

- Midterm we will be in two rooms
- The room you are assigned to depends on the first letter of your SUID (Stanford email handle, e.g **jd**oe@stanford.edu)
- Gates B1 (a-e inclusive)
- Cubberley Auditorium (f-z)


Lecture 10: Policy Gradient III & Midterm Review ¹

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2019

- Additional reading: Sutton and Barto 2018 Chp. 13

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

Class Structure

- Last time: Policy Search
- **This time: Policy Search & Midterm Review**
- Next time: Midterm

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}
- Focus on policy gradient methods

We represent a policy using things like softmax or a NN
Then we take a gradient of these policies to learn parameters that give it
a higher Value

"Vanilla" Policy Gradient Algorithm

Initialize policy parameter θ , baseline b

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

Collect a set of trajectories by executing the current policy

At each timestep t in each trajectory τ^i , compute

Return $G_t^i = \sum_{t'=t}^{T-1} r_{t'}^i$, and

Advantage estimate $\hat{A}_t^i = G_t^i - b(s_t)$.

Re-fit the baseline, by minimizing $\sum_i \sum_t \|b(s_t) - G_t^i\|^2$,

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

we do this to
estimate the
gradient
 $dV/d\theta$

how far do we move on our gradient

Choosing the Target

this is the simplest thing we can do!
This is like what we saw in Monte Carlo



- G_t^i is an estimation of the value function at s_t 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 value function approximation
- Estimate of V/Q is done by a **critic** where we maintain an explicit parameterized representation of the policy, AND an explicit parameterized representation of the Value function
- **Actor-critic** methods maintain an explicit representation of policy and the value function, and update both
- A3C (Mnih et al. ICML 2016) is a very popular actor-critic method

"Vanilla" Policy Gradient Algorithm

Initialize policy parameter θ , baseline b

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

Collect a set of trajectories by executing the current policy

At each timestep t in each trajectory τ^i , compute

Target \hat{R}_t^i

Advantage estimate $\hat{A}_t^i = G_t^i - b(s_t)$.

Re-fit the baseline, by minimizing $\sum_i \sum_t ||b(s_t) - \hat{R}_t^i||^2$,

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

we need to figure out how far along that gradient we go to determine a new policy. this is important because whatever new policy we get determines the data we get

Policy Gradient Methods with Auto-Step-Size Selection

We want monotonic improvement! We might want this for high risk applications. But we want to do this without actually deploying/trying out our new policy.

So we use our existing data and policy to help us decide how big of a step to take

- 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 function¹

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

- 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 $\pi_{\theta_{old}}$ (param. by θ_{old})
- Want to predict the value of a different policy (off policy learning!)



this is a challenge b/c we only have access to data from current/old policy!

¹For today we will primarily consider discounted value functions

Objective Function

The value of the new policy can be represented as the value of the old policy plus the expected advantage. The expected advantage can be represented as the distribution of states we would expect to get to if we followed our new policy, times the advantage we would get under the old policy if we were to follow the new policy

- Goal: find policy parameters that maximize value function¹

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

- 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

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

- where $\mu_{\tilde{\pi}}(s)$ 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 $\mu_{\tilde{\pi}}$, the state distribution under the new proposed policy

¹For today we will primarily consider discounted value functions

Table of Contents

- 1 Updating the Parameters Given the Gradient: Local Approximation
- 2 Updating the Parameters Given the Gradient: Trust Regions
- 3 Updating the Parameters Given the Gradient: TRPO Algorithm

Local approximation

So we define a new objective function.

this won't equal our state distribution in our new policy, unless they're identical. we just ignore this potential "issue"

- 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:

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

- Note that $L_{\pi_{\theta_0}}(\pi_{\theta_0}) = V(\theta_0)$ if you plug in the old policy to this, you just get the original Value of that policy, b/c the advantage of the exiting policy over the existing policy is 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

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

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

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

- where $\epsilon = \max_s |\mathbb{E}_{a \sim \pi'(a|s)} [A_{\pi}(s, a)]|$

This is promising, but in general we won't want to just consider mixture policies...

Find the Lower-Bound in General Stochastic Policies

This theorem says we can get a lower bound on any stochastic policy using this new objective function

distance of total variation between 2 policies. The dot notation means that policies have a probability distribution over the actions

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

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

this is saying, over all states, what's the biggest difference the two policies give over a particular action. We won't be able to calculate this

so this is the max difference in the probability of an action under policy 1 compared to policy 2

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

Theorem

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

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

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

D_{\max_TV} isn't particularly easy to work with, so we can use the fact that its square is upper bounded by the KL divergence

- 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

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

KL divergence between two policies

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

Guaranteed Improvement¹

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

¹ $L_{\pi}(\tilde{\pi}) = V(\theta) + \sum_s \mu_{\pi}(s) \sum_a \tilde{\pi}(a|s) A_{\pi}(s, a)$

Guaranteed Improvement¹

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

lower bound



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

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

$$V^{\pi_i} = M_i(\pi_i) \text{ b/c the KL divergence of the same policy} = 0$$

$$V^{\pi_{i+1}} - V^{\pi_i} \geq M_i(\pi_{i+1}) - M_i(\pi_i) \text{ so if our new lower bound is greater than our old lower bound, then we can ensure monotonic improvement. So we can optimize the lower bound function}$$

- 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 monotonically improve!
- The above is a type of Minorization-Maximization (MM) algorithm

we need to make this more practical

¹ $L_{\pi}(\tilde{\pi}) = V(\theta) + \sum_s \mu_{\pi}(s) \sum_a \tilde{\pi}(a|s) A_{\pi}(s, a)$

Guaranteed Improvement¹

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

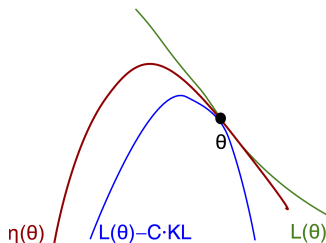


Figure: Source: John Schulman, Deep Reinforcement Learning, 2014

¹ $L_{\pi}(\tilde{\pi}) = V(\theta) + \sum_s \mu_{\pi}(s) \sum_a \tilde{\pi}(a|s) A_{\pi}(s, a)$

Table of Contents

- 1 Updating the Parameters Given the Gradient: Local Approximation
- 2 Updating the Parameters Given the Gradient: Trust Regions
- 3 Updating the Parameters Given the Gradient: TRPO Algorithm

Optimization of Parameterized Policies¹

we just turn this into a hyperparameter

- Goal is to optimize

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

- 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

¹ $L_{\pi}(\tilde{\pi}) = V(\theta) + \sum_s \mu_{\pi}(s) \sum_a \tilde{\pi}(a|s) A_{\pi}(s, a)$

Optimization of Parameterized Policies¹

The idea of TRPO is that there's a trusted region, and we use this to constrain our step sizes

- Goal is to optimize

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

- 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 (Schulman, Levine, Abbeel, Jordan, & Moritz ICML 2015). Do this by imposing a constraint on the KL divergence between the new and old policy.

you put a constraint on how much you can move

$$\begin{aligned} & \max_{\theta} L_{\theta_{old}}(\theta) \\ & \text{subject to } D_{KL}^{s \sim \mu_{\theta_{old}}}(\theta_{old}, \theta) \leq \delta \end{aligned}$$

- 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)

¹ $L_{\pi}(\tilde{\pi}) = V(\theta) + \sum_s \mu_{\pi}(s) \sum_a \tilde{\pi}(a|s) A_{\pi}(s, a)$

From Theory to Practice

Goal: so we have this objective function. Now let's try to put this into an algorithm that we can actually compute.

- Prior objective:

we don't actually know this. all we have is samples that we can use to estimate this.

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

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

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

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

Instead of taking an explicit sum over the state-space, where that state-space might be continuous/infinite, we just look at the states that were actually sampled by our current/old policy and reweight them.

$$\sum_s \mu_{\theta_{old}}(s) \rightarrow \frac{1}{1 - \gamma} \mathbb{E}_{s \sim \mu_{\theta_{old}}}[\dots],$$

From Theory to Practice

this is the probability distribution of actions given a state under our current policy. We have access to this, but the action-space could be continuous/infinite

- Next substitution:

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

Since the action-space could be continuous, we can use importance sampling.

we look at times we've taken a policy under our current policy and we reweight them based on the probability we would have taken those actions under the new policy. This allows us to approximate the expectation using data we already have

from our old policy

- 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}).

switch the Advantage back to the Q-function

- Third substitution:

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

- Note that these 3 substitutions do not change the solution to the above optimization problem

Selecting the Sampling Policy

- Optimize

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

Selecting the Sampling Policy

- Optimize

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

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

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

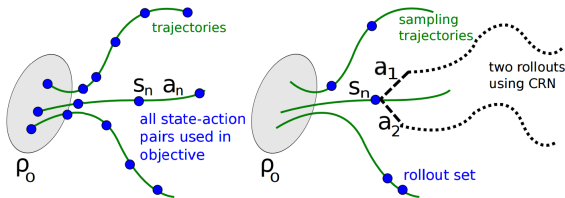


Figure: Trust Region Policy Optimization, Schulman et al, 2015

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

Table of Contents

- 1 Updating the Parameters Given the Gradient: Local Approximation
- 2 Updating the Parameters Given the Gradient: Trust Regions
- 3 Updating the Parameters Given the Gradient: TRPO Algorithm

Practical Algorithm: TRPO

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

Practical Algorithm: TRPO

Applied to

- Locomotion controllers in 2D

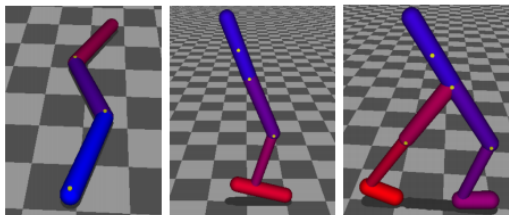


Figure: Trust Region Policy Optimization, Schulman et al, 2015

- Atari games with pixel input

TRPO Results

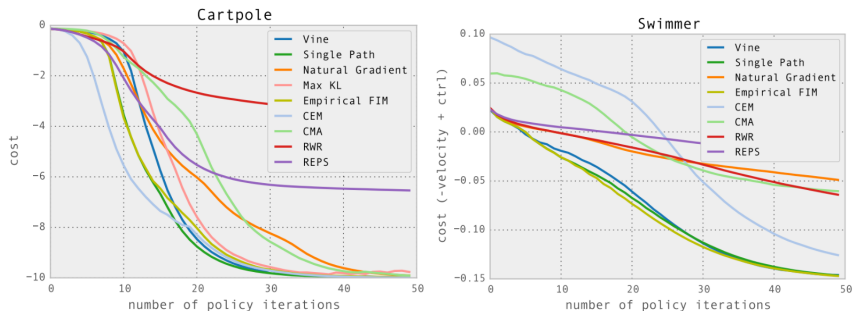


Figure: Trust Region Policy Optimization, Schulman et al, 2015

TRPO Results

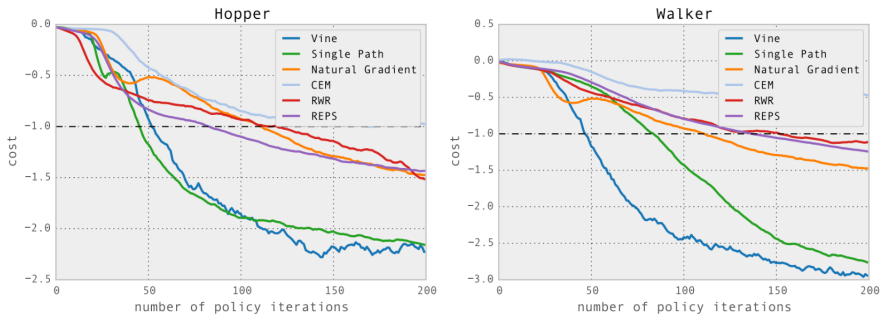


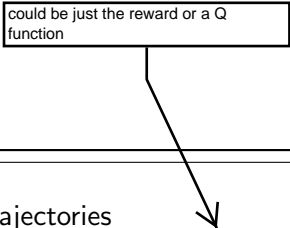
Figure: Trust Region Policy Optimization, Schulman et al, 2015

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

could be just the reward or a Q function



-
- 1: **for** iteration=1,2,... **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**
-

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

Class Structure

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