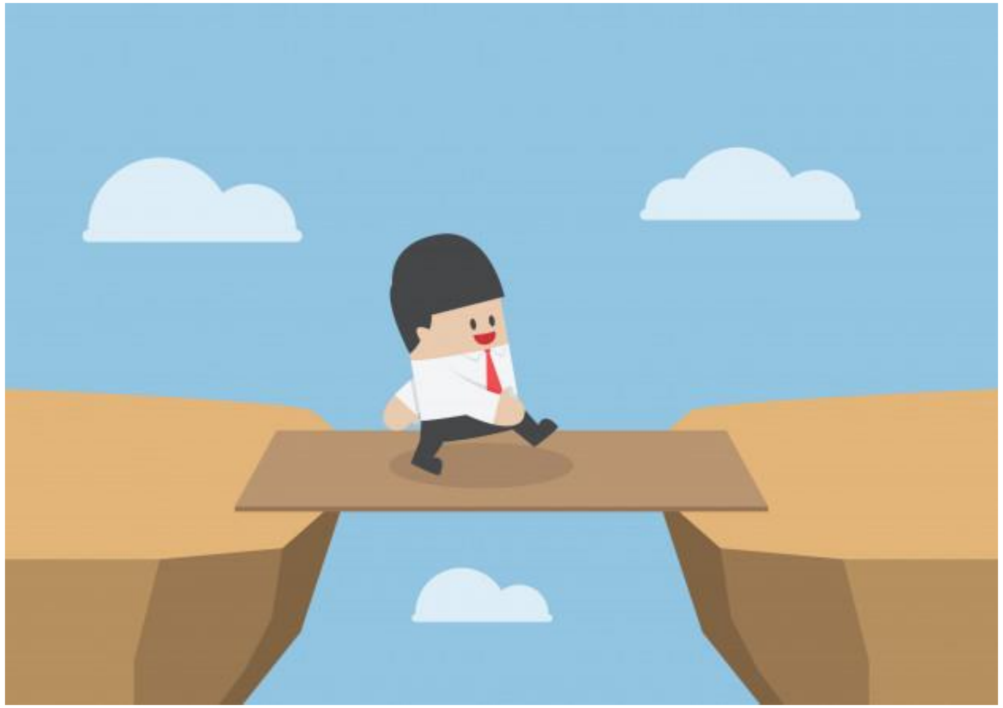
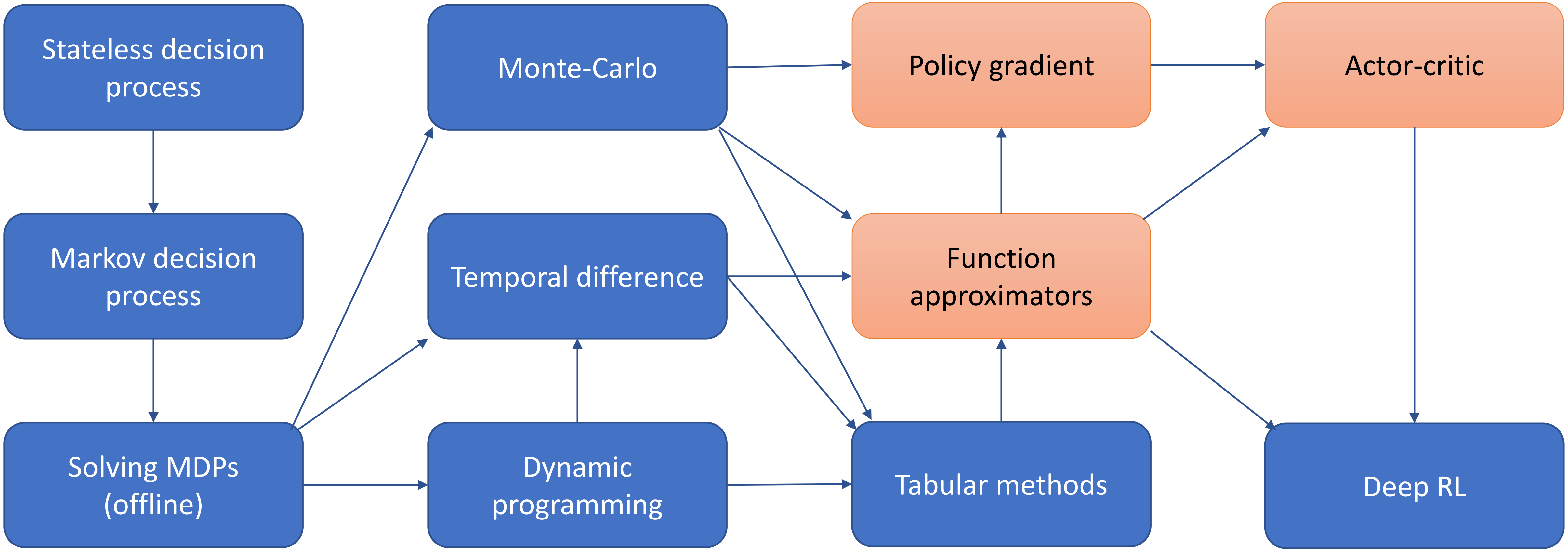
CSCE-642 Reinforcement Learning
Trust Regions



Instructor: Guni Sharon
Based on slides by: Pascal Poupart and John Schulman

CSCE-689, Reinforcement Learning

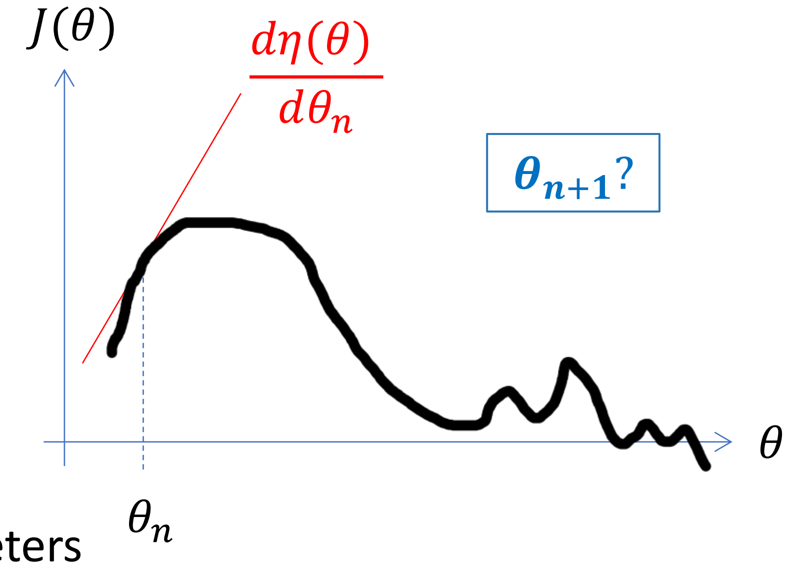


RL as an optimization problem

- ML: reduce learning to numerical optimization problem
  - Supervised learning: minimize training error
- RL: use all observations (environment interactions) to compute the optimal policy
  - Q learning: can (in principle) consider all transitions seen so far, however, we're optimizing a proxy objective (q instead of π)
  - PG methods: optimizes π but are not sample efficient (on policy learning)
  - We would like to formalize a gradient update step as an optimization over all previous samples
    - Input: data sampled from some policy
    - Output: a new policy which maximizes the expected return

Limitations of “vanilla” PG methods

- Hard to choose step size
  - Input data is nonstationary due to changing policy: state and reward distributions are non-stationary
  - Bad step size is more damaging than in supervised learning, since it affects the trajectory distribution
    - Step too far -> bad policy
    - Next batch: collected under bad policy
    - Can’t recover – collapse in performance
- Sample efficiency
  - Only one gradient step per environment sample
    - Need to better utilize previous experience
  - Dependent on scaling of coordinates
    - Uses the same learning rate for all states and all tunable parameters



Find the optimal point

- **PG:** Line search methods (e.g., gradient ascend/decent)
  - Find direction of improvement
  - Select step length
- **Today:** Trust region methods
  - Select a trust region radius (analog to max step length)
  - Find optimal point within trust region

Line search
(like gradient ascent)

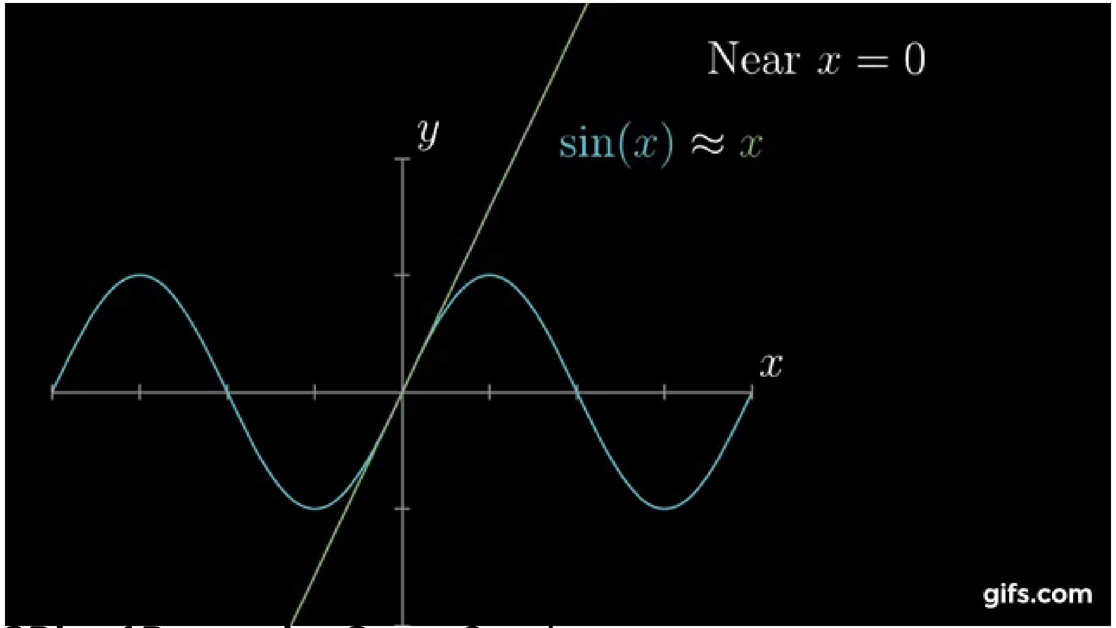
Trust region

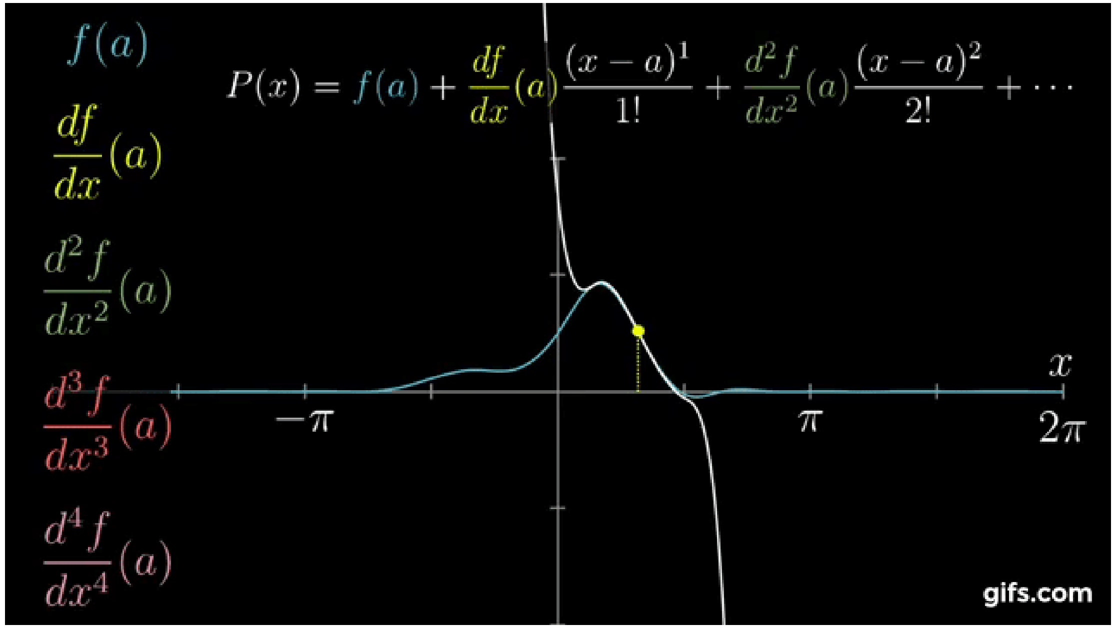
Trust region methods

- Idea:
  - Approximate objective f with a simpler function $\tilde{f}$
  - Solve: $\tilde{x}^* = \arg\min_x \tilde{f}(x)$
- Problem: The optimum $\tilde{x}^*$ might be in region where $\tilde{f}$ poorly approximates f so $\tilde{x}^*$ might be far from x^*
- Solution: restrict the search to a region where we trust $\tilde{f}$ to approximate f well
  - Solve: $\tilde{x}^* = argmin_{x \in trustRegion} \tilde{f}(x)$

Taylor series for approximation

- Approximate an unknown point $f(x')$ around a known point x
  - Assuming known derivatives $\frac{d^n f}{dx^n}$
- $f(x') = f(x) + \frac{df}{dx}(x)\frac{(x'-x)^1}{1!} + \frac{d^2 f}{dx^2}(x)\frac{(x'-x)^2}{2!} + \cdots + \frac{d^n f}{dx^n}(x)\frac{(x'-x)^n}{n!}$





3Blue1Brown by Grant Sanderson

Newton's method

- When considering a high dimensional (n) optimization problem, the k order partial derivatives is size n^k (grows exponentially!)
- Approximate $f(x)$ by a second order Tylor approximation around a know point c
- $f(X) \approx f(c) + \nabla f(c)^{\top}(X - c) + \frac{1}{2}(X - c)H(c)(X - c)$
  - Where $\nabla f(c)$ is the gradient and $H(c)$ is the Hessian at a known point c
- The delta of c from optimum X in the approximated function is found by setting its gradient (with respect to $(X - c)$) equal to zero, which gives:
- $\nabla f(c)^{\top} + H(c)(X - c) = 0$
- Newton step direction $= \underbrace{(X - c)}_{\text{Move from } c \text{ towards the approximated optimum } x} = -H^{-1}(c)\nabla f(c)$

Add a trust region

- Assume $\tilde{f}$ is a second-order Taylor polynomial
- $f(X) \approx \tilde{f}(X) = f(c) + \nabla f(c)^{\top}(X - c) + \frac{1}{2!}(X - c)H(c)(X - c)$
- The closer we are to c the more accurate
- Trust regions are often chosen to be a hypersphere (e.g., l2 norm)

$$\|X - c\|_2 \leq \delta$$

- But it might be that there is no local optimum within the trust region (no point with $\nabla f = [0]^d$)
    - Use Lagrange method to set the trust region boundaries!
    - (Lagrange method provides the optimum under constraint)

Generic algorithm

trustRegionMethod

init $\delta, x_0^*, n = 0$

Repeat:

$n \leftarrow n + 1$

Solve: $x_n^* = \arg\min_x \tilde{f}(x)$ s.t. $\|x - x_{n-1}^*\|_2 \leq \delta$

If $\tilde{f}(x_n^*) \approx f(x_n^*)$ than increase δ

Else decrease δ

Until convergence

Finding the optimal point

- Assume $\tilde{f}$ is a second-order Taylor polynomial
- $f(x) \approx \tilde{f}(x) = f(c) + \nabla f(c)^\top (x - c) + \frac{1}{2!}(x - c)H(c)(x - c)$
  - Subject to: $\|x - c\|_2 \leq \delta$
- If H is a positive (semi-)definite matrix
  - (Strictly) convex optimization
  - Can find the global optimum in closed form
  - Usually not the case in practical problems
- If H is not a positive semi-definite matrix
  - Non-convex optimization
  - Use gradient-based approaches that guarantee improvement per step
  - Find local optima

Non-convex optimization

- Solve: $x_n^* = \arg\min_x \tilde{f}(x)$ s.t. $\|x - x_{n-1}^*\|_2 \leq \delta$
- First: define an appropriate loss function

What loss do we minimize in PG?

- $\theta_{t+1} = \theta_t + \alpha \nabla L$
- Policy gradient (working backwards)
  - $\nabla \hat{J}(\theta) = \hat{A}(a_t, s_t) \nabla_\theta \ln \pi(a_t | s_t; \theta)$
  - $L^{PG}(\theta) = \hat{A}(a_t, s_t) \ln \pi(a_t | s_t; \theta)$
- Recall that: $\nabla_\theta \ln \pi(a_t | s_t; \theta) = \frac{\nabla_\theta \pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta)} = \nabla_\theta \frac{\pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta_{old})}$
- $L^{IS}(\theta) = \hat{A}(a_t, s_t) \frac{\pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta_{old})}$

$\pi(a_t | s_t; \theta_{old})$ is a constant

Important sampling ratio (see slide 28 from “4.Monte-Carlo”)

What do we actually want to optimize in PG?

- $\max_{\theta} \mathbb{E}_{s_t \sim \pi_\theta, a_t \sim \pi_\theta}[A(s_t, a_t)]$ (but we don't observe $s_t \sim \pi_\theta$ and $a_t \sim \pi_\theta$)
  - We sample from θ_{old} instead… Let's use importance sampling!
- $= \mathbb{E}_{s_t \sim \pi_{\theta_{old}}, a_t \sim \pi_{\theta_{old}}} \left[\frac{\pi(A_t|S_t;\theta)}{\pi(A_t|S_t;\theta_{old})} A(s_t, a_t) \right] = L^{IS}(\theta)$
- $L^{IS}(\theta)$ approximates the performance difference between θ and θ_{old}
- Similarity between π_θ and $\pi_{\theta_{old}}$ affects the variance in the approximated value
  - When π_θ and $\pi_{\theta_{old}}$ are far from each other than the $L^{IS}(\theta)$ approximation is usually untrustworthy

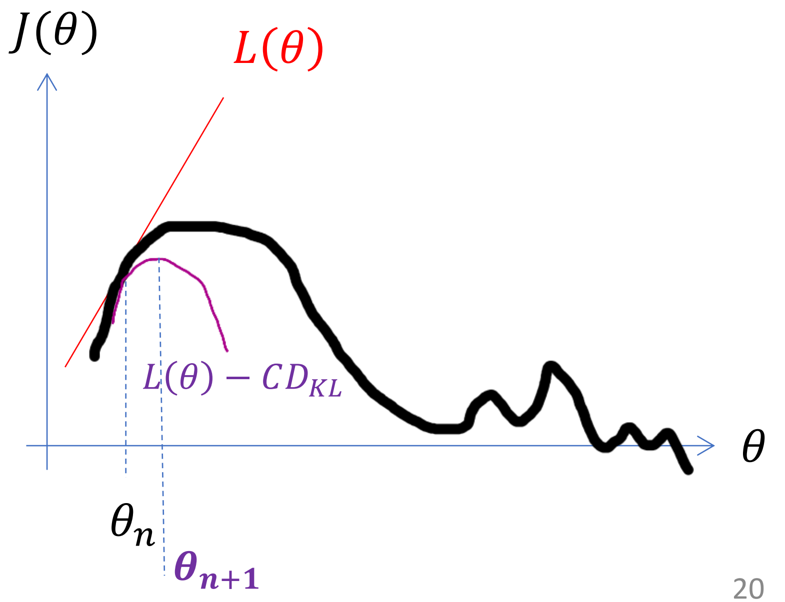
Trust Region Policy Optimization

- Define the following trust region update:
  - $\max_{\theta} \widehat{\mathbb{E}}_t \left[\hat{A}(A_t, S_t) \frac{\pi(A_t|S_t;\theta)}{\pi(A_t|S_t;\theta_{old})} \right]$
  - s.t. $\widehat{\mathbb{E}}_t \left[D_{KL}[\pi(\cdot|S_t;\theta_{old}), \pi(\cdot|S_t;\theta)] \right] \leq \delta$
  - That is, find the best next policy based on data from current policy while bounding the distance between the policies' distributions (KL divergence)
  - Off policy (IS-based) estimations are usually noisier the further they are from the behavior policy hence the necessity of bounding the change
- The constraint can be written as a penalty in the objective function (a Lagrangian)
  - $\max_{\theta} \widehat{\mathbb{E}}_t \left[\hat{A}(A_t, S_t) \frac{\pi(A_t|S_t;\theta)}{\pi(A_t|S_t;\theta_{old})} \right] - \beta \widehat{\mathbb{E}}_t \left[D_{KL}[\pi(\cdot|S_t;\theta_{old}), \pi(\cdot|S_t;\theta)] \right]$
  - For any δ in the constraint variant there is a β value such that both approaches result in the same optimality point [follows from the method of Lagrange multipliers]

Monotonic improvement results

- $J(\theta) \geq L^{IS}_{\theta_{old}}(\theta) - C \max_s D_{KL}[\pi(\cdot|s;\theta_{old}), \pi(\cdot|s;\theta)]$
- Where: $J(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_t]$ is the expected return, $C = \frac{2\epsilon\gamma}{(1-\gamma)^2}$, $\epsilon = \max_s \left| \mathbb{E}_{a\sim\pi(a|s;\theta)}\left[A_{\theta_{old}}(s,a)\right]\right|$
- The optimal point in $L(\theta) - CD_{KL}$
  - Guaranteed to improve $J(\theta)$ over $J(\theta_{old})$

Note that the original paper considers costs instead of rewards so \ge becomes \le



Schulman, John, et al. "Trust region policy optimization." *International conference on machine learning*. 2015.

TRPO algorithm

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

Run policy for T timesteps or N trajectories

Estimate advantage function at all timesteps

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \sum_{n=1}^{N} \frac{\pi_\theta(a_n \mid s_n)}{\pi_{\theta_{\text{old}}}(a_n \mid s_n)} \hat{A}_n \\ \text{subject to} \quad & \overline{\mathrm{KL}}_{\pi_{\theta_{\text{old}}}}(\pi_\theta) \le \delta \end{aligned}$$

end for

Take the mean or the max?
The monotonic improvement guarantees hold only for max but in practice mean works better (max is too conservative)

Solving KL penalized problem

- $\underset{\theta}{\text{maximize}}\, L^{IS}(\pi_\theta) - \beta \cdot \overline{KL}\left(\pi_{\theta_{old}}, \pi_\theta\right)$
- Use a linear approximation for L and a quadratic approximation for $\overline{KL}$
- $\underset{\theta}{\text{maximize}}\, g(\theta - \theta_{old}) - \frac{\beta}{2} \cdot (\theta - \theta_{old})^\top F(\theta - \theta_{old})$
- $g = \frac{\partial L}{\partial \theta}$ (the policy gradient), $F = \frac{\partial^2 \overline{KL}}{\partial \theta^2}$ (the fisher information matrix)
- Newton step direction: $F^{-1} g$
- AKA: natural policy gradient [Kakade, 2002]

TRPO as a general PG algorithm

$$\underset{\theta}{\text{maximize}} \sum_{n=1}^{N} \frac{\pi_\theta(a_n \mid s_n)}{\pi_{\theta_{\text{old}}}(a_n \mid s_n)} \hat{A}_n$$

$$\text{subject to} \quad \overline{\mathrm{KL}}_{\pi_{\theta_{\text{old}}}}(\pi_\theta) \leq \delta$$

- Linear-quadratic approximation + penalty = Natural gradient
- No constraint ($\delta = \infty$): policy iteration
  - New policy updated to be the best one step policy over states visited during the previous policy
- Euclidean constraint instead of KL (corresponds to learning rate) = vanilla policy gradient

Review

- Optimize a surrogate loss L^{PG} or L^{IS}
  - The loss that results in the policy gradient
- Add a constraint in the form of bounded KL divergence
- Under linear (for L^{IS}) and quadratic (for KL) Tylor approximations it corresponds to natural gradient step $F^{-1}g$
- We can approximate the Fisher matrix relatively fast using Conjugate gradient [Kakade, 2002]
- The Fisher matrix can be very big ($|\theta| \times |\theta|$)

Avoiding the Fisher matrix computation

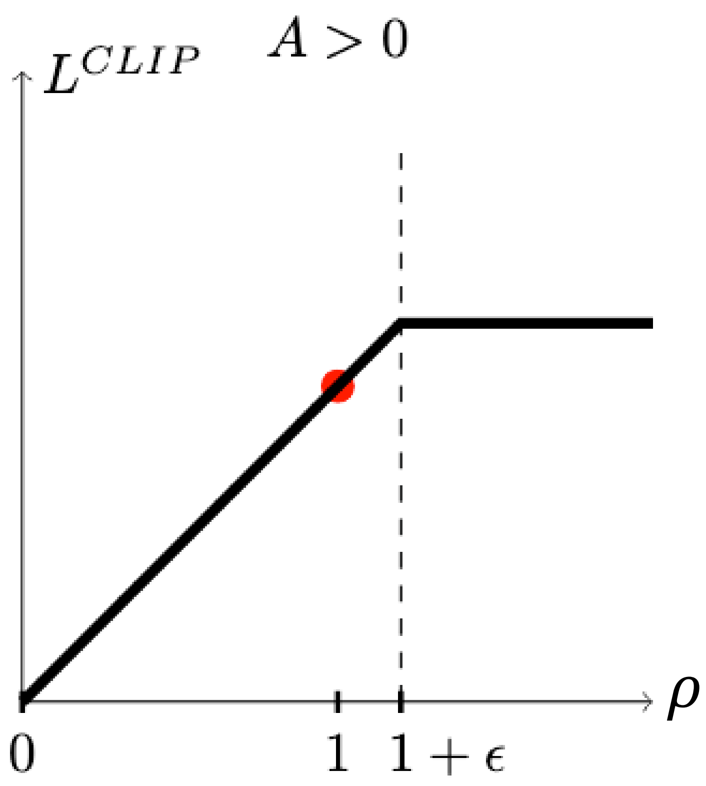
- Use only first order approximation + Euclidian bound
- Proximal Policy Optimization [Schulman et al., 2017]
- $\max_{\theta} \widehat{\mathbb{E}}_t \left[\hat{A}(A_t, S_t) \frac{\pi(A_t|S_t;\theta)}{\pi(A_t|S_t;\theta_{old})} - \beta D_{KL}[\pi(\cdot|S_t;\theta_{old}), \pi(\cdot|S_t;\theta)] \right]$
- Becomes:
- $\max_{\theta} \widehat{\mathbb{E}}_t \left[\min\left(\rho_t(\theta)\hat{A}_t, \mathrm{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right) \right]$
- Where $\rho_t(\theta) = \frac{\pi(A_t|S_t;\theta)}{\pi(A_t|S_t;\theta_{old})}$, $\hat{A}_t = \hat{A}(A_t, S_t)$, ϵ is a hyperparameter

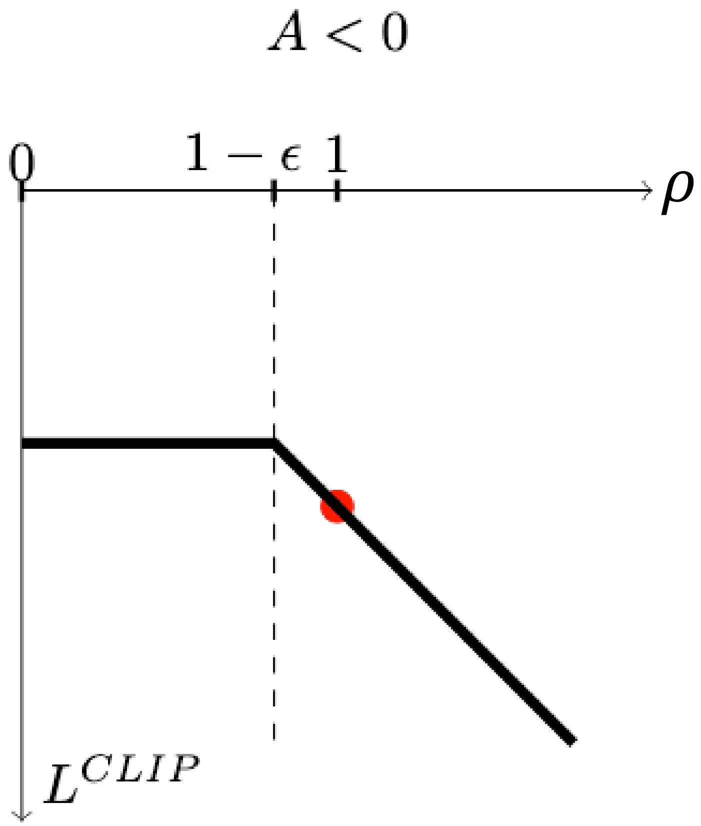
Proximal Policy Optimization [Schulman et al., 2017]

- $\max_\theta \hat{\mathbb{E}}_t \left[\min\left(\rho_t(\theta)\hat{A}_t, \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$
- The **first** term will find the policy that maximizes return assuming the state distribution is similar to the old policy (policy iteration)
- The **second** modifies the surrogate objective by clipping $\rho_t(\theta)$, which removes the incentive for moving ρ_t outside of the interval
- Take the minimum so the final objective is a lower bound (i.e., a pessimistic bound) on the unclipped objective
- We only ignore the change in probability ratio when it would make the objective improve

Proximal Policy Optimization [Schulman et al., 2017]

- $\max_{\theta} \hat{\mathbb{E}}_t \left[\min\left(\rho_t(\theta)\hat{A}_t, \mathrm{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$





Results

No clipping or penalty: $L_t(\theta) = r_t(\theta)\hat{A}_t$

Clipping: $L_t(\theta) = \min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta)), 1-\epsilon, 1+\epsilon)\hat{A}_t$

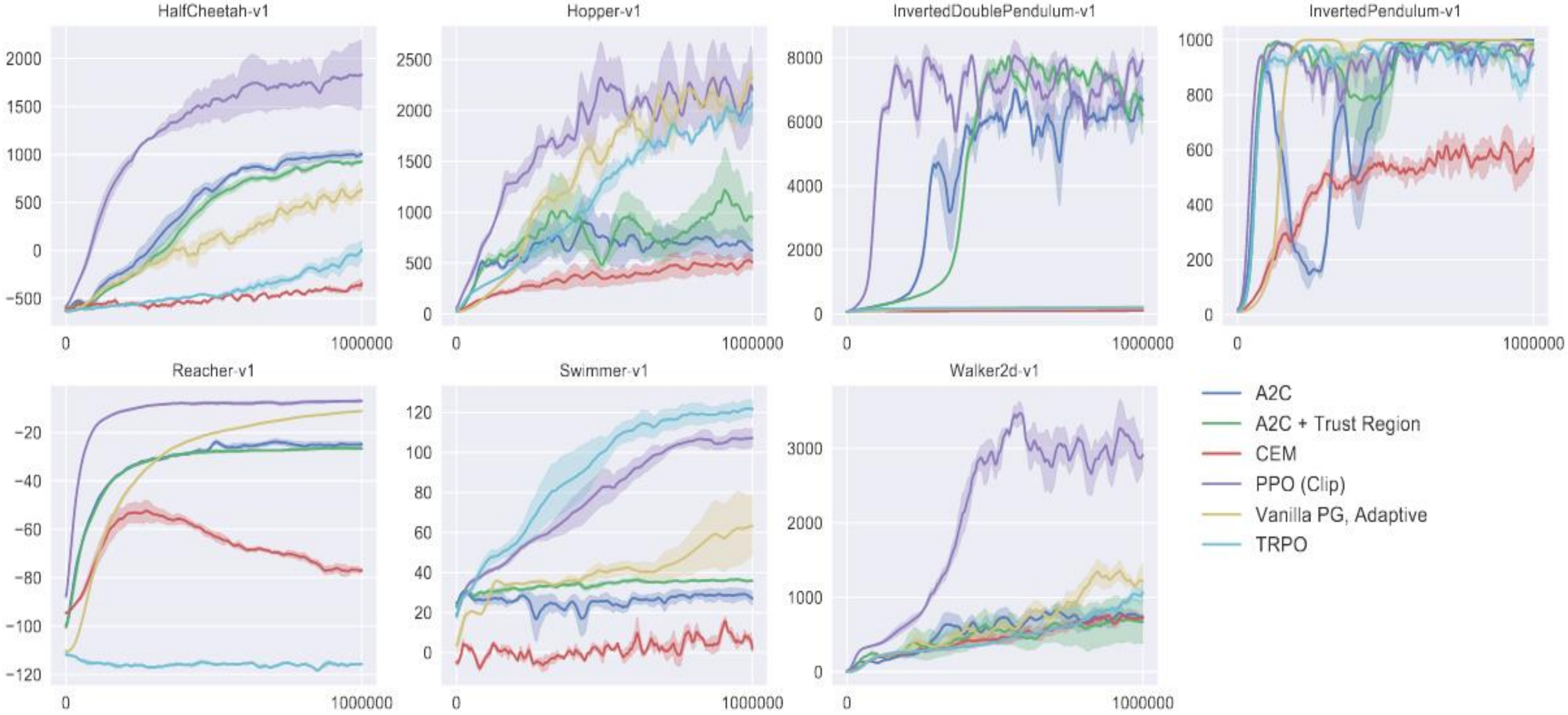
KL penalty (fixed or adaptive) $L_t(\theta) = r_t(\theta)\hat{A}_t - \beta\,\text{KL}[\pi_{\theta_{\text{old}}}, \pi_\theta]$

Results from continuous control benchmark. Average normalized scores (over 21 runs of the algorithm, on 7 environments) for each algorithm / hyperparameter setting .

algorithm	avg. normalized score
No clipping or penalty	-0.39
Clipping, $\epsilon = 0.1$	0.76
Clipping, $\epsilon = 0.2$	**0.82**
Clipping, $\epsilon = 0.3$	0.70
Adaptive KL $d_{\text{targ}} = 0.003$	0.68
Adaptive KL $d_{\text{targ}} = 0.01$	0.74
Adaptive KL $d_{\text{targ}} = 0.03$	0.71
Fixed KL, $\beta = 0.3$	0.62
Fixed KL, $\beta = 1.$	0.71
Fixed KL, $\beta = 3.$	0.72
Fixed KL, $\beta = 10.$	0.69

RESULTS 2

Comparison on several MuJoCo environments training for one million time steps



What did we learn?

- PG is sensitive to the policy update step size
- A big step at some direction might seem like a good idea but in practice it will send us over a cliff
- When observing policy π and trying to evaluate a policy π'
  - The more similar π and π' are (measured with KL divergence) the more trust we have in the evaluation of π'
- Limit the magnitude of change between successive policies such that you are guaranteed to see an improvement (natural gradient)
- Improvement guarantees require too conservative steps sizes
  - Forsake such guarantees to get better performance in practice

What next?

- **Lecture**: Soft Actor-Critic
- **Assignments**:
  - A2C
  - REINFORCE
  - Deep Q-Learning
- **Quiz (on Canvas)**:
  - Policy Gradient
  - Deep Q-Learning
- **Project**:
  - Literature survey