Lecture 8

Value based RL	Policy based RL
Difference	
• SARSA, Q-learning • Find optimal value function and find the policy of optimal value (policy extraction) • Slow convergence but solves harder control problems	• REINFORCE • Policy evaluation and then policy improvement • Relatively faster convergence but appropriate for simpler control problems

https://www.researchgate.net/publication/329368817_Deep_Reinforcement_Learning_for_Soft_Robotic_Applications_Brief_Overview_with_Impending_Challenges

Value based RL	Policy based RL

Difference in procedure (iteration)

Value based RL

1. Approximate value function using parameters
2. Policy generated from value function using e-greedy

Policy based RL

- Collect set of data (trajectories) using the current policy
- Compute the policy gradient
- Apply gradient on SGD or ADAM

lecture 8

Value based RL	Policy based RL

Difference (simple)

Value based RL

- Learnt value function
- Implicit policy (cannot be certain with e chance of randomity)

Policy based RL

- No value function
- Learnt policy (current policy?)

Value based RL

Policy based RL

Advantages and disadvantages (in policy based RL perspective)

Disadvantages

1. Policy based RL typically converges to local optimum rather than the global optimum
2. Policy based RL typically is inefficient and high variance

Advantages

1. Better convergence properties
2. Effective in high-dimensional or continuous action spaces
3. Can learn stochastic properties (important)

Q1. Really? Why?

Value based RL

Policy based RL

Advantages and disadvantages (in policy based RL perspective)

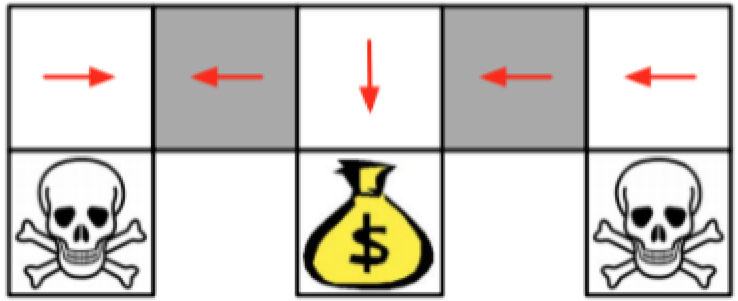
Disadvantages

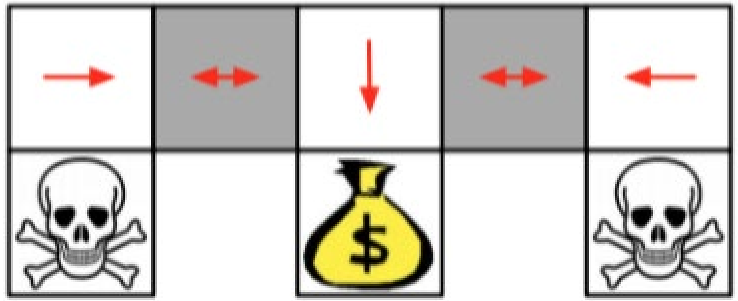
1. Policy based RL typically converges to local optimum rather than the global optimum
2. Policy based RL typically is inefficient and high variance

Advantages

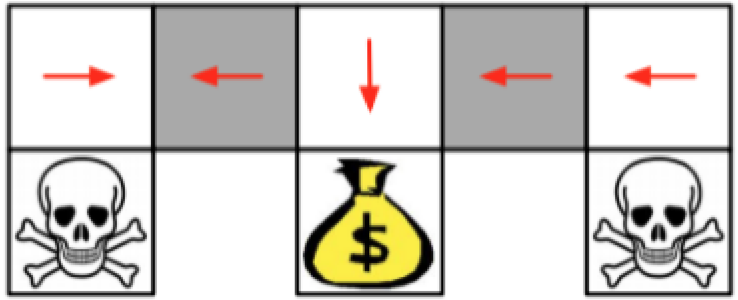
1. Better convergence properties
2. Effective in high-dimensional or continuous action spaces
3. Can learn stochastic properties (important)

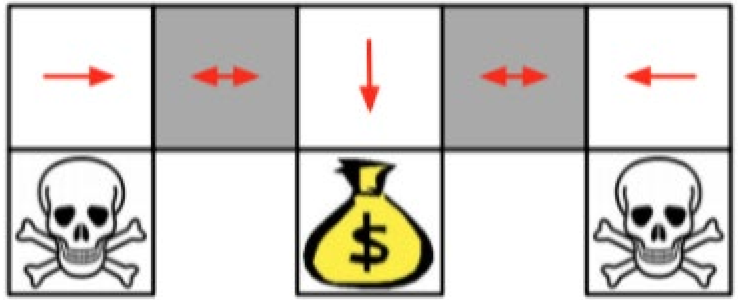
3. Policy based RL can learn stochastic properties





3. Policy based RL can learn stochastic properties





Q1. Why can't value based reinforcement learning do stochastic policy learning to solve this problem?

Trying to Understand Vanilla Policy Gradient Algorithm

Initialize policy parameter θ, baseline b
for iteration$=1, 2, \cdots$ **do**
  Collect a set of trajectories by executing the current policy
  At each timestep in each trajectory, compute
    the *return* $R_t = \sum_{t'=t}^{T-1} r_{t'}$, and
    the *advantage estimate* $\hat{A}_t = R_t - b(s_t)$.
  Re-fit the baseline, by minimizing $||b(s_t) - R_t||^2$,
    summed over all trajectories and timesteps.
  Update the policy, using a policy gradient estimate $\hat{g}$,
    which is a sum of terms $\nabla_\theta \log \pi(a_t|s_t, \theta)\hat{A}_t$.
    (Plug $\hat{g}$ into SGD or ADAM)
endfor

Can be modified better performance

Estimate gradient = delta value function over delta parameters

Trying to Understand Vanilla Policy Gradient Algorithm

1. Take one policy
2. Get data from it
3. Fit baseline (advantage)
4. Calculate policy gradient estimate
5. Update policy by step size using

Can be modified better performance

Trying to Understand Vanilla Policy Gradient Algorithm

1. Before estimation (calculation) of policy gradient, need to quantify the quality of the current policy = policy objective functions

Initialize policy parameter θ, baseline b
for iteration$=1, 2, \cdots$ **do**
Collect a set of trajectories by executing the current policy
At each timestep in each trajectory, compute
the *return* $R_t = \sum_{t'=t}^{T-1} r_{t'}$, and
the *advantage estimate* $\hat{A}_t = R_t - b(s_t)$.
Re-fit the baseline, by minimizing $||b(s_t) - R_t||^2$,
summed over all trajectories and timesteps.
Update the policy, using a policy gradient estimate $\hat{g}$,
which is a sum of terms $\nabla_\theta \log \pi(a_t|s_t, \theta)\hat{A}_t$.
(Plug $\hat{g}$ into SGD or ADAM)
endfor

Estimate gradient = delta value function over delta parameters

Different possible policy objective functions

1. Before estimation (calculation) of policy gradient, need to quantify the quality of the current policy = policy objective functions

$$J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

for episodic environments

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

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

For continuous environments

Stationary distribution

Trying to Understand Vanilla Policy Gradient Algorithm

1. Policy objective functions
2. Under assumption of episodic MDP, extend the idea with the following particular policy objective function…

$$J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

Trying to Understand Vanilla Policy Gradient Algorithm

1. Policy objective functions
2. Under assumption of episodic MDP, extend the idea with the following particular policy objective function…

$$V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

Trying to Understand Vanilla Policy Gradient Algorithm

1. Policy objective functions
2. Under assumption of episodic MDP, extend the idea with the following particular policy objective function… then the objective is to optimize the given function

$$V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

Trying to Understand Vanilla Policy Gradient Algorithm

1. Policy objective functions
2. Under assumption of episodic MDP, extend the idea with the following particular policy objective function… then the objective is to optimize the given function. Although there are other methods that don't involve using gradient approaches, let's focus on the **policy gradient** approach

$$V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

Learning rate

$$\nabla\theta = \alpha\nabla_\theta V(\theta)$$

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

Trying to Understand Vanilla Policy Gradient Algorithm

1. Policy objective functions
2. Optimization using policy gradient → first calculate policy gradient

$$V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

$$\nabla\theta = \alpha\nabla_\theta V(\theta)$$

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

Similar to Q/V based approach but instead of deriving respect to parameters that define Q function, derive respect to parameters that define the policy

Little confusing with notation V(theta) but theta is parameter that defines pi and thus the value function becomes a function of theta (= value function for the policy)

Trying to Understand Vanilla Policy Gradient Algorithm

1. Policy objective functions
2. Calculate policy gradient (case: **finite difference policy gradient:** works even when policy is not differentiable, simple, may be inefficient but sometimes effective)

$$V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

$$\nabla\theta = \alpha \nabla_\theta V(\theta)$$

$$\nabla_\theta V(\theta) = \begin{pmatrix} \frac{\delta V(\theta)}{\delta \theta_1} \\ \vdots \\ \frac{\delta V(\theta)}{\delta \theta_n} \end{pmatrix} \longrightarrow \frac{\delta V(\theta)}{\delta \theta_k} \approx \frac{V(\theta + \epsilon u_k - V(\theta)}{\epsilon}$$

1. Perturb theta by epsilon in kth dimension
2. uk is a unit vector in kth component

Trying to Understand ~~Vanilla Policy Gradient Algorithm~~ Likelihood Ratio Policy

1. **Policy objective functions**
2. Calculate policy gradient

(= policy value)

reset

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

Expected discounted sum of rewards from given policy

Probability of observing particular trajectory

Reward of that trajectory

Trying to Understand ~~Vanilla Policy Gradient Algorithm~~ Likelihood Ratio Policy

1. **Policy objective functions (goal)**
2. Calculate policy gradient

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

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

Trying to Understand ~~Vanilla Policy Gradient Algorithm~~ Likelihood Ratio Policy

1. Policy objective function
2. **Calculate policy gradient**

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

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

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

After some calculations …

$$\nabla_\theta V(\theta) = \sum P(\tau; \theta) R(\tau) \nabla_\theta \log P(\tau; \theta)$$

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

Approximated with empirical estimates for m sample paths under given policy

Trying to Understand ~~Vanilla Policy Gradient Algorithm~~ Score Function

1. Policy objective function
2. **Calculate policy gradient**

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

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

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

Score function

$$\nabla_\theta \log P(\tau^{(i);\theta}) = \sum_{t=0}^{T-1} \underbrace{\nabla_\theta \log \pi_\theta(a_t|s_t)}_{\text{no dynamics model required!}}$$

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

Trying to Understand ~~Vanilla Policy Gradient Algorithm~~ Score Function

1. Policy objective function
2. **Calculate policy gradient**

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

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

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

$$\nabla_\theta \log P(\tau^{(i);\theta}) = \sum_{t=0}^{T-1} \underbrace{\nabla_\theta \log \pi_\theta(a_t|s_t)}_{\text{no dynamics model required!}}$$

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

: only require analytic form for derivative policy with respect to parameters

Trying to Understand ~~Vanilla Policy Gradient Algorithm~~ Score Function

1. Policy objective function
2. **Calculate policy gradient**

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

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

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

$$\nabla_\theta \log P(\tau^{(i);\theta}) = \sum_{t=0}^{T-1} \underbrace{\nabla_\theta \log \pi_\theta(a_t|s_t)}_{\text{no dynamics model required!}}$$

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

Q1. meaning and importance of the score function? (review the lecture...)

Trying to Understand ~~Vanilla Policy Gradient Algorithm~~ Score Function

1. Policy objective function
2. **Calculate policy gradient**

$$\nabla_\theta \log \pi_\theta(s, a)$$

A1. Not that much meaning, just labeling a specific part of an equation.

Q1. meaning and importance of the score function? (review the lecture…)

Trying to Understand ~~Vanilla Policy Gradient Algorithm~~ Policy Gradient Function

1. Decide policy objective function
2. Calculate policy gradient

Theorem

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

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

Interesting that policy gradient can be independent on the type of policy objective function

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION1 : temporal structure

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

Sum of rewards expressed.

From every single one of the rewards – product with sum of the full trajectory of the derivative of the policy parameters

To product with sum of only the ones relevant to that particular reward

→ Lower variance achieved

$$\hat{g} = (1/m)\sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) G_t^{(i)}$$

Reward at some time point is not influenced by later decisions made

= don't require the complete product of probability of action at given states

= future actions don't effect earlier rewards

Lecture 8, 1:00:18
Lecture 9, 17:25

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION1 : temporal structure

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

$$\left(\sum_{t=0}^{T-1} r_t\right)\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)$$

$$\sum_{t=0}^{T-1} r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

reorder

$$\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \sum_{t'=t}^{T-1} r_{t'}$$

$$\hat{g} = (1/m)\sum_{i=1}^{m}\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) G_t^{(i)}$$

Reward at some time point is not influenced by later decisions made

= don't require the complete product of probability of action at given states

= future actions don't effect earlier rewards

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION1 : temporal structure

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

$$\left(\sum_{t=0}^{T-1} r_t\right)\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)$$

$$\sum_{t=0}^{T-1} r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

reorder

$$\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \sum_{t'=t}^{T-1} r_{t'}$$

$$\hat{g} = (1/m)\sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) G_t^{(i)}$$

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION1 : temporal structure

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

$$\left(\sum_{t=0}^{T-1} r_t\right)\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)$$

$$\sum_{t=0}^{T-1} r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

reorder

$$\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \sum_{t'=t}^{T-1} r_{t'}$$

$$\hat{g} = (1/m)\sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) G_t^{(i)}$$

$\sum_{t'=t}^{T-1} r_{t'}^{(i)}$ is the return $G_t^{(i)}$

Q2. Given equation is confusing… shouldn't it be from 0 up to T-1 or t'? If it is an equation of reward up until given timestep??

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION1 : temporal structure = REINFORCE

Initialize policy parameters θ arbitrarily

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

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

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

  endfor

endfor

return θ

A9. 바깥쪽 for문에서는 theta를 활용한 policy로 쫙 termination까지 돌린 결과가 있고, 안쪽 for 문에서 theta를 업데이트를 한다. 이를 다시 말하자면, 안쪽에서는 변화는 되지만, 바깥 루프에서는 실제 적용이 되는 것이니까 MC라 봐도 무방해 보인다.

Q9. for REINFORCE: Update after each episode because it uses MC methods, or as the algorithm implies, update each timestep???

lecture 9

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION2 : baseline

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

$$\hat{g} = (1/m)\sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) G_t^{(i)}$$

$$\hat{g} = (1/m)\sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \sum_{t'=t}^{T-1} r_{t'}$$

$$\hat{g} = (1/m)\sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right)$$

Subtract by baseline that depends only on the state

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION2 : baseline

We can look at increasing log probability of an action proportional to how much better it is than a baseline

$$\hat{g} = (1/m) \sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right)$$

Good example of baseline can be a value function V(s)

Advantage estimate

Where baseline is simply the expected sum of rewards

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION2 : baseline

We can look at increasing log probability of an action proportional to how much better it is than a baseline

$$\hat{g} = (1/m) \sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right)$$

Good example of baseline can be a value function V(s)

Advantage estimate

Where baseline is simply the expected sum of rewards

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION2 : baseline

$$\hat{g} = (1/m)\sum_{i=1}^{m}\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t, s_t)\left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t)\right)$$

Refit baseline by...

$$\sum_i \sum_t \left\| b(s_t^i) - G_t^i \right\|$$

Q4. Is the intuition here that we are trying to develop baseline function to expected sum of rewards?

Therefore the advantage estimate will converge to 0 as the parameters become updated and the baseline becomes the (true)expected sum of rewards.

And the advantage estimate acts like a learning rate, which changes size relatively to how far off the sum of rewards from trajectory is from the (not true yet)expected sum of rewards?

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION2 : baseline

$$\hat{g} = (1/m) \sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right)$$

N-step estimator

Tradeoff between variance and bias

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, evaluate the possible new policy with the current data from using the current policy to find step size that guarantees monotonic improvement.

$$V(\tilde{\theta}) = V(\theta) + \sum_{s} \rho_{\tilde{\pi}}(s) \sum_{a} \tilde{\pi}(a|s) A_{\pi}(s, a)$$

But cannot calculate the stationary distribution given from the new policy

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, evaluate using **local approximation**

$$V(\tilde{\theta}) = V(\theta) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$$

$$L_\pi(\tilde{\pi}) = V(\theta) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$$

lecture 10

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, find the **lower-bound**

Theorem

Let $D_{TV}^{\max}(\pi_1, \pi_2) = \max_s D_{TV}(\pi_1(\cdot|s), \pi_2(\cdot|s))$. *Then*

$$V^{\pi_{new}} \geq L_{\pi_{old}}(\pi_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}(D_{TV}^{\max}(\pi_{old}, \pi_{new}))^2$$

where $\epsilon = \max_{s,a} |A_\pi(s, a)|$.

$$D_{TV}(\pi_1(\cdot\,|s), \pi_2(\cdot\,|s)) = \max_a |\pi_1(a|s) - \pi_2(a|s)|$$

Maximum difference in the probability of an action under one policy versus another policy

Then D max is the biggest difference the two policies give for a particular action over all states

= where the two policies most differ

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, find the **lower-bound**

Theorem

Let $D_{TV}^{\max}(\pi_1, \pi_2) = \max_s D_{TV}(\pi_1(\cdot|s), \pi_2(\cdot|s))$. *Then*

$$V^{\pi_{new}} \geq L_{\pi_{old}}(\pi_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}(D_{TV}^{\max}(\pi_{old}, \pi_{new}))^2$$

where $\epsilon = \max_{s,a} |A_\pi(s, a)|$.

Q5. so this is the step size?

$$D_{TV}\big(\pi_1(\cdot\,|s), \pi_2(\cdot\,|s)\big) = \max_a |\pi_1(a|s) - \pi_2(a|s)|$$

change to KL divergence

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, find the **lower-bound**

Theorem

Let $D_{TV}^{\max}(\pi_1, \pi_2) = \max_s D_{TV}(\pi_1(\cdot|s), \pi_2(\cdot|s))$. *Then*

$$V^{\pi_{new}} \geq L_{\pi_{old}}(\pi_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}(D_{TV}^{\max}(\pi_{old}, \pi_{new}))^2$$

where $\epsilon = \max_{s,a} |A_\pi(s, a)|$.

$$D_{TV}(\pi_1(\cdot\,|s), \pi_2(\cdot\,|s)) = \max_a |\pi_1(a|s) - \pi_2(a|s)|$$

$$D_{TV}(p, q)^2 \leq D_{KL}(p, q)$$ Change to KL divergence

$$D_{KL}^{\max}(\pi_1, \pi_2) = \max_s D_{KL}(\pi_1(\cdot|s), \pi_2(\cdot|s))$$

$$V^{\pi_{new}} \geq L_{\pi_{old}}(\pi_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\pi_{old}, \pi_{new})$$

More practical

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **guarantee monotonic improvement**

Theorem

Let $D_{TV}^{\max}(\pi_1, \pi_2) = \max_s D_{TV}(\pi_1(\cdot|s), \pi_2(\cdot|s))$. *Then*

$$V^{\pi_{new}} \geq L_{\pi_{old}}(\pi_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\pi_{old}, \pi_{new})$$

where $\epsilon = \max_{s,a} |A_\pi(s, a)|$.

$$M_i(\pi) = L_{\pi_i}(\pi) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\pi_i, \pi)$$

$$V^{\pi_{i+1}} - V^{\pi_i} \geq M_i(\pi_{i+1}) - M_i(\pi_i)$$

If the lower bound improves then there is monotonic improvement

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **problem & being more practical**

$$V^{\pi_{new}} \geq L_{\pi_{old}}(\pi_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\pi_{old}, \pi_{new})$$

Optimize...

$$\max_{\theta} L_{\theta_{old}}(\theta_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\theta_{old}, \theta_{new}) = L_{\theta_{old}}(\theta_{new}) - C D_{KL}^{\max}(\theta_{old}, \theta_{new})$$

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **problem & being more practical**

Optimize…

$$\max_\theta L_{\theta_{old}}(\theta_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\theta_{old}, \theta_{new}) = L_{\theta_{old}}(\theta_{new}) - C D_{KL}^{\max}(\theta_{old}, \theta_{new})$$

Too small of a step size

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **problem & being more practical**

Optimize…

$$\max_{\theta} L_{\theta_{old}}(\theta_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\theta_{old}, \theta_{new}) = L_{\theta_{old}}(\theta_{new}) - C D_{KL}^{\max}(\theta_{old}, \theta_{new})$$

Instead

$$\max_{\theta} L_{\theta_{old}}(\theta)$$

$$\text{subject to } D_{KL}^{s\sim\rho_{\theta_{old}}}(\theta_{old}, \theta) \leq \delta$$

Constrain KL divergence on some trusted region

Use average KL instead of max KL divergence for practicality

Basically this means to maximize objective function (value of policy) subject to KL convergence bounded by some delta (range of trusted region)

Q6. would this be correct?

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **being more & more practical**

Optimize...

$$\max_{\theta} L_{\theta_{old}}(\theta_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\theta_{old}, \theta_{new}) = L_{\theta_{old}}(\theta_{new}) - C D_{KL}^{\max}(\theta_{old}, \theta_{new})$$

$$L_\pi(\tilde{\pi}) = V(\theta) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$$

1. Reweight according to states actually sampled by current policy

$$\sum_s \rho_\pi(s) \rightarrow \frac{1}{1-\gamma} \mathbb{E}_{s \sim \rho_{\theta_{old}}}[\ldots]$$

$$\sum_a \pi_\theta(a|s_n) A_{\theta_{old}}(s_n, a) \rightarrow \mathbb{E}_{a \sim q}\left[\frac{\pi_\theta(a|s_n)}{q(a|s_n)} A_{\theta_{old}}(s_n, a)\right]$$

$$A_{\theta_{old}} \rightarrow Q_{\theta_{old}}$$

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **being more & more practical**

Optimize…

$$\max_\theta L_{\theta_{old}}(\theta_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\theta_{old}, \theta_{new}) = L_{\theta_{old}}(\theta_{new}) - C D_{KL}^{\max}(\theta_{old}, \theta_{new})$$

$$L_\pi(\tilde{\pi}) = V(\theta) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$$

$$\sum_s \rho_\pi(s) \rightarrow \frac{1}{1-\gamma} \mathbb{E}_{s\sim\rho_{\theta_{old}}}[\ldots]$$

$$\sum_a \pi_\theta(a|s_n) A_{\theta_{old}}(s_n, a) \rightarrow \mathbb{E}_{a\sim q}\left[\frac{\pi_\theta(a|s_n)}{q(a|s_n)} A_{\theta_{old}}(s_n, a)\right]$$

$$A_{\theta_{old}} \rightarrow Q_{\theta_{old}}$$

2. Importance sampling : using how many time a particular action is taken with current policy and reweight them according to how many times the action would have been taken according to the new policy

lecture 10

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **being more & more practical**

Optimize...

$$\max_\theta L_{\theta_{old}}(\theta_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\theta_{old}, \theta_{new}) = L_{\theta_{old}}(\theta_{new}) - C D_{KL}^{\max}(\theta_{old}, \theta_{new})$$

$$L_\pi(\tilde{\pi}) = V(\theta) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s, a)$$

3. yeah

$$\sum_s \rho_\pi(s) \rightarrow \frac{1}{1-\gamma} \mathbb{E}_{s \sim \rho_{\theta_{old}}}[\ldots]$$

$$\sum_a \pi_\theta(a|s_n) A_{\theta_{old}}(s_n, a) \rightarrow \mathbb{E}_{a \sim q}\left[\frac{\pi_\theta(a|s_n)}{q(a|s_n)} A_{\theta_{old}}(s_n, a)\right]$$

$$A_{\theta_{old}} \rightarrow Q_{\theta_{old}}$$

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **being more & more practical**

Optimize…

$$L_\pi(\tilde{\pi}) = V(\theta) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$$

$$\max_\theta \mathbb{E}_{s\sim\rho_{\theta_{old}}, a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)} Q_{\theta_{old}}(s,a)\right]$$

$$\text{subject to } \mathbb{E}_{s\sim\rho_{\theta_{old}}} D_{KL}(\pi_{\theta_{old}}(\cdot|s), \pi_\theta(\cdot|s)) \leq \delta$$

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **being more & more practical**

Optimize...

$$\max_{\theta} L_{\theta_{old}}(\theta)$$

$$\text{subject to } D_{KL}^{s\sim\rho_{\theta_{old}}}(\theta_{old}, \theta) \leq \delta$$

$$L_{\pi}(\tilde{\pi}) = V(\theta) + \sum_s \rho_{\pi}(s) \sum_a \tilde{\pi}(a|s) A_{\pi}(s,a)$$

Q7. where did this go?
Doesn't matter because it is constant no matter how the policy is changed?

$$\max_{\theta} \mathbb{E}_{s\sim\rho_{\theta_{old}}, a\sim q}\left[\frac{\pi_{\theta}(a|s)}{q(a|s)} Q_{\theta_{old}}(s,a)\right]$$

$$\text{subject to } \mathbb{E}_{s\sim\rho_{\theta_{old}}} D_{KL}(\pi_{\theta_{old}}(\cdot|s), \pi_{\theta}(\cdot|s)) \leq \delta$$

Trying to Understand Vanilla Policy Gradient Algorithm

Initialize policy parameter θ, baseline b
for iteration$=1, 2, \cdots$ **do**
  Collect a set of trajectories by executing the current policy
  At each timestep in each trajectory, compute
    the *return* $R_t = \sum_{t'=t}^{T-1} r_{t'}$, and
    the *advantage estimate* $\hat{A}_t = R_t - b(s_t)$.
  Re-fit the baseline, by minimizing $||b(s_t) - R_t||^2$,
    summed over all trajectories and timesteps.
  Update the policy, using a policy gradient estimate $\hat{g}$,
    which is a sum of terms $\nabla_\theta \log \pi(a_t|s_t, \theta)\hat{A}_t$.
    (Plug $\hat{g}$ into SGD or ADAM)
endfor

Trying to Understand Vanilla Policy Gradient Algorithm

1: **for** iteration$=1, 2, \dots$ **do**
2: Run policy for T timesteps or N trajectories
3: At each timestep in each trajectory, compute target $Q^{\pi}(s_t, a_t)$, and baseline $b(s_t)$
4: Compute estimated policy gradient $\hat{g}$
5: Update the policy using $\hat{g}$, potentially constrained to a local region
6: **end for**