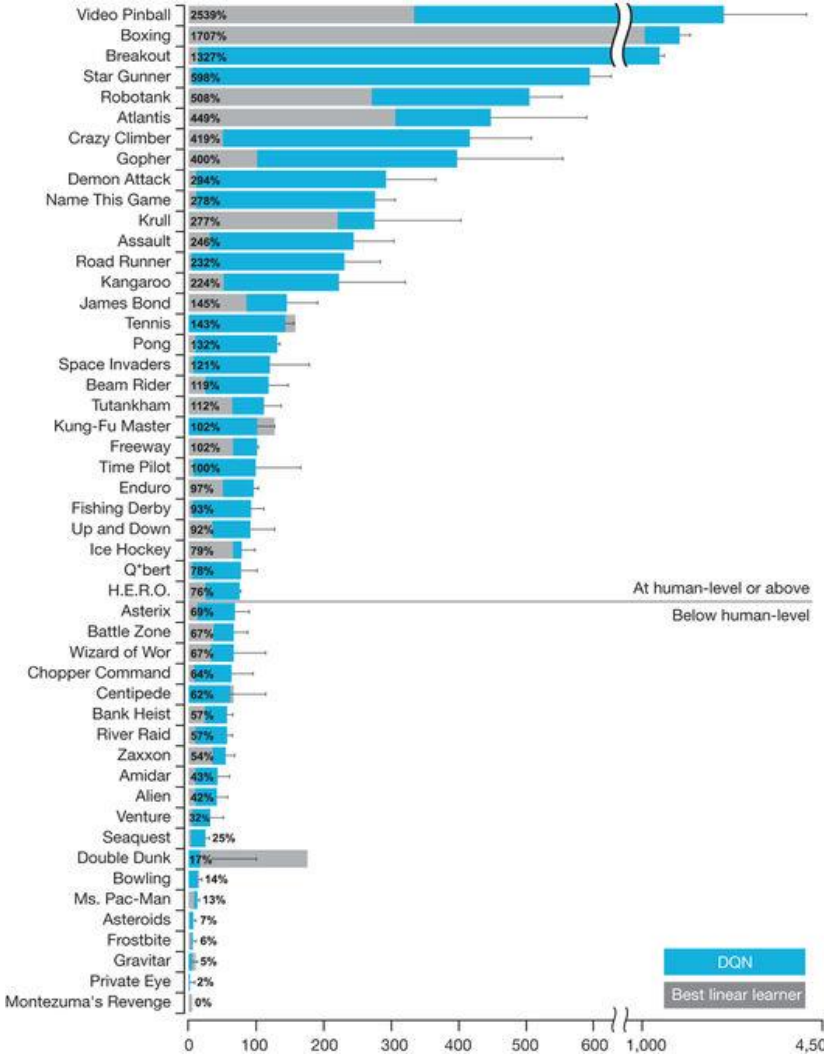
Other notable board games

Deep Blue1996

Arthur Samuel 1962

Playing Atari Games





Controlling Robots

Today Outline: Overview

- Basics
  - Goal of Reinforcement Learning
  - Mathematical Model (MDP)
- Planning
  - Value iteration
  - Policy iteration
- Learning Algorithms
  - Model based
  - Model Free
- Large state space
  - Function approx
  - Policy gradient

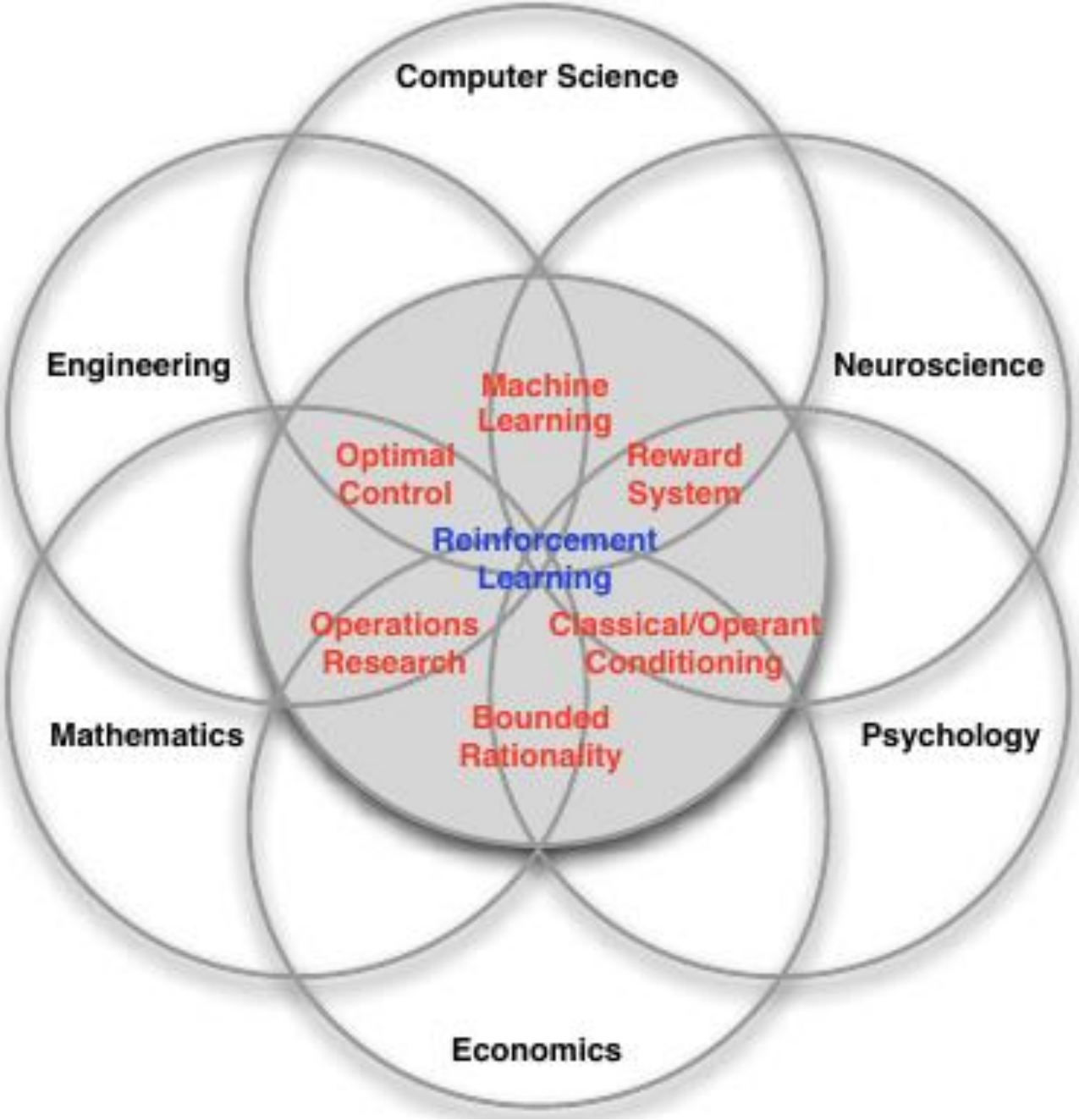
Goal of Reinforcement Learning

Goal oriented learning through interaction

Control of large scale stochastic environments with partial knowledge.

Supervised / Unsupervised Learning

Learn from labeled / unlabeled examples


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

Mathematical Model - Motivation

Model of uncertainty:

Environment, actions, our knowledge.

Focus on decision making.

Maximize long term reward.

Markov Decision Process (MDP)

Contrast with Supervised Learning

The system has a “state”.

The algorithm influences the state distribution.

Inherent Tradeoff: Exploration versus Exploitation.
* There is a cost to discover information !

Mathematical Model - MDP

Markov Decision Processes

S- set of states

A- set of actions

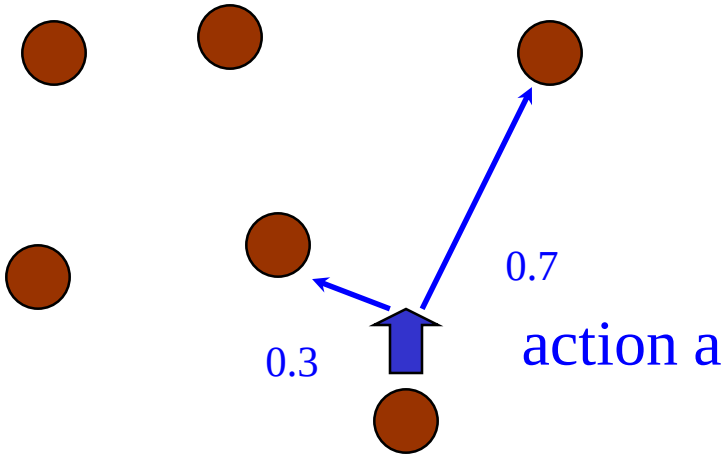
δ - Transition probability

R - Reward function

Similar to DFA!

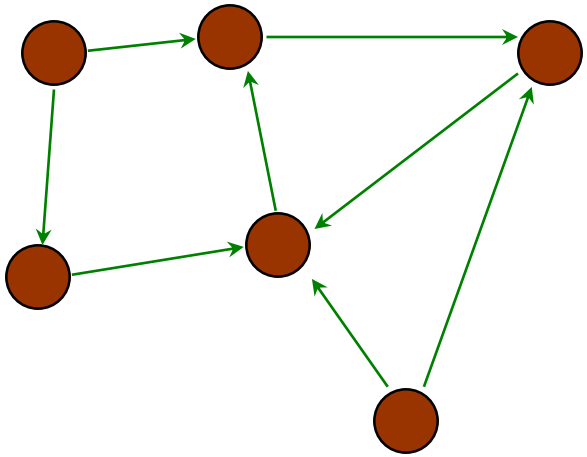
MDP model - states and actions

Environment = states



Actions = transitions $\delta(s, a, s')$

MDP model - rewards



R(s,a) = reward at state ***s***

for doing action ***a***

(a random variable).

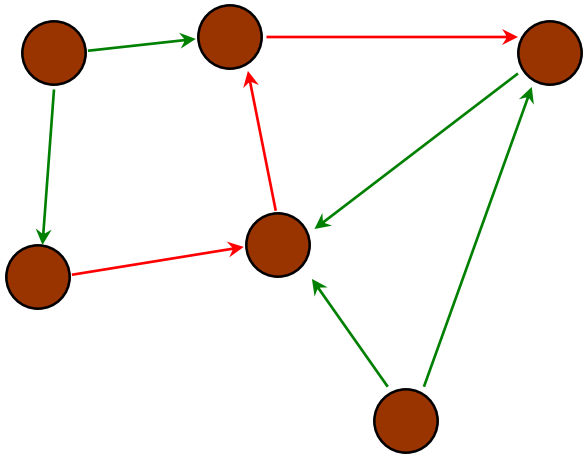
Example:

R(s,a) = -1 with probability 0.5

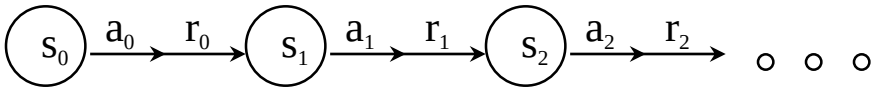
+10 with probability 0.35

+20 with probability 0.15

MDP model - trajectories



trajectory:



MDP - Return function.

Combining all the immediate rewards to a single value.

Modeling issues:

Are early rewards more valuable than later rewards?

Is the system “terminating” or continuous?

Usually the return is linear in the immediate rewards.

MDP model - return functions

Finite Horizon - parameter H

$$return = \sum_{1 \le i \le H} R(s_i, a_i)$$

Infinite Horizon

discounted - parameter $\gamma<1$.

$$return = \sum_{i=0}^{\infty} \gamma^i R(s_i, a_i)$$

undiscounted

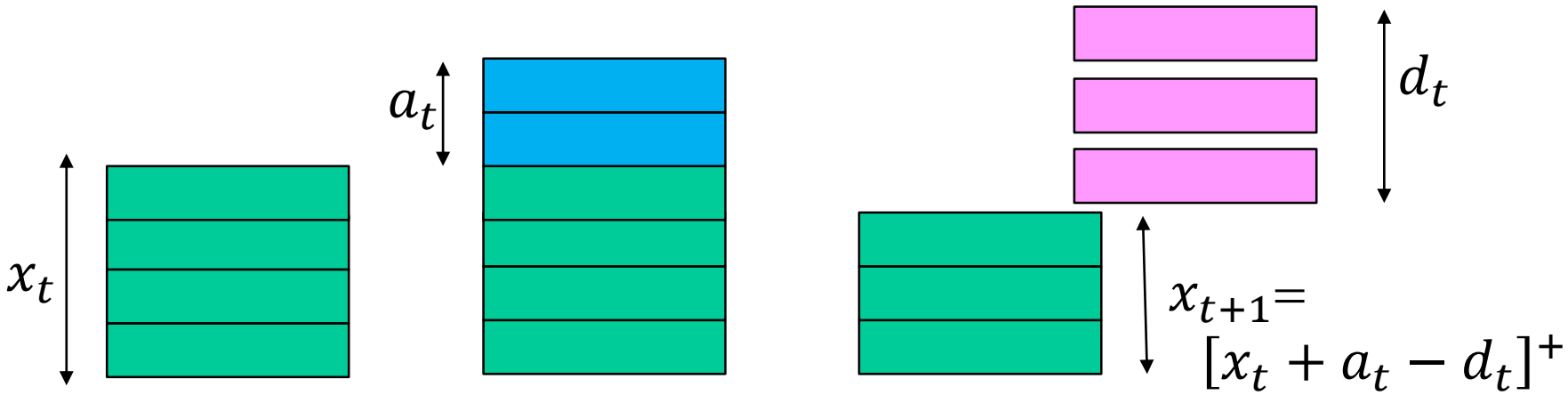
$$\frac{1}{N} \sum_{i=0}^{N-1} R(s_i, a_i) \xrightarrow{N \to \infty} return$$

Terminating MDP

MDP Example: Inventory

P= Price/item,
J(.)=order cost
C(.)=inventory cost
$s_t = \min(x_t + a_t, d_t)$

$$R_t = P\, s_t + J(a_t) - C(x_{t+1})$$



MDP model - action selection

AIM: Maximize the expected return.

This talk: discounted return

Fully Observable - can "see" the "exact" state.

Policy - mapping from states to actions

Optimal policy: optimal from any start state.

THEOREM: There exists a deterministic optimal policy

MDP model - summary

$s \in S$ - set of states, $|S|=n$.

$a \in A$ - set of k actions, $|A|=k$.

$\delta(s_1, a, s_2)$ - transition function.

R(s,a) - immediate reward function.

$\pi : S \to A$ - policy.

$\sum_{i=0}^{\infty} \gamma^i r_i$ - discounted cumulative return.

Contrast with Supervised Learning

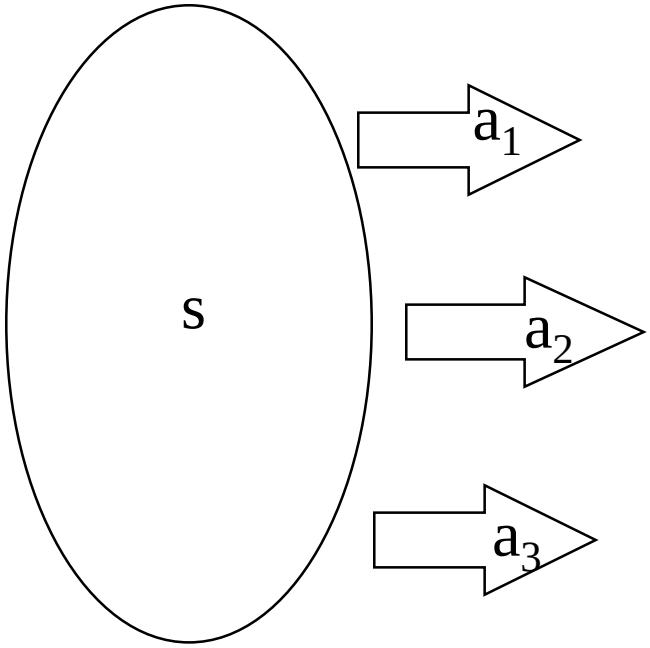
Supervised Learning:
Fixed distribution on examples.

Reinforcement Learning:
The state distribution is policy dependent!!!

A small local change in the policy can make a huge global change in the return.

Simple setting: Multi-armed bandit

Single state.



Goal: Maximize sum of immediate rewards.

Given the model: Greedy action.

Difficulty: unknown rewards.

Today Outline: Overview

- Basics
  - Goal of Reinforcement Learning
  - Mathematical Model (MDP)
- Planning
  - Value iteration
  - Policy iteration
- Learning Algorithms
  - Model based
  - Model Free
- Large state space
  - Function Approx
  - Policy gradient

Planning - Basic Problems.

Given a complete MDP model.

Policy evaluation - Given a policy π, evaluate its return.

Optimal control - Find an **optimal policy π^*** (maximizes the return from any start state).

Planning - Value Functions

$V^{\pi}(s)$ The expected return starting at state s following π.

$Q^{\pi}(s,a)$ The expected return starting at state s with action a and then following π.

$V^*(s)$ and $Q^*(s,a)$ are define using an optimal policy π^*.

$$V^*(s) = max_{\pi} V^{\pi}(s)$$

Planning - Policy Evaluation

Discounted infinite horizon (Bellman Eq.)

$$V^{\pi}(s) = E_{s' \sim (s,\pi(s))} [\, R(s,\pi(s)) + \gamma V^{\pi}(s') \,]$$

Rewrite the expectation

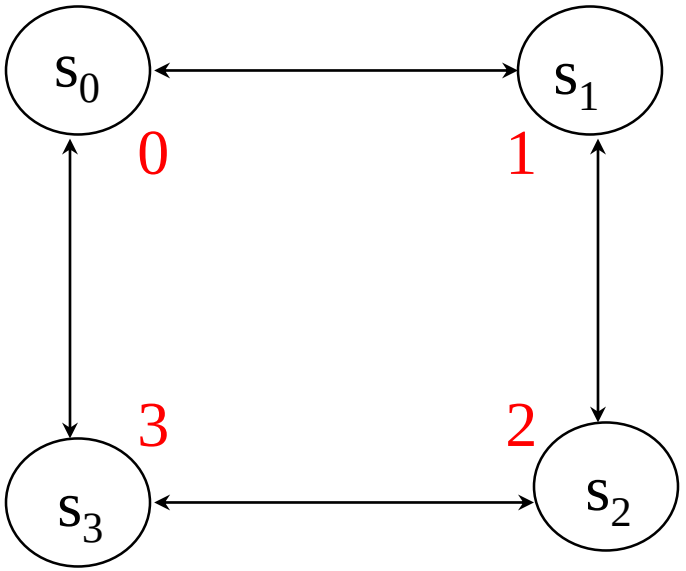
$$V^{\pi}(s) = E[R(s,\pi(s))] + \gamma \sum_{s'} \delta(s,\pi(s),s') V^{\pi}(s')$$

Linear system of equations.

Algorithms -Policy Evaluation
Example

$A=\{+1,-1\}$
$\gamma = 1/2$
$\delta(s_i,a)= s_{i+a}$
π random

$\forall a: R(s_i,a) = i$

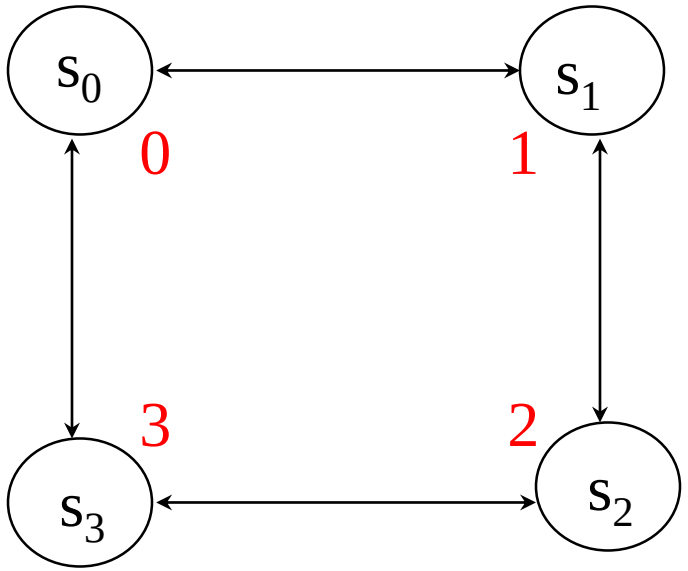


$$V^{\pi}(s_0) = 0 + \gamma [\pi(s_0,+1)V^{\pi}(s_1) + \pi(s_0,-1)\, V^{\pi}(s_3)\ ]$$

Algorithms -Policy Evaluation
Example

$A=\{+1,-1\}$
$\gamma = 1/2$
$\delta(s_i,a)= s_{i+a}$
π random

$\forall a: R(s_i,a) = i$



$V^\pi(s_0) = 5/3$
$V^\pi(s_1) = 7/3$
$V^\pi(s_2) = 11/3$
$V^\pi(s_3) = 13/3$

$V^\pi(s_0) = 0 + (V^\pi(s_1) + V^\pi(s_3))/4$
$V^\pi(s_1) = 1 + (V^\pi(s_0) + V^\pi(s_2))/4$
$V^\pi(s_2) = 2 + (V^\pi(s_1) + V^\pi(s_3))/4$
$V^\pi(s_3) = 3 + (V^\pi(s_0) + V^\pi(s_2))/4$

Algorithms - optimal control

State-Action Value function:

$$Q^{\pi}(s,a) = E\,[\,R(s,a)] + \gamma E_{s' \sim (s,\pi(s))}\,[\,V^{\pi}(s')]$$

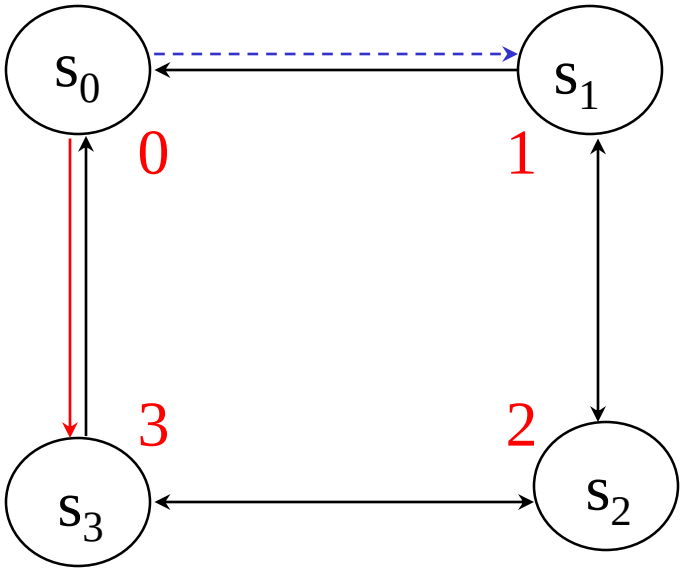
Note $V^{\pi}(s) = Q^{\pi}(s, \pi(s))$

For a deterministic policy π.

Algorithms -Optimal control
Example

$A=\{+1,-1\}$
$\gamma = 1/2$
$\delta(s_i,a)= s_{i+a}$
π random

$R(s_i,a) = i$



$Q^{\pi}(s_0,+1) = 7/6$
$Q^{\pi}(s_0,-1) = 13/6$

$Q^{\pi}(s_0,+1) = 0 + \gamma V^{\pi}(s_1)$

Algorithms - optimal control

CLAIM: A policy π is optimal if and only if at each state *s*:

$$V^{\pi}(s) = MAX_a \{Q^{\pi}(s,a)\} \qquad \textit{(Bellman Eq.)}$$

PROOF: (only if) Assume there is a state *s* and action *a* s.t.,

$$V^{\pi}(s) < Q^{\pi}(s,a).$$

Then the strategy of performing *a* at state *s* (the first time) is better than π.

This is true each time we visit *s*, so the policy that performs action *a* at state *s* is better than π. □

Algorithms -optimal control
Example

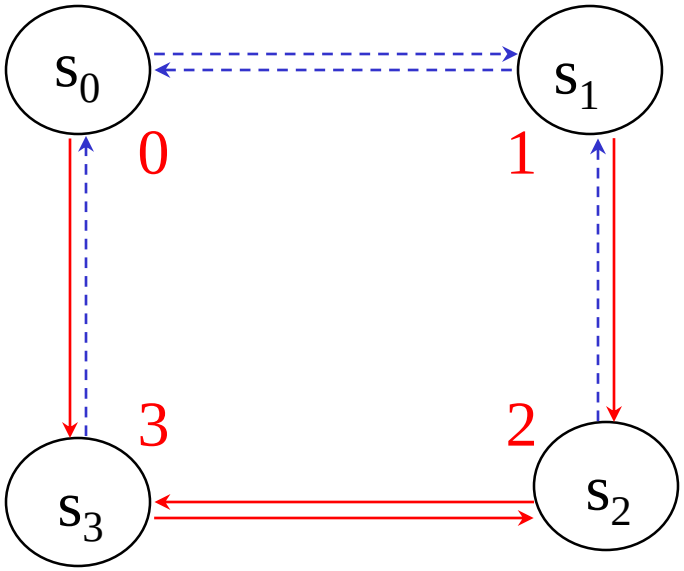
$A=\{+1,-1\}$

$\gamma = 1/2$

$\delta(s_i,a)= s_{i+a}$

π random

$R(s_i,a) = i$



Changing the policy using the state-action value function.

MDP - computing optimal policy

1. Linear Programming

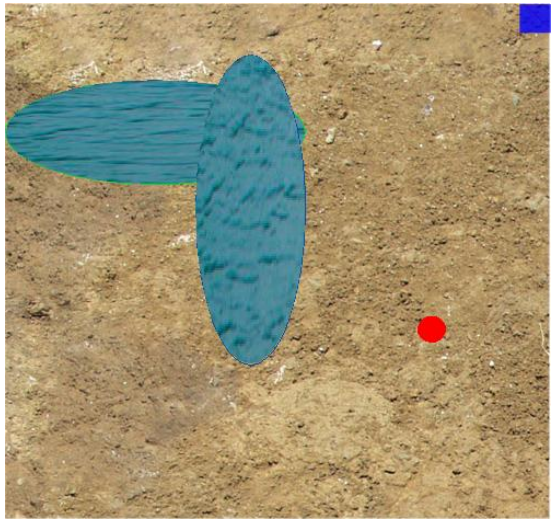
2. Value Iteration method.

$$V^{t+1}(s) \leftarrow \max_a \{R(s,a) + \gamma \sum_{s'} \delta(s,a,s') V^t(s')\}$$

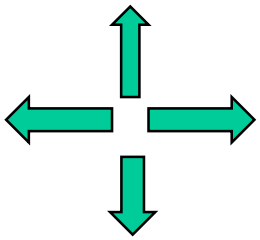
3. Policy Iteration method.

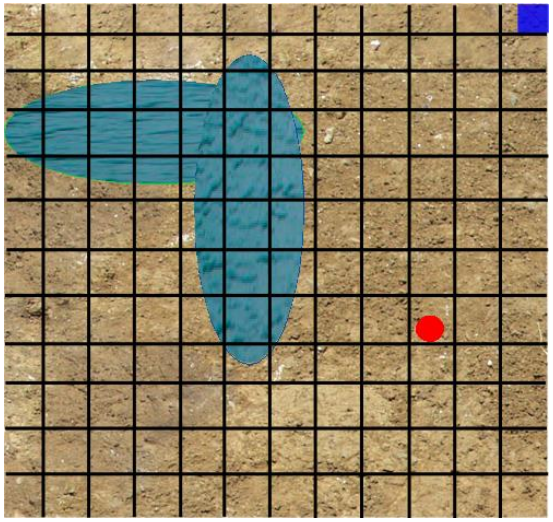
$$\pi_i(s) = \arg\max_a \{Q^{\pi_{i-1}}(s,a)\}$$

Example: Grid world



ACTIONS





Initial State = Red
Final State = Blue
COST = steps to target (blue)

Example: Grid world

∞	∞	∞	∞	∞	∞	∞	∞	∞	0
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

Initial Values
t=0

Example: Grid world

∞	∞	∞	∞	∞	∞	∞	∞	**1**	**0**
∞	∞	∞	∞	∞	∞	∞	∞	∞	**1**
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

One Iteration
t=1

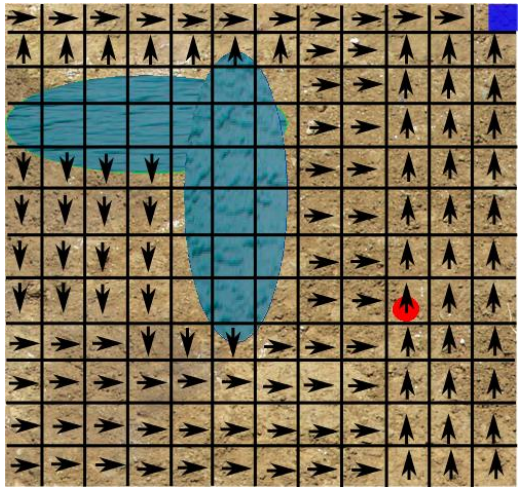
Example: Grid world

∞	∞	∞	∞	∞	∞	∞	**2**	**1**	**0**
∞	∞	∞	∞	∞	∞	∞	∞	**2**	**1**
∞	∞	∞	∞	∞	∞	∞	∞	∞	**2**
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

Two Iterations
t=2

Example: Grid world

- Optimal Policy



Q(s,→)= walk →
and add distance to target

Optimal policy
Derive from Q(s, -)

Convergence

- Value Iteration
  - Decrease the distance from optimal
    - By a factor of γ
- Policy Iteration
  - Policy improves
  - Number of iteration less than Value Iteration

Today Outline: Overview

- Basics
  - Goal of Reinforcement Learning
  - Mathematical Model (MDP)
- Planning
  - Value iteration
  - Policy iteration
- Learning Algorithms
  - Model based
  - Model Free
- Large state space
  - Function approx
  - Policy gradient

Learning Algorithms

Given access only to actions perform:

1. policy evaluation.
2. control - find optimal policy.

Two approaches:

1. Model based.
2. Model free.

Learning - Model Based

Estimate the model from the observation.
(Both transition probability and rewards.)

Use the estimated model as a true model,
and find a near optimal policy.

If we have a “good” estimated model, we should have a “good” estimation.

Learning - Model Based: off policy

- Let the policy run for a “long” time.
  - what is “long” ?!
  - Assuming some “exploration”
- Build an “observed model”:
  - Transition probabilities
  - Rewards
    - Both are independent!
- Use the “observed model” learn optimal policy.

Learning - Model Based
off-policy algorithm

- Observe a trajectory generated by a policy π
  - off-policy: not need to control
- For every $s, s' \in S$ and $a \in A$:
  - $\delta'(s, a, s') = \#(s, a, s') \,/\, \#(s, a, \cdot)$
  - $R'(s, a) = AVG(r \mid s, a)$
- Find an optimal policy for (S, a, δ', R')

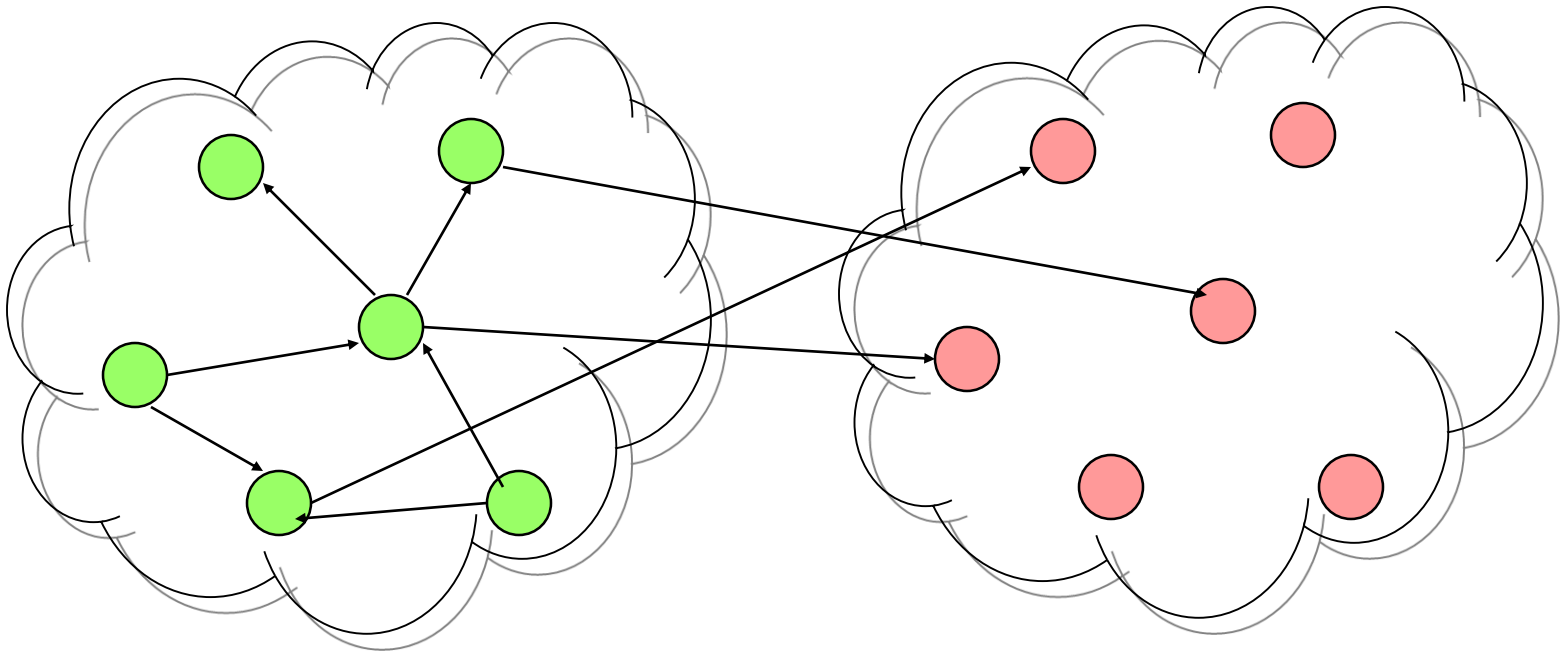
Learning - Model Based

- Claim: if the model is “accurate” we will compute a near-optimal policy.
- Hidden assumption:
  - Each $(s, a, \cdot)$ is sampled many times.
  - This is the “responsibility” of the policy π
    - Off-policy
- Simple question: How many samples we need for each $(s, a, \cdot)$

Learning - Model Based: on policy

- The learner has control over the action.
  - The immediate goal is to learn a model
- As before:
  - Build an “observed model”:
    - Transition probabilities and Rewards
  - Use the “observed model” to estimate value of the policy.
- Accelerating the learning:
  - How to reach “unexplored” states ?!

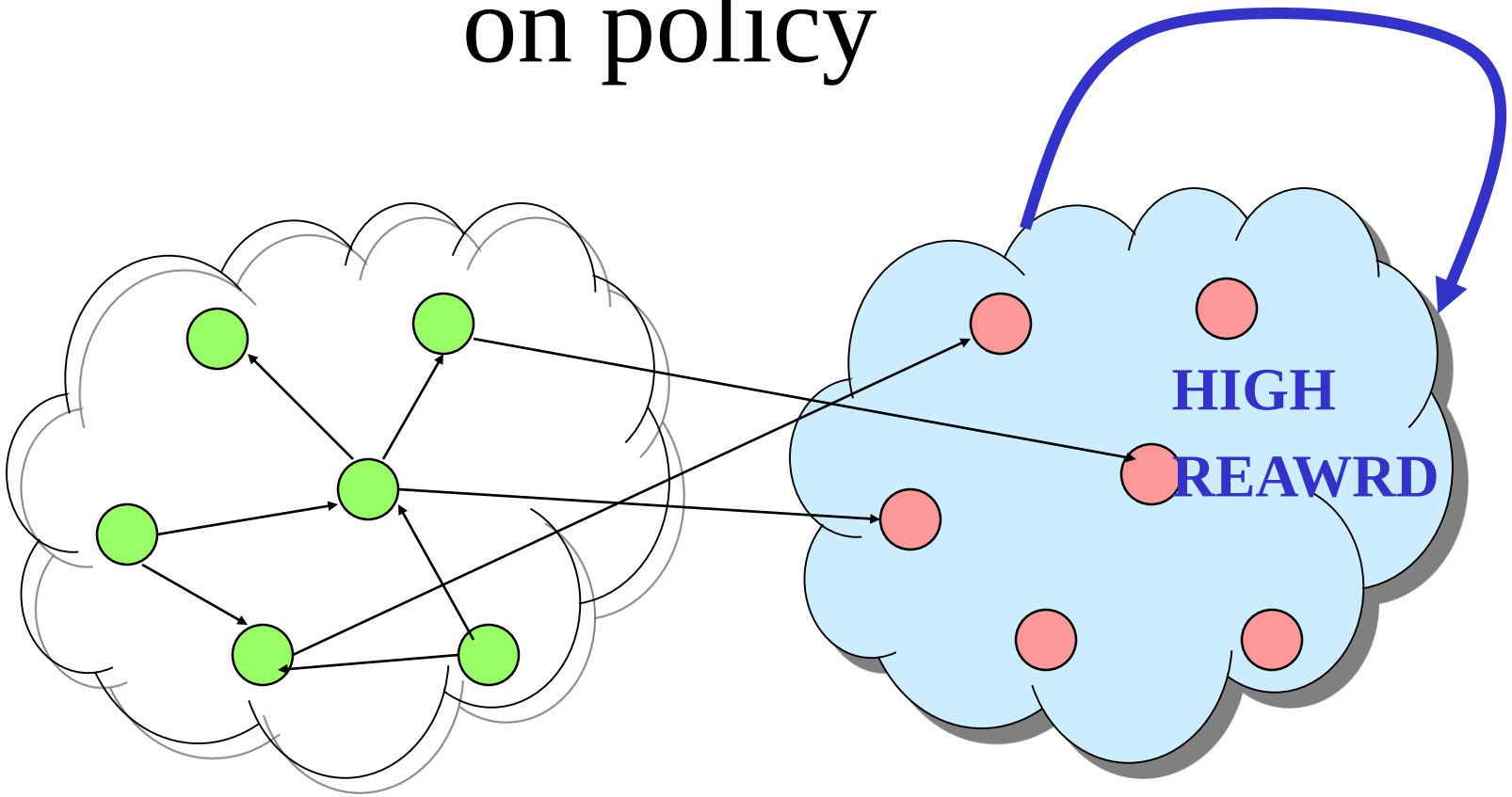
Learning - Model Based: on policy



Well sampled states

Relatively unknown states

Learning - Model Based: on policy



Well sampled states

Relatively unknown states

Exploration → Planning in new MDP

Learning: Policy improvement

- Assume that we can perform:
  - Given a policy π,
  - Compute V^{π} and Q^{π} functions of π
- Can run policy improvement:
  - π = Greedy (Q^{π})
- Process converges if estimates are accurate.

Model-Free learning

Q-Learning: off policy

Basic Idea: Learn the Q-function.

On a move (s,a) → s' update:

Old estimate

$$Q_{t+1}(s,a) = (1-\alpha_t(s,a))Q_t(s,a) +$$

$$\alpha_t(s,a)[R_t(s,a) + \gamma \max_u Q_t(s',u)]$$

New estimate

Learning rate $\alpha_t(s,a)=1/t^w$

Q-Learning: update equation

learning rate

update

$$Q_{t+1}(s,a) = Q_t(s,a) - \alpha_t(s,a)\Delta$$

$$\Delta = Q_t(s,a) - R_t(s,a) + \gamma \max_u Q_t(s',u)$$

Old estimate

New estimate

Q-Learning: Intuition

- Updates based on the difference:

$$\Delta = Q_t(s,a) - [R_t(s,a) + \gamma \max_u Q_t(s',u)]$$

- Assume we have the optimal Q^* function
- Good news: The expectation is Zero !!!
- Challenge: understand the dynamics
  - stochastic process

Learning - Model Free
Policy evaluation: TD(0)

An online view:
At state s_t we performed action a_t,
received reward r_t and moved to state s_{t+1}.

Our "estimation error" is $Err_t = r_t + \gamma V(s_{t+1}) - V(s_t)$,
The update:

$$V_{t+1}(s_t) = V_t(s_t) + \alpha\, Err_t$$

Note that for the correct value function we have:

$$E[r + \gamma V(s') - V(s)] = E[Err_t] = 0$$

Learning - Model Free
Policy evaluation: TD(λ)

Again: At state s_t we performed action a_t,
received reward r_t and moved to state s_{t+1}.
Our "estimation error" $\mathrm{Err}_t = r_t + \gamma V(s_{t+1}) - V(s_t)$,

Update **every** state ***s***:

$$V_{t+1}(s) = V_t(s) + \alpha \, Err_t \, e(s)$$

Update of e(s) :

When visiting *s*: incremented by 1: $e(s) = e(s)+1$

For all other *s'*:

decremented by $\gamma\lambda$ factor: $e(s') = \gamma\lambda\, e(s')$

Model-Free Algorithms

- On-policy versus Off-policy
- The function approximated (V vs Q)
- Fairly similar general methodologies
- Challenges: controlling stochastic process

Today Outline: Overview

- Basics
  - Goal of Reinforcement Learning
  - Mathematical Model (MDP)
- Planning
  - Value iteration
  - Policy iteration
- Learning Algorithms
  - Model based
  - Model Free
- Large state space
  - Function approx
  - Policy gradient

Large state MDP

Previous Methods: Tabular. (small state space).

Large state Exponential size.

Similar to basic view of learning.

Large state MDP

Approaches:

1. Restricted Value function.
2. Restricted policy class.
3. Restricted model of MDP.
4. Different MDP representation: Generative Model

Large MDP - Restricted Value Function

(Value) Function Approximation:
Use a limited class of functions to estimate the value function.

Given a good approximation of the value function, we can estimate the optimal policy.

Vague Idea: reduce to a supervised (deep) learning.

applications: most of the recent work (AlphaGo, Atari etc.)

Large MDP: Restricted Policy

- Fix a policy class $\Pi = \{\pi: S \to A\}$
  - Given a policy $\pi \in \Pi$
  - Approximate V^{π} and Q^{π}
- Run policy improvement:
  - π = Greedy (Q^{π})
- Quality depends on approximation.
  - Convergence not guaranteed.

Large MDP: Policy Gradient

- Improving the parameter of a policy
  - Taking the gradient
- Challenge:
  - The update impacts both:
    - Action probabilities
    - Distribution over states
  - Off-policy
    - compute the gradient from policy history

Large MDP: Generative Model.

Clearly we can not use matrix representation.

Generator representation:

Algorithm for estimating optimal policy (for discounted infinite horizon) in time independent of the number of states

Large state MDP: Generative Model.

- Use generative model for “look ahead”
- Sample a tree of a shallow depth
  - Using generative model
- Compute optimal policy on the tree
- Result in approximate optimal policy in the MDP

Today Outline: Overview

- Basics
  - Goal of Reinforcement Learning
  - Mathematical Model (MDP)
- Planning
  - Value iteration
  - Policy iteration
- Learning Algorithms
  - Model based
  - Model Free
- Large state space
  - Function approx
  - Policy gradient

Course (tentative) Outline

- Part 1: MDP basics and planning
- Part 2: MDP learning
  - Model Based and Model free
- Part 3: Large state MDP
  - Policy Gradient, Deep Q-Network
- Part 4: Special MDPs
  - Bandits and Partially Observable MDP
- Part 5: Advanced topics