

Stanford·CS234 | Reinforcement Learning (2019)

CS234 (2019)· 课程资料包 @ShowMeAI



视频
中英双语字幕



课件
一键打包下载



笔记
官方笔记翻译



代码
作业项目解析



视频·B站【扫码或点击链接】

<https://www.bilibili.com/video/BV1H64y1x7GH>



课件 & 代码·博客【扫码或点击链接】

<http://blog.showmeai.tech/cs234>

斯坦福

reinforcement learning
马尔可夫决策过程

DQN

强化学习

值函数方法

policy
gradient

Q-learning

梯度策略

Model free
蒙特卡洛搜索树

Awesome AI Courses Notes Cheatsheets 是 [ShowMeAI](#) 资料库的分支系列，覆盖最具知名度的 **TOP20+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程资料包页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets · 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094



微信公众号

资料下载方式 2: 扫码点击底部菜单栏

称为 AI 内容创作者? 回复 [添砖加瓦]

Lecture 8: Policy Gradient I ¹

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2019

- Additional reading: Sutton and Barto 2018 Chp. 13

¹With many slides from or derived from David Silver and John Schulman and Pieter Abbeel

Last Time: We want RL Algorithms that Perform

- Optimization
- Delayed consequences
- Exploration
- Generalization
- And do it statistically and computationally efficiently

Class Structure

- Last time: Imitation Learning
- **This time: Policy Search**
- Next time: Policy Search Cont.

Table of Contents

- 1 Introduction
- 2 Policy Gradient
- 3 Score Function and Policy Gradient Theorem
- 4 Policy Gradient Algorithms and Reducing Variance

Policy-Based Reinforcement Learning

- In the last lecture we approximated the value or action-value function using parameters θ , or we often used ω

$$V_{\theta}(s) \approx V^{\pi}(s)$$

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

- A policy was generated directly from the value function
 - e.g. using ϵ -greedy

$$\pi: S \rightarrow A$$

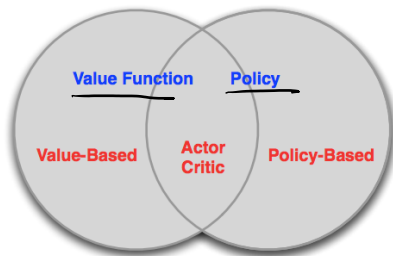
- In this lecture we will directly parametrize the policy

$$\pi_{\theta}(s, a) = \mathbb{P}[a|s; \overset{\omega}{\theta}]$$

- Goal is to find a policy π with the highest value function V^{π}
- We will focus again on model-free reinforcement learning

Value-Based and Policy-Based RL

- Value Based
 - Learnt Value Function
 - Implicit policy (e.g. ϵ -greedy)
- Policy Based
 - No Value Function
 - Learnt Policy
- Actor-Critic π Q/V
 - Learnt Value Function
 - Learnt Policy



Advantages of Policy-Based RL

Advantages:

- Better convergence properties
- Effective in high-dimensional or continuous action spaces
- Can learn stochastic policies

↪ computation can matter

Disadvantages:

- Typically converge to a local rather than global optimum
- Evaluating a policy is typically inefficient and high variance

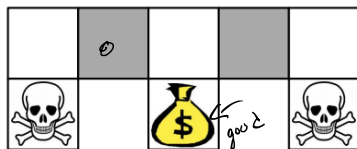
why do we want this?
tabular MDP $\exists \pi$ which is
deterministic
& optimal

Example: Rock-Paper-Scissors



- Two-player game of rock-paper-scissors
 - Scissors beats paper
 - Rock beats scissors
 - Paper beats rock
- Consider policies for iterated rock-paper-scissors
 - A deterministic policy is easily exploited
 - A uniform random policy is optimal (i.e. Nash equilibrium)

Example: Aliased Gridworld (1)



- The agent cannot differentiate the grey states
- Consider features of the following form (for all N , \underline{E} , \underline{S} , \underline{W})

$$\phi(s, a) = \mathbb{1}(\text{wall to } N, a = \text{move } E)$$

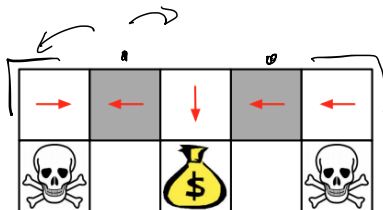
- Compare value-based RL, using an approximate value function

$$Q_{\theta}(s, a) = f(\phi(s, a); \theta)$$

- To policy-based RL, using a parametrized policy

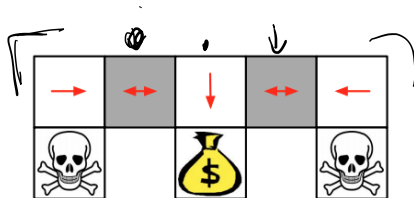
$$\pi_{\theta}(s, a) = g(\phi(s, a); \theta)$$

Example: Aliased Gridworld (2)



- Under aliasing, an optimal **deterministic** policy will either
 - move W in both grey states (shown by red arrows)
 - move E in both grey states
- Either way, it can get stuck and never reach the money
- Value-based RL learns a near-deterministic policy
 - e.g. greedy or ϵ -greedy
- So it will traverse the corridor for a long time

Example: Aliased Gridworld (3)



not Markov
partially
observable

- An optimal **stochastic** policy will randomly move E or W in grey states

$$\pi_{\theta}(\text{wall to N and S, move E}) = 0.5$$

$$\pi_{\theta}(\text{wall to N and S, move W}) = 0.5$$

- It will reach the goal state in a few steps with high probability
- Policy-based RL can learn the optimal stochastic policy

Policy Objective Functions

- Goal: given a policy $\pi_\theta(s, a)$ with parameters θ , find best θ
- But how do we measure the quality for a policy π_θ ?
- In episodic environments we can use the **start** value of the policy

H steps
terminal state

$$J_1(\theta) = V^{\pi_\theta}(s_1)$$

- In continuing environments we can use the **average value**
online

$$J_{avV}(\theta) = \sum_s \underbrace{d^{\pi_\theta}(s)} V^{\pi_\theta}(s)$$

- where $d^{\pi_\theta}(s)$ is the **stationary distribution** of Markov chain for π_θ .
- Or the **average reward per time-step**

$$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s, a) R(a, s) \Big]$$

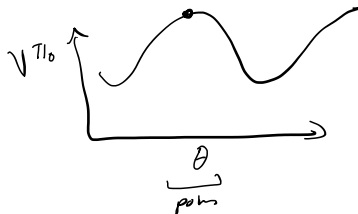
- For simplicity, today will mostly discuss the episodic case, but can easily extend to the continuing / infinite horizon case

Policy optimization

- Policy based reinforcement learning is an **optimization** problem
- Find policy parameters θ that maximize V^{π_θ}

Policy optimization

- Policy based reinforcement learning is an **optimization** problem
- Find policy parameters θ that maximize V^{π_θ}
- Can use gradient free optimization
 - Hill climbing
 - Simplex / amoeba / Nelder Mead
 - Genetic algorithms
 - Cross-Entropy method (CEM)
 - Covariance Matrix Adaptation (CMA)



Human-in-the-Loop Exoskeleton Optimization (Zhang et al. Science 2017)

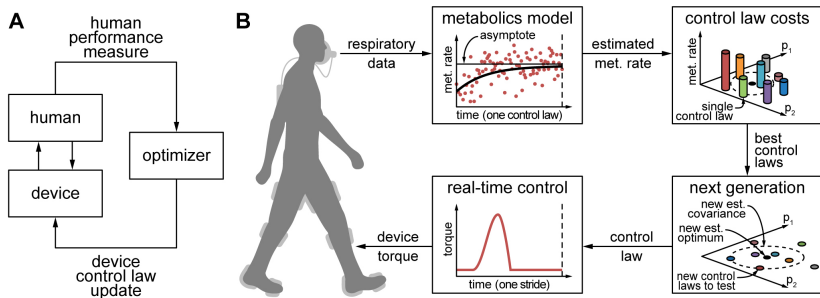


Figure: Zhang et al. Science 2017

- Optimization was done using CMA-ES, variation of covariance matrix evaluation

Gradient Free Policy Optimization

- Can often work embarrassingly well: "discovered that evolution strategies (ES), an optimization technique that's been known for decades, rivals the performance of standard reinforcement learning (RL) techniques on modern RL benchmarks (e.g. Atari/MuJoCo)" (<https://blog.openai.com/evolution-strategies/>)

Gradient Free Policy Optimization

- Often a great simple baseline to try
- Benefits
 - Can work with any policy parameterizations, including non-differentiable
 - Frequently very easy to parallelize
- Limitations
 - Typically not very sample efficient because it ignores temporal structure

Policy optimization

- Policy based reinforcement learning is an **optimization** problem
- Find policy parameters θ that maximize V^{π_θ}
- Can use gradient free optimization:
- Greater efficiency often possible using gradient
 - Gradient descent
 - Conjugate gradient
 - Quasi-newton
- We focus on gradient descent, many extensions possible
- And on methods that exploit sequential structure

Table of Contents

1 Introduction

2 Policy Gradient

3 Score Function and Policy Gradient Theorem

4 Policy Gradient Algorithms and Reducing Variance

- Define $V(\theta) = V^{\pi_\theta}$ to make explicit the dependence of the value on the policy parameters
- Assume episodic MDPs (easy to extend to related objectives, like average reward)

Policy Gradient

- Define $V(\theta) = V^{\pi_\theta}$ to make explicit the dependence of the value on the policy parameters
- Assume episodic MDPs
- Policy gradient algorithms search for a *local* maximum in $V(\theta)$ by ascending the gradient of the policy, w.r.t parameters θ

$$\Delta\theta = \alpha \nabla_{\theta} V(\theta)$$

\uparrow
warning!

very similar to
Q/V based
optimization

- Where $\nabla_{\theta} V(\theta)$ is the **policy gradient**

$$\nabla_{\theta} V(\theta) = \begin{pmatrix} \frac{\partial V(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial V(\theta)}{\partial \theta_n} \end{pmatrix}$$

- and α is a step-size parameter

Computing Gradients by Finite Differences

- To evaluate policy gradient of $\pi_{\theta}(s, a)$
- For each dimension $k \in [1, n]$
 - Estimate k th partial derivative of objective function w.r.t. θ
 - By perturbing θ by small amount ϵ in k th dimension

$$\frac{\partial V(\theta)}{\partial \theta_k} \approx \frac{V(\theta + \epsilon u_k) - V(\theta)}{\epsilon}$$

where u_k is a unit vector with 1 in k th component, 0 elsewhere.

Computing Gradients by Finite Differences

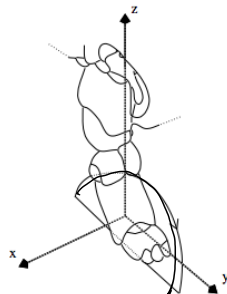
- To evaluate policy gradient of $\pi_{\theta}(s, a)$
- For each dimension $k \in [1, n]$
 - Estimate k th partial derivative of objective function w.r.t. θ
 - By perturbing θ by small amount ϵ in k th dimension

$$\frac{\partial V(\theta)}{\partial \theta_k} \approx \frac{V(\theta + \epsilon u_k) - V(\theta)}{\epsilon}$$

where u_k is a unit vector with 1 in k th component, 0 elsewhere.

- Uses n evaluations to compute policy gradient in n dimensions
- Simple, noisy, inefficient - but sometimes effective
- Works for arbitrary policies, even if policy is not differentiable

Training AIBO to Walk by Finite Difference Policy Gradient¹



- Goal: learn a fast AIBO walk (useful for Robocup)
- Adapt these parameters by finite difference policy gradient
- Evaluate performance of policy by field traversal time

¹Kohl and Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. ICRA 2004. <http://www.cs.utexas.edu/ai-lab/pubs/icra04.pdf>

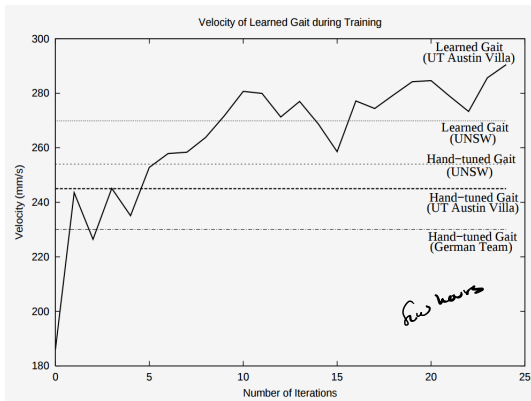
AIBO Policy Parameterization

- AIBO walk policy is open-loop policy
- No state, choosing set of action parameters that define an ellipse
- Specified by 12 continuous parameters (elliptical loci)
 - The front locus (3 parameters: height, x-pos., y-pos.)
 - The rear locus (3 parameters)
 - Locus length
 - Locus skew multiplier in the x-y plane (for turning)
 - The height of the front of the body
 - The height of the rear of the body
 - The time each foot takes to move through its locus
 - The fraction of time each foot spends on the ground
- New policies: for each parameter, randomly add (ϵ , 0, or $-\epsilon$)

AIBO Policy Experiments

- "All of the policy evaluations took place on actual robots... only human intervention required during an experiment involved replacing discharged batteries ... about once an hour."
- Ran on 3 Aibos at once
- Evaluated 15 policies per iteration.
- Each policy evaluated 3 times (to reduce noise) and averaged
- Each iteration took 7.5 minutes
- Used $\eta = 2$ (learning rate for their finite difference approach)

Training AIBO to Walk by Finite Difference Policy Gradient Results



- Authors discuss that performance is likely impacted by: initial starting policy parameters, ϵ (how much policies are perturbed), η (how much to change policy), as well as policy parameterization

AIBO Walk Policies

- [link](#)

Table of Contents

- 1 Introduction
- 2 Policy Gradient
- 3 Score Function and Policy Gradient Theorem**
- 4 Policy Gradient Algorithms and Reducing Variance

Computing the gradient analytically

only converge local optima

- We now compute the policy gradient *analytically*
- Assume policy π_θ is differentiable whenever it is non-zero
- and we know the gradient $\nabla_\theta \pi_\theta(s, a)$

compute b/c

Likelihood Ratio Policies

- Denote a state-action trajectory as


$$\tau = (s_0, a_0, r_0, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$$

- Use $R(\tau) = \sum_{t=0}^T R(s_t, a_t)$ to be the sum of rewards for a trajectory τ

↙ terminate

Likelihood Ratio Policies

- Denote a state-action trajectory as
$$\tau = (s_0, a_0, r_0, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$$
- Use $R(\tau) = \sum_{t=0}^T R(s_t, a_t)$ to be the sum of rewards for a trajectory τ
- Policy value is

$$V(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^T R(s_t, a_t); \pi_\theta \right] = \sum_{\tau} P(\tau; \theta) R(\tau)$$


- where $P(\tau; \theta)$ is used to denote the probability over trajectories when executing policy $\pi(\theta)$
- In this new notation, our goal is to find the policy parameters θ :

$$\arg \max_{\theta} V(\theta) = \arg \max_{\theta} \underbrace{\sum_{\tau} P(\tau; \theta)}_{\text{probability}} \underbrace{R(\tau)}_{\text{reward}}$$

Likelihood Ratio Policy Gradient

- Goal is to find the policy parameters θ :

$$\arg \max_{\theta} V(\theta) = \arg \max_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)$$

- Take the gradient with respect to θ :

$$\nabla_{\theta} V(\theta) = \nabla_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)$$

$$= \sum_{\tau} R(\tau) \nabla_{\theta} P(\tau; \theta)$$

$$= \sum_{\tau} R(\tau) \frac{P(\tau; \theta)}{P(\tau; \theta)} \nabla_{\theta} P(\tau; \theta)$$

$$= \sum_{\tau} R(\tau) P(\tau; \theta) \nabla_{\theta} \log P(\tau; \theta)$$

$$\begin{aligned} & \nabla_{\theta} \log P(\tau; \theta) \\ &= \frac{1}{P(\tau; \theta)} \nabla_{\theta} P(\tau; \theta) \end{aligned}$$

Likelihood Ratio Policy Gradient

- Goal is to find the policy parameters θ :

$$\arg \max_{\theta} V(\theta) = \arg \max_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)$$

- Take the gradient with respect to θ :

$$\begin{aligned} \nabla_{\theta} V(\theta) &= \nabla_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau) \\ &= \sum_{\tau} \nabla_{\theta} P(\tau; \theta) R(\tau) \\ &= \sum_{\tau} \frac{P(\tau; \theta)}{P(\tau; \theta)} \nabla_{\theta} P(\tau; \theta) R(\tau) \\ &= \sum_{\tau} P(\tau; \theta) R(\tau) \underbrace{\frac{\nabla_{\theta} P(\tau; \theta)}{P(\tau; \theta)}}_{\text{likelihood ratio}} \\ &= \sum_{\tau} P(\tau; \theta) R(\tau) \nabla_{\theta} \log P(\tau; \theta) \end{aligned}$$

Likelihood Ratio Policy Gradient

- Goal is to find the policy parameters θ :

$$\arg \max_{\theta} V(\theta) = \arg \max_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)$$

- Take the gradient with respect to θ :

$$\nabla_{\theta} V(\theta) = \sum_{\tau} \underbrace{P(\tau; \theta) R(\tau)} \nabla_{\theta} \log P(\tau; \theta)$$

- Approximate with empirical estimate for m sample paths under policy π_{θ} :

$$\nabla_{\theta} V(\theta) \approx \underbrace{\hat{g} = (1/m) \sum_{i=1}^m R(\tau^{(i)}) \nabla_{\theta} \log P(\tau^{(i)}; \theta)}$$

Score Function Gradient Estimator: Intuition

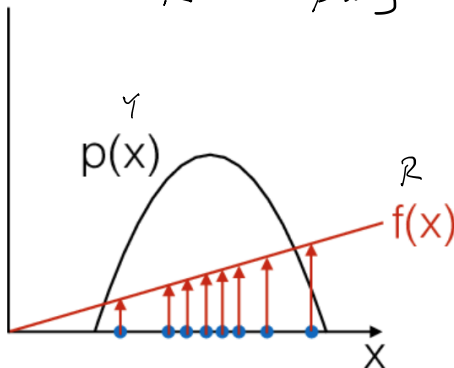
- Consider generic form of $R(\tau^{(i)})\nabla_{\theta} \log P(\tau^{(i)}; \theta)$:
 $\hat{g}_i = f(x_i)\nabla_{\theta} \log p(x_i|\theta)$
- $f(x)$ measures how good the sample x is.
- Moving in the direction \hat{g}_i pushes up the logprob of the sample, in proportion to how good it is
- *Valid even if $f(x)$ is discontinuous, and unknown, or sample space (containing x) is a discrete set*



Score Function Gradient Estimator: Intuition

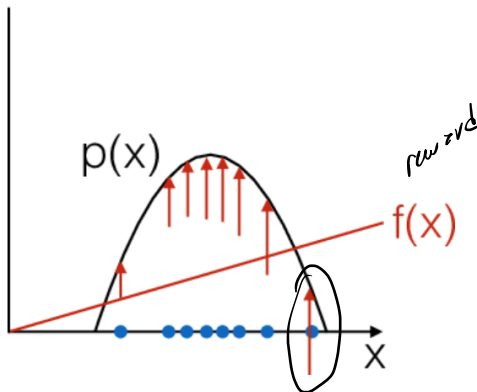
$$\hat{g}_i = f(x_i) \nabla_{\theta} \log p(x_i | \theta)$$

\mathcal{R} *policy*



Score Function Gradient Estimator: Intuition

$$\hat{g}_i = f(x_i) \nabla_{\theta} \log p(x_i | \theta)$$



Decomposing the Trajectories Into States and Actions

- Approximate with empirical estimate for m sample paths under policy π_θ :

$$\begin{aligned}
 \nabla_\theta V(\theta) &\approx \hat{g} = (1/m) \sum_{i=1}^m R(\tau^{(i)}) \nabla_\theta \log P(\tau^{(i)}) \quad \text{unknown} \\
 \downarrow \\
 \nabla_\theta \log P(\tau^{(i)}; \theta) &= \nabla_\theta \log \left(\underbrace{\mu(s_0)}_{\text{initial state}} \prod_{j=0}^{\tau-1} p(s_{j+1}^i | s_j^i, a_j^i) \pi_\theta(a_j^i | s_j^i) \right) \\
 &= \underbrace{\nabla_\theta \log \mu(s_0)}_{\textcircled{1} \rightarrow 0} + \underbrace{\sum_{j=0}^{\tau-1} \nabla_\theta \log p(s_{j+1}^i | s_j^i, a_j^i)}_{\textcircled{2} \rightarrow 0} + \underbrace{\sum_{j=0}^{\tau-1} \nabla_\theta \log \pi_\theta(a_j^i | s_j^i)}_{\textcircled{3}}
 \end{aligned}$$

Decomposing the Trajectories Into States and Actions

- Approximate with empirical estimate for m sample paths under policy π_θ :

$$\nabla_\theta V(\theta) \approx \hat{g} = (1/m) \sum_{i=1}^m R(\tau^{(i)}) \nabla_\theta \log P(\tau^{(i)})$$

$$\begin{aligned} \nabla_\theta \log P(\tau^{(i)}; \theta) &= \nabla_\theta \log \left[\underbrace{\mu(s_0)}_{\text{Initial state distrib.}} \prod_{t=0}^{T-1} \underbrace{\pi_\theta(a_t|s_t)}_{\text{policy}} \underbrace{P(s_{t+1}|s_t, a_t)}_{\text{dynamics model}} \right] \\ &= \nabla_\theta \left[\log \mu(s_0) + \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t) + \log P(s_{t+1}|s_t, a_t) \right] \\ &= \sum_{t=0}^{T-1} \underbrace{\nabla_\theta \log \pi_\theta(a_t|s_t)}_{\text{no dynamics model required!}} \end{aligned}$$

↙ have to evaluate if

Score Function

- Define **score function** as $\underbrace{\nabla_{\theta} \log \pi_{\theta}(s, a)}$

Likelihood Ratio / Score Function Policy Gradient

- Putting this together
- Goal is to find the policy parameters θ :

$$\arg \max_{\theta} V(\theta) = \arg \max_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)$$

- Approximate with empirical estimate for m sample paths under policy π_{θ} using score function:

$$\begin{aligned} \nabla_{\theta} V(\theta) &\approx \hat{g} = (1/m) \sum_{i=1}^m R(\tau^{(i)}) \nabla_{\theta} \log P(\tau^{(i)}; \theta) \\ &= (1/m) \sum_{i=1}^m R(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) \end{aligned}$$

- Do not need to know dynamics model

Policy Gradient Theorem

- The policy gradient theorem generalizes the likelihood ratio approach

Theorem

*For any differentiable policy $\pi_\theta(s, a)$,
for any of the policy objective function $J = J_1$, (episodic reward), J_{avR} (average reward per time step), or $\frac{1}{1-\gamma} J_{avV}$ (average value),
the policy gradient is*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s, a) Q^{\pi_\theta}(s, a)]$$

Table of Contents

- 1 Introduction
- 2 Policy Gradient
- 3 Score Function and Policy Gradient Theorem
- 4 Policy Gradient Algorithms and Reducing Variance

Likelihood Ratio / Score Function Policy Gradient

$$\nabla_{\theta} V(\theta) \approx (1/m) \sum_{i=1}^m R(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)})$$

- Unbiased but very noisy
- Fixes that can make it practical
 - Temporal structure
 - Baseline
- Next time will discuss some additional tricks

Policy Gradient: Use Temporal Structure

- Previously:

$$\nabla_{\theta} \mathbb{E}_{\tau}[R] = \mathbb{E}_{\tau} \left[\left(\sum_{t=0}^{T-1} r_t \right) \left(\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right) \right]$$

- We can repeat the same argument to derive the gradient estimator for a single reward term $r_{t'}$.

$$\nabla_{\theta} \mathbb{E}[r_{t'}] = \mathbb{E} \left[r_{t'} \sum_{t=0}^{t'} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right]$$

- Summing this formula over t, we obtain

$$\begin{aligned} V(\theta) = \nabla_{\theta} \mathbb{E}[R] &= \mathbb{E} \left[\sum_{t'=0}^{T-1} r_{t'} \sum_{t=0}^{t'} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right] \\ &= \mathbb{E} \left[\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t, s_t) \sum_{t'=t}^{T-1} r_{t'} \right] \end{aligned}$$

Policy Gradient: Use Temporal Structure

- Recall for a particular trajectory $\tau^{(i)}$, $\sum_{t'=t}^{T-1} r_{t'}^{(i)}$ is the return $G_t^{(i)}$

$$\nabla_{\theta} \mathbb{E}[R] \approx (1/m) \sum_{i=1}^m \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t, s_t) G_t^{(i)}$$

Monte-Carlo Policy Gradient (REINFORCE)

- Leverages likelihood ratio / score function and temporal structure

$$\Delta\theta_t = \alpha \nabla_{\theta} \log \pi_{\theta}(s_t, a_t) G_t$$

REINFORCE:

Initialize policy parameters θ arbitrarily

for each episode $\{s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_{\theta}$ **do**

for $t = 1$ to $T - 1$ **do**

$\theta \leftarrow \theta + \alpha \nabla_{\theta} \log \pi_{\theta}(s_t, a_t) G_t$

endfor

endfor

return θ

Differentiable Policy Classes

- Many choices of differentiable policy classes including:
 - Softmax
 - Gaussian
 - Neural networks

- Weight actions using linear combination of features $\phi(s, a)^T \theta$
- Probability of action is proportional to exponentiated weight

$$\pi_{\theta}(s, a) = e^{\phi(s, a)^T \theta} / (\sum_a e^{\phi(s, a)^T \theta})$$

- The score function is

$$\nabla_{\theta} \log \pi_{\theta}(s, a) = \phi(s, a) - \mathbb{E}_{\pi_{\theta}}[\phi(s, \cdot)]$$

Gaussian Policy

- In continuous action spaces, a Gaussian policy is natural
- Mean is a linear combination of state features $\mu(s) = \phi(s)^T \theta$
- Variance may be fixed σ^2 , or can also be parametrised
- Policy is Gaussian $a \sim \mathcal{N}(\mu(s), \sigma^2)$
- The score function is

$$\nabla_{\theta} \log \pi_{\theta}(s, a) = \frac{(a - \mu(s))\phi(s)}{\sigma^2}$$

Likelihood Ratio / Score Function Policy Gradient

$$\nabla_{\theta} V(\theta) \approx (1/m) \sum_{i=1}^m R(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)})$$

- Unbiased but very noisy
- Fixes that can make it practical
 - Temporal structure
 - **Baseline**
- Next time will discuss some additional tricks

Policy Gradient: Introduce Baseline

- Reduce variance by introducing a *baseline* $b(s)$

$$\nabla_{\theta} \mathbb{E}_{\tau}[R] = \mathbb{E}_{\tau} \left[\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi(a_t | s_t, \theta) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right) \right]$$

- For any choice of $b(s)$, gradient estimator is unbiased.
- Near optimal choice is expected return,
 $b(s_t) \approx \mathbb{E}[r_t + r_{t+1} + \dots + r_{T-1}]$
- Interpretation: increase logprob of action a_t proportionally to how much returns $\sum_{t'=t}^{T-1} r_{t'}$ are better than expected

Baseline $b(s)$ Does Not Introduce Bias–Derivation

$$\begin{aligned} & \mathbb{E}_{\tau} [\nabla_{\theta} \log \pi(a_t | s_t, \theta) b(s_t)] \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} \left[\mathbb{E}_{s_{(t+1):T}, a_{t:(T-1)}} [\nabla_{\theta} \log \pi(a_t | s_t, \theta) b(s_t)] \right] \end{aligned}$$

Baseline $b(s)$ Does Not Introduce Bias–Derivation

$$\begin{aligned} & \mathbb{E}_{\tau} [\nabla_{\theta} \log \pi(a_t | s_t, \theta) b(s_t)] \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} \left[\mathbb{E}_{s_{(t+1):T}, a_{t:(T-1)}} [\nabla_{\theta} \log \pi(a_t | s_t, \theta) b(s_t)] \right] \text{ (break up expectation)} \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} \left[b(s_t) \mathbb{E}_{s_{(t+1):T}, a_{t:(T-1)}} [\nabla_{\theta} \log \pi(a_t | s_t, \theta)] \right] \text{ (pull baseline term out)} \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} \left[b(s_t) \mathbb{E}_{a_t} [\nabla_{\theta} \log \pi(a_t | s_t, \theta)] \right] \text{ (remove irrelevant variables)} \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} \left[b(s_t) \sum_a \pi_{\theta}(a_t | s_t) \frac{\nabla_{\theta} \pi(a_t | s_t, \theta)}{\pi_{\theta}(a_t | s_t)} \right] \text{ (likelihood ratio)} \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} \left[b(s_t) \sum_a \nabla_{\theta} \pi(a_t | s_t, \theta) \right] \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} \left[b(s_t) \nabla_{\theta} \sum_a \pi(a_t | s_t, \theta) \right] \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} [b(s_t) \nabla_{\theta} 1] \\ &= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} [b(s_t) \cdot 0] = 0 \end{aligned}$$

"Vanilla" Policy Gradient Algorithm

Initialize policy parameter θ , baseline b

for iteration=1, 2, \dots **do**

Collect a set of trajectories by executing the current policy

At each timestep in each trajectory, compute

the *return* $R_t = \sum_{t'=t}^{T-1} r_{t'}$, and

the *advantage estimate* $\hat{A}_t = R_t - b(s_t)$.

Re-fit the baseline, by minimizing $\|b(s_t) - R_t\|^2$,
summed over all trajectories and timesteps.

Update the policy, using a policy gradient estimate \hat{g} ,
which is a sum of terms $\nabla_{\theta} \log \pi(a_t|s_t; \theta) \hat{A}_t$.

(Plug \hat{g} into SGD or ADAM)

endfor

Practical Implementation with Autodiff

- Usual formula $\sum_t \nabla_{\theta} \log \pi(a_t|s_t; \theta) \hat{A}_t$ is inefficient—want to batch data
- Define "surrogate" function using data from current batch

$$L(\theta) = \sum_t \log \pi(a_t|s_t; \theta) \hat{A}_t$$

- Then policy gradient estimator $\hat{g} = \nabla_{\theta} L(\theta)$
- Can also include value function fit error

$$L(\theta) = \sum_t \left(\log \pi(z_t|s_t; \theta) \hat{A}_t - \|V(s_t) - \hat{R}_t\|^2 \right)$$

Value Functions

- Recall Q-function / state-action-value function:

$$Q^{\pi,\gamma}(s, a) = \mathbb{E}_{\pi} [r_0 + \gamma r_1 + \gamma^2 r_2 \cdots | s_0 = s, a_0 = a]$$

- State-value function can serve as a great baseline

$$\begin{aligned} V^{\pi,\gamma}(s) &= \mathbb{E}_{\pi} [r_0 + \gamma r_1 + \gamma^2 r_2 \cdots | s_0 = s] \\ &= \mathbb{E}_{a \sim \pi} [Q^{\pi,\gamma}(s, a)] \end{aligned}$$

- Advantage function: Combining Q with baseline V

$$A^{\pi,\gamma}(s, a) = Q^{\pi,\gamma}(s, a) - V^{\pi,\gamma}(s)$$

N-step estimators

- Can also consider blending between TD and MC estimators for the target to substitute for the true state-action value function.

$$\hat{R}_t^{(1)} = r_t + \gamma V(s_{t+1})$$

$$\hat{R}_t^{(2)} = r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) \quad \dots$$

$$\hat{R}_t^{(\text{inf})} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+1} + \dots$$

- If subtract baselines from the above, get advantage estimators

$$\hat{A}_t^{(1)} = r_t + \gamma V(s_{t+1}) - V(s_t)$$

$$\hat{A}_t^{(2)} = r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) - V(s_t)$$

$$\hat{A}_t^{(\text{inf})} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+1} + \dots - V(s_t)$$

- $\hat{A}_t^{(a)}$ has low variance & high bias. $\hat{A}_t^{(\infty)}$ high variance but low bias. (Why? Like which model-free policy estimation techniques?)
- Using intermediate k (say, 20) can give an intermediate amount of bias and variance.

Learning to Walk in 20 Minutes

Russ Tedrake

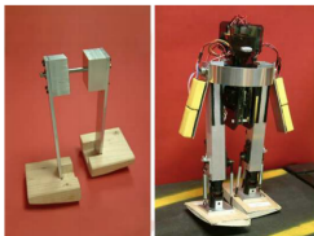
Brain & Cognitive Sciences
Center for Bits and Atoms
Massachusetts Inst. of Technology
Cambridge, MA 02139
russt@csail.mit.edu

Teresa Weirui Zhang

Mechanical Engineering
Department
University of California, Berkeley
Berkeley, CA 94270
resa@berkeley.edu

H. Sebastian Seung

Howard Hughes Medical Institute
Brain & Cognitive Sciences
Massachusetts Inst. of Technology
Cambridge, MA 02139
seung@mit.edu



Class Structure

- Last time: Imitation Learning
- **This time: Policy Search**
- Next time: Policy Search Cont.

Stanford·CS234 | Reinforcement Learning (2019)

CS234 (2019)· 课程资料包 @ShowMeAI



视频
中英双语字幕



课件
一键打包下载



笔记
官方笔记翻译



代码
作业项目解析



视频·B站【扫码或点击链接】

<https://www.bilibili.com/video/BV1H64y1x7GH>



课件 & 代码·博客【扫码或点击链接】

<http://blog.showmeai.tech/cs234>

斯坦福

reinforcement learning
马尔可夫决策过程

DQN

强化学习

值函数方法

policy
gradient

Q-learning

梯度策略

Model free
蒙特卡洛搜索树

Awesome AI Courses Notes Cheatsheets 是 [ShowMeAI](#) 资料库的分支系列，覆盖最具知名度的 **TOP20+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程资料包页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets · 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094



微信公众号

资料下载方式 2: 扫码点击底部菜单栏

称为 AI 内容创作者? 回复【添砖加瓦】