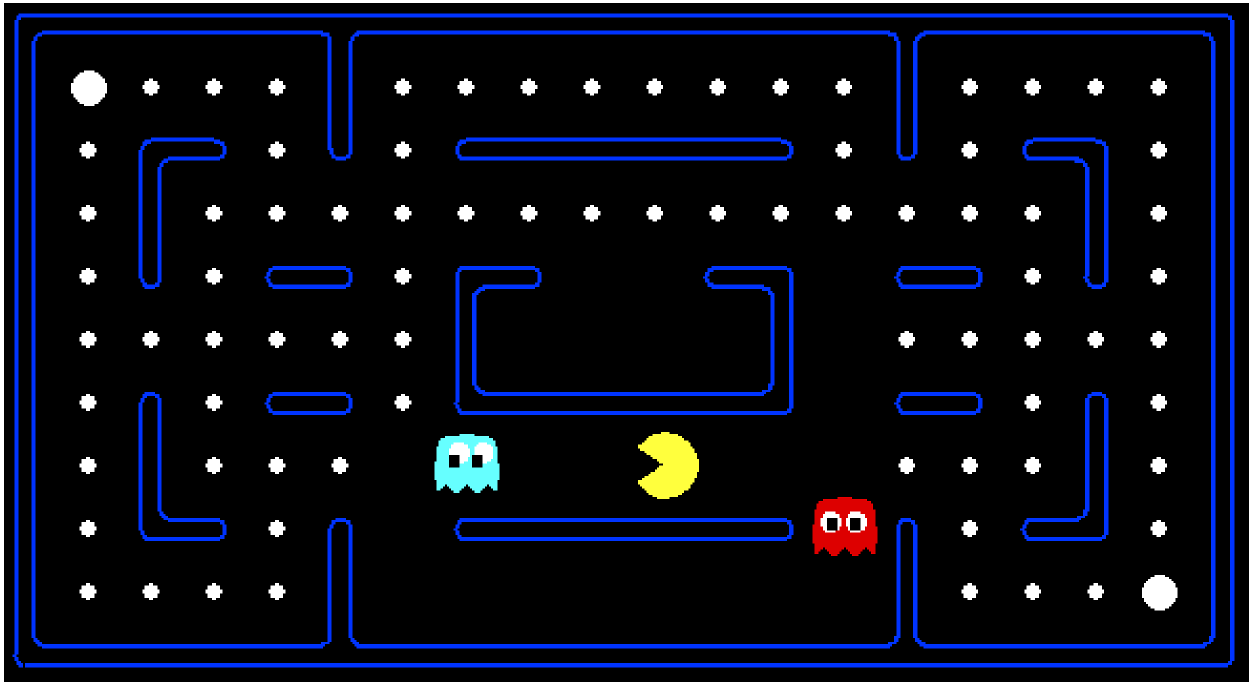
Warm up

- Pick an agent among {Pacman, Blue Ghost, Red Ghost}. Design an algorithm to control your agent. Assume they can see each others' location but can't talk. Assume they move simultaneously in each step.

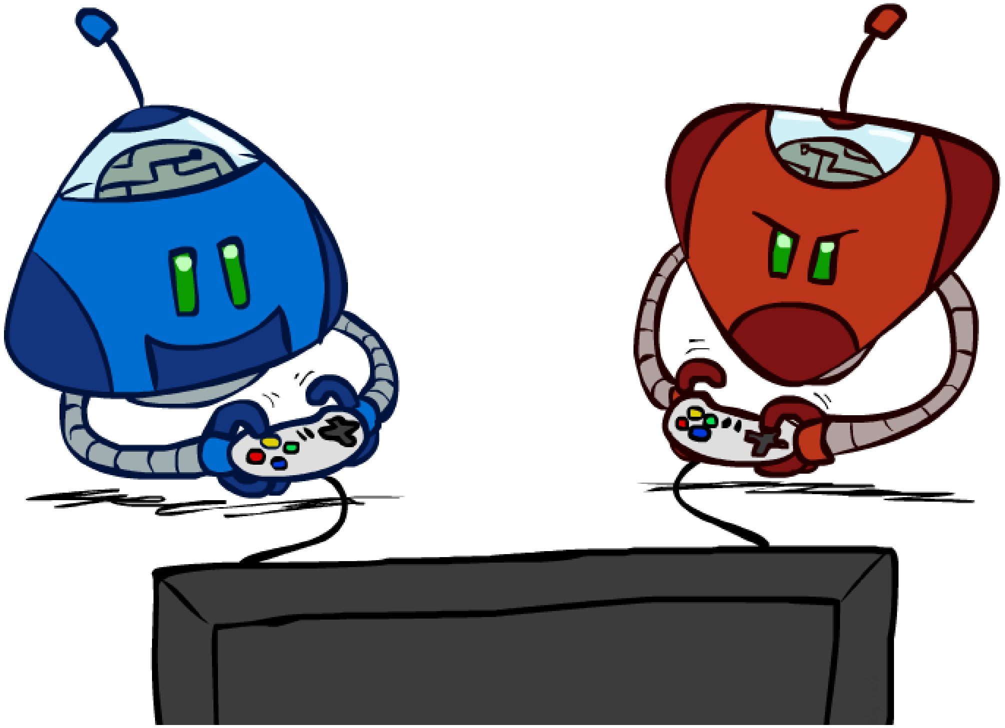


Announcement

- Assignments
  - HW12 (written) ~~due 12/4 Wed, 10 pm~~
    Due 12/6 Fri, 10 pm
- Final exam
  - 12/12 Thu, 1pm-4pm
- Piazza post for in-class questions

AI: Representation and Problem Solving

Multi-Agent Reinforcement Learning



Instructors: Fei Fang & Pat Virtue

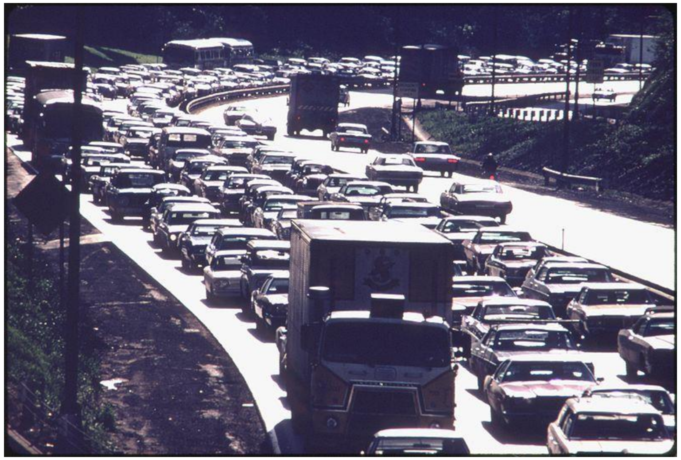
Slide credits: CMU AI and http://ai.berkeley.edu

Learning objectives

- Compare single-agent RL with multi-agent RL
- Describe the definition of Markov games
- Describe and implement
  - Minimax-Q algorithm
  - Fictitious play
- Explain at a high level how fictitious play and double-oracle framework can be combined with single-agent RL algorithms for multi-agent RL

Single-Agent → Multi-Agent

- Many real-world scenarios have more than one agent!
  - Autonomous driving

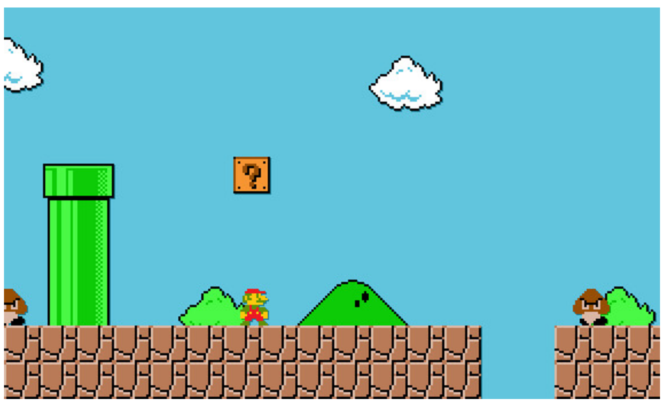


Single-Agent → Multi-Agent

- Many real-world scenarios have more than one agent!
  - Autonomous driving
  - Humanitarian Assistance / Disaster Response

Single-Agent → Multi-Agent

- Many real-world scenarios have more than one agent!
  - Autonomous driving
  - Humanitarian Assistance / Disaster Response
  - Entertainment

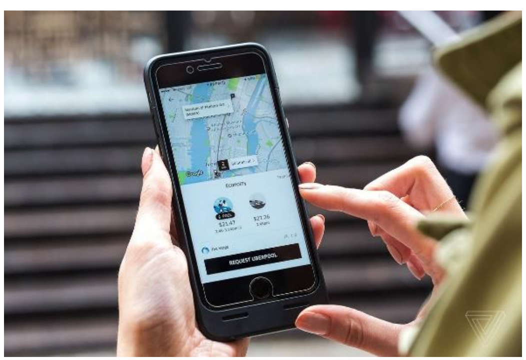


Single-Agent → Multi-Agent

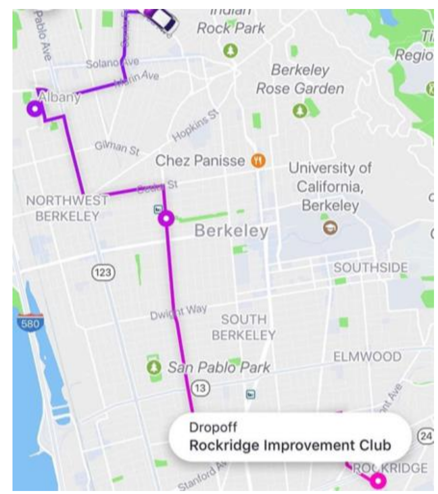
- Many real-world scenarios have more than one agent!
  - Autonomous driving
  - Humanitarian Assistance / Disaster Response
  - Entertainment
  - Infrastructure security / green security / cyber security

Single-Agent → Multi-Agent

- Many real-world scenarios have more than one agent!
  - Autonomous driving
  - Humanitarian Assistance / Disaster Response
  - Entertainment
  - Infrastructure security / green security / cyber security
  - Ridesharing



+



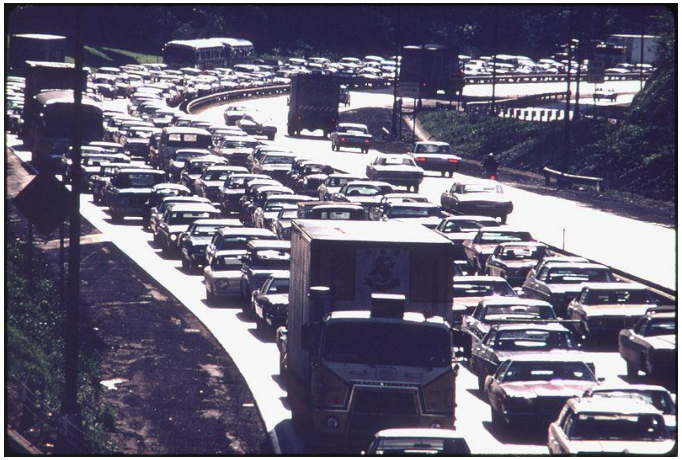
Recall: Normal-Form/Extensive-Form games

- Games are specified by
  - Set of players
  - Set of actions for each player (at each decision point)
  - Payoffs for all possible game outcomes
  - (Possibly imperfect) information each player has about the other player's moves when they make a decision
- Solution concepts
  - Nash equilibrium, dominant strategy equilibrium, Minimax/Maximin strategy, Stackelberg equilibrium
- Approaches to solve the game
  - Iterative removal, Solving linear systems, Linear programming

Single-Agent → Multi-Agent

- Can we use these approaches to previous problems?

Alex	Berry: Football	Berry: Concert
Football	2,1	0,0
Concert	0,0	1,2



- Limitations of classic approaches in game theory
  - Scalability: Can hardly handle complex problems
  - Need to specify payoff for all outcomes
  - Often need domain knowledge for improvement (e.g., abstraction)

Recall: Reinforcement learning

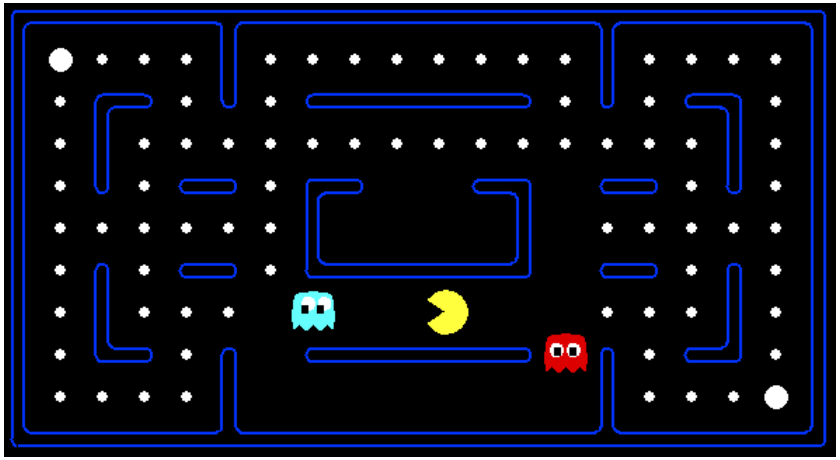
- Assume a Markov decision process (MDP):
  - A set of states $s \in S$
  - A set of actions (per state) A
  - A model T(s,a,s')
  - A reward function R(s,a,s')

- Looking for a policy $\pi(s)$ without knowing T or R

- Learn the policy through experience in the environment

Single-Agent → Multi-Agent

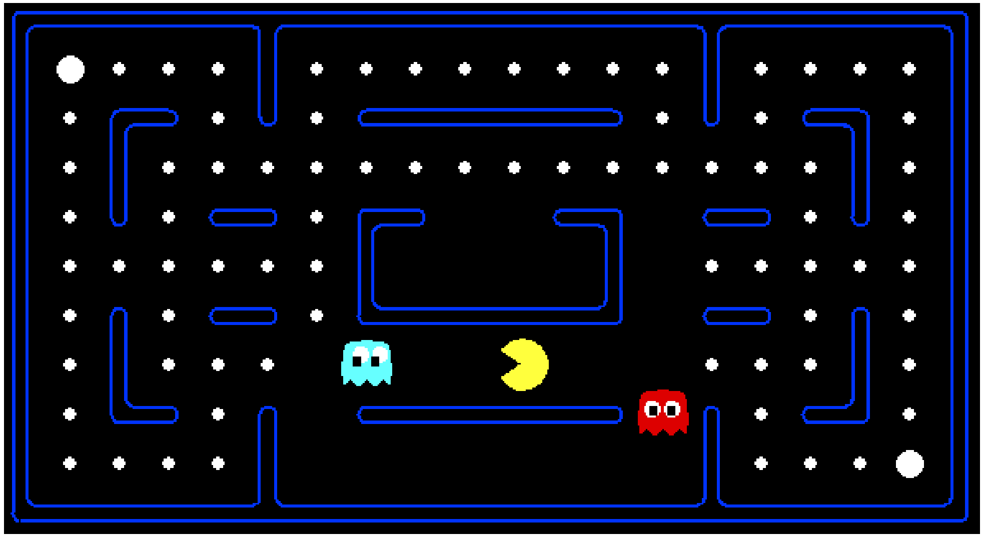
- Can we apply single-agent RL to previous problems? How?
  - Simultaneously independent single-agent RL, i.e., let every agent i use Q-learning to learn $Q(s, a_i)$ at the same time
  - Effective only in some problems (limited agent interactions)
  - Limitations of single-agent RL in multi-agent setting
    - Instability and adapatability: Agents are co-evolving



If treat other agents as part of environment, this environment is changing over time!

Single-Agent → Multi-Agent

- Multi-Agent Reinforcement Learning
  - Let the agents learn through interacting with the environment and with each other
  - Simplest approach: Simultaneously independent single-agent RL (suffer from instability and adapatability)
  - Need better approaches



Multi-Agent Reinforcement Learning

- Assume a Markov game:
  - A set of N agents
  - A set of states S
    - Describing the possible configurations for all agents
  - A set of actions for each agent $A_1, \ldots, A_N$
  - A transition function $T(s, a_1, a_2, \ldots, a_n, s')$
    - Probability of arriving at state s' after all the agents taking actions $a_1, a_2, \ldots, a_n$ respectively
  - A reward function for each agent $R_i(s, a_1, a_2, \ldots, a_n)$

Piazza Poll 1

- You know that the state at time t is s_t and the actions taken by the players at time t is $a_{t,1}, \dots, a_{t,N}$. The reward for agent i at time $t+1$ is dependent on which factors?
  - A: s_t
  - B: $a_{t,i}$
  - C: $a_{t,-i} \triangleq a_{t,1}, \dots, a_{t,i-1}, a_{t,i+1}, \dots, a_{t,N}$
  - D: None
  - E: I don't know

Multi-Agent Reinforcement Learning

- Assume a Markov game

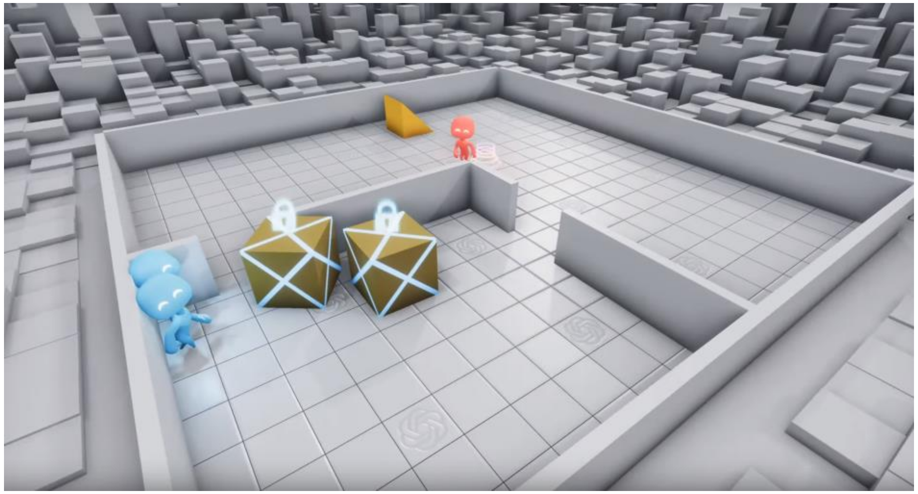
- Looking for a set of policies $\{\pi_i\}$, one for each agent, without knowing $T, R_i, \forall i$
  - $\pi_i(s, a)$ is the probability of choosing action a at state s

- Each agent's total expected return is $\sum_t \gamma^t r_i^t$ where γ is the discount factor

- Learn the policies through experience in the environment and interact with each other

Multi-Agent Reinforcement Learning

- Descriptive
  - What would happen if agents learn in a certain way?
  - Propose a model of learning that mimics learning in real life
  - Analyze the emergent behavior with this learning model (expecting them to agree with the behavior in real life)
  - Identify interesting properties of the learning model



Multi-Agent Reinforcement Learning

- Prescriptive (our main focus today)
  - How agents should learn?
  - Not necessary to show a match with real-world phenomena
  - Design a learning algorithm to get a “good” policy
    (e.g., high total reward against a broad class of other agents)

DeepMind's AlphaStar beats 99.8% of human

Recall: Value Iteration and Bellman Equation

- Value iteration

$$V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_k(s')], \forall s$$

- With reward function $R(s,a)$

$$V_{k+1}(s) = \max_a R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_k(s'), \forall s$$

- When converges (Bellman Equation)

$$V^*(s) = \max_a Q^*(s,a), \forall s$$

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s'), \forall a, s$$

Value Iteration in Markov Games

$$V^*(s) = \max_a Q^*(s,a), \forall s$$

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s'), \forall a, s$$

- In two-player zero-sum Markov game
  - Let $V^*(s)$ be state value for player 1 ($-V^*(s)$ for player 2)
  - Let $Q^*(s, a_1, a_2)$ be action-state value for player 1 when player 1 chooses a_1 and player 2 chooses a_2 in state s

$$Q^*(s, a_1, a_2) =$$

$$V^*(s) =$$

Minimax-Q Algorithm

- Value iteration requires knowing T, R_i

- Minimax-Q [Littman94]
  - Extension of Q-learning
  - For two-player zero-sum Markov games
  - Provably converges to Nash equilibria in self play

A learning agent learns through interacting with another learning agent using the same learning algorithm

Minimax-Q Algorithm

Initialize $Q(s, a_1, a_2) \leftarrow 1, V(s) \leftarrow 1, \pi_1(s, a_1) \leftarrow \frac{1}{|A_1|}, \alpha \leftarrow 1$

Take actions: At state s, with prob. ϵ choose a random action, and with prob. $1 - \epsilon$ choose action according to $\pi_1(s, a)$

Learn: after receiving r_1 for moving from s to s' via a_1, a_2

$$Q(s, a_1, a_2) \leftarrow (1 - \alpha)Q(s, a_1, a_2) + \alpha\big(r_1 + \gamma V(s')\big)$$

$$\pi_1(s, \cdot) \leftarrow \underset{\pi_1'(s,\cdot) \in \Delta(A_1)}{\operatorname{argmax}} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \pi_1'(s, a_1) Q(s, a_1, a_2)$$

$$V(s) \leftarrow \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \pi_1(s, a_1) Q(s, a_1, a_2)$$

Update α

Minimax-Q Algorithm

- How to solve the maximin problem?

$$\pi_1(s,\cdot) \leftarrow \underset{\pi_1'(s,\cdot)\in\Delta(A_1)}{\operatorname{argmax}} \min_{a_2\in A_2} \sum_{a_1\in A_1} \pi_1'(s,a_1) Q(s,a_1,a_2)$$

$$V(s) \leftarrow \min_{a_2\in A_2} \sum_{a_1\in A_1} \pi_1(s,a_1) Q(s,a_1,a_2)$$

Linear Programming: $\max_{\pi_1'(s,\cdot),v} \; v$

Get optimal solution $\pi_1'^*(s,\cdot), v^*$, update $\pi_1(s,\cdot) \leftarrow \pi_1'^*(s,\cdot), V(s) \leftarrow v^*$

Minimax-Q Algorithm

- How does player 2 chooses action a_2?

- If player 2 is also using the minimax-Q algorithm
  - Self-play
  - Proved to converge to NE

- If player 2 chooses actions uniformly randomly, the algorithm still leads to a good policy empirically in some games

Minimax-Q for Matching Pennies

- A simple Markov game: Repeated Matching Pennies

Player 2

Player 1	Heads	Tails
Heads	1,-1	-1,1
Tails	-1,1	1,-1

- Let state to be dummy: Player's strategy is not dependent on past actions. Just play a mixed strategy as in the one-shot game

- Discount factor $\gamma = 0.9$

Minimax-Q for Matching Pennies

	Heads	Tails
Heads	1,-1	-1,1
Tails	-1,1	1,-1

Simplified version for this games with only one state

Initialize $Q(a_1, a_2) \leftarrow 1, V \leftarrow 1, \pi_1(a_1) \leftarrow 0.5, \alpha \leftarrow 1$

Take actions: With prob. ϵ choose a random action, and with prob. $1-\epsilon$ choose action according to $\pi_1(a)$

Learn: after receiving r_1 with actions a_1, a_2

$$Q(a_1, a_2) \leftarrow (1-\alpha)Q(a_1, a_2) + \alpha(r_1 + \gamma V)$$

$$\pi_1(\cdot) \leftarrow \underset{\pi_1'(\cdot)\in\Delta^2}{\operatorname{argmax}} \min_{a_2\in A_2} \sum_{a_1\in A_1} \pi_1'(a_1)Q(a_1, a_2)$$

$$V \leftarrow \min_{a_2\in A_2} \sum_{a_1\in A_1} \pi_1(a_1)Q(a_1, a_2)$$

Update $\alpha = 1/$ #times (a_1, a_2) visited

Minimax-Q for Matching Pennies

	Heads	Tails
Heads	1,-1	-1,1
Tails	-1,1	1,-1

t	Actions	Reward_1	$Q_t(H,H)$	$Q_t(H,T)$	$Q_t(T,H)$	$Q_t(T,T)$	$V(s)$	$\pi_1(H)$
0			1	1	1	1	1	0.5
1	(H*,H)	1	1.9	1	1	1	1	0.5

$$Q(a_1, a_2) \leftarrow (1-\alpha)Q(a_1, a_2) + \alpha(r_1 + \gamma V)$$

$$\max_{\pi_1'(s,\cdot),v} v$$

$$v \leq \sum_{a_1 \in A_1} \pi_1'(s, a_1) Q(s, a_1, a_2), \forall a_2$$

$$\sum_{a_1 \in A_1} \pi_1'(s, a_1) = 1$$

$$\pi_1'(s, a_1) \geq 0, \forall a_1$$

Piazza Poll 2

- If the actions are (H,T) in round 1 with a reward of -1 to player 1, what would be the updated value of $Q(H,T)$ with $\gamma = 0.9$?
- A: 0.9
- B: 0.1
- C: -0.1
- D: 1.9
- E: I don't know

t	Actions	$\mathbf{Reward_1}$	$\mathbf{Q_t(H,H)}$	$\mathbf{Q_t(H,T)}$	$\mathbf{Q_t(T,H)}$	$\mathbf{Q_t(T,T)}$	$\mathbf{V(s)}$	$\pi_1(\mathbf{H})$
0			1	1	1	1	1	0.5
1	(H*,H)	1	1.9	1	1	1	1	0.5

Minimax-Q for Matching Pennies

	Heads	Tails
Heads	1,-1	-1,1
Tails	-1,1	1,-1

t	Actions	Reward_1	$Q_t(H,H)$	$Q_t(H,T)$	$Q_t(T,H)$	$Q_t(T,T)$	$V(s)$	$\pi_1(H)$
0			1	1	1	1	1	0.5
1	(H*,H)	1	1.9	1	1	1	1	0.5
2	(T,H)	-1	1.9	1	-0.1	1	1	0.55
3	(T,T)	1	1.9	1	-0.1	1.9	1.279	0.690
4	(H*,T)	-1	1.9	0.151	-0.1	1.9	0.967	0.534
5	(T,H)	-1	1.9	0.151	-0.115	1.9	0.964	0.535
6	(T,T)	1	1.9	0.151	-0.115	1.884	0.960	0.533
7	(T,H)	-1	1.9	0.151	-0.122	1.884	0.958	0.534
8	(H,T)	-1	1.9	0.007	-0.122	1.884	0.918	0.514
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
100	(H,H)	1	1.716	-0.269	-0.277	1.730	0.725	0.503
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
000	(T,T)	1	1.564	-0.426	-0.415	1.564	0.574	0.500
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

How to Evaluate a MARL algorithm (prescriptive)?

- Brainstorming: how to evaluate minimiax-Q?
  - Recall: Design a learning algorithm Alg to get a "good" policy (e.g., high total expected return against a broad class of other agents)

How to Evaluate a MARL algorithm (prescriptive)?

- Training: Find a policy for agent 1 through minimax-Q
- Let an agent 1 learn with minimax-Q while agent 2 is
  - Also learning with minimax-Q (Self-play)
  - Using a heuristic strategy, e.g., random
  - Learning using a different learning algorithm, e.g., vanilla Q-learning or a variant of minimax-Q
  - Exemplary resulting policy:
    - π_1^{MM} (Minimax-Q-trained-against-selfplay)
    - π_1^{MR} (Minimax-Q-trained-against-Random)
    - π_1^{MQ} (Minimax-Q-trained-against-Q)

Co-evolving!

How to Evaluate a MARL algorithm (prescriptive)?

- Testing: Fix agent 1's strategy π_1, no more change
- Test again an agent 2's strategy π_2, which can be
  - A heuristic strategy, e.g., random
  - Trained using a different learning algorithm, e.g., vanilla Q-learning or a variant of minimax-Q
    - Need to specify agent 1's behavior during training agent 2 (random? Minimax-Q? Q-learning?), can be different from π_1 or even co-evolving
  - Best response to player 1's strategy π_1
    - Worst case for player 1
    - Fix π_1, treat player 1 as part of the environment, find the optimal policy for player 2 through single-agent RL

How to Evaluate a MARL algorithm (prescriptive)?

- Testing: Fix agent 1's strategy π_1, no more change
- Test again an agent 2's strategy π_2, which can be
  - Exemplary policy for agent 2:
    - π_2^{MM} (Minimax-Q-trained-against-selfplay)
    - π_2^{MR} (Minimax-Q-trained-against-Random)
    - π_2^{R} (Random)
    - $\pi_2^{BR} = BR(\pi_1)$ (Best response to π_1)

Piazza Poll 3

- Only consider strategies resulting from minimax-Q algorithm and random strategy. How many different tests can we run? An example test can be:

$$\pi_1^{MM}\text{(Minimax-Q-trained-against-selfplay) vs } \pi_2^{R}\text{(Random)}$$

- A: 1
- B: 2
- C: 4
- D: 9
- E: Other
- F: I don't know

Piazza Poll 3

- Only consider strategies resulting from minimax-Q algorithm and random strategy. How many different tests can we run?
- π_1 can be
  - π_1^{MM} (Minimax-Q-trained-against-selfplay)
  - π_1^{MR} (Minimax-Q-trained-against-Random)
  - π_1^{R} (Random)
- π_2 can be
  - π_2^{MM} (Minimax-Q-trained-against-selfplay)
  - π_2^{MR} (Minimax-Q-trained-against-Random)
  - π_2^{R} (Random)
- So 3*3=9

Fictitious Play

- A simple learning rule
  - An iterative approach for computing NE in two-player zero-sum games
  - Learner explicitly maintain belief about opponent's strategy
  - In each iteration, learner
    - Best responds to current belief about opponent
    - Observe the opponent's actual play
    - Update belief accordingly
  - Simplest way of forming the belief: empirical frequency!

Fictitious Play

- One-shot matching pennies

Player 2

Player 1

	Heads	Tails
Heads	1,-1	-1,1
Tails	-1,1	1,-1

Let $w(a)$= #times opponent play a

Agent believes opponent's strategy is choosing a with prob. $\frac{w(a)}{\sum_{a'} w(a')}$

Round	1's action	2's action	1's belief in $w(a)$	2's belief in $w(a)$
0			(1.5,2)	(2,1.5)
1	T	T	(1.5,3)	(2,2.5)
2				
3				
4				

Fictitious Play

- How would actions change from iteration t to $t+1$?
  - Steady state: whenever a pure strategy profile $\mathbf{a}=(a_1,a_2)$ is played in t, it will be played in $t+1$
  - If $\mathbf{a}=(a_1,a_2)$ is a strict NE (deviation leads to lower utility), then it is a steady state of FP
  - If $\mathbf{a}=(a_1,a_2)$ is a steady state of FP, then it is a (possibly weak) NE in the game

Fictitious Play

- Will this process converge?
  - Assume agents use empirical frequency to form the briefs

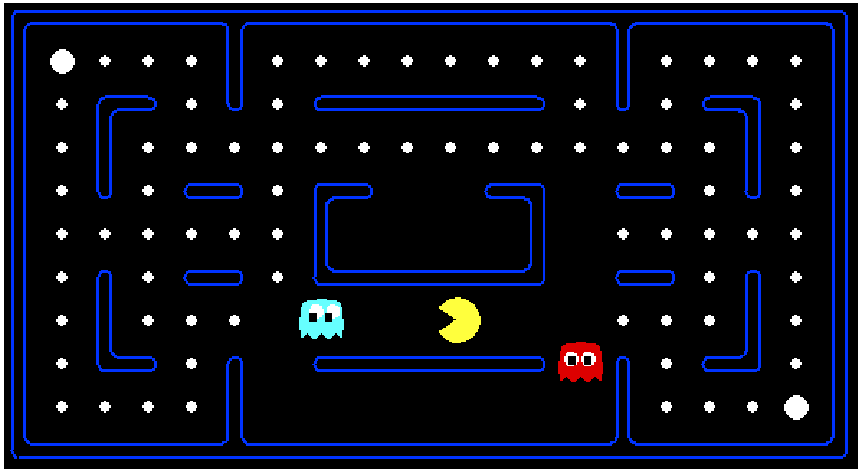
  - Empirical frequencies of play converge to NE if the game is
    - Two-player zero-sum
    - Solvable by iterative removal
    - Some other cases

Fictitious Play with Reinforcement Learning

- In each iteration, best responds to opponents' historical average strategy
- Find best response through single-agent RL

Basic implementation: Perform a complete RL process until convergence for each agent in each iteration

Time consuming ☹



(Optional) MARL with Partial Observation

- Assume a Markov game with partial observation (imperfect information):
  - A set of N agents
  - A set of states S
    - Describing the possible configurations for all agents
  - A set of actions for each agent $A_1, \ldots, A_N$
  - A transition function $T(s, a_1, a_2, \ldots, a_n, s')$
    - Probability of arriving at state s' after all the agents taking actions $a_1, a_2, \ldots, a_n$ respectively
  - A reward function for each agent $R_i(s, a_1, a_2, \ldots, a_n)$
  - A set of observations for each agent $O_1, \ldots, O_N$
  - A observation function for each agent $\Omega_i(s)$

(Optional) MARL with Partial Observation

- Assume a Markov game with partial observation

- Looking for a set of policies $\{\pi_i(o_i)\}$, one for each agent, without knowing T, R_i or Ω_i

- Learn the policies through experience in the environment and interact with each other

- Many algorithm can be applied, e.g., use a simple variant of Minimax-Q

Patrol with Real-Time Information

- Sequential interaction
  - Players make flexible decisions instead of sticking to a plan
  - Players may leave traces as they take actions

- Example domain: Wildlife protection

Footprints

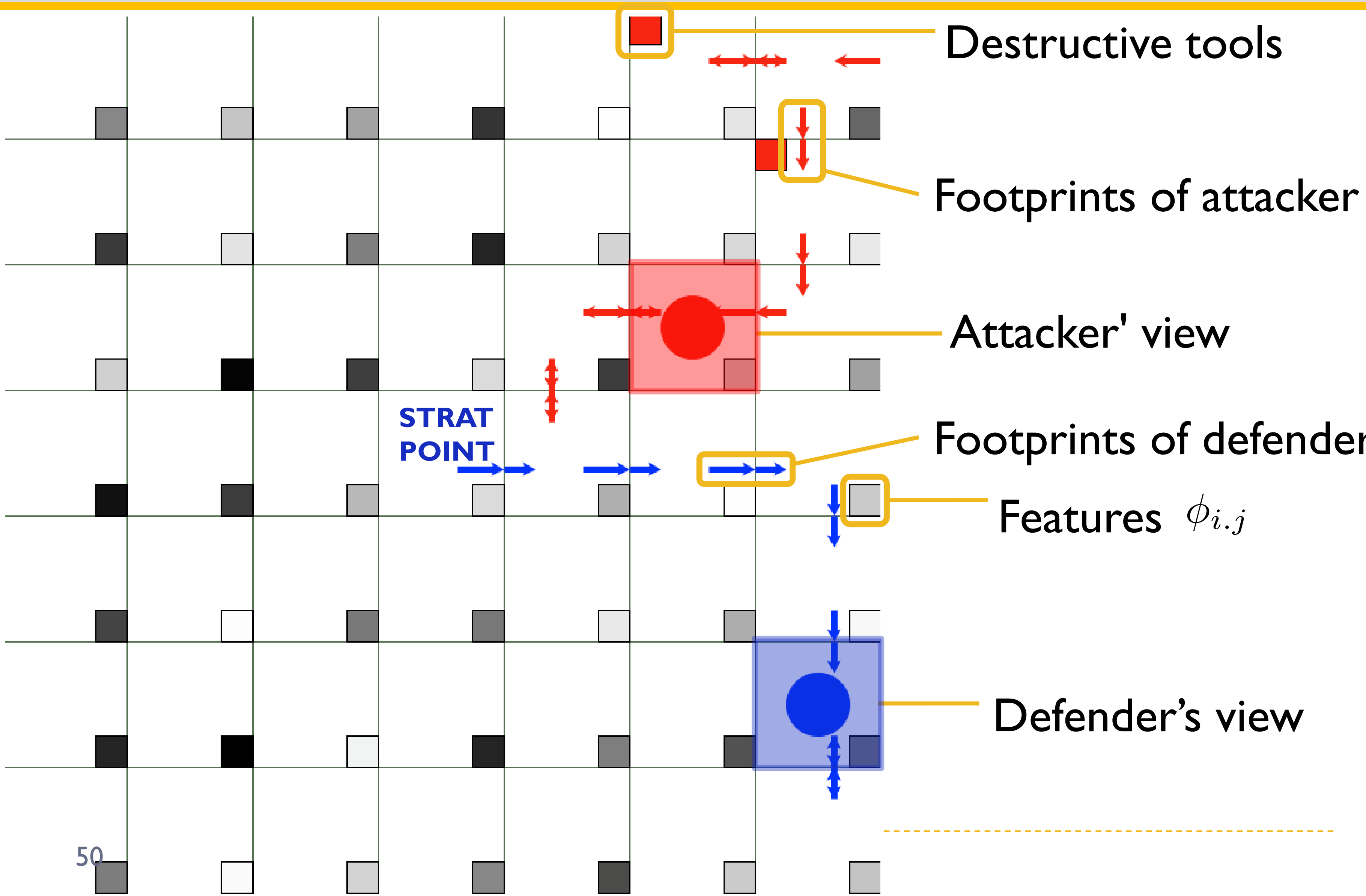
Lighters

Poacher camp

Tree marking

Deep Reinforcement Learning for Green Security Games with Real-Time Information Yufei Wang, Zheyuan Ryan Shi, Lantao Yu, Yi Wu, Rohit Singh, Lucas Joppa, Fei Fang In AAAI-19

Patrol with Real-Time Information

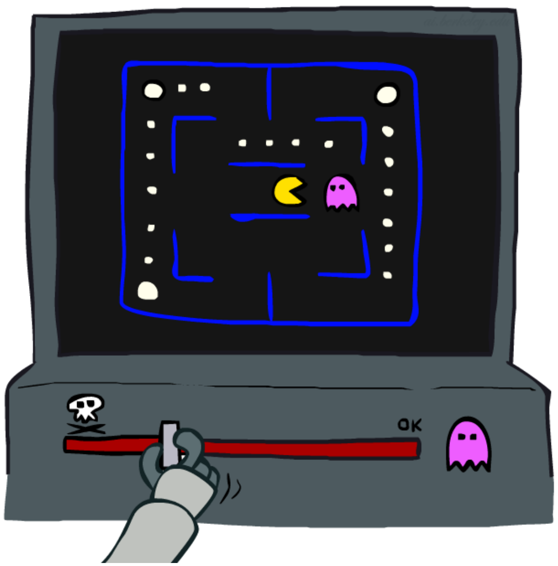


Recall: Approximate Q-Learning

- Features are functions from q-state (s, a) to real numbers, e.g.,
  - $f_1(s,a)$=Distance to closest ghost
  - $f_2(s,a)$=Distance to closest food
  - $f_3(s,a)$=Whether action leads to closer distance to food

- Aim to learn the q-value for any (s,a)
  - Assume the q-value can be approximated by a parameterized Q-function

$$Q(s,a) \approx Q_w(s,a)$$



If $Q_w(s,a)$ is a linear function of features:

$$Q_w(s,a) = w_1 f_1(s,a) + \ldots + w_n f_n(s,a)$$

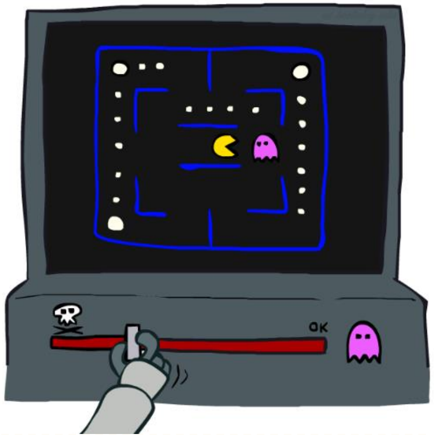
Recall: Approximate Q-Learning

Need to learn parameters w through interacting with the environment

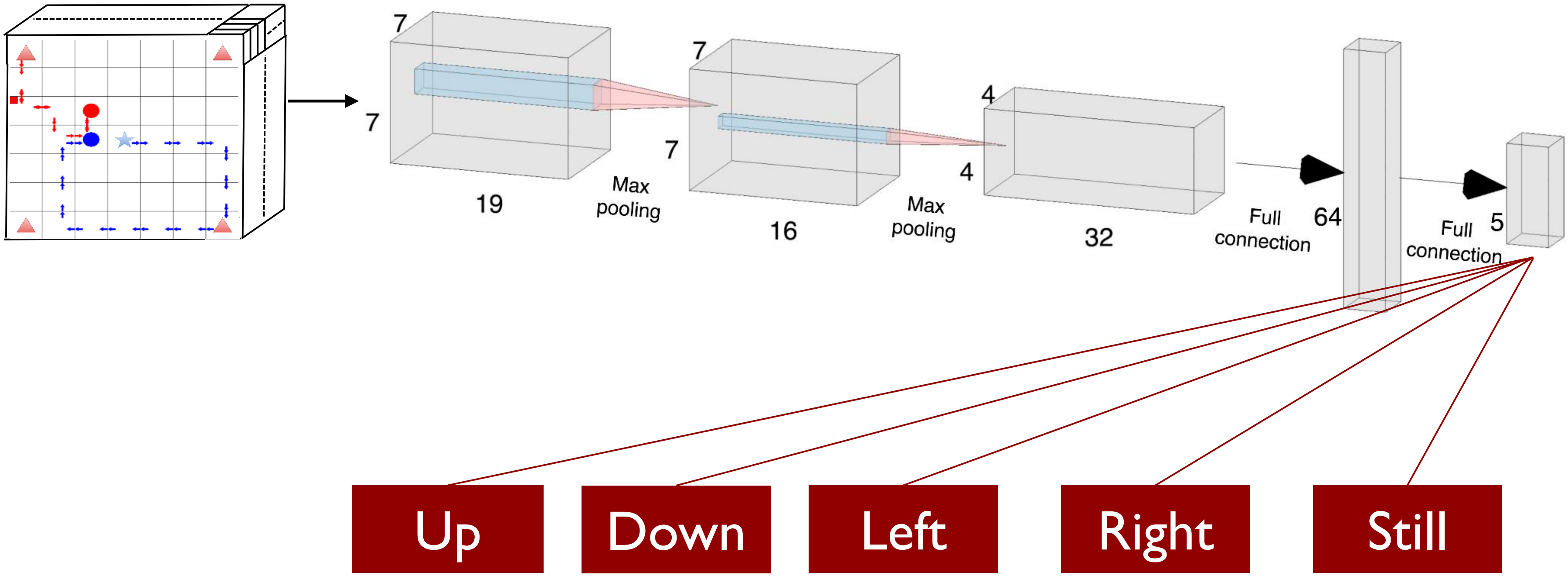
Update Rule for Approximate Q-Learning with Q-Value Function:

$$w_i \leftarrow w_i + \alpha \left(\underbrace{r + \gamma \max_{a'} Q_w(s', a')}_{\text{Latest sample}} - \underbrace{Q_w(s, a)}_{\text{Previous estimate}} \right) \frac{\partial Q_w(s, a)}{\partial w_i}$$

If latest sample higher than previous estimate:
adjust weights to increase the estimated Q-value

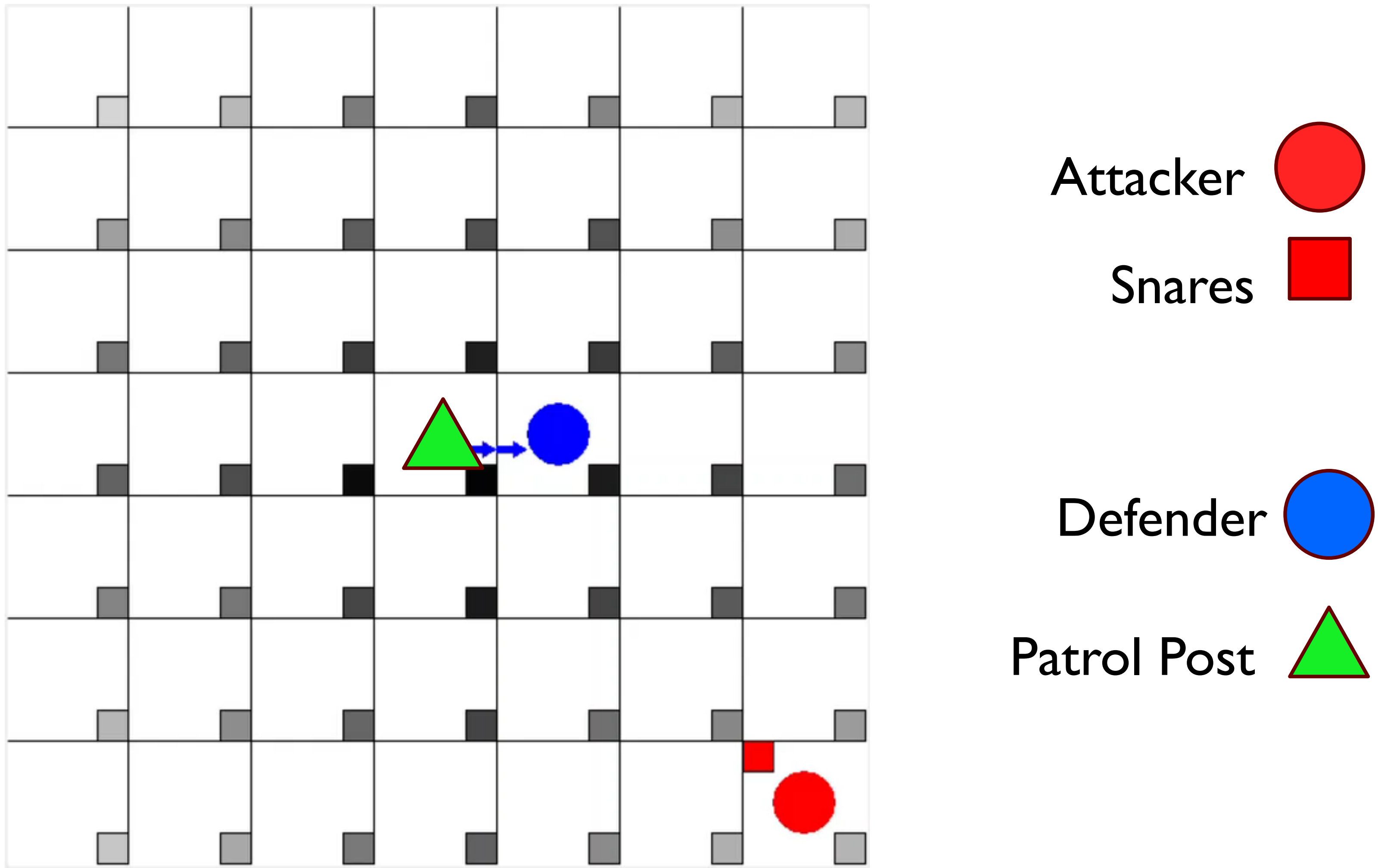


(Optional) Train Defender Against Heuristic Attacker



- Through single-agent RL
- Use neural network to represent a parameterized Q function $Q(o_i, a_i)$ where o is the observation

(Optional) Train Defender Against Heuristic Attacker



Compute Equilibrium: RL + Double Oracle

Compute $\sigma^d, \sigma^a = Nash(G^d, G^a)$

Compute Nash/Minimax

Train $f^d = RL(\sigma^a)$

Find Best Response to attacker's strategy

Add f^d,f^a to G^d, G^a

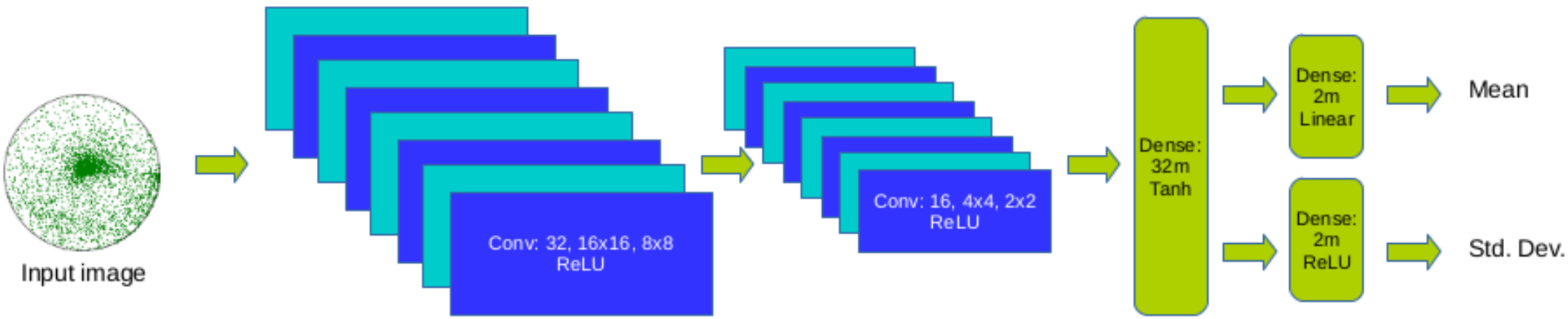
Update bags of strategies

Train$f^a = RL(\sigma^a)$

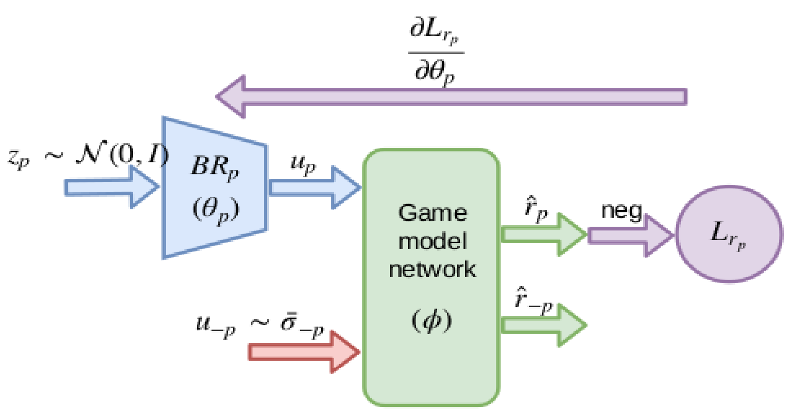
Find Best Response to defender's strategy

(Optional) Other Domains: Patrol in Continuous Area

OptGradFP: CNN + Fictitious Play



DeepFP: Generative network + Fictitious Play



Policy Learning for Continuous Space Security Games using Neural Networks. Nitin Kamra, Umang Gupta, Fei Fang, Yan Liu, Milind Tambe. In AAAI-18

DeepFP for Finding Nash Equilibrium in Continuous Action Spaces. Nitin Kamra, Umang Gupta, Kai Wang, Fei Fang, Yan Liu, Milind Tambe. In GameSec-19

AI Has Great Potential for Social Good

Artificial Intelligence

Machine Learning

Computational Game Theory

Societal Challenges

Security & Safety

Environmental Sustainability

Mobility