CS 760: Machine Learning
Reinforcement Learning II

Kirthi Kandasamy

University of Wisconsin-Madison

April 17, 2023

Announcements

- HW7 (last HW) is out, due on May 1.

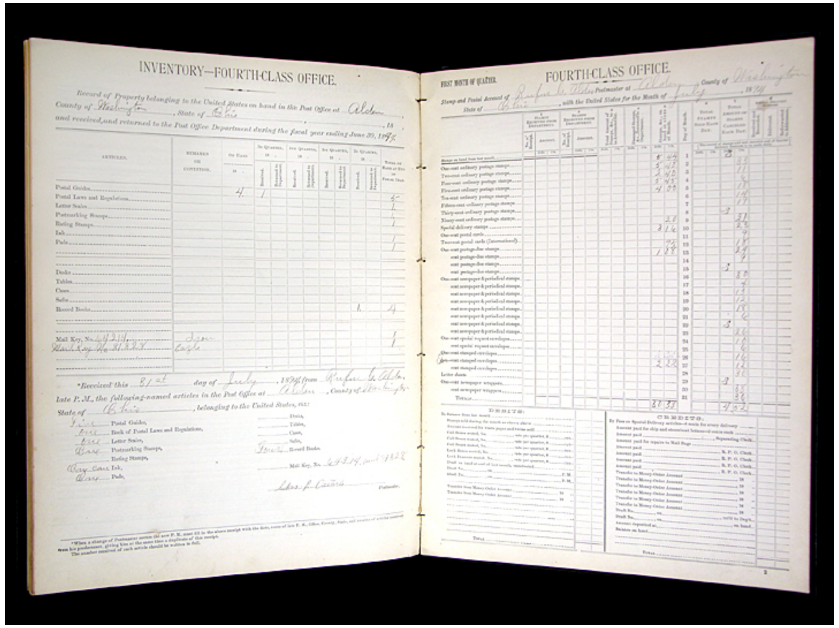
Outline

- **Function Approximation**
  - Value & Q-function approximations, linear, nonlinear
- **Policy-based RL**
  - Policy gradient, policy gradient theorem, REINFORCE algorithm

Outline

- **Function Approximation**
  - Value & Q-function approximations
- **Policy-based RL**
  - Policy gradient, policy gradient theorem, REINFORCE algorithm

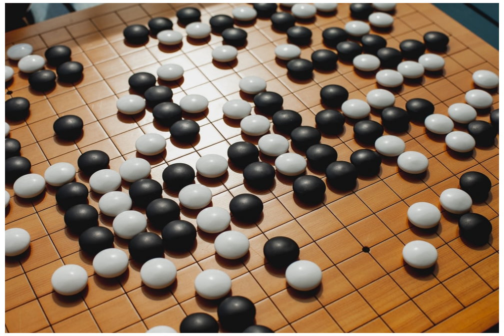
Beyond Tables



So far:

- Represent everything with a table
  - Value function **V**: table size $|S| \times 1$
  - **Q** function: table size $|S| \times |A|$

- Too big to store in memory for many tasks
  - Backgammon: 10^{20} states. Go: 3^{361} states
  - Need some other approach



Beyond Tables: Function Approximation

Both V and Q are functions…

- Can approximate them with models, ie, neural networks
- So we write $V^{\pi}(s) \approx \hat{V}_{\theta}(s)$

- New goal: find the weights θ
- Loss function:

$$J(\theta) = \mathbb{E}_{\pi}[(V^{\pi}(s) - \hat{V}_{\theta}(s))^2]$$

State **Representations** & **Models**

How do we represent a state?

- As usual, feature vectors, i.e.,

$$x(s) = \begin{bmatrix} x_1(s) \\ x_2(s) \\ \vdots \\ x_d(s) \end{bmatrix}$$

- E.g Linear models:

$$\hat{V}_\theta(s) = x(s)^T \theta$$

Linear VFA With an Oracle

$$J(\theta) = \mathbb{E}_{\pi}[(V^{\pi}(s) - \hat{V}_{\theta}(s))^2]$$

- SGD update (change) to current estimate is is

$$\alpha[(V^{\pi}(s) - \hat{V}_{\theta}(s))\nabla_{\theta}\hat{V}_{\theta}(s)]$$

- For linear models, we get

$$\underset{\substack{\uparrow \\ \text{Step Size}}}{\alpha}\underbrace{(V^{\pi}(s) - \hat{V}_{\theta}(s))}_{\text{Prediction Error}}\underset{\substack{\uparrow \\ \text{Feature Value}}}{x(s)}$$

What if We **Don’t Have** an Oracle?

Use Monte-Carlo!

- We won’t know $V^{\pi}(s_t)$
- Run the policy to obtain rewards: $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

- Can just run episodes and estimate, ie, get some noisy estimates. Data:

$$(s_1, G_1), (s_2, G_2), \ldots, (s_T, G_T)$$

Q-Function Approximation

Similar idea for Q-function

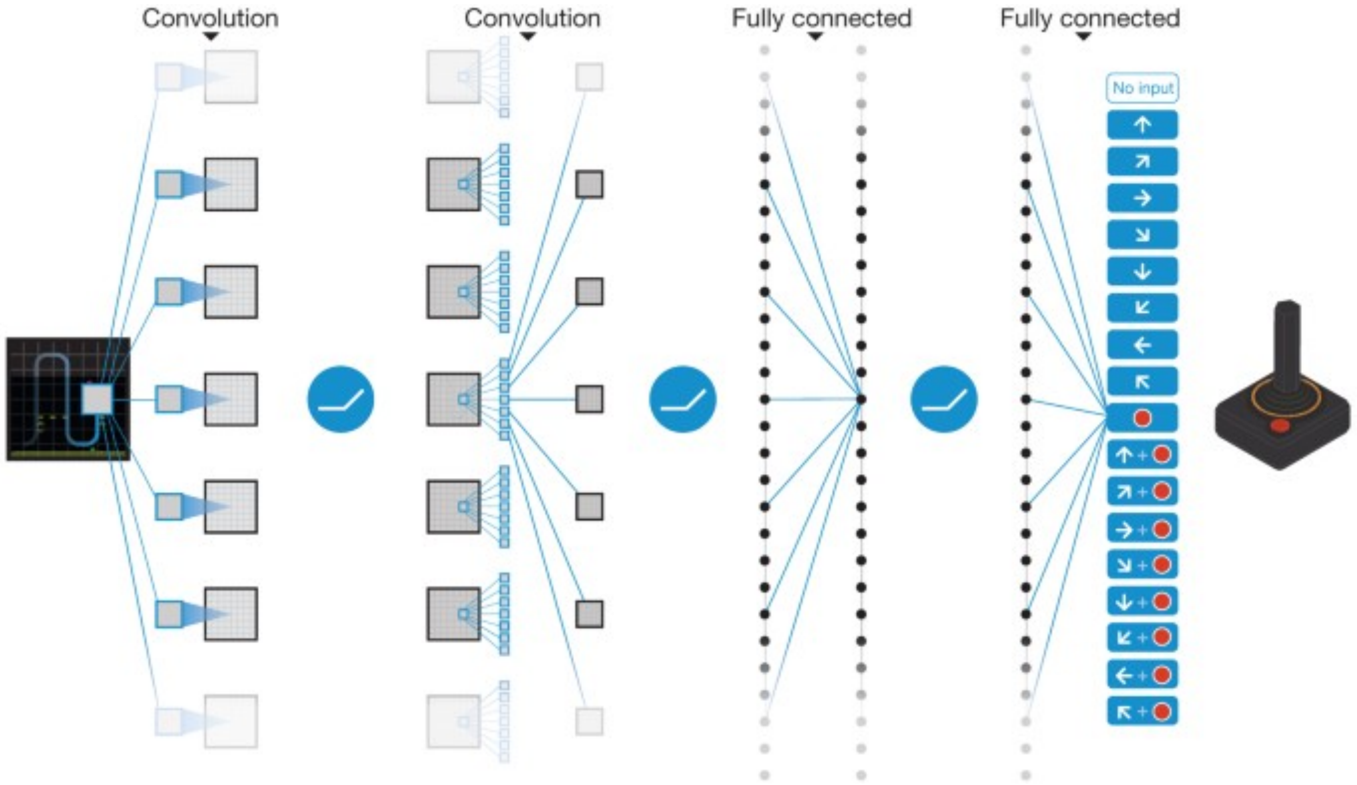
$$Q^{\pi}(s,a) \approx \hat{Q}_{\theta}(s,a)$$

Representation: use both states and values

- Can still use linear models
- Note: quite popular to use **deep models**

Q-Function Approximation: Deep Models

- Note: quite popular to use **deep models**
  - E.g., CNNs if the states are images (e.g: video games)



Mnih et al, "Human-level control through deep reinforcement learning"

Outline

- **Function Approximation**
  - Value & Q-function approximations, linear, nonlinear
- **Policy-based RL**
  - Policy gradient, policy gradient theorem, REINFORCE algorithm

Policy-Based RL

So far, we either approximated V or Q

- Then use these to extract the optimal policy

But we can directly model and update the policy

- Note: so far our policies were deterministic, now we'll allow a distribution over actions, ie,

$$\pi_\theta(s, a) = P_\theta(a|s)$$

Policy Gradient

Use the same idea. We'll define an objective $J(\theta)$

- And then can get gradients:

$$\nabla_\theta \pi_\theta(s,a) = \pi_\theta(s,a) \nabla_\theta \underbrace{\log \pi_\theta(s,a)}_{\text{Score Function}}$$

Policy Gradient

Set our objective to be

$$J(\theta) = \sum_s \underbrace{P(s|\pi_\theta)}_{\text{Stationary distribution}} \sum_a \pi_\theta(s,a) Q^\pi(s,a)$$

- Compute the gradient via the **policy gradient theorem**

$$\nabla_\theta J(\theta) = \sum_s P(s|\pi_\theta) \sum_a \nabla_\theta \pi_\theta(s,a) Q^\pi(s,a)$$

REINFORCE Algorithm

So, to learn a policy, we can run SGD (actually ascent)

- Compute gradients via policy gradient theorem

$$\nabla_\theta J(\theta) = \sum_s P(s|\pi_\theta) \sum_a \nabla_\theta \pi_\theta(s,a) Q^\pi(s,a)$$

  - Just need $Q^\pi(s,a)$ estimates.
  - How? Monte-Carlo again: Use G_t for our estimates.

Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, Pedro Domingos, Jerry Zhu, Yingyu Liang, Volodymyr Kuleshov, David Silver, Emma Brunskill