

Bootcamp 6: Reinforcement Learning



William H. Guss, James Bartlett
{wguss, james}@ml.berkeley.edu
Machine Learning at Berkeley

April 22, 2016

1 Introduction

2 Theory

3 Algorithms

4 Questions

Problem: ML for Pacman.



Reinforcement
Learning

Guss &
Bartlett

Introduction

Theory

Algorithms

Questions

*How would you solve pacman
with machine learning?*



Problem: ML for Pacman.



Reinforcement
Learning

Guss &
Bartlett

Introduction

Theory

Algorithms

Questions

*How would you solve pacman
with machine learning?*

**Find a model which takes
screen pixels to actions:**

$$\pi_{\theta} : s_t \mapsto a_t.$$



Problem: ML for Pacman.



Reinforcement
Learning

Guss &
Bartlett

Introduction

Theory

Algorithms

Questions

*How would you solve pacman
with machine learning?*

**Find a model which takes
screen pixels to actions:**

$$\pi_{\theta} : s_t \mapsto a_t.$$

*What is your loss function?
Data?*



Problem: ML for Pacman.



Reinforcement
Learning

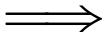
Guss &
Bartlett

Introduction

Theory

Algorithms

Questions



Solution: Reinforcement Learning



Reinforcement
Learning

Guss &
Bartlett

Introduction

Theory

Algorithms

Questions

- Supervised learning is *not* the most general formulation of learning.



Solution: Reinforcement Learning



Reinforcement
Learning

Guss &
Bartlett

Introduction

Theory

Algorithms

Questions

- Supervised learning is *not* the most general formulation of learning.
- Humans learn through reward and penalty



Solution: Reinforcement Learning



Reinforcement
Learning

Guss &
Bartlett

Introduction

Theory

Algorithms

Questions

- Can we make algorithms which improve with crude reward signals?

**Machine learning without
explicit objective functions**



Reinforcement Learning (RL)



The Core Idea



Reinforcement
Learning

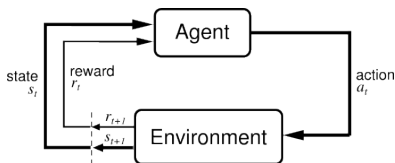
Guss &
Bartlett

Introduction

Theory

Algorithms

Questions



- Models (agents) take action a_t in some environment.
- Environment provides state s_t , reward r_t .
- Models learn to maximize reward r_t , $\forall t$.

Markov Decision Process (MDP)



Reinforcement
Learning

Guss &
Bartlett

Introduction

Theory

Algorithms

Questions

Environment, $E = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \rho, r)$.

- 1 State space, \mathcal{S}
- 2 Action space, \mathcal{A}
- 3 Reward space, \mathcal{R}
- 4 Transition distribution, $\rho(s' \mid s, a)$. Given a previous state s and action a , environment gives s' .
- 5 Reward function $r(s, a) \in \mathcal{R}$.

Markov Property: $\rho(s' \mid s, a)$ depends only on s, a not previous states!

Markov Decision Process (MDP)



Reinforcement
Learning

Guss &
Bartlett

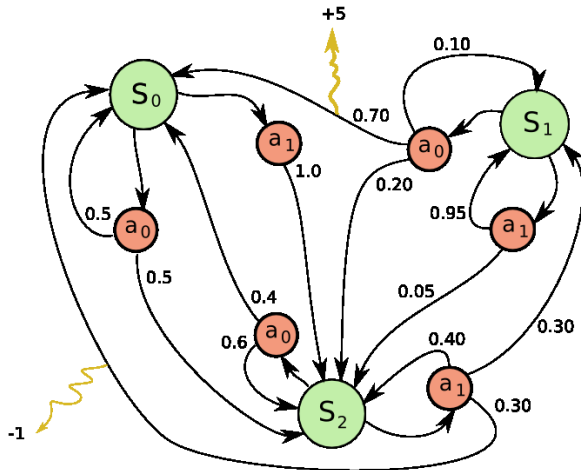
Introduction

Theory

Algorithms

Questions

Example MDP



Pacman as an MDP



Reinforcement
Learning

Guss &
Bartlett

Introduction

Theory

Algorithms

Questions

- $\mathcal{S} = \mathbb{R}^{256 \times 256}$, images as state space.
- $\mathcal{A} = \{\uparrow, \downarrow, \rightarrow, \leftarrow\}$, joystick as action space.
- $r(s_t, a_t)$ = change in score.
- $\rho(s_{t+1} \mid s_t, a_t)$ = next frame of game after joystick action a_t .



Two different types of agents

- Deterministic policy $a = \pi(s)$ acts in E .
- Stochastic policy $a \sim \pi(a|s)$ gives a probability distribution over actions.

Policy Trajectories

$$s_1 \xrightarrow{\pi} a_1 \xrightarrow{\rho, r} s_2, r_2 \xrightarrow{\pi} a_2 \xrightarrow{\rho, r} \dots$$

The **state value** is a function of a given state for an agent π defined as

$$V^\pi(s_t) = \mathbb{E} \left[\sum_{n=t+1}^{\infty} \gamma^n r(s_n, \pi(s_n)) \right]$$

- 1 γ is the discount factor
- 2 $\pi(s_n)$ is the action the agent π makes after seeing state s_n .
- 3 $r(s_n, \pi(s_n))$ is the reward the agent gets from taking that action.

The **state-action value** for an agent π is defined such that

$$Q^{\pi}(s_t, a_t) = \mathbb{E} \left[\underbrace{r(s_t, a_t)}_{\text{reward for } a_t} + V^{\pi}(s_t) \right]$$

- Given some state s_t , the *best* agent, π^* is one that take action

$$a_t = \operatorname{argmax}_a Q(s_t, a).$$

- **Policy Optimization:** maximize the expected reward with respect to a policy π ;

$$\pi^* = \operatorname{argmax}_{\pi} \mathbb{E} \left[\sum_{t=0}^{\infty} r_t \right]$$

- **Policy Evaluation:** Given some fixed policy π compute expected return.
 - Computing Q^{π} , V^{π} , and other expectations on policy rollout.
 - Lets us perform policy optimization!

Reinforcement
Learning

Guss &
Bartlett

Introduction

Theory

Algorithms

Questions

Algorithms

Behavioral Cloning: Supervised learning in MDPs using and expert agent expert π^* !

Behavioral Cloning: Supervised learning in MDPs using and expert agent expert π^* !

Given expert examples $\mathcal{D} = (s_t, a_t = \pi^*(s_t))$ and a model π_θ find θ^* st

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(a_t, \pi_\theta(s_t)).$$

where \mathcal{L} is some loss function.

Behavioral Cloning: Supervised learning in MDPs using and expert agent expert π^* !

Given expert examples $\mathcal{D} = (s_t, a_t = \pi^*(s_t))$ and a model π_θ find θ^* st

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(a_t, \pi_\theta(s_t)).$$

where \mathcal{L} is some loss function.

- Show, don't tell!

Behavioral Cloning: Supervised learning in MDPs using and expert agent expert π^* !

Given expert examples $\mathcal{D} = (s_t, a_t = \pi^*(s_t))$ and a model π_θ find θ^* st

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(a_t, \pi_\theta(s_t)).$$

where \mathcal{L} is some loss function.

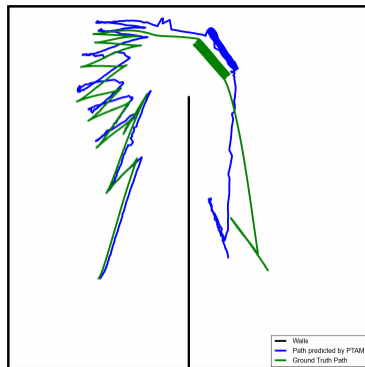
- Show, don't tell!
- No complicated machinery, just standard ML.

Issue: Compounding Error

Given some irreducible error

$$\epsilon = 0.001$$

- $\mathcal{L}(a_0, \pi_\theta(s_0)) = \epsilon$
- $\mathcal{L}(a_1, \pi_\theta(s_1)) = 2\epsilon$
- $\mathcal{L}(a_2, \pi_\theta(s_2)) = 3\epsilon$
- $\mathcal{L}(a_3, \pi_\theta(s_3)) = 4\epsilon$
- $\mathcal{L}(a_4, \pi_\theta(s_4)) = 5\epsilon$



Issue: Distribution Mismatch

- States expert dataset \mathcal{D} generated by π^* have different distribution than those generated by π_θ .
⇒ No self correction.



Issue: Distribution Mismatch

- States expert dataset \mathcal{D} generated by π^* have different distribution than those generated by π_θ .
 \implies No self correction.

Solution: DAgger.

- Do BC on \mathcal{D} and generate E_0 states generated by π_θ .
- Label E_0 with expert level actions and add to \mathcal{D} .



The action-value function (simplified).

- 1 The future expected reward of an agent π is

$$Q^\pi(s_t, a_t) = \underbrace{r(s_t, a_t)}_{\text{reward for } a_t} + \sum_{n=t+1}^{\infty} \gamma^n r(s_n, \pi(s_n))$$

- 2 The Bellman equation gives us

$$Q^\pi(s_t, a_t) = r_t + \gamma Q^\pi(s_{t+1}, \pi(s_{t+1}))$$

..

- 3 Given some state s_t , the **best** agent, π^* is one that take action

$$a_t = \arg \max_a Q(s_t, a).$$

TODO: DO THESE SLIDES The action-value function (simplified).

- 1 The Q function for π^* is

$$Q^*(s_t, a_t) = r_t + \gamma \arg \max_a Q^\pi(s_{t+1}, a).$$

- 2 We can *approximate* this with deep learning!

- 1 Make a neural network $\mathcal{N} : \mathcal{S} \rightarrow \mathbb{R}^n$ which predicts the future reward of taking each possible action

$$\mathcal{N}(s_t) = \begin{pmatrix} Q^*(s_t, a_1) \\ Q^*(s_t, a_2) \\ \vdots \\ Q^*(s_t, a_n) \end{pmatrix}$$

Q-Learning (State-action Value Iteration)



Reinforcement
Learning

Guss &
Bartlett

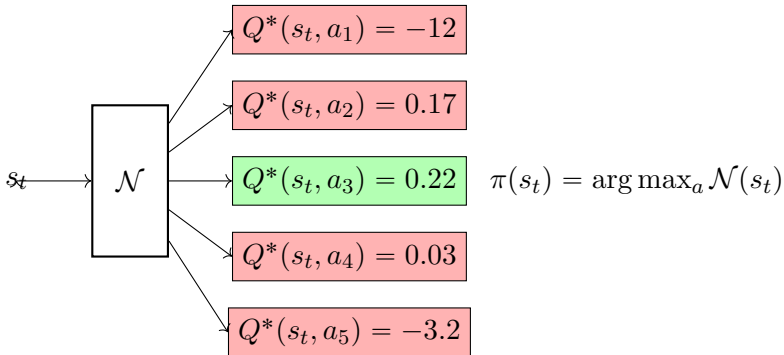
Introduction

Theory

Algorithms

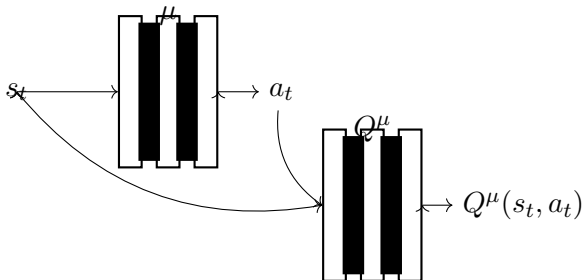
Questions

Deep Q-Learning



Deep Deterministic Policy Gradient

- 1 Actor neural network $\mu : \mathcal{S} \rightarrow \mathcal{A}$
- 2 Critic network $Q^\mu : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$
- 3 Performance of μ is $Q^\mu(s_t, \mu(s_t))$. **Maximize performance!** $\nabla_W Q^\mu(s_t, a_t) = \nabla_a Q^\mu(s_t, a) \cdot \nabla_W \mu(s_t)$



Reinforcement
Learning

Guss &
Bartlett

Introduction

Theory

Algorithms

Questions

Questions?