CS420, Machine Learning, Lecture 12

Approximation Methods in Reinforcement Learning

Weinan Zhang

Shanghai Jiao Tong University

http://wnzhang.net

http://wnzhang.net/teaching/cs420/index.html

Reinforcement Learning Materials

Our course on RL is mainly based on the materials from these masters.

Prof. Richard Sutton

- University of Alberta, Canada
- http://incompleteideas.net/sutton/index.html
- Reinforcement Learning: An Introduction (2nd edition)
- http://incompleteideas.net/sutton/book/the-book-2nd.html

Dr. David Silver

- Google DeepMind and UCL, UK
- http://www0.cs.ucl.ac.uk/staff/d.silver/web/Home.html
- UCL Reinforcement Learning Course
- http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

Prof. Andrew Ng

- Stanford University, US
- http://www.andrewng.org/
- Machine Learning (CS229) Lecture Notes 12: RL
- http://cs229.stanford.edu/materials.html

Last Lecture

- Model-based dynamic programming
  - Value iteration $V(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V(s')$
  - Policy iteration $\pi(s) = \arg\max_{a \in A} \sum_{s' \in S} P_{sa}(s') V(s')$
- Model-free reinforcement learning
  - On-policy MC $V(s_t) \leftarrow V(s_t) + \alpha(G_t - V(s_t))$
  - On-policy TD $V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$
  - On-policy TD SARSA

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

  - Off-policy TD Q-learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

Key Problem to Solve in This Lecture

- In all previous models, we have created a lookup table to maintain a variable $V(s)$ for each state or $Q(s,a)$ for each state-action

- What if we have a large MDP, i.e.
  - the state or state-action space is too large
  - or the state or action space is continuous

  to maintain $V(s)$ for each state or $Q(s,a)$ for each state-action?
  - For example
    - Game of Go (10^{170} states)
    - Helicopter, autonomous car (continuous state space)

Content

- Solutions for large MDPs
  - Discretize or bucketize states/actions
  - Build parametric value function approximation

- Policy gradient

- Deep reinforcement learning

Content

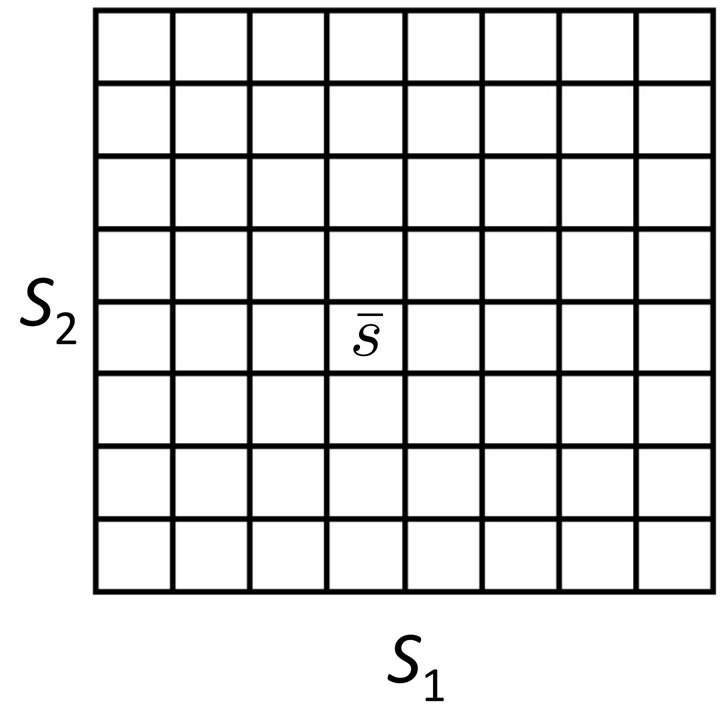
- Solutions for large MDPs
  - Discretize or bucketize states/actions
  - Build parametric value function approximation
- Policy gradient
- Deep reinforcement learning

Discretization Continuous MDP

- For a continuous-state MDP, we can discretize the state space
  - For example, if we have 2D states (s_1, s_2), we can use a grid to discretize the state space
  - The discrete state $\bar{s}$
  - The discretized MDP:

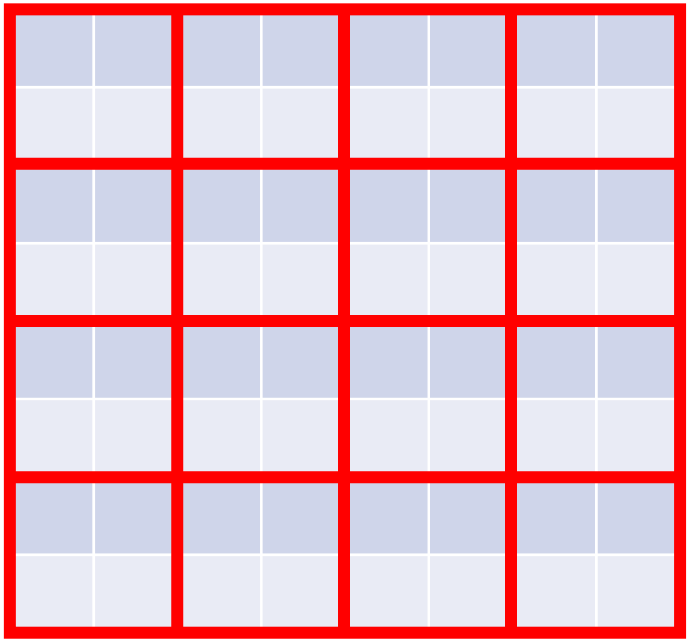
$$(\bar{S}, A, \{P_{\bar{s}a}\}, \gamma, R)$$

  - Then solve this MDP with any previous solutions



Bucketize Large Discrete MDP

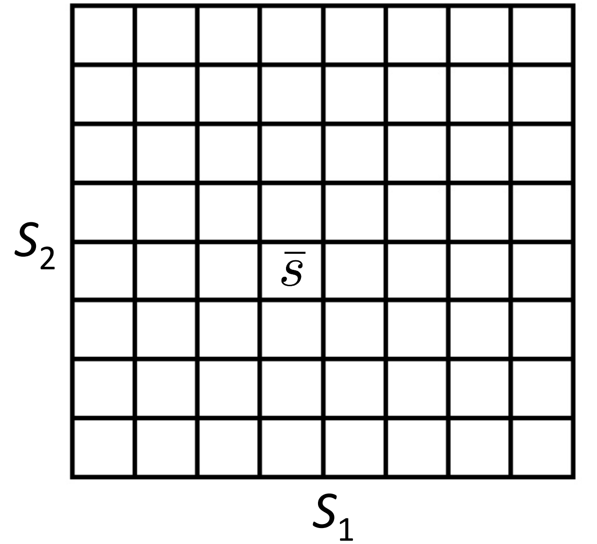
- For a large discrete-state MDP, we can bucketize the states to ‘down sample’ the states
  - To use domain knowledge to merge similar discrete states
    - For example, clustering using state features extracted from domain knowledge

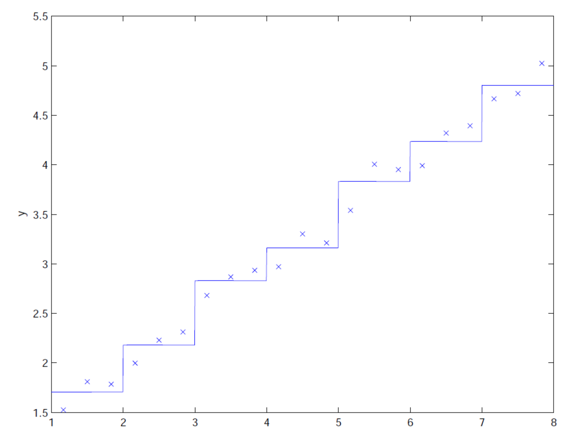


Discretization/Bucketization

- Pros
  - Straightforward and off-the-shelf
  - Efficient
  - Can work well for many problems
- Cons
  - A fairly naïve representation for *V*
  - Assumes a constant value over each discretized cell
  - Curse of dimensionality

$$S = \mathbb{R}^n \Rightarrow \bar{S} = \{1, \ldots, k\}^n$$





Parametric Value Function Approximation

- Create parametric (thus learnable) functions to approximate the value function

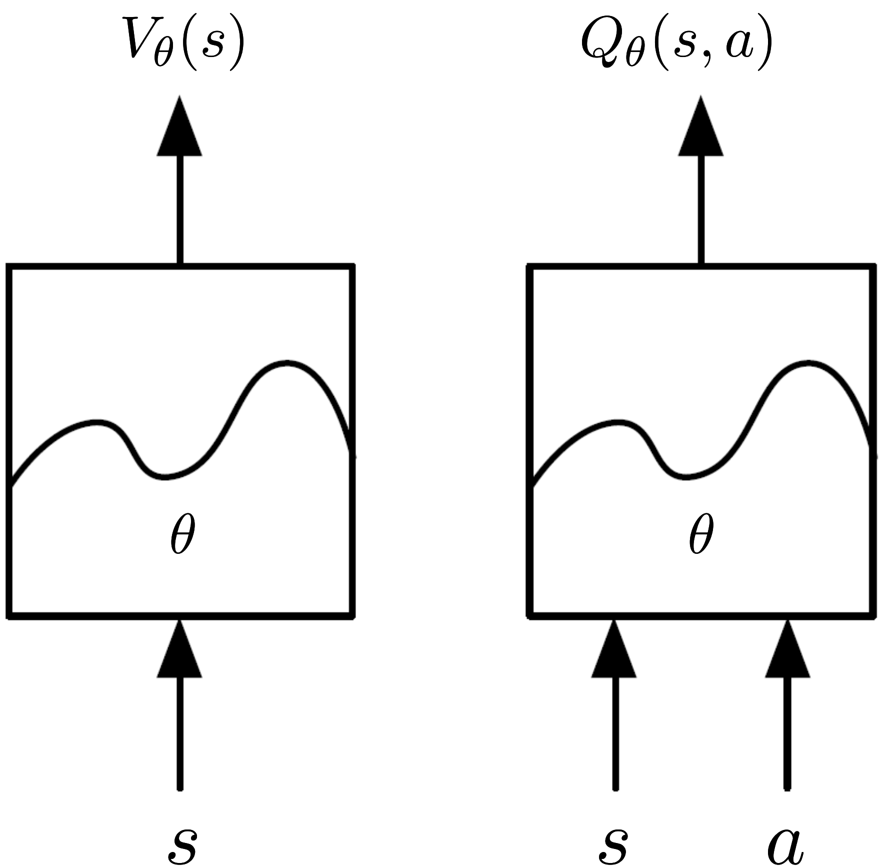
$$V_\theta(s) \simeq V^\pi(s)$$

$$Q_\theta(s, a) \simeq Q^\pi(s, a)$$

  - ϑ is the parameters of the approximation function, which can be updated by reinforcement learning

  - Generalize from seen states to unseen states

Main Types of Value Function Approx.



Many function approximations

- (Generalized) linear model
- Neural network
- Decision tree
- Nearest neighbor
- Fourier / wavelet bases

Differentiable functions

- (Generalized) linear model
- Neural network

We assume the model is suitable to be trained for non-stationary, non-iid data

Value Function Approx. by SGD

- Goal: find parameter vector ϑ minimizing mean-squared error between approximate value function $V_\vartheta(s)$ and true value $V^\pi(s)$

$$J(\theta) = \mathbb{E}_\pi\left[\frac{1}{2}(V^\pi(s) - V_\theta(s))^2\right]$$

- Gradient to minimize the error

$$-\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_\pi[V^\pi(s) - V_\theta(s)]\frac{\partial V_\theta(s)}{\partial \theta}$$

- Stochastic gradient descent on one sample

$$\begin{aligned}\theta &\leftarrow \theta - \alpha\frac{\partial J(\theta)}{\partial \theta} \\ &= \theta + \alpha(V^\pi(s) - V_\theta(s))\frac{\partial V_\theta(s)}{\partial \theta}\end{aligned}$$

Featurize the State

- Represent state by a feature vector

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

- For example of a helicopter
  - 3D location
  - 3D speed (differentiation of location)
  - 3D acceleration (differentiation of speed)

Linear Value Function Approximation

- Represent value function by a linear combination of features

$$V_\theta(s) = \theta^\top x(s)$$

- Objective function is quadratic in parameters $\boldsymbol{\vartheta}$

$$J(\theta) = \mathbb{E}_\pi\left[\frac{1}{2}(V^\pi(s) - \theta^\top x(s))^2\right]$$

- Thus stochastic gradient descent converges on global optimum

$$\begin{aligned}\theta &\leftarrow \theta - \alpha\frac{\partial J(\theta)}{\partial \theta} \\ &= \theta + \underbrace{\alpha}_{\text{Step size}}(\underbrace{V^\pi(s) - V_\theta(s)}_{\text{Prediction error}})\underbrace{x(s)}_{\text{Feature value}}\end{aligned}$$

Monte-Carlo with Value Function Approx.

$$\theta \leftarrow \theta + \alpha(V^{\pi}(s) - V_{\theta}(s))x(s)$$

- Now we specify the target value function $V^{\pi}(s)$
- We can apply supervised learning to "training data"

$$\langle s_1, G_1 \rangle, \langle s_2, G_2 \rangle, \ldots, \langle s_T, G_T \rangle$$

- For each data instance $<s_t, G_t>$

$$\theta \leftarrow \theta + \alpha(G_t - V_{\theta}(s))x(s_t)$$

- MC evaluation at least converges to a local optimum
  - In linear case it converges to a global optimum

TD Learning with Value Function Approx.

$$\theta \leftarrow \theta + \alpha(V^{\pi}(s) - V_{\theta}(s))x(s)$$

- TD target $r_{t+1} + \gamma V_{\theta}(s_{t+1})$ is a biased sample of true target value $V^{\pi}(s_t)$
- Supervised learning from "training data"

$$\langle s_1, r_2 + \gamma V_{\theta}(s_2) \rangle, \langle s_2, r_3 + \gamma V_{\theta}(s_3) \rangle, \ldots, \langle s_T, r_T \rangle$$

- For each data instance $\langle s_t, r_{t+1} + \gamma V_{\theta}(s_{t+1}) \rangle$

$$\theta \leftarrow \theta + \alpha(r_{t+1} + \gamma V_{\theta}(s_{t+1}) - V_{\theta}(s))x(s_t)$$

- Linear TD(0) converges (close) to global optimum

Action-Value Function Approximation

- Approximate the action-value function

$$Q_\theta(s,a) \simeq Q^\pi(s,a)$$

- Minimize mean squared error

$$J(\theta) = \mathbb{E}_\pi\Big[\frac{1}{2}(Q^\pi(s,a) - Q_\theta(s,a))^2\Big]$$

- Stochastic gradient descent on one sample

$$\begin{aligned}\theta &\leftarrow \theta - \alpha\frac{\partial J(\theta)}{\partial\theta}\\ &= \theta + \alpha(Q^\pi(s,a) - Q_\theta(s,a))\frac{\partial Q_\theta(s,a)}{\partial\theta}\end{aligned}$$

Linear Action-Value Function Approx.

- Represent state-action pair by a feature vector

$$x(s,a) = \begin{bmatrix} x_1(s,a) \\ \vdots \\ x_k(s,a) \end{bmatrix}$$

- Parametric Q function, e.g., the linear case

$$Q_\theta(s,a) = \theta^\top x(s,a)$$

- Stochastic gradient descent update

$$\begin{aligned} \theta &\leftarrow \theta - \alpha \frac{\partial J(\theta)}{\partial \theta} \\ &= \theta + \alpha (Q^\pi(s,a) - \theta^\top x(s,a)) x(s,a) \end{aligned}$$

TD Learning with Value Function Approx.

$$\theta \leftarrow \theta + \alpha(Q^\pi(s,a) - Q_\theta(s,a))\frac{\partial Q_\theta(s,a)}{\partial \theta}$$

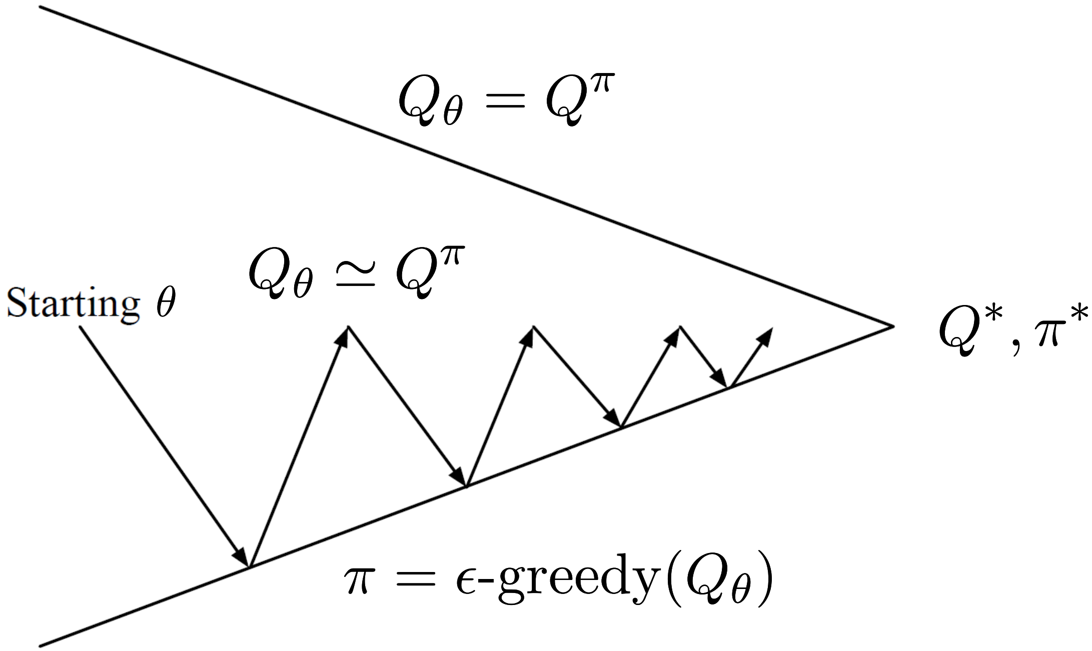
- For MC, the target is the return G_t

$$\theta \leftarrow \theta + \alpha(G_t - Q_\theta(s,a))\frac{\partial Q_\theta(s,a)}{\partial \theta}$$

- For TD(0), the TD target is $r_{t+1} + \gamma Q_\theta(s_{t+1}, a_{t+1})$

$$\theta \leftarrow \theta + \alpha(r_{t+1} + \gamma Q_\theta(s_{t+1}, a_{t+1}) - Q_\theta(s,a))\frac{\partial Q_\theta(s,a)}{\partial \theta}$$

Control with Value Function Approx.



- Policy evaluation: approximately policy evaluation $Q_\theta \simeq Q^\pi$
- Policy improvement: ε-greedy policy improvement

NOTE of TD Update

- For TD(0), the TD target is
  - State value

$$\theta \leftarrow \theta + \alpha(V^{\pi}(s_t) - V_{\theta}(s_t))\frac{\partial V_{\theta}(s_t)}{\partial \theta}$$

$$= \theta + \alpha(r_{t+1} + \gamma V_{\theta}(s_{t+1}) - V_{\theta}(s))\frac{\partial V_{\theta}(s_t)}{\partial \theta}$$

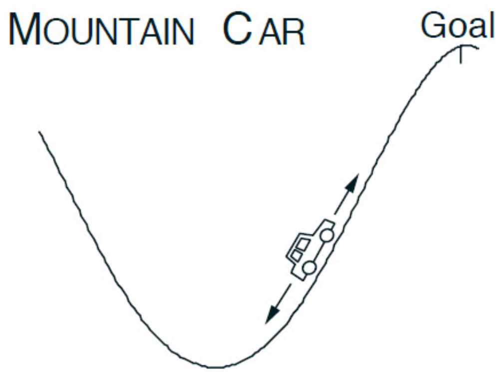
  - Action value

$$\theta \leftarrow \theta + \alpha(Q^{\pi}(s,a) - Q_{\theta}(s,a))\frac{\partial Q_{\theta}(s,a)}{\partial \theta}$$

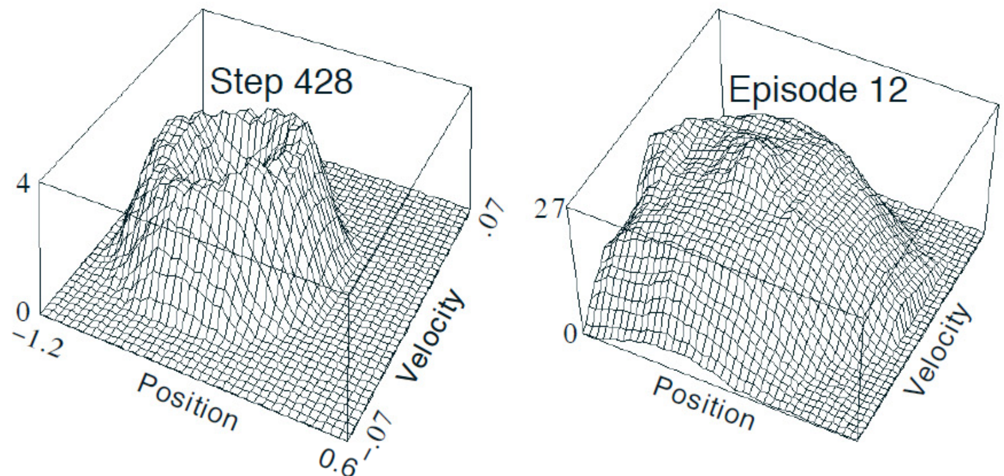
$$= \theta + \alpha(r_{t+1} + \gamma Q_{\theta}(s_{t+1}, a_{t+1}) - Q_{\theta}(s,a))\frac{\partial Q_{\theta}(s,a)}{\partial \theta}$$

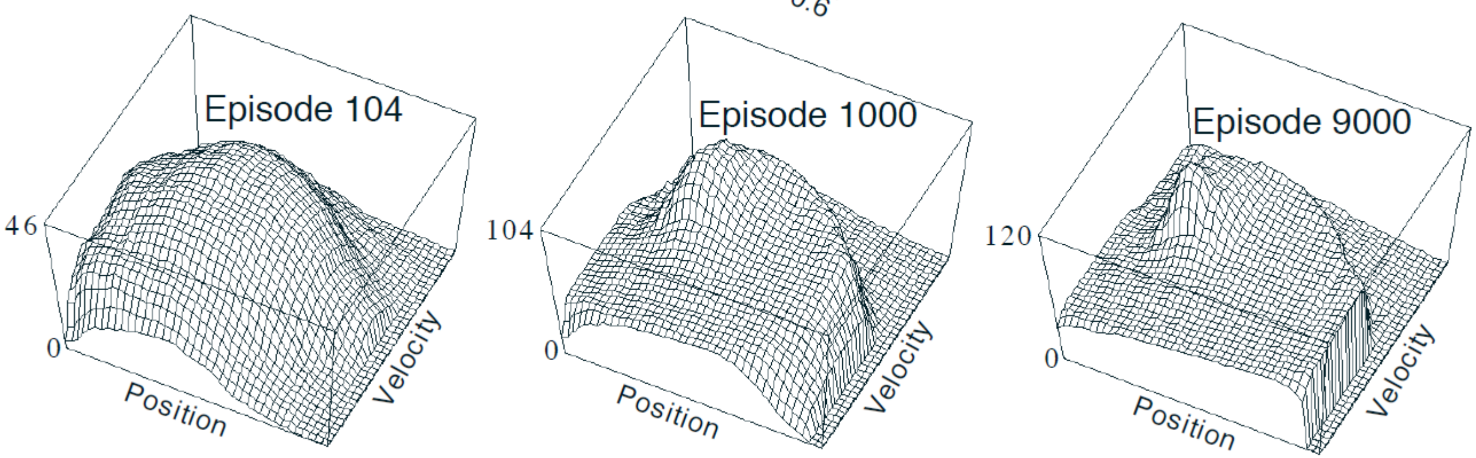
- Although ϑ is in the TD target, we don't calculate gradient from the target. Think about why.

Case Study: Mountain Car



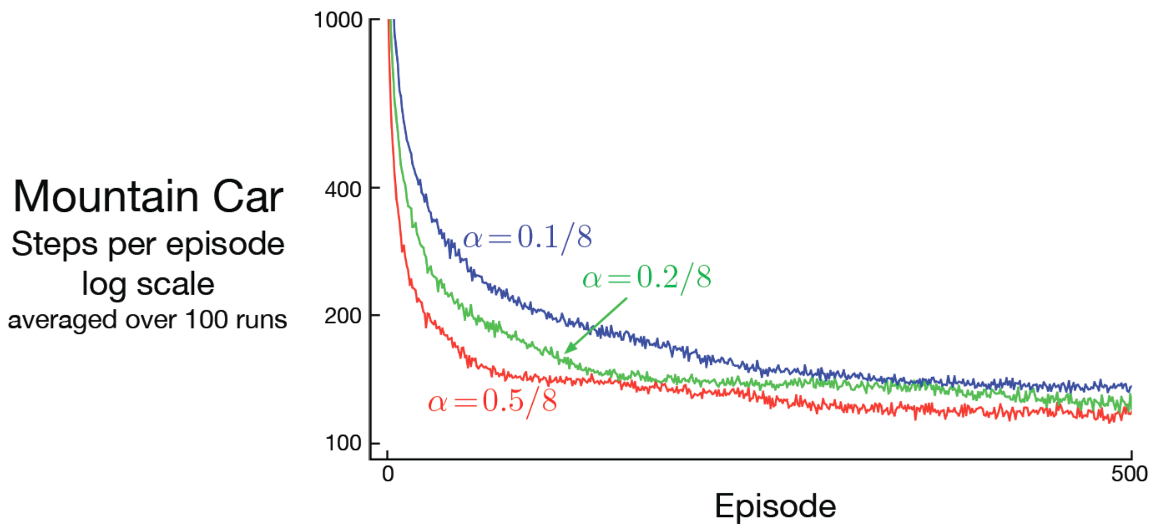
The gravity is stronger than the car's engine

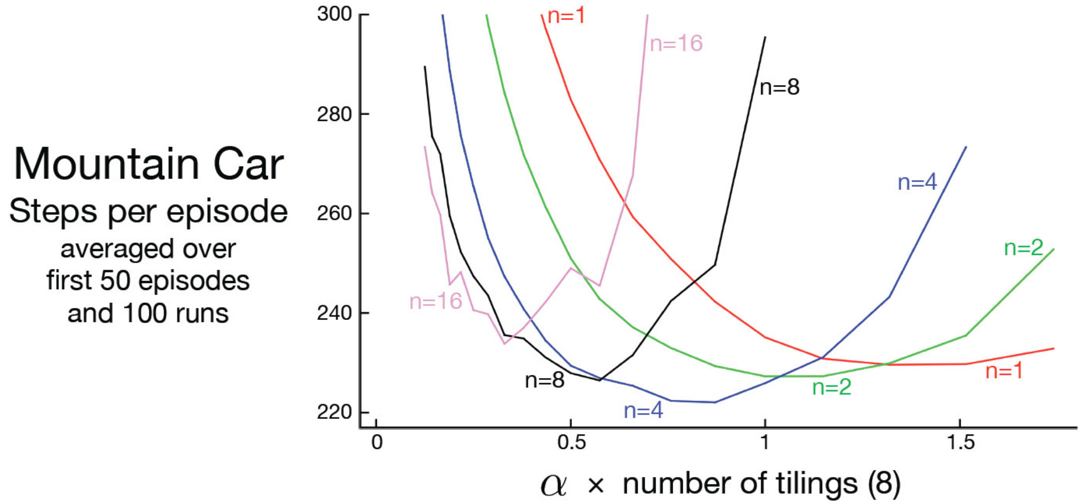




Cost-to-go function

Case Study: Mountain Car





Content

- Solutions for large MDPs
    - Discretize or bucketize states/actions
    - Build parametric value function approximation

- Policy gradient

- Deep reinforcement learning

Parametric Policy

- We can parametrize the policy

$$\pi_\theta(a|s)$$

which could be deterministic

$$a = \pi_\theta(s)$$

or stochastic

$$\pi_\theta(a|s) = P(a|s;\theta)$$

  - ϑ is the parameters of the policy
  - Generalize from seen states to unseen states
  - We focus on model-free reinforcement learning

Policy-based RL

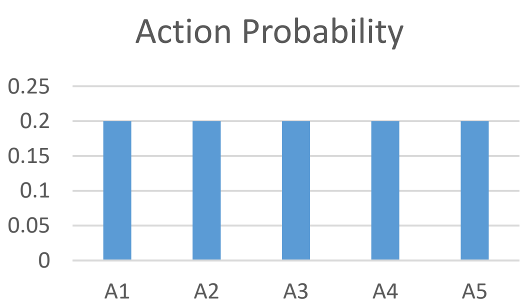
- Advantages
  - Better convergence properties
  - Effective in high-dimensional or continuous action spaces
    - No.1 reason: for value function, you have to take a max operation
  - Can learn stochastic polices

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

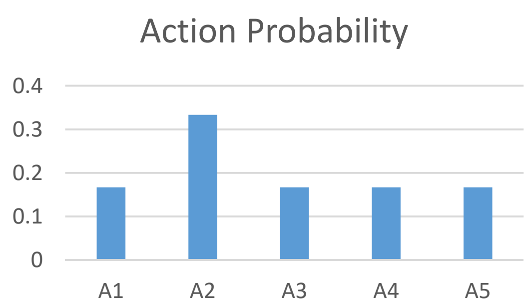
Policy Gradient

- For stochastic policy $\pi_\theta(a|s) = P(a|s;\theta)$
- Intuition
  - lower the probability of the action that leads to low value/reward
  - higher the probability of the action that leads to high value/reward
- A 5-action example

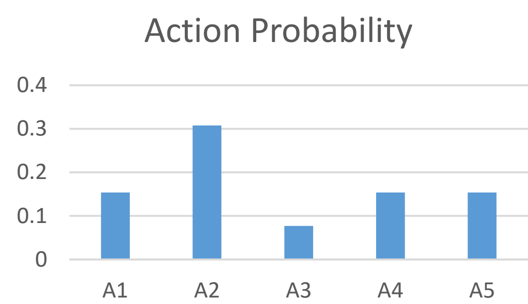
1. Initialize ϑ



3. Update ϑ by policy gradient



5. Update ϑ by policy gradient



2. Take action A2
Observe positive reward

4. Take action A3
Observe negative reward

Policy Gradient in One-Step MDPs

- Consider a simple class of one-step MDPs
  - Starting in state $s \sim d(s)$
  - Terminating after one time-step with reward r_{sa}

- Policy expected value

$$J(\theta) = \mathbb{E}_{\pi_\theta}[r] = \sum_{s \in S} d(s) \sum_{a \in A} \pi_\theta(a|s) r_{sa}$$

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{s \in S} d(s) \sum_{a \in A} \frac{\partial \pi_\theta(a|s)}{\partial \theta} r_{sa}$$

Likelihood Ratio

- Likelihood ratios exploit the following identity

$$\begin{aligned}\frac{\partial \pi_\theta(a|s)}{\partial \theta} &= \pi_\theta(a|s)\frac{1}{\pi_\theta(a|s)}\frac{\partial \pi_\theta(a|s)}{\partial \theta} \\ &= \pi_\theta(a|s)\frac{\partial \log \pi_\theta(a|s)}{\partial \theta}\end{aligned}$$

- Thus the policy's expected value

$$J(\theta) = \mathbb{E}_{\pi_\theta}[r] = \sum_{s\in S} d(s) \sum_{a\in A} \pi_\theta(a|s) \boxed{r_{sa}}$$

$$\begin{aligned}\frac{\partial J(\theta)}{\partial \theta} &= \sum_{s\in S} d(s) \sum_{a\in A} \frac{\partial \pi_\theta(a|s)}{\partial \theta} r_{sa} \\ &= \sum_{s\in S} d(s) \sum_{a\in A} \pi_\theta(a|s) \boxed{\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} r_{sa}} \\ &= \mathbb{E}_{\pi_\theta}\left[\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} r_{sa}\right]\end{aligned}$$

This can be approximated by sampling state *s* from *d*(*s*) and action *a* from π_ϑ

Policy Gradient Theorem

- The policy gradient theorem generalizes the likelihood ratio approach to multi-step MDPs
  - Replaces instantaneous reward r_{sa} with long-term value $Q^{\pi_\theta}(s,a)$
- Policy gradient theorem applies to
  - start state objective J_1, average reward objective J_{avR}, and average value objective J_{avV}
- Theorem
  - For any differentiable policy $\pi_\theta(a|s)$, for any of policy objective function $J = J_1, J_{avR}, J_{avV}$, the policy gradient is

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

Please refer to appendix of the slides for detailed proofs

Monte-Carlo Policy Gradient (REINFORCE)

- Update parameters by stochastic gradient ascent
- Using policy gradient theorem
- Using return v_t as an unbiased sample of $Q^{\pi_\theta}(s,a)$

$$\Delta\theta_t = \alpha \frac{\partial \log \pi_\theta(a_t|s_t)}{\partial \theta} v_t$$

- REINFORCE Algorithm

Initialize ϑ arbitrarily

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

for t=1 to T-1 do

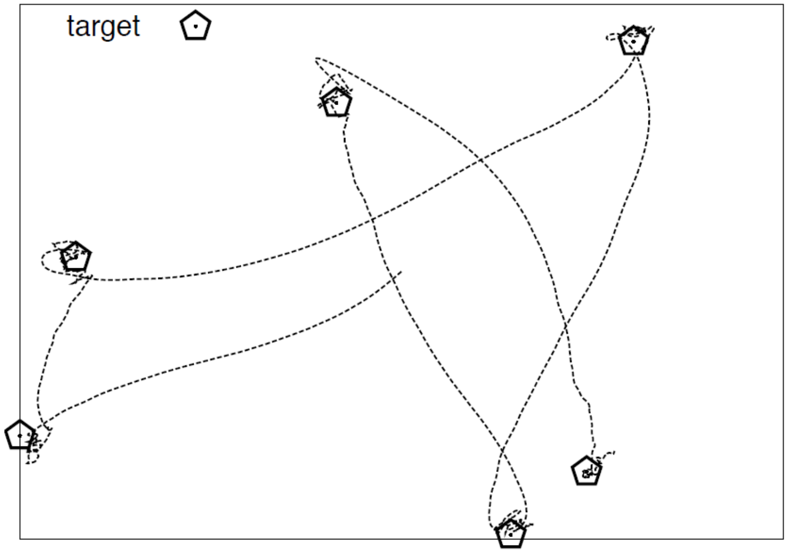
$$\theta \leftarrow \theta + \alpha \frac{\partial}{\partial \theta} \log \pi_\theta(a_t|s_t) v_t$$

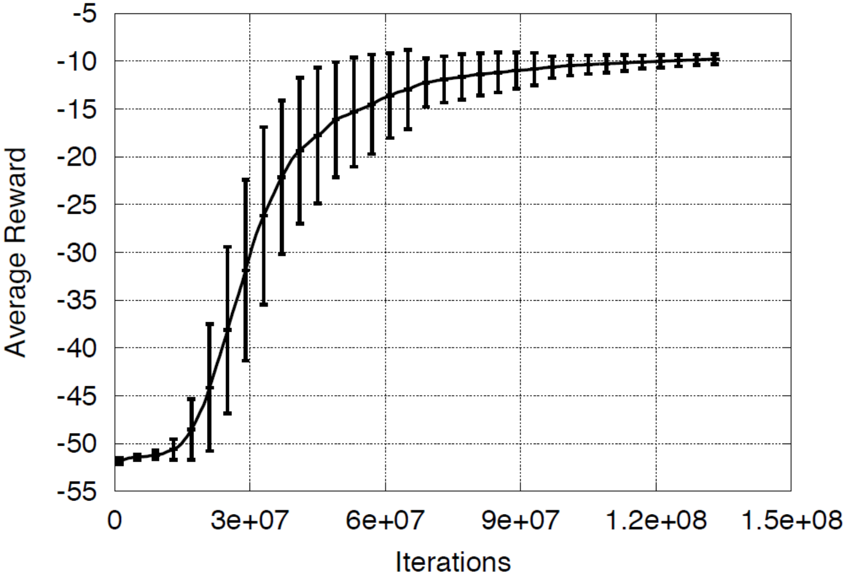
end for

end for

return ϑ

Puck World Example





- Continuous actions exert small force on puck
- Puck is rewarded for getting close to target
- Target location is reset every 30 seconds
- Policy is trained using variant of MC policy gradient

Softmax Stochastic Policy

- Softmax policy is a very commonly used stochastic policy

$$\pi_\theta(a|s) = \frac{e^{f_\theta(s,a)}}{\sum_{a'} e^{f_\theta(s,a')}}$$

  - where $f_\vartheta(s,a)$ is the score function of a state-action pair parametrized by ϑ, which can be defined with domain knowledge

- The gradient of its log-likelihood

$$\begin{aligned}\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} &= \frac{\partial f_\theta(s,a)}{\partial \theta} - \frac{1}{\sum_{a'} e^{f_\theta(s,a')}} \sum_{a''} e^{f_\theta(s,a'')} \frac{\partial f_\theta(s,a'')}{\partial \theta} \\ &= \frac{\partial f_\theta(s,a)}{\partial \theta} - \mathbb{E}_{a' \sim \pi_\theta(a'|s)}\left[\frac{\partial f_\theta(s,a')}{\partial \theta}\right]\end{aligned}$$

Softmax Stochastic Policy

- Softmax policy is a very commonly used stochastic policy

$$\pi_\theta(a|s) = \frac{e^{f_\theta(s,a)}}{\sum_{a'} e^{f_\theta(s,a')}}$$

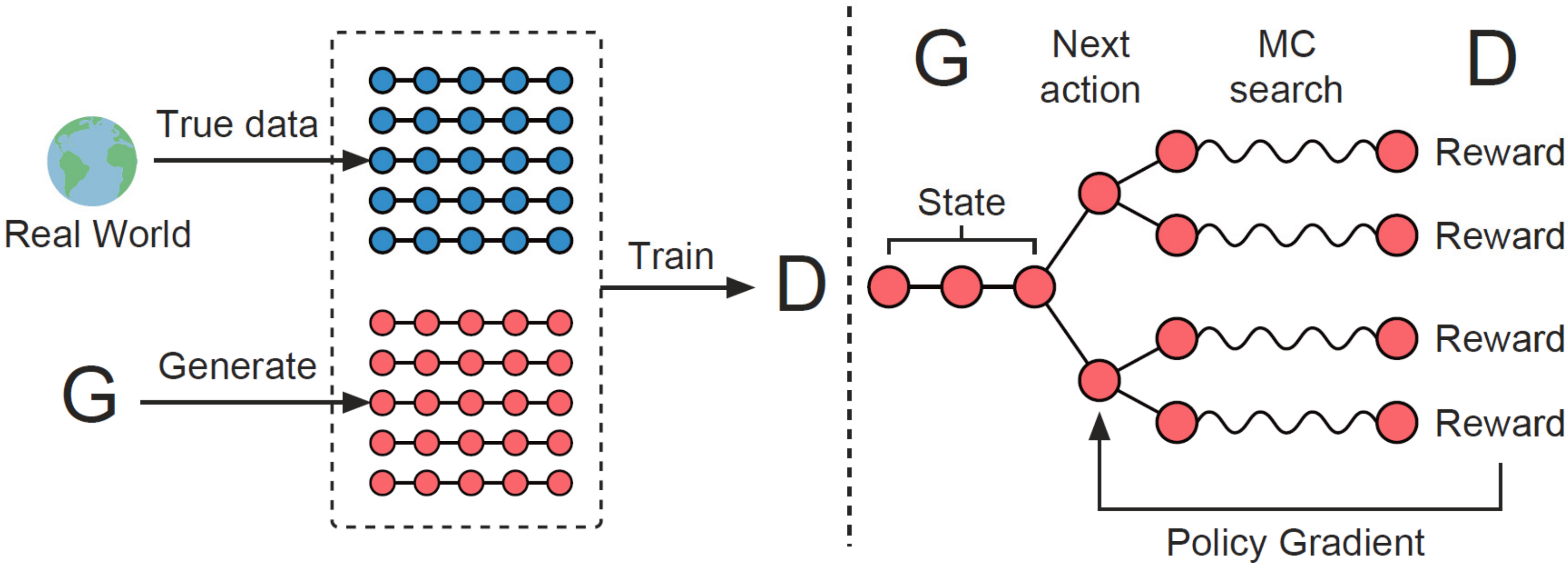
  - where $f_\vartheta(s,a)$ is the score function of a state-action pair parametrized by ϑ, which can be defined with domain knowledge

- For example, we define the linear score function

$$f_\theta(s,a) = \theta^\top x(s,a)$$

$$\begin{aligned}\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} &= \frac{\partial f_\theta(s,a)}{\partial \theta} - \mathbb{E}_{a' \sim \pi_\theta(a'|s)}\Big[\frac{\partial f_\theta(s,a')}{\partial \theta}\Big] \\ &= x(s,a) - \mathbb{E}_{a' \sim \pi_\theta(a'|s)}[x(s,a')]\end{aligned}$$

Sequence Generation Example



- Generator is a reinforcement learning policy $G_\theta(y_t|Y_{1:t-1})$ of generating a sequence
  - Decide the next word (discrete action) to generate given the previous ones, implemented by softmax policy
  - Discriminator provides the reward (i.e. the probability of being true data) for the whole sequence
  - G is trained by MC policy gradient (REINFORCE)

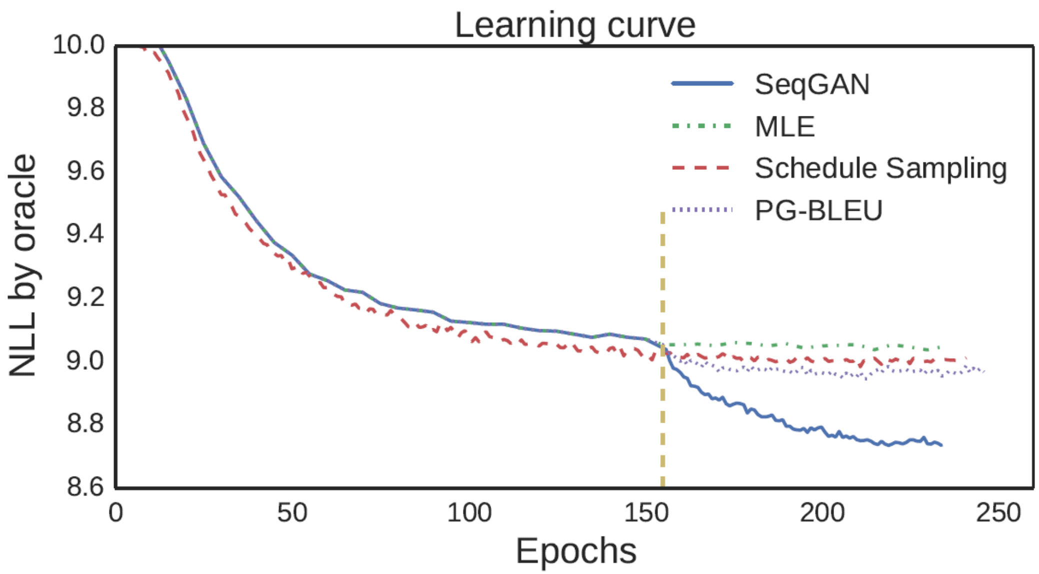
[Lantao Yu, Weinan Zhang, Jun Wang, Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. AAAI 2017.]

Experiments on Synthetic Data

- Evaluation measure with Oracle

$$\mathrm{NLL}_{\mathrm{oracle}} = -\mathbb{E}_{Y_{1:T} \sim G_\theta}\left[\sum_{t=1}^{T} \log G_{\mathrm{oracle}}(y_t | Y_{1:t-1})\right]$$

Algorithm	Random	MLE	SS	PG-BLEU	SeqGAN
NLL	10.310	9.038	8.985	8.946	**8.736**
p-value	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	



[Lantao Yu, Weinan Zhang, Jun Wang, Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. AAAI 2017.]

Content

- Solutions for large MDPs
  - Discretize or bucketize states/actions
  - Build parametric value function approximation

- Policy gradient

- Deep reinforcement learning
  - By our invited speakers Yuhuai Wu, Shixiang Gu and Ying Wen

APPENDIX

Policy Gradient Theorem: Average Reward Setting

- Average reward objective

$$J(\pi) = \lim_{n\to\infty} \frac{1}{n}\mathbb{E}\Big[r_1 + r_2 + \cdots + r_n | \pi\Big] = \sum_s d^\pi(s) \sum_a \pi(s,a) r(s,a)$$

$$Q^\pi(s,a) = \sum_{t=1}^{\infty} \mathbb{E}\Big[r_t - J(\pi) | s_0 = s, a_0 = a, \pi\Big]$$

$$\begin{aligned}
\frac{\partial V^\pi(s)}{\partial\theta} &\stackrel{\text{def}}{=} \frac{\partial}{\partial\theta}\sum_a \pi(s,a) Q^\pi(s,a), \quad \forall s \\
&= \sum_a \Big[\frac{\partial\pi(s,a)}{\partial\theta} Q^\pi(s,a) + \pi(s,a)\frac{\partial}{\partial\theta} Q^\pi(s,a)\Big] \\
&= \sum_a \Big[\frac{\partial\pi(s,a)}{\partial\theta} Q^\pi(s,a) + \pi(s,a)\frac{\partial}{\partial\theta}\Big(r(s,a) - J(\pi) + \sum_{s'} P^a_{ss'} V^\pi(s')\Big)\Big] \\
&= \sum_a \Big[\frac{\partial\pi(s,a)}{\partial\theta} Q^\pi(s,a) + \pi(s,a)\Big(-\frac{\partial J(\pi)}{\partial\theta} + \frac{\partial}{\partial\theta}\sum_{s'} P^a_{ss'} V^\pi(s')\Big)\Big]
\end{aligned}$$

$$\Rightarrow \frac{\partial J(\pi)}{\partial\theta} = \sum_a \Big[\frac{\partial\pi(s,a)}{\partial\theta} Q^\pi(s,a) + \pi(s,a)\sum_{s'} P^a_{ss'}\frac{\partial V^\pi(s')}{\partial\theta}\Big] - \frac{\partial V^\pi(s)}{\partial\theta}$$

Please refer to Chapter 13 of Rich Sutton's Reinforcement Learning: An Introduction (2nd Edition)

APPENDIX

Policy Gradient Theorem: Average Reward Setting

- Average reward objective

$$\frac{\partial J(\pi)}{\partial \theta} = \sum_a \Big[\frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) + \pi(s,a) \sum_{s'} P^a_{ss'} \frac{\partial V^\pi(s')}{\partial \theta}\Big] - \frac{\partial V^\pi(s)}{\partial \theta}$$

$$\sum_s d^\pi(s) \frac{\partial J(\pi)}{\partial \theta} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) + \sum_s d^\pi(s) \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \frac{\partial V^\pi(s')}{\partial \theta} - \sum_s d^\pi(s) \frac{\partial V^\pi(s)}{\partial \theta}$$

$$\sum_s d^\pi(s) \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \frac{\partial V^\pi(s')}{\partial \theta} = \sum_s \sum_a \sum_{s'} d^\pi(s) \pi(s,a) P^a_{ss'} \frac{\partial V^\pi(s')}{\partial \theta}$$

$$= \sum_s \sum_{s'} d^\pi(s) \Big(\sum_a \pi(s,a) P^a_{ss'} \Big) \frac{\partial V^\pi(s')}{\partial \theta} = \sum_s \sum_{s'} d^\pi(s) P_{ss'} \frac{\partial V^\pi(s')}{\partial \theta}$$

$$= \sum_{s'} \Big(\sum_s d^\pi(s) P_{ss'} \Big) \frac{\partial V^\pi(s')}{\partial \theta} = \sum_{s'} d^\pi(s') \frac{\partial V^\pi(s')}{\partial \theta}$$

$$\Rightarrow \sum_s d^\pi(s) \frac{\partial J(\pi)}{\partial \theta} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) + \sum_{s'} d^\pi(s') \frac{\partial V^\pi(s')}{\partial \theta} - \sum_s d^\pi(s) \frac{\partial V^\pi(s)}{\partial \theta}$$

$$\Rightarrow \frac{\partial J(\pi)}{\partial \theta} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a)$$

Please refer to Chapter 13 of Rich Sutton's Reinforcement Learning: An Introduction (2nd Edition)

APPENDIX

Policy gradient theorem: Start Value Setting

- Start state value objective

$$J(\pi) = \mathbb{E}\Big[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \Big| s_0, \pi\Big]$$

$$Q^\pi(s,a) = \mathbb{E}\Big[\sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k} \Big| s_t = s, a_t = a, \pi\Big]$$

$$\begin{aligned}
\frac{\partial V^\pi(s)}{\partial \theta} &\stackrel{\text{def}}{=} \frac{\partial}{\partial \theta} \sum_a \pi(s,a) Q^\pi(s,a), \quad \forall s \\
&= \sum_a \Big[\frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) + \pi(s,a) \frac{\partial}{\partial \theta} Q^\pi(s,a)\Big] \\
&= \sum_a \Big[\frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) + \pi(s,a) \frac{\partial}{\partial \theta} \Big(r(s,a) + \sum_{s'} \gamma P^a_{ss'} V^\pi(s') \Big)\Big] \\
&= \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) + \sum_a \pi(s,a) \gamma \sum_{s'} P^a_{ss'} \frac{\partial V^\pi(s')}{\partial \theta}
\end{aligned}$$

Please refer to Chapter 13 of Rich Sutton's Reinforcement Learning: An Introduction (2nd Edition)

APPENDIX

Policy gradient theorem: Start Value Setting

- Start state value objective

$$\frac{\partial V^{\pi}(s)}{\partial \theta} = \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^{\pi}(s,a) + \sum_a \pi(s,a)\gamma \sum_{s_1} P^a_{ss_1} \frac{\partial V^{\pi}(s_1)}{\partial \theta}$$

$$\sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^{\pi}(s,a) = \gamma^0 Pr(s \to s, 0, \pi) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^{\pi}(s,a)$$

$$\begin{aligned}\sum_a \pi(s,a)\gamma \sum_{s_1} P^a_{ss_1} \frac{\partial V^{\pi}(s_1)}{\partial \theta} &= \sum_{s_1}\sum_a \pi(s,a)\gamma P^a_{ss_1} \frac{\partial V^{\pi}(s_1)}{\partial \theta} \\ &= \sum_{s_1} \gamma P_{ss_1} \frac{\partial V^{\pi}(s_1)}{\partial \theta} = \gamma^1 \sum_{s_1} Pr(s \to s_1, 1, \pi) \frac{\partial V^{\pi}(s_1)}{\partial \theta}\end{aligned}$$

$$\frac{\partial V^{\pi}(s_1)}{\partial \theta} = \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^{\pi}(s,a) + \gamma^1 \sum_{s_2} Pr(s_1 \to s_2, 1, \pi) \frac{\partial V^{\pi}(s_2)}{\partial \theta}$$

Please refer to Chapter 13 of Rich Sutton's Reinforcement Learning: An Introduction (2nd Edition)

APPENDIX

Policy gradient theorem: Start Value Setting

- Start state value objective

$$
\begin{aligned}
\frac{\partial V^\pi(s)}{\partial \theta} &= \gamma^0 Pr(s \to s, 0, \pi) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) + \gamma^1 \sum_{s_1} Pr(s \to s_1, 1, \pi) \sum_a \frac{\partial \pi(s_1,a)}{\partial \theta} Q^\pi(s_1,a) \\
&\quad + \gamma^2 \sum_{s_1} Pr(s \to s_1, 1, \pi) \sum_{s_2} Pr(s_1 \to s_2, 1, \pi) \frac{\partial V^\pi(s_2)}{\partial \theta} \\
&= \gamma^0 Pr(s \to s, 0, \pi) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) + \gamma^1 \sum_{s_1} Pr(s \to s_1, 1, \pi) \sum_a \frac{\partial \pi(s_1,a)}{\partial \theta} Q^\pi(s_1,a) \\
&\quad + \gamma^2 \sum_{s_2} Pr(s \to s_2, 2, \pi) \frac{\partial V^\pi(s_2)}{\partial \theta} \\
&= \sum_{k=0}^{\infty} \sum_x \gamma^k Pr(s \to x, k, \pi) \sum_a \frac{\partial \pi(x,a)}{\partial \theta} Q^\pi(x,a) = \sum_x \sum_{k=0}^{\infty} \gamma^k Pr(s \to x, k, \pi) \sum_a \frac{\partial \pi(x,a)}{\partial \theta} Q^\pi(x,a)
\end{aligned}
$$

$$
\Rightarrow \frac{\partial J(\pi)}{\partial \theta} = \frac{\partial V^\pi(s_0)}{\partial \theta} = \sum_s \sum_{k=0}^{\infty} \gamma^k Pr(s_0 \to s, k, \pi) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a)
$$

Please refer to Chapter 13 of Rich Sutton's Reinforcement Learning: An Introduction (2nd Edition)