Policy Gradient

16-831 Statistical Techniques in Robotics

Carnegie Mellon University (Bolun Dai, Kyungzun Rim, Haosen Xing)

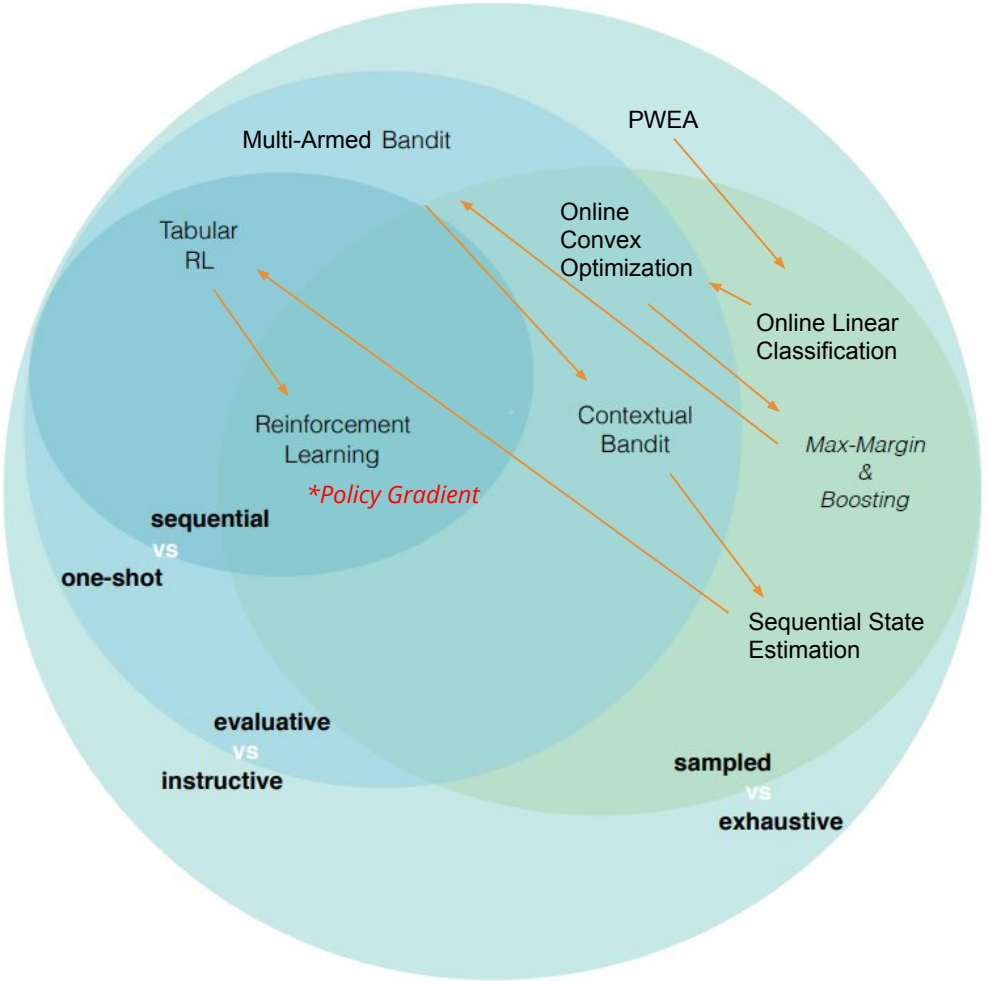
Outline

- Overview
- Policy Approximation (Motivation) and its Advantage
- Reinforcement Learning Tasks
  - Episodic Task
    - The Policy Gradient Theorem
    - Policy Gradient Learning algorithm
      - REINFORCE: Monte Carlo Policy Gradient
      - REINFORCE with Baseline
      - Actor-Critic Methods
  - Continuous Task
    - Policy Gradient for Continuing Problems
    - Policy Parameterization for Continuous Actions
- Applications

Where we are in the class now:

Difference from Bandit: outcome of action affects next state (**sequential**);

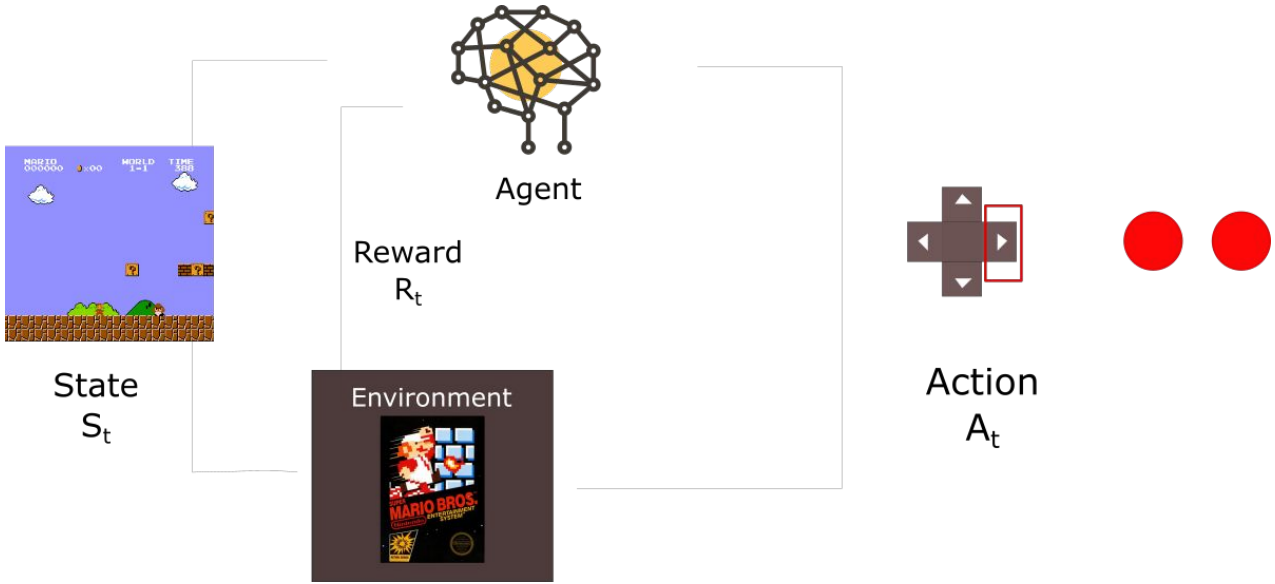
Difference from Supervised Learning: unclear relationship between actions and rewards (**evaluative**)



Littman (2015)

Reinforcement Learning

Evaluative and Sequential



Policy: a map from state space to action space
Goal: maximize the expected cumulative reward

Simonino (2018)

Reinforcement Learning

Tabular (RL) Solution Methods
Exhaustive

State/Action space: *small enough*

Value function/Policy: *optimal*

Approximate Solution Methods
Sampled

State/Action space: *arbitrarily large*

Value function/Policy: *good approximate solution*

Reinforcement Learning

Value-Based

- Learn a Value Function
- Implicit policy (e.g. **ε**-greedy)

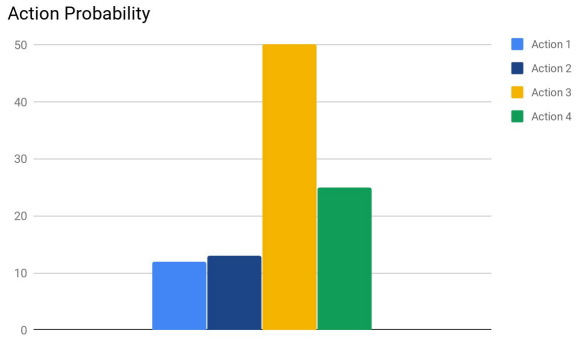
Model-based RL
(covered in last class)

Policy-Based

- Learn a Policy directly (Policy Gradient method)
- $\pi_\theta(s,a) = \mathbb{P}[a|s,\theta]$

a policy parameterized by θ
s: state
a: action

probability



Policy Gradient

- Goal: given $\pi_\theta(s, a)$ with parameters θ, find the "best" θ

$$\theta^* = argmax_\theta J(\theta)$$

- In episodic environments, objective function J_θ is the value of start state s_1

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

the true value function

- In continuing environments, objective function J_θ can be the average value

$$J(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$$

stationary distribution of
Markov Chain of the policy

Policy Gradient

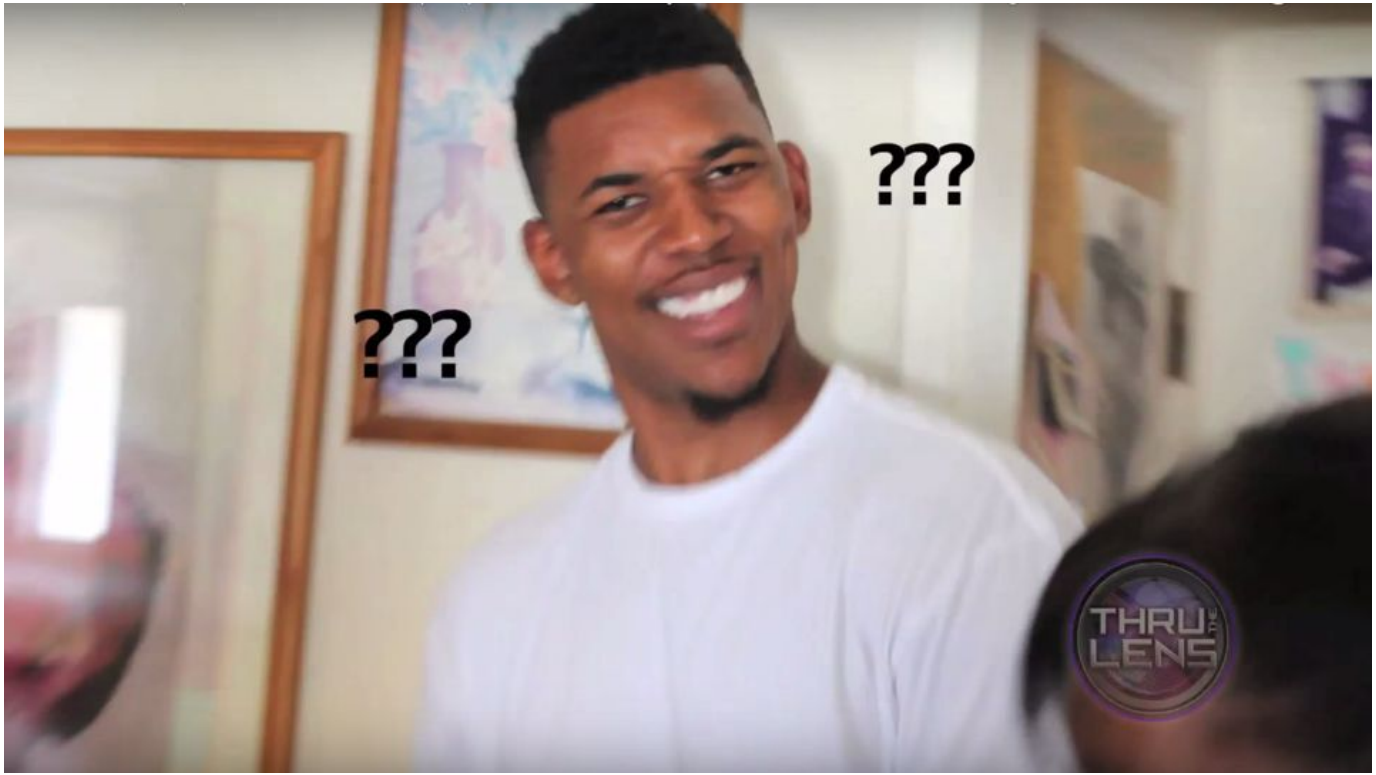
Policy-based RL is an optimization problem, where we focus on gradient ascent

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t)$$

Step size

Gradient of performance measurement of the objective function

But Why?



Advantages

- Can approach a deterministic policy (compared to epsilon-greedy)

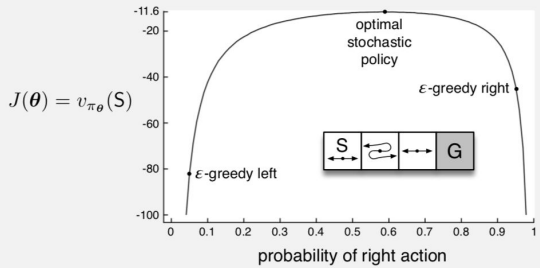
	Optimal Action	Suboptimal Action
Epsilon-Greedy (epsilon = 0.1)	0.90 + 0.05 = 0.95	0.05
Policy Gradient	0.999	0.001

Advantages

- Can approach a deterministic policy (compared to epsilon-greedy)
- Can find stochastic policies

Example 13.1 Short corridor with switched actions

Consider the small corridor gridworld shown inset in the graph below. The reward is -1 per step, as usual. In each of the three nonterminal states there are only two actions, right and left. These actions have their usual consequences in the first and third states (left causes no movement in the first state), but in the second state they are reversed, so that right moves to the left and left moves to the right. The problem is difficult because all the states appear identical under the function approximation. In particular, we define $\mathbf{x}(s, \text{right}) = [1, 0]^\top$ and $\mathbf{x}(s, \text{left}) = [0, 1]^\top$, for all s. An action-value method with ε-greedy action selection is forced to choose between just two policies: choosing right with high probability $1 - \varepsilon/2$ on all steps or choosing left with the same high probability on all time steps. If $\varepsilon = 0.1$, then these two policies achieve a value (at the start state) of less than -44 and -82, respectively, as shown in the graph. A method can do significantly better if it can learn a specific probability with which to select right. The best probability is about 0.59, which achieves a value of about -11.6.



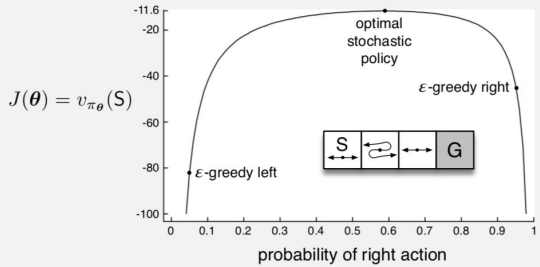
R. Sutton (2017)

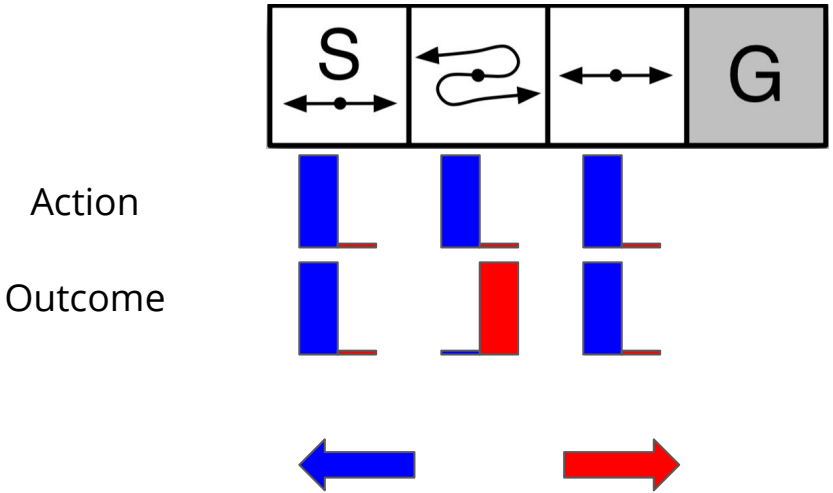
Advantages

- Can approach a deterministic policy (compared to epsilon-greedy)
- Can find stochastic policies

Example 13.1 Short corridor with switched actions

Consider the small corridor gridworld shown inset in the graph below. The reward is -1 per step, as usual. In each of the three nonterminal states there are only two actions, right and left. These actions have their usual consequences in the first and third states (left causes no movement in the first state), but in the second state they are reversed, so that right moves to the left and left moves to the right. The problem is difficult because all the states appear identical under the function approximation. In particular, we define $\mathbf{x}(s, \text{right}) = [1, 0]^\top$ and $\mathbf{x}(s, \text{left}) = [0, 1]^\top$, for all s. An action-value method with ε-greedy action selection is forced to choose between just two policies: choosing right with high probability $1 - \varepsilon/2$ on all steps or choosing left with the same high probability on all time steps. If $\varepsilon = 0.1$, then these two policies achieve a value (at the start state) of less than -44 and -82, respectively, as shown in the graph. A method can do significantly better if it can learn a specific probability with which to select right. The best probability is about 0.59, which achieves a value of about -11.6.





Epsilon greedy left

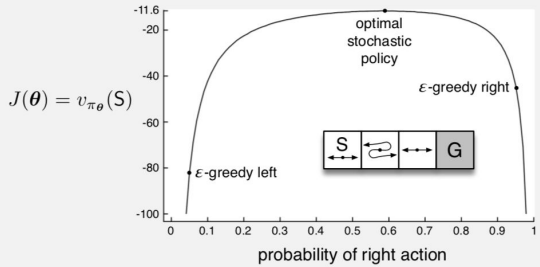
R. Sutton (2017)

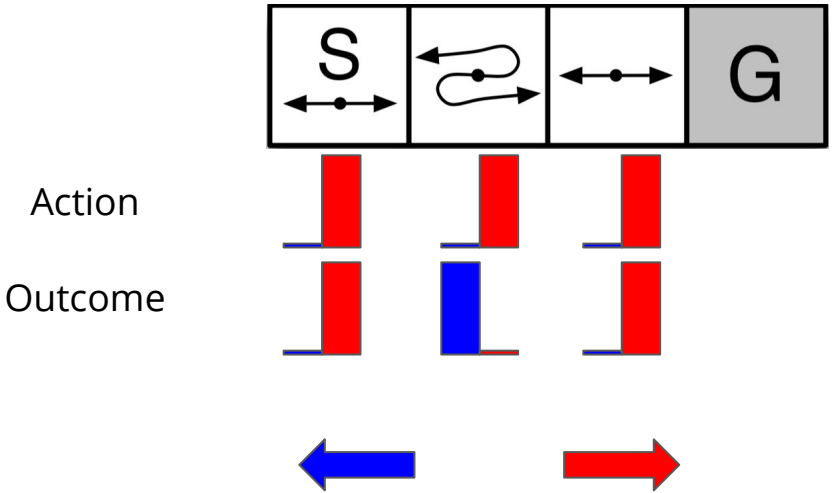
Advantages

- Can approach a deterministic policy (compared to epsilon-greedy)
- Can find stochastic policies

Example 13.1 Short corridor with switched actions

Consider the small corridor gridworld shown inset in the graph below. The reward is -1 per step, as usual. In each of the three nonterminal states there are only two actions, right and left. These actions have their usual consequences in the first and third states (left causes no movement in the first state), but in the second state they are reversed, so that right moves to the left and left moves to the right. The problem is difficult because all the states appear identical under the function approximation. In particular, we define $\mathbf{x}(s, \text{right}) = [1, 0]^\top$ and $\mathbf{x}(s, \text{left}) = [0, 1]^\top$, for all s. An action-value method with ε-greedy action selection is forced to choose between just two policies: choosing right with high probability $1 - \varepsilon/2$ on all steps or choosing left with the same high probability on all time steps. If $\varepsilon = 0.1$, then these two policies achieve a value (at the start state) of less than -44 and -82, respectively, as shown in the graph. A method can do significantly better if it can learn a specific probability with which to select right. The best probability is about 0.59, which achieves a value of about -11.6.





Epsilon greedy right

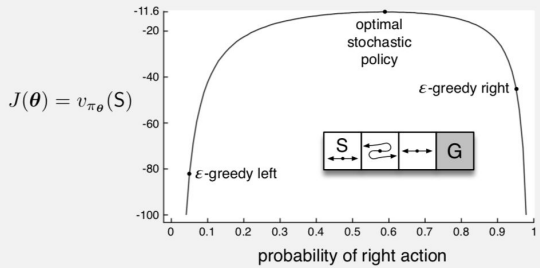
R. Sutton (2017)

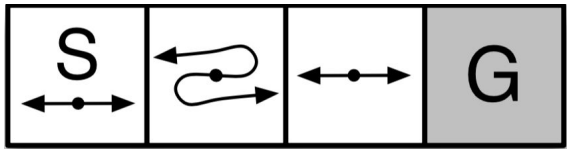
Advantages

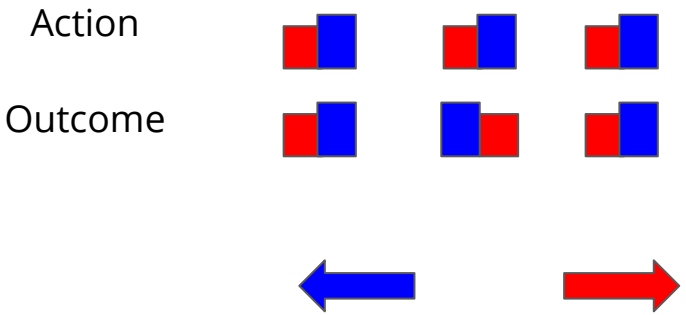
- Can approach a deterministic policy (compared to epsilon-greedy)
- Can find stochastic policies

Example 13.1 Short corridor with switched actions

Consider the small corridor gridworld shown inset in the graph below. The reward is -1 per step, as usual. In each of the three nonterminal states there are only two actions, right and left. These actions have their usual consequences in the first and third states (left causes no movement in the first state), but in the second state they are reversed, so that right moves to the left and left moves to the right. The problem is difficult because all the states appear identical under the function approximation. In particular, we define $\mathbf{x}(s, \text{right}) = [1, 0]^\top$ and $\mathbf{x}(s, \text{left}) = [0, 1]^\top$, for all s. An action-value method with ε-greedy action selection is forced to choose between just two policies: choosing right with high probability $1 - \varepsilon/2$ on all steps or choosing left with the same high probability on all time steps. If $\varepsilon = 0.1$, then these two policies achieve a value (at the start state) of less than -44 and -82, respectively, as shown in the graph. A method can do significantly better if it can learn a specific probability with which to select right. The best probability is about 0.59, which achieves a value of about -11.6.





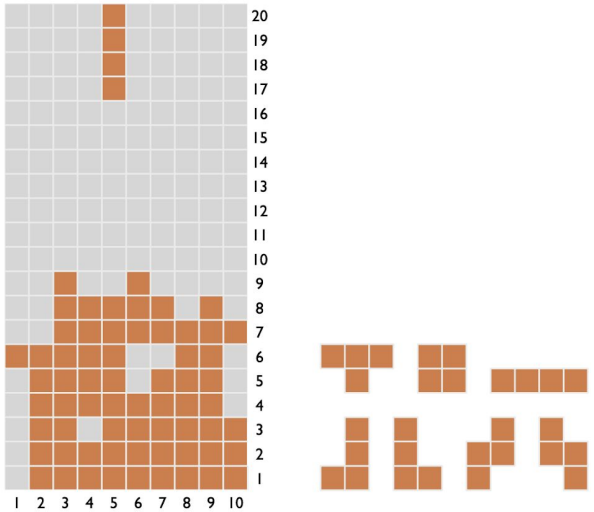


Optimal stochastic policy

R. Sutton (2017)

Advantages

- Can approach a deterministic policy (compared to epsilon-greedy)
- Can find stochastic policies
- The policy function is simpler than the value function



ID	Feature	Weight	Sample value
1	Rows with holes	−24.04	6
2	Column transitions	−19.77	20
3	Holes	−13.08	12
4	Landing height	−12.63	10.5
5	Cumulative wells	−10.49	26
6	Row transitions	−9.22	56
7	Eroded piece cells	+6.60	0
8	Hole depth	−1.61	12

O. Simsek (2016)

Advantages

- Can approach a deterministic policy (compared to epsilon-greedy)
- Can find stochastic policies
- The policy function is simpler than the value function
- Good way to inject prior knowledge

Advantages

- Can approach a deterministic policy (compared to epsilon-greedy)
- Can find stochastic policies
- The policy function is simpler than the value function
- Good way to inject prior knowledge
- With **continuous policy parameterization**, the action probability changes **smoothly** as a function of the learned parameters

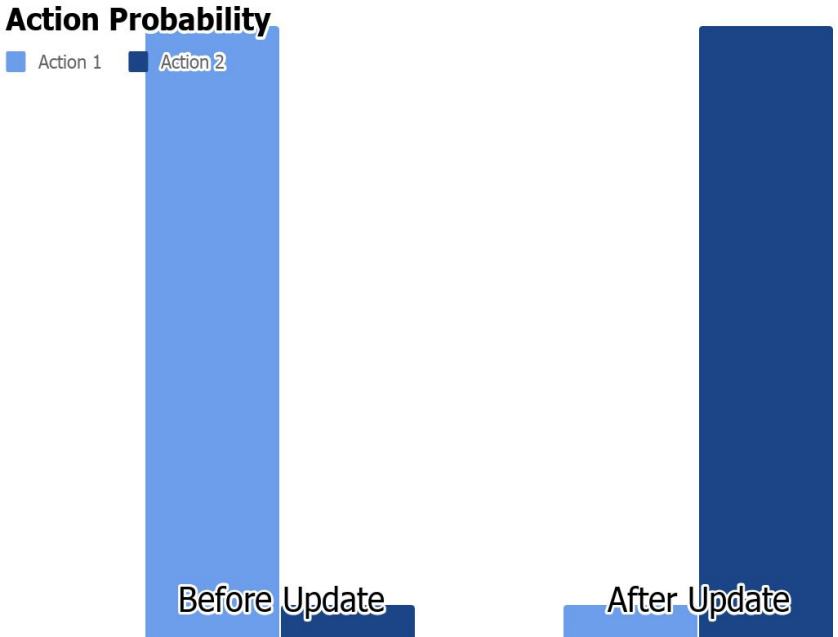
Advantages

- Can approach a deterministic policy (compared to epsilon-greedy)
- Can find stochastic policies
- The policy function is simpler than the value function
- Good way to inject prior knowledge
- With **continuous policy parameterization**, the action probability changes **smoothly** as a function of the learned parameters

$$\pi(a|s_0, \theta) = \begin{bmatrix} 0.50 \\ 0.49 \end{bmatrix} \xrightarrow{\text{update}} \pi(a|s_0, \theta') = \begin{bmatrix} 0.50 \\ 0.51 \end{bmatrix}$$

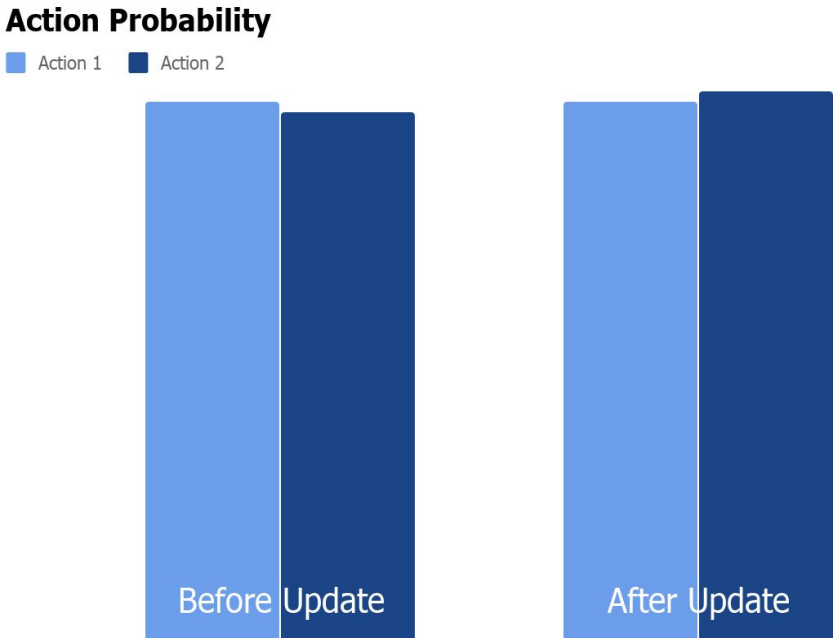
Epsilon Greedy

$$\pi(a|s_0,\theta) = \begin{bmatrix} 0.50 \\ 0.49 \end{bmatrix} \xrightarrow{\text{update}} \pi(a|s_0,\theta') = \begin{bmatrix} 0.50 \\ 0.51 \end{bmatrix}$$



Policy Gradient

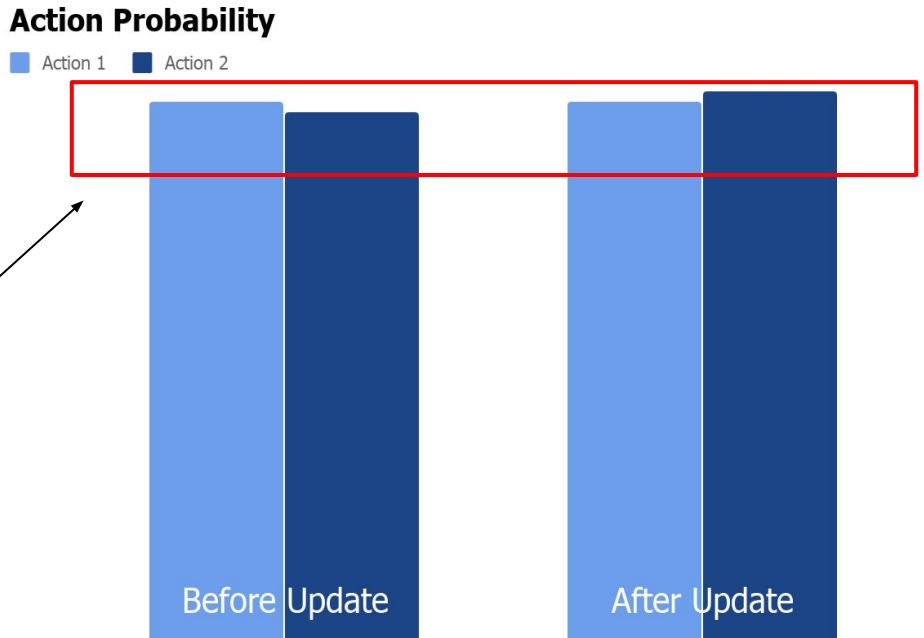
$$\pi(a|s_0,\theta) = \begin{bmatrix} 0.50 \\ 0.49 \end{bmatrix} \xrightarrow{\text{update}} \pi(a|s_0,\theta') = \begin{bmatrix} 0.50 \\ 0.51 \end{bmatrix}$$



Policy Gradient

$$\pi(a|s_0,\theta) = \begin{bmatrix} 0.50 \\ 0.49 \end{bmatrix} \xrightarrow{\text{update}} \pi(a|s_0,\theta') = \begin{bmatrix} 0.50 \\ 0.51 \end{bmatrix}$$

The change is much smoother given that we are sampling the action from this possibility distribution



Reinforcement Learning Tasks

Episodic Task

Continuous Task

Episodic Task

We have the objective function (performance measurement) as

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

True value function of the policy given the current parameters

Particular, non-random start state

Policy Parameterization for Discrete Actions

Softmax:

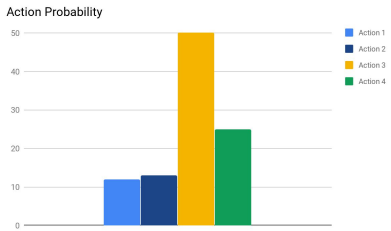
$$\pi(a|s,\theta) = \frac{\exp h(s,a,\theta)}{\sum_b \exp h(s,b,\theta)}$$

sum to one

where h is any function of s, a with params θ

e.g. h is a linear combination of features:

$$h(s,a,\theta) = \theta^T \phi(s,a)$$



e.g. h is the output of trained neural network

Performance

Action selection

$$\pi(a|s,\theta)$$

Easy to compute

State distribution

$$E(\theta)$$

Unknown

Performance

$$\Delta\theta \rightarrow \Delta J(\theta)$$

Action selection

$$\pi(a|s,\theta)$$

Easy to compute

State distribution

$$E(\theta)$$

Unknown

How can we tell how our change in the policy parameters affected the **action selections** and the performance

Performance

Δr

Action selection

$\pi(a|s,\theta)$

Easy to compute

State distribution

$E(\theta)$

Unknown

How can we tell how our change in the policy parameters affected the **action selections** and the performance

Performance

Δr

Action selection

$\pi(a|s,\theta)$

Easy to compute

State distribution

$E(\theta)$

Unknown

Getting more reward

Performance

How can we tell how our change in the policy parameters affected the **action selections** and the performance

How can we tell how our change in the policy parameters affected the **state distribution** and the performance

Δr

Action selection

$\pi(a|s,\theta)$

Easy to compute

Getting more reward

State distribution

$E(\theta)$

Unknown

$\Delta \mu(s)$

Performance

How can we tell how our change in the policy parameters affected the **action selections** and the performance

How can we tell how our change in the policy parameters affected the **state distribution** and the performance

Δr

Action selection

$\pi(a|s,\theta)$

Easy to compute

Getting more reward

State distribution

$E(\theta)$

Unknown

We can't!

$\Delta \mu(s)$

Policy Gradient Theorem

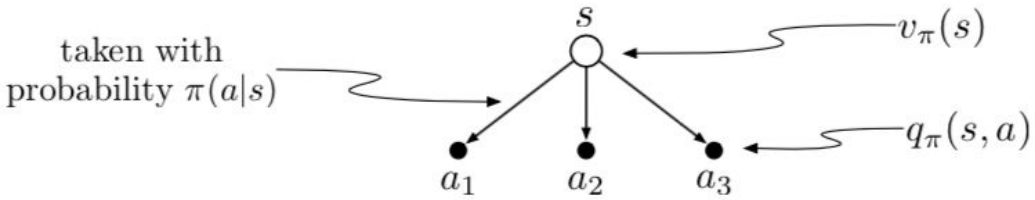
Gradient of performance with respect to the policy parameter **does not involve the derivative of the state distribution**.

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_\theta \pi(a|s,\theta)$$

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t)$$

$$\nabla v_\pi(s) = \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right]$$

Definition of value function



$$\nabla v_\pi(s) = \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right]$$

Definition of value function

R. Sutton (2017)

$$\nabla v_\pi(s) = \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right]$$

Definition of value function

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \right]$$

Bring the gradient inside the sum

$$\nabla v_\pi(s) = \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right]$$

Definition of value function

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \right]$$

Bring the gradient inside the sum

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a)(r + v_\pi(s')) \right]$$

Definition of action-value function

$$\nabla v_\pi(s) = \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right]$$

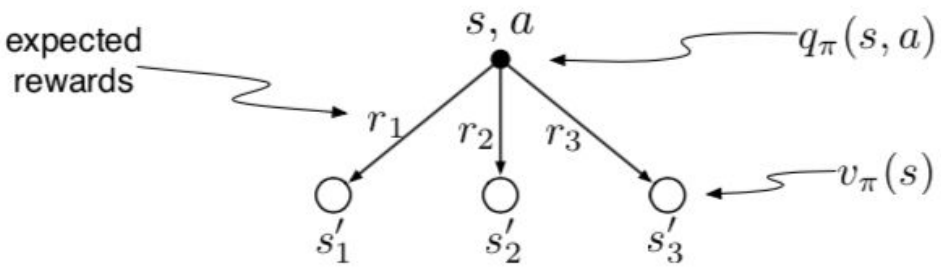
Definition of value function

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \right]$$

Bring the gradient inside the sum

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a)(r + v_\pi(s')) \right]$$

Definition of action-value function



R. Sutton (2017)

$$\nabla v_\pi(s) = \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right]$$

Definition of value function

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \right]$$

Bring the gradient inside the sum

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a)(r + v_\pi(s')) \right]$$

Definition of action-value function

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \right]$$

Gradient is with respect to policy parameters

$$\nabla v_\pi(s) = \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right]$$

Definition of value function

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \right]$$

Bring the gradient inside the sum

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a)(r + v_\pi(s')) \right]$$

Definition of action-value function

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \right]$$

Gradient is with respect to policy parameters

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \sum_a \left[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \right] \right]$$

Substitution

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \left[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \right] \right]$$

$$= \sum_a \nabla \pi(a|s) q_\pi(s,a) + \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \nabla \pi(a'|s') q_\pi(s',a')$$

$$+ \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'')$$

Regrouping terms with respect to the steps it takes starting from current state s

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \left[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \right] \right]$$

$$= \sum_a \nabla \pi(a|s) q_\pi(s,a) + \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \nabla \pi(a'|s') q_\pi(s',a')$$

$$+ \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'')$$

Zero step terms

Regrouping terms with respect to the steps it takes starting from current state s

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \left[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \right] \right]$$

$$= \sum_a \nabla \pi(a|s) q_\pi(s,a) + \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \nabla \pi(a'|s') q_\pi(s',a')$$

$$+ \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'')$$

Zero step terms

One step terms

Regrouping terms with respect to the steps it takes starting from current state s

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \left[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \right] \right]$$

$$= \sum_a \nabla \pi(a|s) q_\pi(s,a) + \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \nabla \pi(a'|s') q_\pi(s',a')$$

$$+ \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'')$$

Zero step terms

One step terms

Two step terms

Regrouping terms with respect to the steps it takes starting from current state s

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \left[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \right] \right]$$

$$= \sum_a \nabla \pi(a|s) q_\pi(s,a) + \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \nabla \pi(a'|s') q_\pi(s',a')$$

$$+ \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'')$$

Zero step terms

One step terms

Two step terms

Regrouping terms with respect to the steps it takes starting from current state s

We can expand the equation till the terminal state, where we have

$$\nabla v_\pi(s) = 0$$

$$= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \big[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \big] \Big]$$

$$= \sum_a \nabla \pi(a|s) q_\pi(s,a) + \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \nabla \pi(a'|s') q_\pi(s',a')$$

$$+ \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'')$$

Zero step terms

One step terms

Two step terms

Regrouping terms with respect to the steps it takes starting from current state s

$$\nabla v_\pi(s) = \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla \pi(a|x) q_\pi(x,a)$$

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \left[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \right] \right]$$

$$= \sum_a \nabla \pi(a|s) q_\pi(s,a) + \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \nabla \pi(a'|s') q_\pi(s',a')$$

$$+ \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'')$$

Regrouping terms with respect to the steps it takes starting from current state s

Zero step terms

One step terms

Two step terms

$$\nabla v_\pi(s) = \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla \pi(a|x) q_\pi(x,a)$$

All possible states x

$$= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \left[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \right] \right]$$

$$= \sum_a \nabla \pi(a|s) q_\pi(s,a) + \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \nabla \pi(a'|s') q_\pi(s',a')$$

$$+ \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'')$$

Regrouping terms with respect to the steps it takes starting from current state s

Zero step terms

One step terms

Two step terms

$$\nabla v_\pi(s) = \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla \pi(a|x) q_\pi(x,a)$$

All possible states x

Possibility of reaching state x from state s in k steps under current policy

$$\nabla J(\theta) = \nabla v_\pi(s_0)$$

Definition

$$\nabla J(\theta) = \nabla v_\pi(s_0)$$

$$= \sum_s \sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \sum_a \nabla \pi(a|s) q_\pi(s, a)$$

Definition

From previous proof

$$
\begin{aligned}
\nabla J(\theta) &= \nabla v_\pi(s_0) && \text{Definition} \\
&= \sum_s \sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \sum_a \nabla \pi(a|s) q_\pi(s,a) && \text{From previous proof} \\
&= \sum_s \eta(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) && \text{Replace variable}
\end{aligned}
$$

$$\nabla J(\theta) = \nabla v_\pi(s_0)$$

Definition

$$= \sum_s \sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \sum_a \nabla \pi(a|s) q_\pi(s, a)$$

From previous proof

$$= \sum_s \eta(s) \sum_a \nabla \pi(a|s) q_\pi(s, a)$$

Replace variable

Denotes the number of time step spent, on average, in state s in a single episode

$$\nabla J(\theta) = \nabla v_\pi(s_0)$$

Definition

$$= \sum_s \sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

From previous proof

$$= \sum_s \eta(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

Replace variable

$$= \Big(\sum_s \eta(s) \Big) \sum_s \frac{\eta(s)}{\sum_s \eta(s)} \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

Algebra

$$\nabla J(\theta) = \nabla v_\pi(s_0)$$

Definition

$$= \sum_s \sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

From previous proof

$$= \sum_s \eta(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

Replace variable

$$= \Big(\sum_s \eta(s) \Big) \sum_s \frac{\eta(s)}{\sum_s \eta(s)} \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

Algebra

$$\propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

Q.E.D

$$\nabla J(\theta) = \nabla v_\pi(s_0)$$

Definition

$$= \sum_s \sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

From previous proof

$$= \sum_s \eta(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

Replace variable

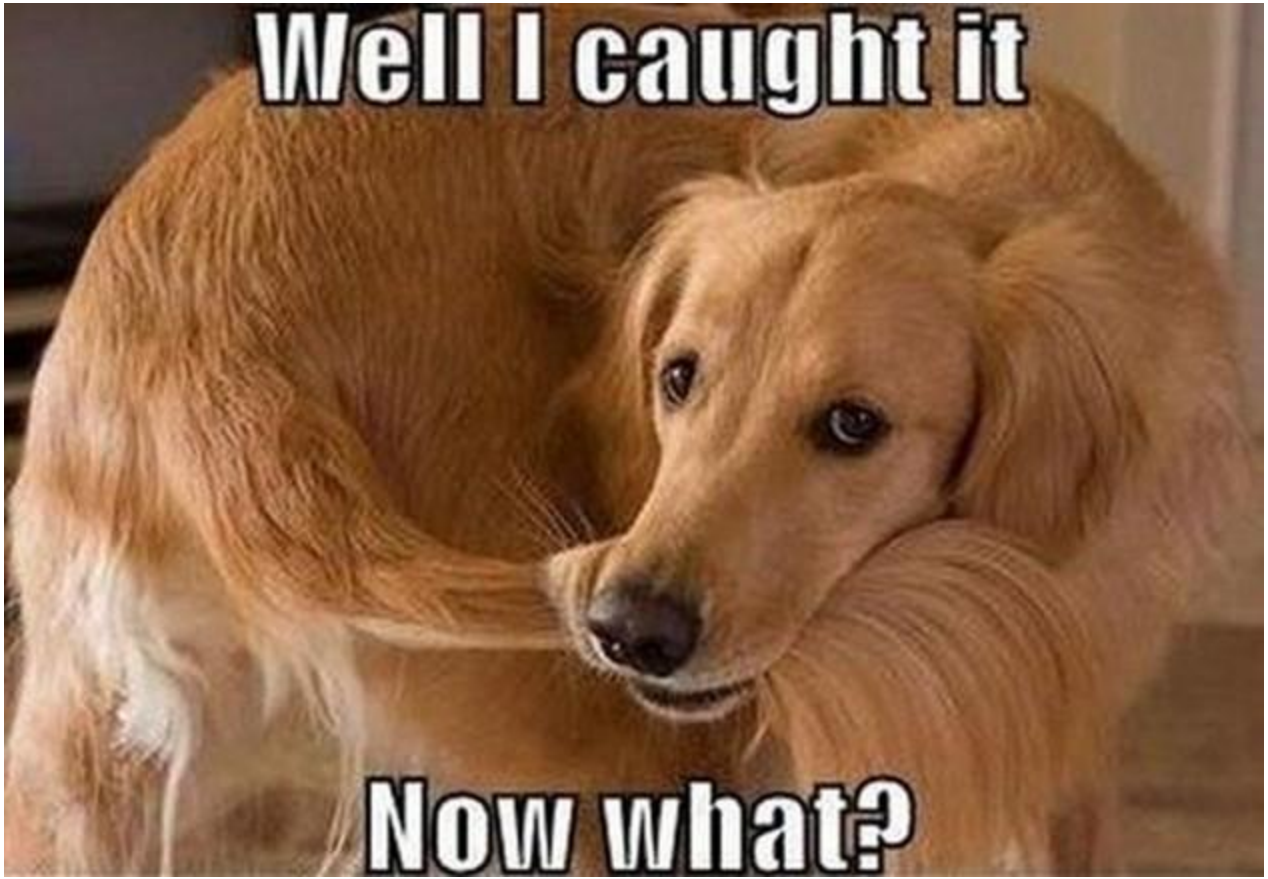
$$= \Big(\sum_s \eta(s)\Big) \sum_s \frac{\eta(s)}{\sum_s \eta(s)} \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

Algebra

$$\propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

Q.E.D

On-policy distribution: the fraction of time spent in each state normalized to one


Well I caught it
Now what?

So far ….

1. Advantages of using policy gradient.
2. Policy gradient theorem

First policy gradient learning algorithm:

REINFORCE: Monte Carlo Policy Gradient

1. Overall Strategy of Policy Gradient

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}$$

What's the requirement of this equation?

First policy gradient learning algorithm:

REINFORCE: Monte Carlo Policy Gradient

1. Overall Strategy of Policy Gradient

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}$$

What's the requirement of this equation?

$E_\pi\left[\widehat{\nabla J(\theta_t)}\right]$ should be proportional to $\nabla J(\theta_t)$

First policy gradient learning algorithm:

REINFORCE: Monte Carlo Policy Gradient

Calculate $\widehat{\nabla J(\theta_t)}$

Start from policy gradient theorem

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_\theta \pi(a|s,\theta)$$

→ Proportional to $\nabla J(\theta_t)$

We are going to change right hand term to sample gradient, $E_\pi \left[\widehat{\nabla J(\theta_t)} \right]$

How?

First policy gradient learning algorithm:

REINFORCE: Monte Carlo Policy Gradient

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_\theta \pi(a|s,\theta)$$

how often the states occur under the target policy, π

$$\sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_\theta \pi(a \mid s,\theta) = E_\pi\left[\sum_a q_\pi(S_t,a) \nabla_\theta \pi(a \mid S_t,\theta)\right]$$

First policy gradient learning algorithm:

REINFORCE: Monte Carlo Policy Gradient

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_\theta \pi(a|s,\theta)$$

$$= E_\pi \left[\sum_a q_\pi(S_t, a) \nabla_\theta \pi(a \mid S_t, \theta) \right]$$

$$\nabla J(\theta) = E_\pi \left[\sum_a \pi(a \mid S_t, \theta) q_\pi(S_t, a) \frac{\nabla_\theta \pi(a|S_t,\theta)}{\pi(a|S_t,\theta)} \right]$$

$$= E_\pi \left[q_\pi(S_t, A_t) \frac{\nabla_\theta \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)} \right] (a \to A_t \sim \pi)$$

$$= E_\pi \left[G_t \frac{\nabla_\theta \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)} \right] \qquad (E_\pi[G_t \mid S_t, A_t] = q_\pi(S_t, A_t))$$

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

complete return from time t, including all future rewards up until the end of episode (Monte Carlo Algorithm)

$$\widehat{\nabla J(\theta_t)} = G_t \frac{\nabla_\theta \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}$$

First policy gradient learning algorithm:

REINFORCE: Monte Carlo Policy Gradient

$$\nabla J(\theta_t) = E\left[G_t \frac{\nabla_\theta \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}\right]$$

$$\widehat{\nabla J(\theta_t)} = G_t \frac{\nabla_\theta \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}$$

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}$$

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla_\theta \pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}$$

$$\theta_{t+1} = \theta_t + \alpha G_t \nabla_\theta \ln \pi(A_t \mid S_t, \theta_t) \quad \frac{\nabla_\theta \pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)} = \nabla_\theta \ln \pi(A_t \mid S_t, \theta_t) \quad \left(\nabla \ln x = \frac{\nabla x}{x}\right)$$

First policy gradient learning algorithm:

REINFORCE: Monte Carlo Policy Gradient

2. Algorithm

Monte-Carlo Policy-Gradient Method(episodic)

Input: a differentiable policy parameters $\pi(a \mid s, \theta)$
Initialize policy parameter θ
Repeat forever:
  Generate an episode $S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T$,following $\pi(\cdot \mid \cdot, \theta)$
  For each step of the episode $t = 0, ..., T-1$:
    $G \leftarrow$ Return from step t
    $\theta \leftarrow \theta + \alpha \gamma^t G \nabla_\theta \ln \pi(A_t \mid S_t, \theta)$

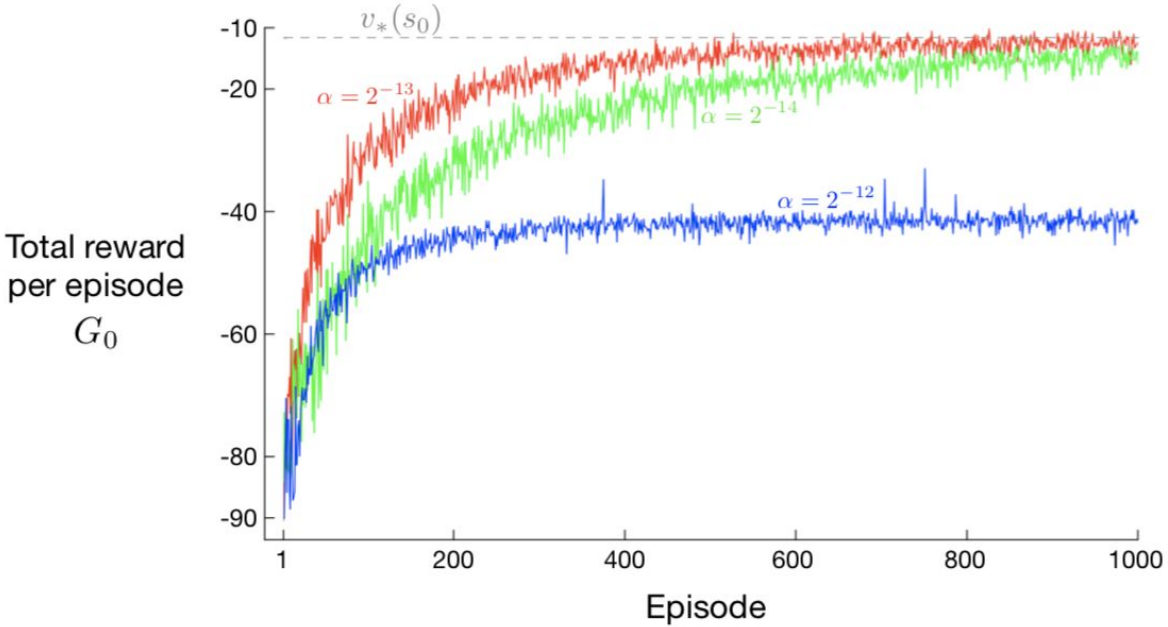
$$\frac{\nabla_\theta \pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)} = \nabla_\theta \ln \pi(A_t \mid S_t, \theta_t) \quad \left(\nabla \ln x = \frac{\nabla x}{x}\right)$$ Eligibility vector

R. Sutton (2017)

First policy gradient learning algorithm:

REINFORCE: Monte Carlo Policy Gradient

3. Analysis



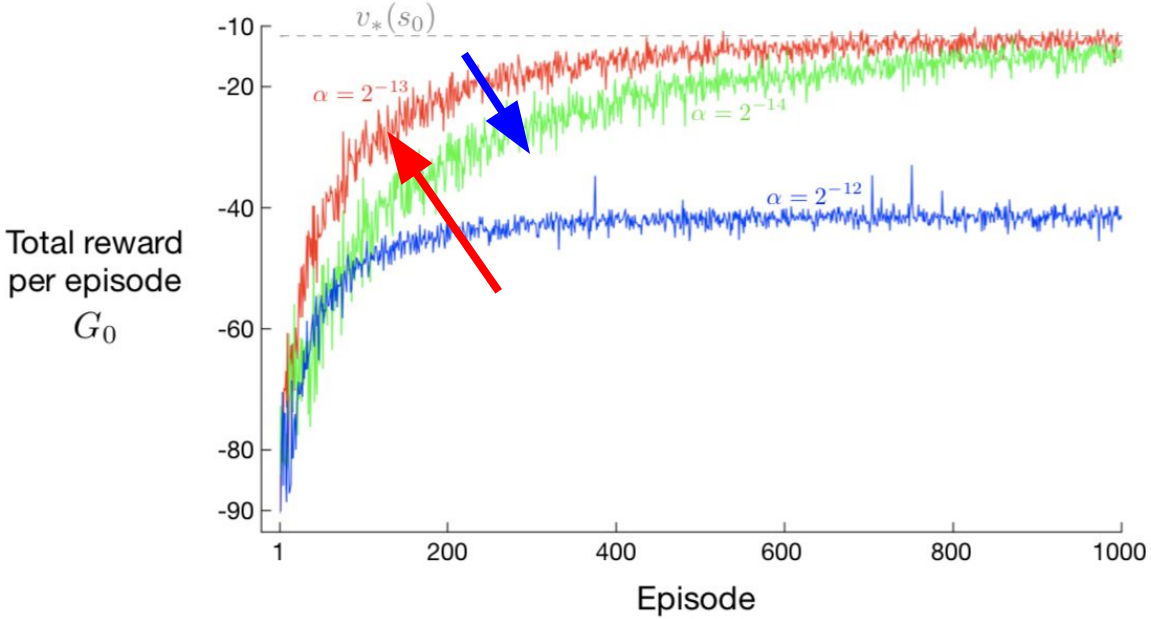
R. Sutton (2017)

First policy gradient learning algorithm:

REINFORCE: Monte Carlo Policy Gradient

3. Analysis

With a good small step size, it approaches to the optimal start state

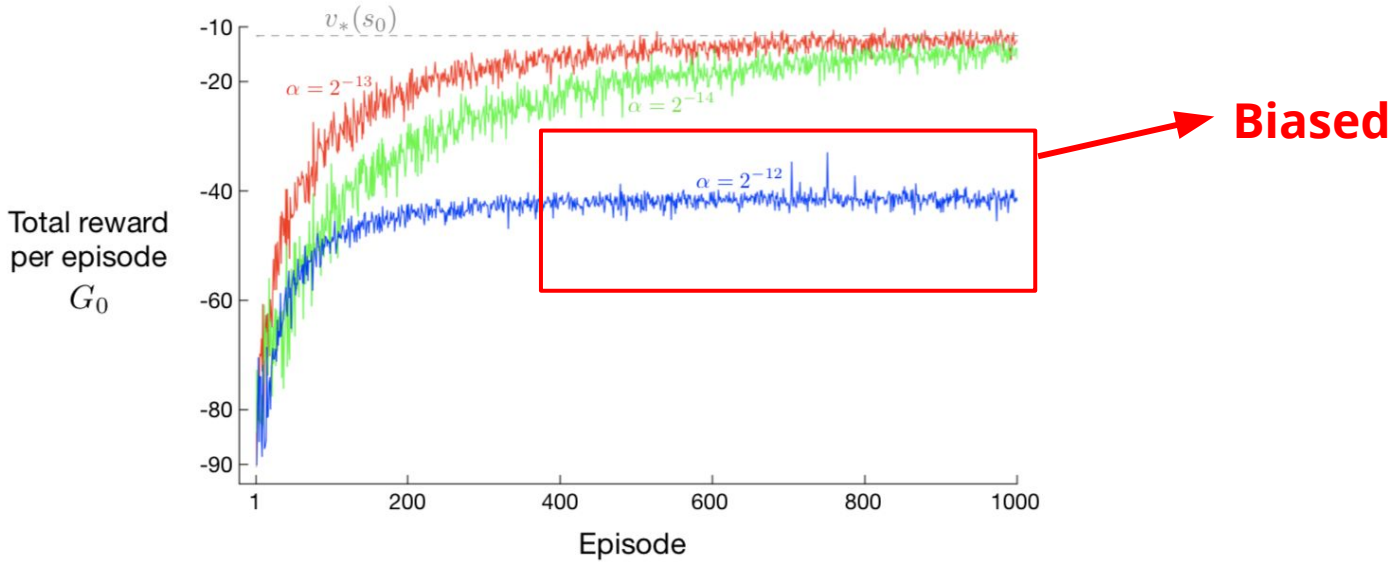


R. Sutton (2017)

First policy gradient learning algorithm:

REINFORCE: Monte Carlo Policy Gradient

3. Analysis

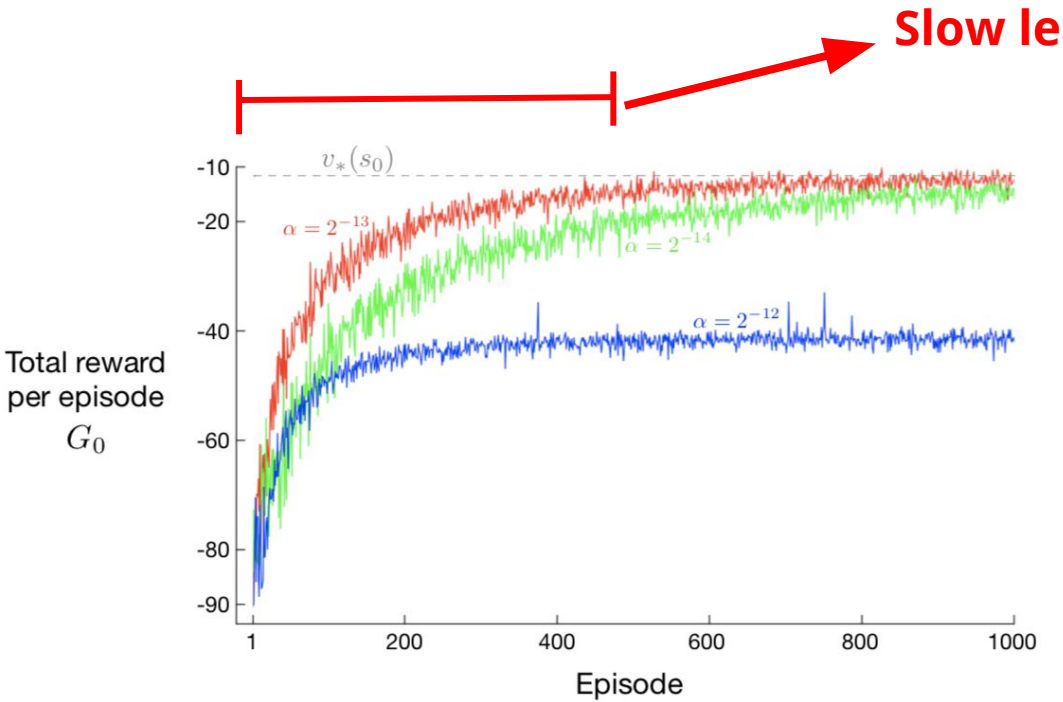


R. Sutton (2017)

First policy gradient learning algorithm:

REINFORCE: Monte Carlo Policy Gradient

3. Analysis



Why?

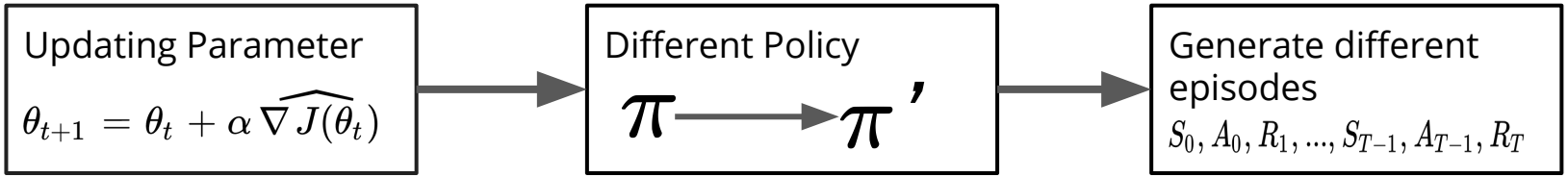
R. Sutton (2017)

First policy gradient learning algorithm:

REINFORCE: Monte Carlo Policy Gradient

3. Analysis

Why this algorithm has high variance?



$$\theta_{t+1} = \theta_t + \alpha G_t \nabla_\theta \ln(A_t \mid S_t, \theta_t)$$

High Variance

R. Sutton (2017)

Second policy gradient learning algorithm:

REINFORCE with Baseline

1. Solution - adding the baseline

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a (q_\pi(s,a) - b(s)) \nabla_\theta \pi(a \mid s, \theta)$$

   a. Constant baseline
      i. High baseline
         1. Actions have high values
      ii. Low baseline
         1. Actions have low values
   b. Variable baseline
      i. Recommended
         1. Estimate of the state value $\hat{v}(S_t, w)$

Second policy gradient learning algorithm:

REINFORCE with Baseline

1. Generalized policy gradient theorem
   a. Add the arbitrary baseline

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a \left(q_\pi(s,a) - b(s)\right) \nabla_\theta \pi(a \mid s, \theta)$$

baseline

Second policy gradient learning algorithm:

REINFORCE with Baseline

1. Generalized policy gradient theorem
   a. Add the arbitrary baseline

baseline

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a (q_\pi(s,a) - b(s)) \nabla_\theta \pi(a \mid s, \theta)$$

Requirement of b(s)

- b(s) should not vary with action a

Second policy gradient learning algorithm:

REINFORCE with Baseline

1. Generalized policy gradient theorem
   a. Add the arbitrary baseline

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a (q_\pi(s,a) - b(s)) \nabla_\theta \pi(a \mid s, \theta)$$

baseline

Requirement of b(s)
- b(s) should not vary with action a

Why?

Second policy gradient learning algorithm:

REINFORCE with Baseline

1. Generalized policy gradient theorem
   a. Add the arbitrary baseline

baseline

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a (q_\pi(s,a) - b(s)) \nabla_\theta \pi(a \mid s, \theta)$$

Requirement of b(s)

- b(s) should not vary with action a

It should not affect to the expectation

$$\sum_a b(s) \nabla_\theta \pi(a \mid s, \theta) = b(s) \nabla_\theta \sum_a \pi(a \mid s, \theta) = b(s) \nabla_\theta 1 = 0$$

Second policy gradient learning algorithm:

REINFORCE with Baseline

1. Generalized policy gradient theorem
   a. Add the arbitrary baseline

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a (q_\pi(s,a) - b(s)) \nabla_\theta \pi(a \mid s, \theta)$$

baseline

New update equation

$$\theta_{t+1} = \theta_t + \alpha(G_t - b(S_t)) \frac{\nabla_\theta \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}$$

Second policy gradient learning algorithm:

REINFORCE with Baseline

2. Algorithm

Monte-Carlo Policy-Gradient Method(episodic)

Input: a differentiable policy parameters $\pi(a \mid s, \theta)$
Input: a differentiable state-value parameterization $\hat{v}(S_t, w)$
Parameters: step sizes $\alpha^\theta > 0$, $\alpha^w > 0$

Initialize policy parameter $\theta \in R^{d'}$ and state_value weights $w \in R^d$
Repeat forever:
Generate an episode $S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot \mid \cdot, \theta)$
For each step of the episode $t = 0, ..., T-1$:

$G \leftarrow$ Return from step t
$\delta \leftarrow G_t - \hat{v}(S_t, w)$
$w \leftarrow w + \alpha^w \gamma^t \delta \nabla_w \hat{v}(S_t, w)$
$\theta \leftarrow \theta + \alpha^\theta \gamma^t \delta \nabla_\theta \ln \pi(A_t \mid S_t, \theta)$

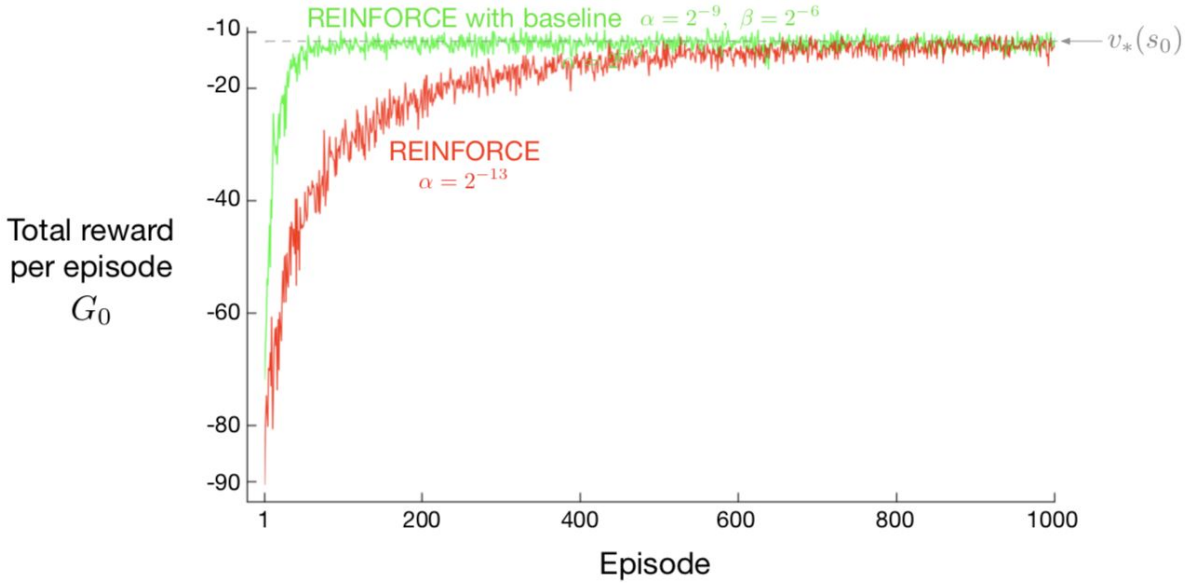
$$\theta_{t+1} = \theta_t + \alpha(G_t - b(S_t)) \frac{\nabla_\theta \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}$$

R. Sutton (2017)

Second policy gradient learning algorithm:

REINFORCE with Baseline

3. Analysis

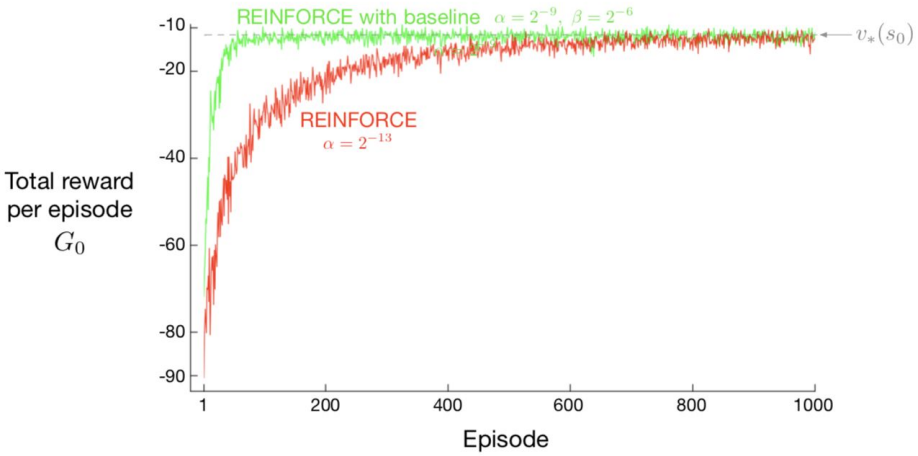


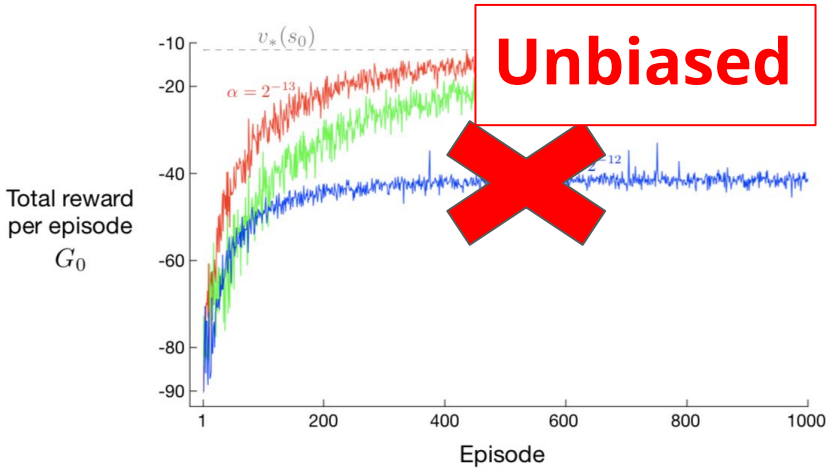
R. Sutton (2017)

Second policy gradient learning algorithm:

REINFORCE with Baseline

3. Analysis



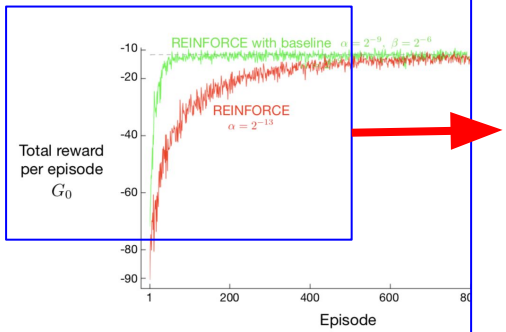


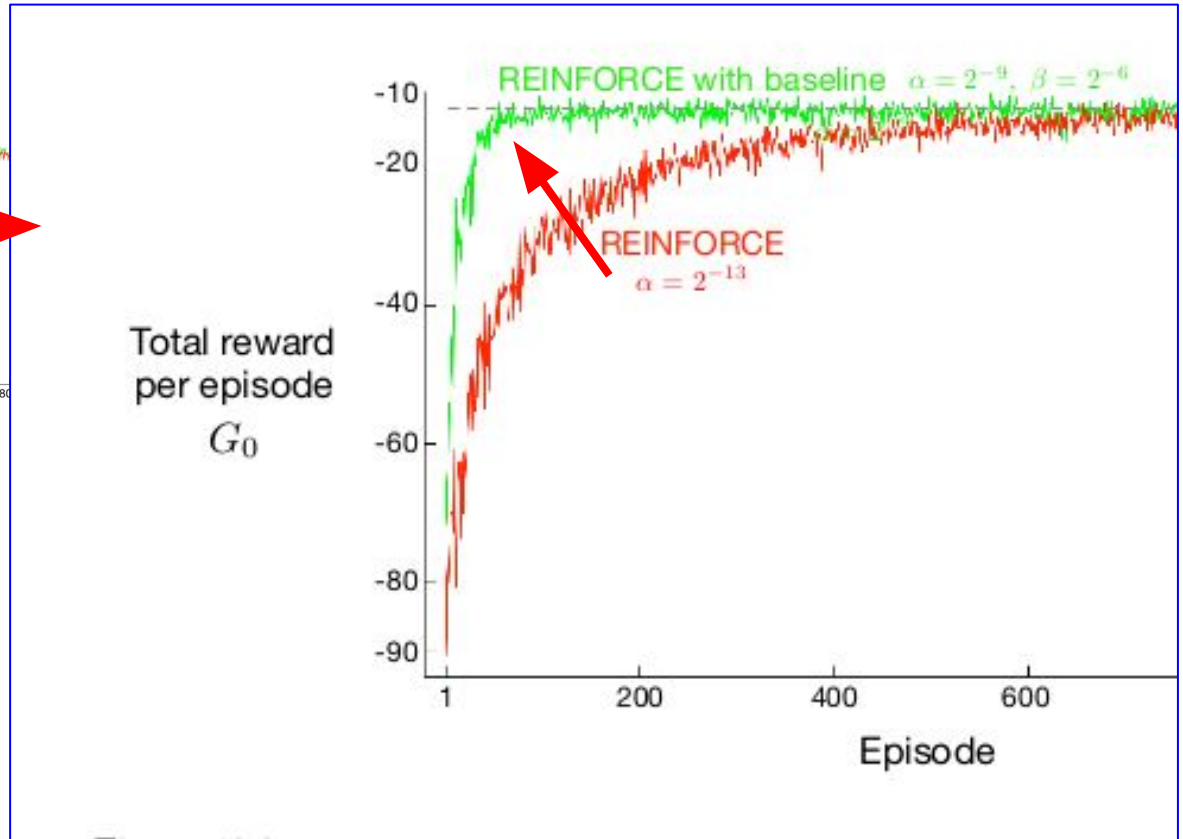
R. Sutton (2017)

Second policy gradient learning algorithm:

REINFORCE with Baseline

3. Analysis





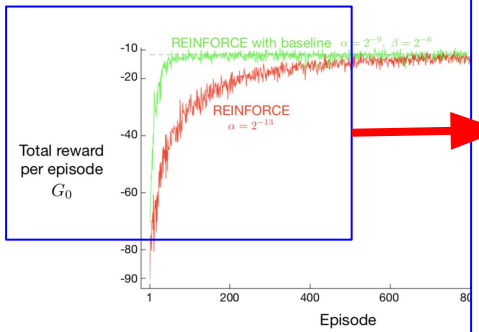
R. Sutton (2017)

Second policy gradient learning algorithm:

REINFORCE with Baseline

3. Analysis

Learns faster than REINFORCE
WHY?

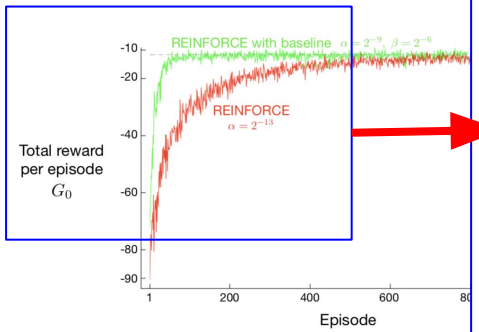


R. Sutton (2017)

Second policy gradient learning algorithm:

REINFORCE with Baseline

3. Analysis



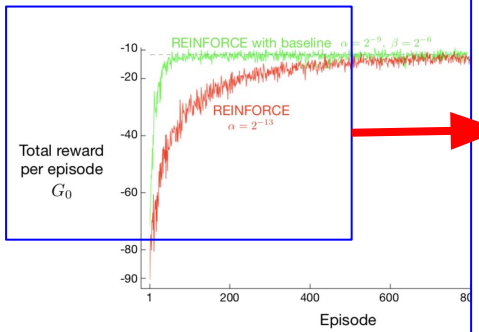
The variance becomes reduced

R. Sutton (2017)

Second policy gradient learning algorithm:

REINFORCE with Baseline

3. Analysis



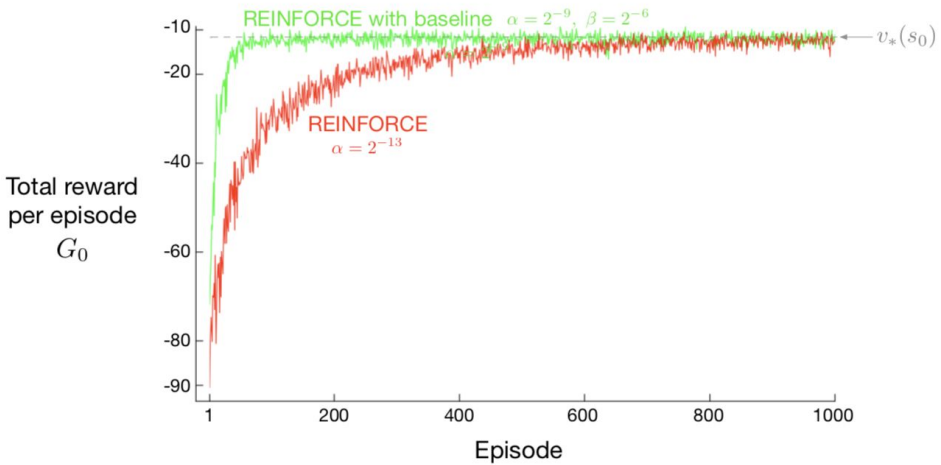
REINFORCE with baseline $\alpha = 2^{-9}$, $\beta = 2^{-6}$
REINFORCE
$\alpha = 2^{-13}$
Total reward per episode
G_0
-20
-40
-60
-80
-90
1
200
400
600
Episode

R. Sutton (2017)

Second policy gradient learning algorithm:

REINFORCE with Baseline

3. Analysis



- Unbiased
- Converge asymptotically to a local minimum
- Slow learning (still high variance)
- Inconvenient to implement online or for continuing problems

⇨Actor-Critic Method

R. Sutton (2017)

Third policy gradient learning algorithm:

Actor-Critic Methods

Difference between Actor-Critic Methods and REINFORCE

Third policy gradient learning algorithm:

Actor-Critic Methods

Difference between Actor-Critic Methods and REINFORCE

Actor: The model updating the policy parameters θ, for $\pi_\theta(a \mid s)$, in the direction suggested by the critic

Critic: The model updating the value function parameters w and depending on the algorithm it could be action-value $Q_w(a \mid s)$or state-value $V_w(s)$

Third policy gradient learning algorithm:

Actor-Critic Methods

1. One-step Actor-Critic

Replace the full return of REINFORCE with the one-step return

$$\theta_{t+1} = \theta_t + \alpha(G_{t:t+1} - \hat{v}(S_t, w)) \frac{\nabla_\theta \pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}$$

$$= \theta_t + \alpha(R_{t+1} + \gamma \hat{v}(S_{t+1}, w) - \hat{v}(S_t, w)) \frac{\nabla_\theta \pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}$$

$$= \theta_t + \alpha \delta_t \frac{\nabla_\theta \pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}$$

one-step return

Third policy gradient learning algorithm:

Actor-Critic Methods

2. Algorithm

: Actor

: Critic

One-step Actor-Critic (episodic)

Input: a differentiable policy parameters $\pi(a \mid s, \theta)$
Input: a differentiable state-value parameterization $\hat{v}(S_t, w)$
Parameters: step sizes $\alpha^\theta > 0$, $\alpha^w > 0$

Initialize policy parameter $\theta \in R^{d'}$ and state_value weights $w \in R^d$
Repeat forever (for each episode):

Initialize S (first state of episode)

$I \leftarrow 1$

Loop while S is not the terminal (for each time step):

$A \sim \pi(\cdot|S, \theta)$

Take action A, observe S', R

$\delta \leftarrow R + \gamma \hat{v}(S', w) - \hat{v}(S, w)$ (if S 0 is terminal, then v̂(S 0 ,w) = 0)

$w \leftarrow w + \alpha^w I \delta \bigtriangledown_w ln\hat{v}(S, w)$

$\theta \leftarrow \theta + \alpha^\theta I \delta \bigtriangledown_\theta ln\pi(A|S, \theta)$

$I \leftarrow \gamma I$

$S \leftarrow S'$

R. Sutton (2017)

Third policy gradient learning algorithm:

Actor-Critic Methods

Actor–Critic with Eligibility Traces (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: trace-decay rates $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\lambda^{\mathbf{w}} \in [0,1]$; step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever (for each episode):
  Initialize S (first state of episode)
  $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
  $\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)
  $I \leftarrow 1$
  While S is not terminal (for each time step):
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{z}^{\mathbf{w}} \leftarrow \gamma\lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + I\nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})$
    $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \gamma\lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + I\nabla_{\boldsymbol{\theta}}\ln\pi(A|S,\boldsymbol{\theta})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

R. Sutton (2017)

Third policy gradient learning algorithm:

Actor-Critic Methods

Actor–Critic with Eligibility Traces (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: trace-decay rates $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\lambda^{\mathbf{w}} \in [0,1]$; step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever (for each episode):
  Initialize S (first state of episode)
  $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
  $\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)
  $I \leftarrow 1$
  While S is not terminal (for each time step):
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{z}^{\mathbf{w}} \leftarrow \gamma\lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + I\nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})$
    $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \gamma\lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + I\nabla_{\boldsymbol{\theta}}\ln\pi(A|S,\boldsymbol{\theta})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Eligibility vector

$$\frac{\nabla_\theta \pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)} = \nabla_\theta \ln \pi(A_t \mid S_t, \theta_t)$$

$$\left(\nabla \ln x = \frac{\nabla x}{x}\right)$$

R. Sutton (2017)

Third policy gradient learning algorithm:

Actor-Critic Methods

Actor–Critic with Eligibility Traces (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: trace-decay rates $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\lambda^{\mathbf{w}} \in [0,1]$; step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever (for each episode):
  Initialize S (first state of episode)
  $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
  $\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)
  $I \leftarrow 1$
  While S is not terminal (for each time step):
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{z}^{\mathbf{w}} \leftarrow \gamma\lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + I\nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})$
    $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \gamma\lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + I\nabla_{\boldsymbol{\theta}}\ln\pi(A|S,\boldsymbol{\theta})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

$$\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$$
$$\mathbf{z}^{\mathbf{w}} \leftarrow \gamma\lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + I\nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})$$
$$\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \gamma\lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + I\nabla_{\boldsymbol{\theta}}\ln\pi(A|S,\boldsymbol{\theta})$$
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$$
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$$

REINFORCE with baseline

$$\delta \leftarrow G_t - \hat{v}(S_t,\mathbf{w})$$
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\gamma^t\delta\nabla_{\mathbf{w}}\hat{v}(S_t,\mathbf{w})$$
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\gamma^t\delta\nabla_{\boldsymbol{\theta}}\ln\pi(A_t|S_t,\boldsymbol{\theta})$$

$$G_t = \sum_{k=t+1}^{T}\gamma^{k-t-1}R_k$$

R. Sutton (2017)

Third policy gradient learning algorithm:

Actor-Critic Methods

Actor–Critic with Eligibility Traces (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: trace-decay rates $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\lambda^{\mathbf{w}} \in [0,1]$; step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever (for each episode):
  Initialize S (first state of episode)
  $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
  $\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)
  $I \leftarrow 1$
  While S is not terminal (for each time step):
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{z}^{\mathbf{w}} \leftarrow \gamma\lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + I\nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})$
    $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \gamma\lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + I\nabla_{\boldsymbol{\theta}}\ln\pi(A|S,\boldsymbol{\theta})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

$$\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$$
$$\mathbf{z}^{\mathbf{w}} \leftarrow \gamma\lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + I\nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})$$
$$\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \gamma\lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + I\nabla_{\boldsymbol{\theta}}\ln\pi(A|S,\boldsymbol{\theta})$$
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$$
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$$

REINFORCE with baseline

$$\delta \leftarrow G_t - \hat{v}(S_t,\mathbf{w})$$
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\gamma^t\delta\nabla_{\mathbf{w}}\hat{v}(S_t,\mathbf{w})$$
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\gamma^t\delta\nabla_{\boldsymbol{\theta}}\ln\pi(A_t|S_t,\boldsymbol{\theta})$$

$$G_t = \sum_{k=t+1}^{T}\gamma^{k-t-1}R_k$$

R. Sutton (2017)

Reinforcement Learning Tasks

Episodic Task

Continuous Task

Policy Gradient for Continuing Problems

What are continuing problems?

Policy Gradient for Continuing Problems

What are continuing problems?

Problems that do not have the episode boundary: the agent can go on forever

Policy Gradient for Continuing Problems

What are continuing problems?

Problems that do not have the episode boundary: the agent can go on forever

What changed?

Policy Gradient for Continuing Problems

In episodic cases we defined the performance measurement as

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

Policy Gradient for Continuing Problems

In episodic cases we defined the performance measurement as

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

Can we do the same for continuing problems?

Policy Gradient for Continuing Problems

In episodic cases we defined the performance measurement as

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

Can we do the same for continuing problems?

No!

Policy Gradient for Continuing Problems

In episodic cases we defined the performance measurement as

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

Can we do the same for continuing problems?

No!

Why not?

Policy Gradient for Continuing Problems

In episodic cases we defined the performance measurement as

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

Can we do the same for continuing problems?

No!

It will go to infinity!

Policy Gradient for Continuing Problems

In episodic cases we defined the performance measurement as

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

Solution:
We can define it as

- the average value

$$J(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$$

- the average rate of reward per time step

stationary distribution of Markov Chain of the policy

$$J(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s, a) \mathrm{R}_s^a$$

Policy Gradient for Continuing Problems

One-Step MDPs

- Consider a simple class of one-step MDPs
  - Starting in state $s \sim d^{\pi_\theta}(s)$
  - Terminating after one time-step with reward $r = \mathrm{R}(s, a)$
- The objective function

$$J(\theta) = \mathbb{E}_{\pi_\theta}[r]$$

$$= \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s, a) \mathrm{R}(s, a)$$

Expansion of expectation value

Policy Gradient for Continuing Problems

One-Step MDPs

- Consider a simple class of one-step MDPs
  - Starting in state $s \sim d^{\pi_\theta}(s)$
  - Terminating after one time-step with reward $r = \mathrm{R}(s,a)$
- The objective function

$$J(\theta) = \mathbb{E}_{\pi_\theta}[r]$$

$$= \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s,a) \mathrm{R}(s,a)$$

Likelihood ratio trick: same as before

$$\nabla J(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \nabla \pi_\theta(s,a) \mathrm{R}(s,a)$$

$$= \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s,a) \nabla ln \pi_\theta(s,a) \mathrm{R}(s,a)$$

Policy Gradient for Continuing Problems

One-Step MDPs

- Consider a simple class of one-step MDPs
  - Starting in state $s \sim d^{\pi_\theta}(s)$
  - Terminating after one time-step with reward $r = \mathrm{R}(s,a)$
- The objective function

$$J(\theta) = \mathbb{E}_{\pi_\theta}[r]$$

$$= \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s,a) \mathrm{R}(s,a)$$

Likelihood ratio trick: same as before

$$\nabla J(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \nabla \pi_\theta(s,a) \mathrm{R}(s,a)$$

$$= \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s,a) \nabla ln \pi_\theta(s,a) \mathrm{R}(s,a)$$

eligibility vector

$$= \mathbb{E}_{\pi_\theta}[\nabla ln \pi_\theta(s,a) r]$$

@David Silver "Policy Gradient"

Policy Gradient for Continuing Problems

Generalized to Multi-Step MDPs

- replace reward r with long-term value $q_\pi(s,a)$
- In fact, policy gradient theorem applies to start state objective, average reward and average value objective

Estimate of reward

Theorem

For any differentiable policy $\pi_\theta(s,a)$,
for any of the policy objective functions $J = J_1, J_{avR}$, *or* $\frac{1}{1-\gamma} J_{avV}$,
the policy gradient is

$$\bigtriangledown J(\theta) = \mathbb{E}_{\pi_\theta}[\bigtriangledown ln \pi_\theta(s,a) q_\pi(s,a)]$$

eligibility vector

For continuing task, different parameterization for $\pi(a|s,\theta)$

@David Silver “Policy Gradient”

Policy Parameterization for Continuous Actions

We learn statistics of the probability distribution.
e.g. actions are chosen from a Gaussian distribution.

mean

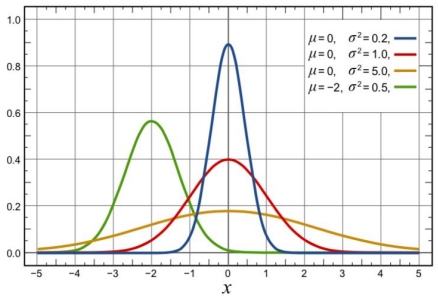
$$\pi(a|s,\theta) \doteq \frac{1}{\sigma(s,\theta)\sqrt{2\pi}} exp\left(-\frac{(a-\mu(s,\theta))^2}{2\sigma(s,\theta)^2}\right)$$

density of action

std

state

The probability density function of the normal distribution for different means and variances



R. Sutton (2017)

Policy Parameterization for Continuous Actions

A policy parameterization with mean and std given by parametric function approximators depend on the state

$$\pi(a|s,\theta) \doteq \frac{1}{\sigma(s,\theta)\sqrt{2\pi}} exp\left(-\frac{(a-\mu(s,\theta))^2}{2\sigma(s,\theta)^2}\right)$$

Mean function $$\mu(s,\theta) \doteq \theta_\mu^T \phi_\mu(s)$$

Std function $$\sigma(s,\theta) \doteq exp(\theta_\sigma^T \phi_\sigma(s))$$

state feature vector

Action $$a \sim \mathcal{N}(\mu(s), \sigma(s)^2)$$

Policy Parameterization for Continuous Actions

A policy parameterization with mean and std given by parametric function approximators depend on the state

$$\pi(a|s,\theta) \doteq \frac{1}{\sigma(s,\theta)\sqrt{2\pi}} exp(-\frac{(a-\mu(s,\theta))^2}{2\sigma(s,\theta)^2})$$

Eligibility Vectors

$$\bigtriangledown ln\pi_{\theta_\mu}(a|s,\theta) = \frac{(a-\mu(s))\phi_\mu(s)}{\sigma(s)^2}$$

$$\bigtriangledown ln\pi_{\theta_\sigma}(a|s,\theta) = (\frac{(a-\mu(s))^2}{\sigma(s)^2} - 1)\phi_\sigma(s)$$

Policy Parameterization for Continuous Actions

A policy parameterization with mean and std given by parametric function approximators depend on the state

$$\pi(a|s,\theta) \doteq \frac{1}{\sigma(s,\theta)\sqrt{2\pi}} exp\left(-\frac{(a-\mu(s,\theta))^2}{2\sigma(s,\theta)^2}\right)$$

Take the gradient

$$\bigtriangledown J(\theta) \approx \sum_i \left(\sum_t \bigtriangledown ln\pi_\theta(a_t^i|s_t^i)\right)\left(\sum_t r(s_t^i, a_t^i)\right]$$

Update the parameter

$$\theta \leftarrow \theta + \alpha \bigtriangledown J(\theta)$$

Policy Parameterization for Continuous Actions

Now, apply all the algorithms learned before to learn real-valued actions

- REINFORCE

$$\bigtriangledown J(\theta) = \mathbb{E}_{\pi_\theta}[\bigtriangledown ln\pi_\theta(s,a) G_t]$$

total return

- REINFORCE with Baseline

$$\bigtriangledown J(\theta) = \mathbb{E}_{\pi_\theta}[\bigtriangledown ln\pi_\theta(s,a) (G_t - b)]$$

add baseline

- One-step TD Actor-Critic

$$\bigtriangledown J(\theta) = \mathbb{E}_{\pi_\theta}[\bigtriangledown ln\pi_\theta(s,a) (R + \gamma\hat{v}(s') - \hat{v}(s))]$$

one-step TD error

- Q Actor-Critic

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

critic

- Advantage Actor-Critic

$$\bigtriangledown J(\theta) = \mathbb{E}_{\pi_\theta}[\bigtriangledown ln\pi_\theta(s,a) (Q^w(s,a) - V(s))]$$

advantage function

Policy Parameterization for Continuous Actions

Now, apply all the algorithms learned before to learn real-valued actions

- Natural Policy Gradient

$$\bigtriangledown J(\theta) = \mathbb{E}_{\pi_\theta}[\bigtriangledown ln\pi_\theta(s,a) \bigtriangledown ln\pi_\theta(s,a)^T w]$$

critic parameter

How to get this?

Policy Parameterization for Continuous Actions

Now, apply all the algorithms learned before to learn real-valued actions

- Natural Policy Gradient

$$\bigtriangledown J(\theta) = \mathbb{E}_{\pi_\theta}[\bigtriangledown ln\pi_\theta(s,a) \bigtriangledown ln\pi_\theta(s,a)^T w]$$

Remember: there is a bias when approximating the policy gradient

Expected: $$\bigtriangledown J(\theta) = \mathbb{E}_{\pi_\theta}[\bigtriangledown ln\pi_\theta(s,a) Q^{\pi_\theta}(s,a)]$$

Approximate: $$\bigtriangledown J(\theta) = \mathbb{E}_{\pi_\theta}[\bigtriangledown ln\pi_\theta(s,a) Q^{w}(s,a)]$$

$$Q_w(s,a) \approx Q^{\pi_\theta}(s,a)$$

A biased policy gradient may not find the right solution

Policy Parameterization for Continuous Actions

Now, apply all the algorithms learned before to learn real-valued actions

- Natural Policy Gradient

$$\bigtriangledown J(\theta) = \mathbb{E}_{\pi_\theta}[\bigtriangledown ln\pi_\theta(s,a) \bigtriangledown ln\pi_\theta(s,a)^T w]$$

bias: $Q_w(s,a) \approx Q^{\pi_\theta}(s,a)$

To reduce the bias, we minimize mean square error

$$\varepsilon = \mathbb{E}_{\pi_\theta}[(Q^{\pi_\theta}(s,a) - Q_w(s,a))^2]$$

Take the gradient of ε respect to w to be 0

Policy Parameterization for Continuous Actions

Now, apply all the algorithms learned before to learn real-valued actions

- Natural Policy Gradient

$$\bigtriangledown J(\theta) = \mathbb{E}_{\pi_\theta}[\bigtriangledown ln\pi_\theta(s,a) \bigtriangledown ln\pi_\theta(s,a)^T w$$

MSE: $$\varepsilon = \mathbb{E}_{\pi_\theta}[(Q^{\pi_\theta}(s,a) - Q_w(s,a))^2]$$

If $$\bigtriangledown_w Q_w(s,a) = \bigtriangledown_\theta ln\pi_\theta(s,a)$$

$$\nabla_w \varepsilon = 0$$

$$\mathbb{E}_{\pi_\theta}\left[(Q^\theta(s,a) - Q_w(s,a))\nabla_w Q_w(s,a)\right] = 0$$

$$\mathbb{E}_{\pi_\theta}\left[(Q^\theta(s,a) - Q_w(s,a))\nabla_\theta \log \pi_\theta(s,a)\right] = 0$$

exactly same $$\mathbb{E}_{\pi_\theta}\left[Q^\theta(s,a)\nabla_\theta \log \pi_\theta(s,a)\right] = \mathbb{E}_{\pi_\theta}\left[Q_w(s,a)\nabla_\theta \log \pi_\theta(s,a)\right]$$

Policy Parameterization for Continuous Actions

Now, apply all the algorithms learned before to learn real-valued actions

- Natural Policy Gradient

$$\bigtriangledown J(\theta) = \mathbb{E}_{\pi_\theta}[\bigtriangledown ln\pi_\theta(s,a) \bigtriangledown ln\pi_\theta(s,a)^T w]$$

How to achieve? $\bigtriangledown_w Q_w(s,a) = \bigtriangledown_\theta ln\pi_\theta(s,a)$

If $Q_w(s,a) = w^T \phi(s,a)$

$ln\pi_\theta(s,a) = \theta^T \phi(s,a)$

then $\bigtriangledown_w Q_w(s,a) = \phi(s,a) = \bigtriangledown_\theta ln\pi_\theta(s,a)$ achieved

Policy Parameterization for Continuous Actions

Now, apply all the algorithms learned before to learn real-valued actions

- Natural Policy Gradient

$$\bigtriangledown J(\theta) = \mathbb{E}_{\pi_\theta}[\bigtriangledown ln\pi_\theta(s,a) \bigtriangledown ln\pi_\theta(s,a)^T w]$$

How to achieve? $\bigtriangledown_w Q_w(s,a) = \bigtriangledown_\theta ln\pi_\theta(s,a)$

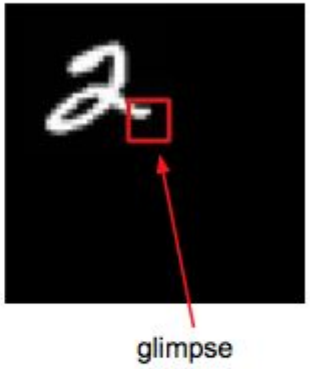
If $Q_w(s,a) = w^T \phi(s,a)$

$ln\pi_\theta(s,a) = \theta^T \phi(s,a)$

then $\bigtriangledown_w Q_w(s,a) = \phi(s,a) = \bigtriangledown_\theta ln\pi_\theta(s,a)$

$Q_w(s,a) = w^T \phi(s,a) = \bigtriangledown_\theta ln\pi_\theta(s,a)^T w$ achieved

Applications



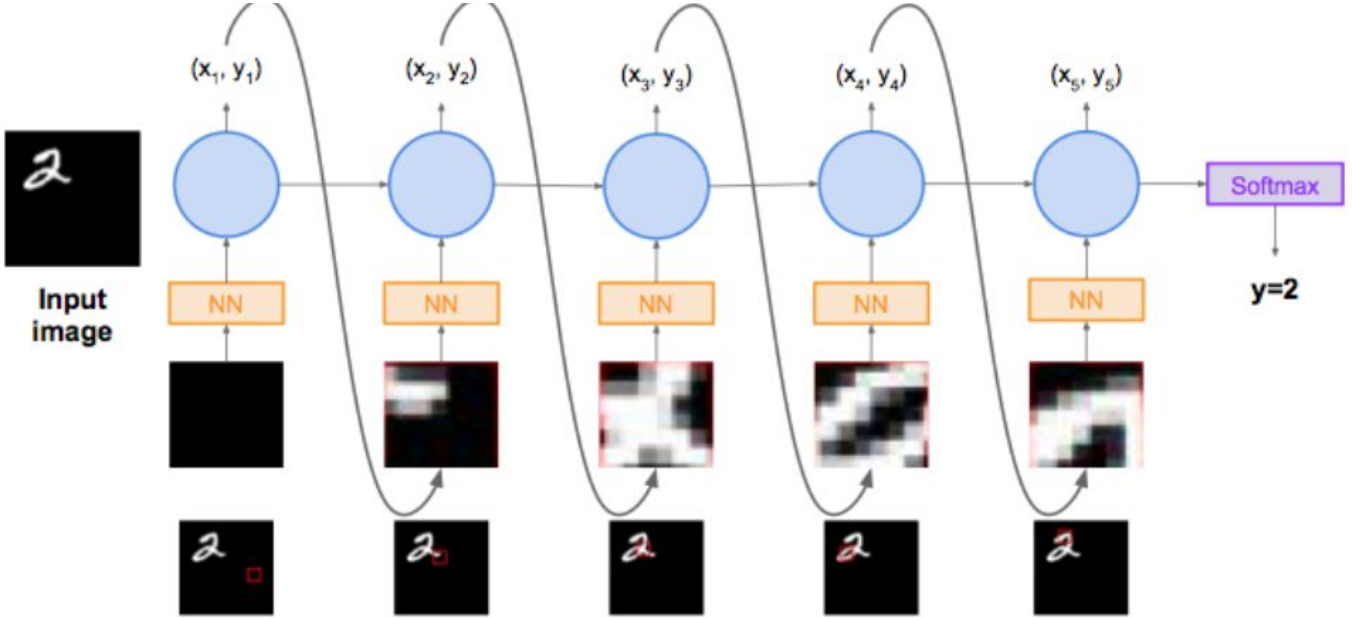
Recurrent Attention Model (RAM)

- Objective: Image Classification
  - Take a sequence of “glimpses” selectively focusing on regions of image to predict class
- State: Glimpses seen so far
- Action: (x,y) coordinates (glimpses center) of where to look next in image
- Reward: 1 at the final timestep if image correctly classified, 0 otherwise

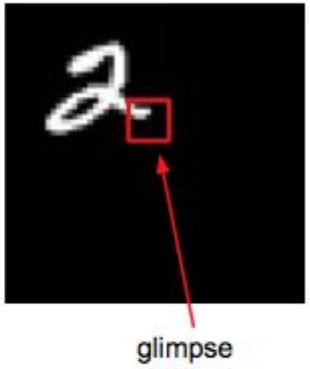
Mnih et al. (2014)

Applications

Recurrent Attention Model (RAM)



Applications



Recurrent Attention Model (RAM)

- Objective: Image Classification
  - Take a sequence of “glimpses” selectively focusing on regions of image to predict class
- State: Glimpses seen so far
- Action: (x,y) coordinates (glimpses center) of where to look next in image
- Reward: 1 at the final timestep if image correctly classified, 0 otherwise

Training Goal: maximize the total reward the agent can expect when interacting with the environment.

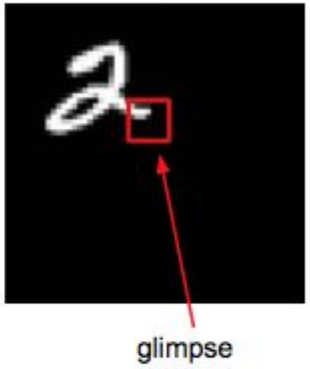
$$J(\theta) = \mathbb{E}_{p(s_{1:T};\theta)}\left[\sum_{t=1}^{T} r_t\right] = \mathbb{E}_{p(s_{1:T};\theta)}\left[R\right]$$

depends on the policy

A distribution over possible interaction sequences

Mnih et al. (2014)

Applications



Recurrent Attention Model (RAM)

- Objective: Image Classification
  - Take a sequence of “glimpses” selectively focusing on regions of image to predict class
- State: Glimpses seen so far
- Action: (x,y) coordinates (glimpses center) of where to look next in image
- Reward: 1 at the final timestep if image correctly classified, 0 otherwise

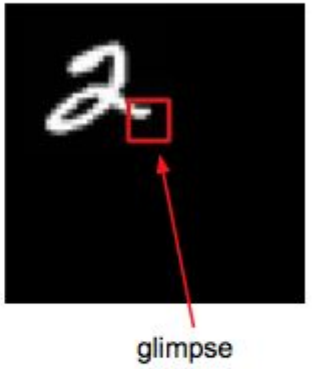
REINFORCE:

$$\nabla_\theta J = \sum_{t=1}^{T} \mathbb{E}_{p(s_{1:T};\theta)} \left[\nabla_\theta \log \pi(u_t | s_{1:t}; \theta) R \right]$$

$$\approx \frac{1}{M} \sum_{i=1}^{M} \sum_{t=1}^{T} \nabla_\theta \log \pi(u_t^i | s_{1:t}^i; \theta) R^i,$$

Mnih et al. (2014)

Applications



Recurrent Attention Model (RAM)

- Objective: Image Classification
  - Take a sequence of “glimpses” selectively focusing on regions of image to predict class
- State: Glimpses seen so far
- Action: (x,y) coordinates (glimpses center) of where to look next in image
- Reward: 1 at the final timestep if image correctly classified, 0 otherwise

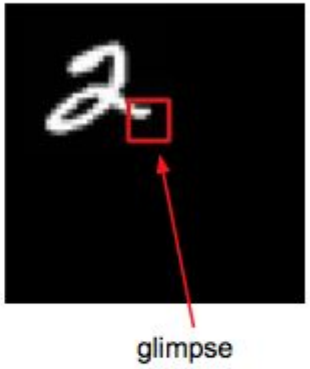
Add Baseline to reduce variance:

$$\frac{1}{M}\sum_{i=1}^{M}\sum_{t=1}^{T}\nabla_\theta \log \pi(u_t^i|s_{1:t}^i;\theta)\left(R_t^i - b_t\right)$$

the gradient of RNN (backpropagation)

Mnih et al. (2014)

Applications



Recurrent Attention Model (RAM)

- Objective: Image Classification
  - Take a sequence of “glimpses” selectively focusing on regions of image to predict class
- State: Glimpses seen so far
- Action: (x,y) coordinates (glimpses center) of where to look next in image
- Reward: 1 at the final timestep if image correctly classified, 0 otherwise

Learning policy for how to take glimpse actions using REINFORCE, given state of glimpses seen so far, use RNN to model the state and output next action

Mnih et al. (2014)

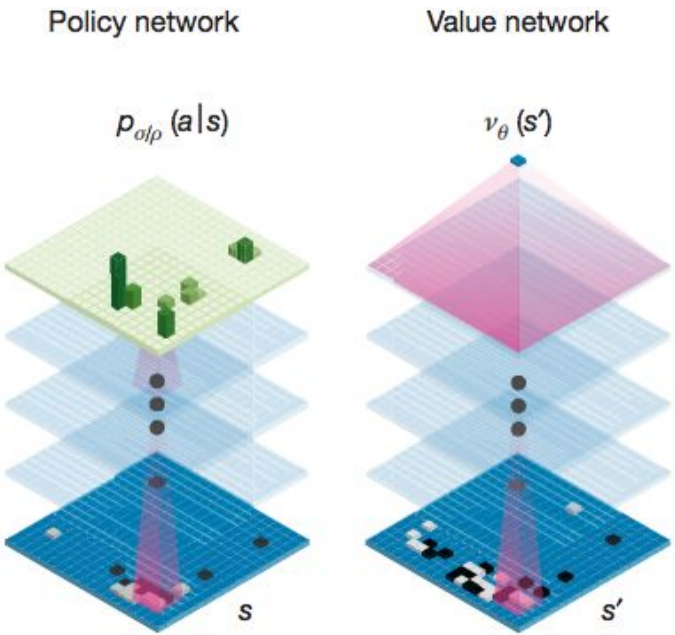
Applications

AlphaGo

March 2016: AlphaGo beats18-time world champion Lee Sedol

Applications

AlphaGo



Silver et al. (2016)

Policy Network: self-play with REINFORCE

The update of the weight:

$$\Delta\rho \propto \frac{\partial \log p_\rho(a_t|s_t)}{\partial \rho} z_t$$

the policy network

the network weight

the terminal reward

Sampling the action from its output probability distribution over actions.

Applications

AlphaGo

AlphaGo Zero [Nature 2017]: beat AlphaGo 100:0

Alpha Zero (Dec 2017): generalized to chess and shogi

Suggested Readings

- Classic papers
  - Williams (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning: introduces REINFORCE algorithm
  - Baxter & Bartlett (2001). Infinite-horizon policy-gradient estimation: temporally decomposed policy gradient (not the first paper on this! see actor-critic section later)
  - Peters & Schaal (2008). Reinforcement learning of motor skills with policy gradients: very accessible overview of optimal baselines and natural gradient
- Deep reinforcement learning policy gradient papers
  - Levine & Koltun (2013). Guided policy search: deep RL with importance sampled policy gradient (unrelated to later discussion of guided policy search)
  - Schulman, L., Moritz, Jordan, Abbeel (2015). Trust region policy optimization: deep RL with natural policy gradient and adaptive step size
  - Schulman, Wolski, Dhariwal, Radford, Klimov (2017). Proximal policy optimization algorithms: deep RL with importance sampled policy gradient

References

- Sutton, Richard S., and Andrew G. Barto. ***Reinforcement learning: An introduction***. MIT press, 2018.
- Tom Mitchell, Katerina Fragkiadaki. ***Deep Reinforcement Learning***. Fall 2018