Lecture 9: Policy Gradient II (Post lecture) [2]

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2018

- Additional reading: Sutton and Barto 2018 Chp. 13

[2] With many slides from or derived from David Silver and John Schulman and Pieter Abbeel

Class Feedback

- Thanks to those that participated!
- Of 70 responses, 54% thought too fast, 43% just right

Class Feedback

- Thanks to those that participated!
- Of 70 responses, 54% thought too fast, 43% just right
- Multiple request to: repeat questions for those watching later on; have more worked examples; have more conceptual emphasis; minimize notation errors

Class Feedback

- Thanks to4 those that participated!
- Of 70 responses, 54% thought too fast, 43% just right
- Multiple request to: repeat questions for those watching later on; have more worked examples; have more conceptual emphasis; minimize notation errors
- Common things people find are helping them learn: assignments, mathematical derivations, checking your understanding/talking to a neighbor

Class Structure

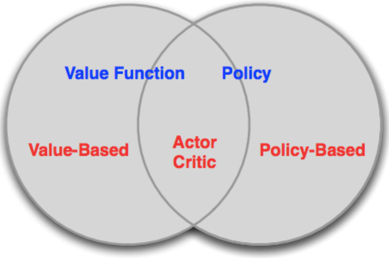
- Last time: Policy Search
- **This time: Policy Search**
- Next time: Midterm review

Recall: Policy-Based RL

- Policy search: directly parametrize the policy

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

- Goal is to find a policy π with the highest value function V^π
- (Pure) Policy based methods
  - No Value Function
  - Learned Policy
- Actor-Critic methods
  - Learned Value Function
  - Learned Policy



Recall: Advantages of Policy-Based RL

Advantages:

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

Disadvantages:

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

Recall: Policy Gradient

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

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

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

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

- and α is a step-size parameter

Desired Properties of a Policy Gradient RL Algorithm

- Goal: Converge as quickly as possible to a local optima
  - Incurring reward / cost as execute policy, so want to minimize number of iterations / time steps until reach a good policy

Desired Properties of a Policy Gradient RL Algorithm

- Goal: Converge as quickly as possible to a local optima
  - Incurring reward / cost as execute policy, so want to minimize number of iterations / time steps until reach a good policy
- During policy search alternating between evaluating policy and changing (improving) policy (just like in policy iteration)
- Would like each policy update to be a monotonic improvement
  - Only guaranteed to reach a local optima with gradient descent
  - Monotonic improvement will achieve this
  - And in the real world, monotonic improvement is often beneficial

Desired Properties of a Policy Gradient RL Algorithm

- Goal: Obtain large monotonic improvements to policy at each update
- Techniques to try to achieve this:
  - Last time and today: Get a better estimate of the gradient (intuition: should improve updating policy parameters)
  - Today: Change how to update the policy parameters given the gradient

Table of Contents

1. Better Gradient Estimates
2. Policy Gradient Algorithms and Reducing Variance
3. Updating the Parameters Given the Gradient: Motivation
4. Need for Automatic Step Size Tuning
5. Updating the Parameters Given the Gradient: Local Approximation
6. Updating the Parameters Given the Gradient: Trust Regions
7. Updating the Parameters Given the Gradient: TRPO Algorithm

Likelihood Ratio / Score Function Policy Gradient

- Recall last time (m is a set of trajectories):

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

- Unbiased estimate of gradient but very noisy
- Fixes that can make it practical
  - Temporal structure (discussed last time)
  - Baseline
  - Alternatives to using Monte Carlo returns $R(\tau^{(i)})$ as targets

Policy Gradient: Introduce Baseline

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

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

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

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

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

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

$$
\begin{aligned}
&\mathbb{E}_\tau[\nabla_\theta \log \pi(a_t|s_t,\theta) b(s_t)] \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[\mathbb{E}_{s_{(t+1):T},a_{t:(T-1)}}[\nabla_\theta \log \pi(a_t|s_t,\theta) b(s_t)]\right] \text{ (break up expectation)} \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[b(s_t)\mathbb{E}_{s_{(t+1):T},a_{t:(T-1)}}[\nabla_\theta \log \pi(a_t|s_t,\theta)]\right] \text{ (pull baseline term out)} \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}[b(s_t)\mathbb{E}_{a_t}[\nabla_\theta \log \pi(a_t|s_t,\theta)]] \text{ (remove irrelevant variables)} \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[b(s_t)\sum_a \pi_\theta(a_t|s_t)\frac{\nabla_\theta \pi(a_t|s_t,\theta)}{\pi_\theta(a_t|s_t)}\right] \text{ (likelihood ratio)} \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[b(s_t)\sum_a \nabla_\theta \pi(a_t|s_t,\theta)\right] \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[b(s_t)\nabla_\theta \sum_a \pi(a_t|s_t,\theta)\right] \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}[b(s_t)\nabla_\theta 1] \\
&= \mathbb{E}_{s_0:t,a_0:(t-1)}[b(s_t)\cdot 0] = 0
\end{aligned}
$$

"Vanilla" Policy Gradient Algorithm

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

Practical Implementation with Autodiff

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

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

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

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

Other choices for baseline?

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

Choosing the Baseline: Value Functions

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

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

- State-value function can serve as a great baseline

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

- Advantage function: Combining Q with baseline V

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

Table of Contents

1. Better Gradient Estimates
2. Policy Gradient Algorithms and Reducing Variance
3. Updating the Parameters Given the Gradient: Motivation
4. Need for Automatic Step Size Tuning
5. Updating the Parameters Given the Gradient: Local Approximation
6. Updating the Parameters Given the Gradient: Trust Regions
7. Updating the Parameters Given the Gradient: TRPO Algorithm

Likelihood Ratio / Score Function Policy Gradient

- Recall last time:

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

- Unbiased estimate of gradient but very noisy
- Fixes that can make it practical
  - Temporal structure (discussed last time)
  - Baseline
  - **Alternatives to using Monte Carlo returns $R * \tau^{(i)}$ as targets**

Choosing the Target

- $R(\tau^{(i)})$ is an estimation of the value function from a single roll out
- Unbiased but high variance
- reduce variance by introducing bias using bootstrapping and function approximation (just like in we saw for TD vs MC, and in the value function approximation lectures)
- Estimate of V/Q is done by a **critic**
- **Actor-critic** methods maintain an explicit representation of both the policy and the value function, and update both
- A3C is very popular an actor-critic method

Policy Gradient Formulas with Value Functions

- Recall:

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

$$\nabla_\theta \mathbb{E}_\tau[R] \approx \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t,\theta) \left(Q(s_t,\boldsymbol{w}) - b(s_t) \right) \right]$$

- Letting the baseline be an estimate of the value V, we can represent the gradient in terms of the state-action advantage function

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

Choosing the Target: N-step estimators

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

Note that critic can select any blend between TD and MC estimators for the target to substitute for the true state-action value function.

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

If subtract baselines from the above, get advantage estimators

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

$\hat{A}_t^{(a)}$ has low variance & high bias. $\hat{A}_t^{(\infty)}$ high variance but low bias.

Table of Contents

1. Better Gradient Estimates
2. Policy Gradient Algorithms and Reducing Variance
3. Updating the Parameters Given the Gradient: Motivation
4. Need for Automatic Step Size Tuning
5. Updating the Parameters Given the Gradient: Local Approximation
6. Updating the Parameters Given the Gradient: Trust Regions
7. Updating the Parameters Given the Gradient: TRPO Algorithm

Updating the Policy Parameters Given the Gradient

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

Table of Contents

1. Better Gradient Estimates
2. Policy Gradient Algorithms and Reducing Variance
3. Updating the Parameters Given the Gradient: Motivation
4. Need for Automatic Step Size Tuning
5. Updating the Parameters Given the Gradient: Local Approximation
6. Updating the Parameters Given the Gradient: Trust Regions
7. Updating the Parameters Given the Gradient: TRPO Algorithm

Policy Gradient and Step Sizes

- Goal: Each step of policy gradient yields an updated policy π' whose value is greater than or equal to the prior policy π: $V^{\pi'} \geq V^{\pi}$
- Gradient descent approaches update the weights a small step in direction of gradient
- **First order** / linear approximation of the value function's dependence on the policy parameterization
- Locally a good approximation, further away less good

Why are step sizes a big deal in RL?

- Step size is important in any problem involving finding the optima of a function
- Supervised learning: Step too far $\rightarrow$ next updates will fix it
- Reinforcement learning
  - Step too far $\rightarrow$ bad policy
  - Next batch: collected under bad policy
  - **Policy is determining data collect!** Essentially controlling exploration and exploitation trade off due to particular poilcy parameters and the stochasticity of the policy
  - May not be able to recover from a bad choice, collapse in performance!

- Simple step-sizing: Line search in direction of gradient
  - Simple but expensive (perform evaluations along the line)
  - Naive: ignores where the first order approximation is good or bad

Policy Gradient Methods with Auto-Step-Size Selection

- Can we automatically ensure the updated policy π' has value greater than or equal to the prior policy π: $V^{\pi'} \geq V^{\pi}$?
- Consider this for the policy gradient setting, and hope to address this by modifying step size

Objective Function

- Goal: find policy parameters that maximize value function[35]

$$V(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t); \pi_\theta\right] \tag{1}$$

- where $s_0 \sim \mu(s_0)$, $a_t \sim \pi(a_t|s_t), s_{t+1} \sim P(s_{t+1}|s_t, a_t)$
- Have access to samples from the current policy π (param. by θ)
- Want to predict the value of a different policy (off policy learning!)

[35] For today we will primarily consider discounted value functions

Objective Function

- Goal: find policy parameters that maximize value function[37]

$$V(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t); \pi_\theta\right] \quad (2)$$

- where $s_0 \sim \mu(s_0)$, $a_t \sim \pi(a_t|s_t)$, $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$
- Express expected return of another policy in terms of the advantage over the original policy

$$V(\tilde{\theta}) = V(\theta) + \mathbb{E}_{\pi_{\tilde{\theta}}}\left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t)\right] = V(\theta) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A_\pi(s, a)$$

- where $\rho_{\tilde{\pi}}(s)$ is defined as the discounted weighted frequency of state s under policy $\tilde{\pi}$ (similar to in Imitation Learning lecture)
- We know the advantage A_π and $\tilde{\pi}$
- But we can't compute the above because we don't know $\rho_{\tilde{\pi}}$, the state distribution under the new proposed policy

[37] For today we will primarily consider discounted value functions

Table of Contents

1. Better Gradient Estimates
2. Policy Gradient Algorithms and Reducing Variance
3. Updating the Parameters Given the Gradient: Motivation
4. Need for Automatic Step Size Tuning
5. Updating the Parameters Given the Gradient: Local Approximation
6. Updating the Parameters Given the Gradient: Trust Regions
7. Updating the Parameters Given the Gradient: TRPO Algorithm

Local approximation

- Can we remove the dependency on the discounted visitation frequencies under the new policy?
- Substitute in the discounted visitation frequencies under the current policy to define a new objective function:

$$L_\pi(\tilde{\pi}) = V(\theta) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a) \tag{4}$$

- Note that $L_{\pi_{\theta_0}}(\pi_{\theta_0}) = V(\theta_0)$
- Gradient of L is identical to gradient of value function at policy parameterized evaluated at θ_0: $\nabla_\theta L_{\pi_{\theta_0}}(\pi_\theta)|_{\theta=\theta_0} = \nabla_\theta V(\theta)|_{\theta=\theta_0}$

Conservative Policy Iteration

- Is there a bound on the performance of a new policy obtained by optimizing the surrogate objective?
- Consider mixture policies that blend between an old policy and a different policy

$$\pi_{new}(a|s) = (1-\alpha)\pi_{old}(a|s) + \alpha\pi'(a|s) \tag{5}$$

- In this case can guarantee a lower bound on value of the new π_{new}:

$$V^{\pi_{new}} \geq L_{\pi_{old}}(\pi_{new}) - \frac{2\epsilon\gamma}{(1-\gamma)^2}\alpha^2 \tag{6}$$

- where $\epsilon = \max_s \left|\mathbb{E}_{a\sim\pi'(a|s)}\left[A_\pi(s,a)\right]\right|$
- Check your understanding: is this bound tight if $\pi_{new} = \pi_{old}$? Can we remove the dependency on the discounted visitation frequencies under the new policy?

Find the Lower-Bound in General Stochastic Policies

- Would like to similarly obtain a lower bound on the potential performance for general stochastic policies (not just mixture policies)
- Recall $L_\pi(\tilde{\pi}) = V(\theta) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$

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

- Note that $D_{TV}(p,q)^2 \leq D_{KL}(p,q)$ for prob. distrib p and q.
- Then the above theorem immediately implies that

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

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

Guaranteed Improvement[43]

- Goal is to compute a policy that maximizes the objective function defining the lower bound:

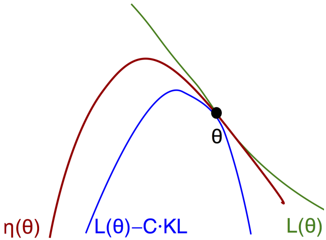
$$
\begin{aligned}
M_i(\pi) &= L_{\pi_i}(\pi) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\pi_i, \pi) && (7)\\
V^{\pi_{i+1}} &\geq L_{\pi_i}(\pi) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\pi_i, \pi) = M_i(\pi_{i+1}) && (8)\\
V^{\pi_i} &= M_i(\pi_i) && (9)\\
V^{\pi_{i+1}} - V^{\pi_i} &\geq M_i(\pi_{i+1}) - M_i(\pi_i) && (10)
\end{aligned}
$$

- So as long as the new policy π_{i+1} is equal or an improvement compared to the old policy π_i with respect to the lower bound, we are guaranteed to to monotonically improve!
- The above is a type of Minorization-maximization (MM) algorithm

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

Guaranteed Improvement[45]

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



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

Table of Contents

1. Better Gradient Estimates
2. Policy Gradient Algorithms and Reducing Variance
3. Updating the Parameters Given the Gradient: Motivation
4. Need for Automatic Step Size Tuning
5. Updating the Parameters Given the Gradient: Local Approximation
6. Updating the Parameters Given the Gradient: Trust Regions
7. Updating the Parameters Given the Gradient: TRPO Algorithm

Optimization of Parameterized Policies[48]

- Goal is to optimize

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

- where C is the penalty coefficient
- In practice, if we used the penalty coefficient recommended by the theory above $C = \frac{4\epsilon\gamma}{(1-\gamma)^2}$, the step sizes would be very small
- New idea: Use a trust region constraint on step sizes. Do this by imposing a constraint on the KL divergence between the new and old policy.

$$\max_\theta L_{\theta_{old}}(\theta) \tag{11}$$

$$\text{subject to } D_{KL}^{s \sim \rho_{\theta_{old}}}(\theta_{old}, \theta) \leq \delta \tag{12}$$

- This uses the average KL instead of the max (the max requires the KL is bounded at all states and yields an impractical number of constraints)

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

From Theory to Practice

- Prior objective:

$$\max_{\theta} L_{\theta_{old}}(\theta) \tag{13}$$

$$\text{subject to } D_{KL}^{s \sim \rho_{\theta_{old}}}(\theta_{old}, \theta) \leq \delta \tag{14}$$

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

- Don't know the visitation weights nor true advantage function
- Instead do the following substitutions:

$$\sum_s \rho_{\pi}(s) \rightarrow \frac{1}{1-\gamma} \mathbb{E}_{s \sim \rho_{\theta_{old}}}[\ldots], \tag{15}$$

From Theory to Practice

- Next substitution:

$$\sum_a \pi_\theta(a|s_n) A_{\theta_{old}}(s_n, a) \rightarrow \mathbb{E}_{a \sim q}\left[\frac{\pi_\theta(a|s_n)}{q(a|s_n)} A_{\theta_{old}}(s_n, a)\right] \tag{16}$$

- where q is some sampling distribution over the actions and s_n is a particular sampled state.
- This second substitution is to use importance sampling to estimate the desired sum, enabling the use of an alternate sampling distribution q (other than the new policy π_θ.
- Third substitution:

$$A_{\theta_{old}} \rightarrow Q_{\theta_{old}} \tag{17}$$

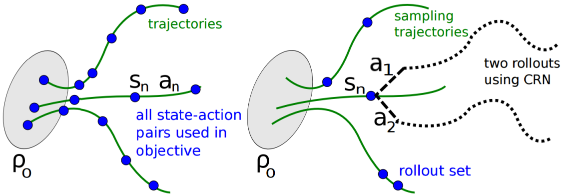
- Note that the above substitutions do not change solution to the above optimization problem

Selecting the Sampling Policy

- Optimize

$$\max_\theta \mathbb{E}_{s\sim\rho_{\theta_{old}}, a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)} Q_{\theta_{old}}(s,a)\right]$$
$$\text{subject to } \mathbb{E}_{s\sim\rho_{\theta_{old}}} D_{KL(}\pi_{\theta_{old}}(\cdot|s), \pi_\theta(\cdot|s)) \leq \delta$$

- Standard approach: sampling distribution is $q(a|s)$ is simply $\pi_{old}(a|s)$
- For the vine procedure see the paper



Searching for the Next Parameter

- Use a linear approximation to the objective function and a quadratic approximation to the constraint
- Constrained optimization problem
- Use conjugate gradient descent

schulman2015trust

Table of Contents

1. Better Gradient Estimates
2. Policy Gradient Algorithms and Reducing Variance
3. Updating the Parameters Given the Gradient: Motivation
4. Need for Automatic Step Size Tuning
5. Updating the Parameters Given the Gradient: Local Approximation
6. Updating the Parameters Given the Gradient: Trust Regions
7. Updating the Parameters Given the Gradient: TRPO Algorithm

Practical Algorithm: TRPO

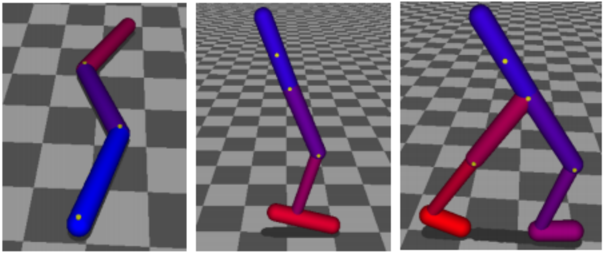
for iteration$=1, 2, \ldots$ **do**
    Run policy for T timesteps or N trajectories
    Estimate advantage function at all timesteps
    Compute policy gradient g
    Use CG (with Hessian-vector products) to compute $F^{-1}g$ where F is the Fisher information matrix
    Do line search on surrogate loss and KL constraint
end for

schulman2015trust

Practical Algorithm: TRPO

Applied to

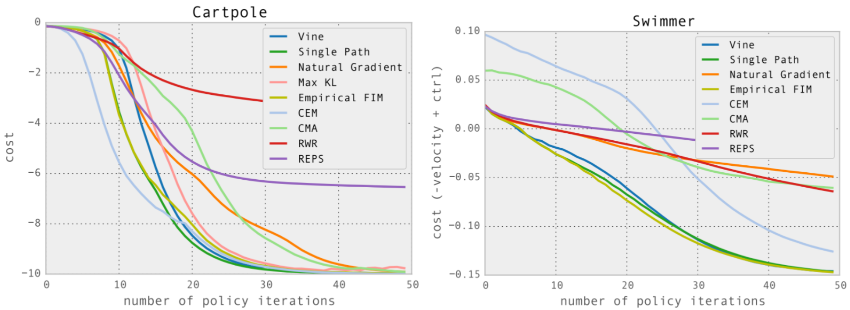
- Locomotion controllers in 2D



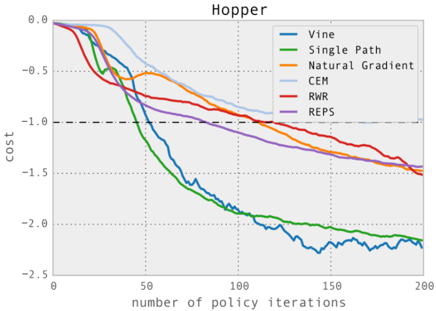
- Atari games with pixel input

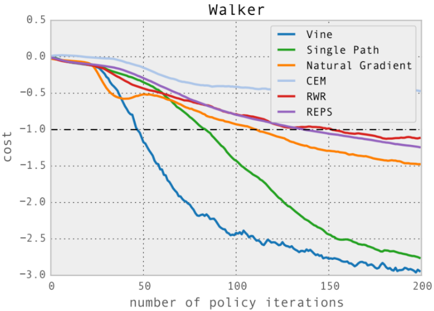
schulman2015trust

TRPO Results



TRPO Results





TRPO Summary

- Policy gradient approach
- Uses surrogate optimization function
- Automatically constrains the weight update to a trusted region, to approximate where the first order approximation is valid
- Empirically consistently does well
- Very influential: +350 citations since introduced a few years ago

Common Template of Policy Gradient Algorithms

---

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

Policy Gradient Summary

- Extremely popular and useful set of approaches
- Can input prior knowledge in the form of specifying policy parameterization
- You should be very familiar with REINFORCE and the policy gradient template on the prior slide
- Understand where different estimators can be slotted in (and implications for bias/variance)
- Don't have to be able to derive or remember the specific formulas in TRPO for approximating the objectives and constraints
- Will have the opportunity to practice with these ideas in homework 3