Trust-Region Policy Optimization

Christopher Mutschler

Policy Gradients so far

- We wanted to do the following:

$$\max_{\theta} J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left(\sum_{t=0}^{\infty} \gamma^t r_t\right)$$

- Then we defined the policy gradient:

$$g = \nabla_\theta J(\pi_\theta) = \nabla_\theta \mathbb{E}_{\pi_\theta} G(\tau) \approx \frac{1}{L} \sum_{\tau} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\, \hat{A}_t$$

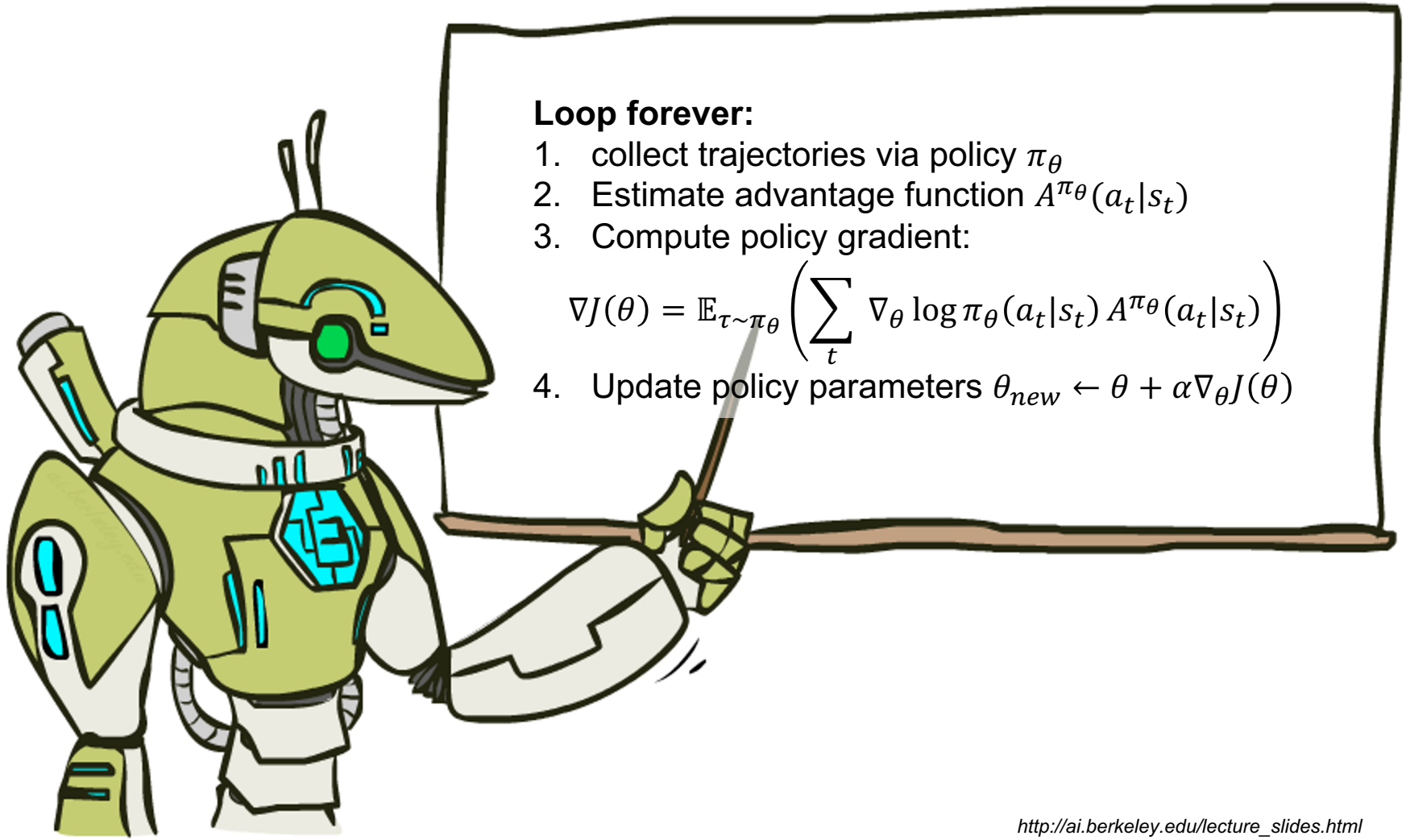
- If $\hat{A}_t > 0$ (< 0):
  - Gradient becomes positive (negative)
  - The probability of taking a_t in s_t is increased (decreased)

- We update our policy parameters by

$$\theta_{k+1} = \theta_k + \alpha \cdot g$$

Take a gradient step in updating the policy (gradient ascent)

Policy Gradients so far



Loop forever:

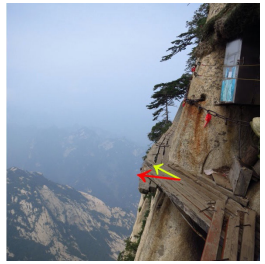
1. collect trajectories via policy π_θ
2. Estimate advantage function $A^{\pi_\theta}(a_t|s_t)$
3. Compute policy gradient:

$$\nabla J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left(\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t)\, A^{\pi_\theta}(a_t|s_t)\right)$$

4. Update policy parameters $\theta_{new} \leftarrow \theta + \alpha\nabla_\theta J(\theta)$

http://ai.berkeley.edu/lecture_slides.html

Policy Gradients so far



Problem

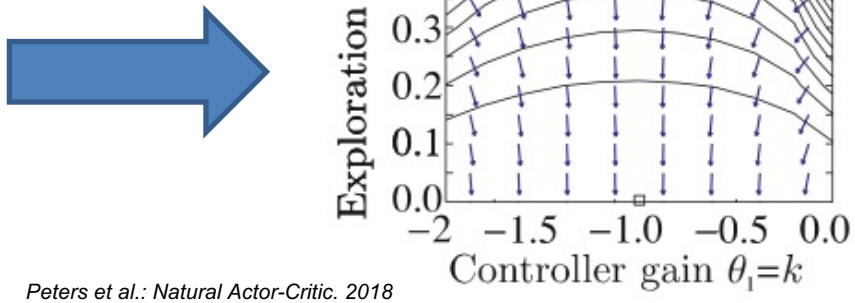
- Run gradient descent/ascent on one batch of collected experience
- Note: the advantage function (which is a noisy estimate) may not be accurate
  - Too large steps may lead to a disaster (even *if* the gradient is *correct*)
  - Too small steps are also bad
- Definition and scheduling of learning rates in RL is tricky as the underlying data distribution changes with updates to the policy
  - Mathematical formulization:
    - First-order derivatives approximate the (parameter) surface to be flat
    - But if the surface exhibits high curvature it gets dangerous
    - Projection: small changes in parameter space might lead to large changes in policy space!
- Parameters θ get updated to areas too far out of the range from where previous data was collected *(note: a bad policy leads to bad data)*

Images taken from https://medium.com/@jonathan_hui/rl-trust-region-policy-optimization-trpo-explained-a6ee04eeeee9 and http://www.taiwanoffthebeatentrack.com/2012/08/23/mount-hua-华山-the-most-dangerous-hike-in-the-world/

Primer: Natural Policy Gradient

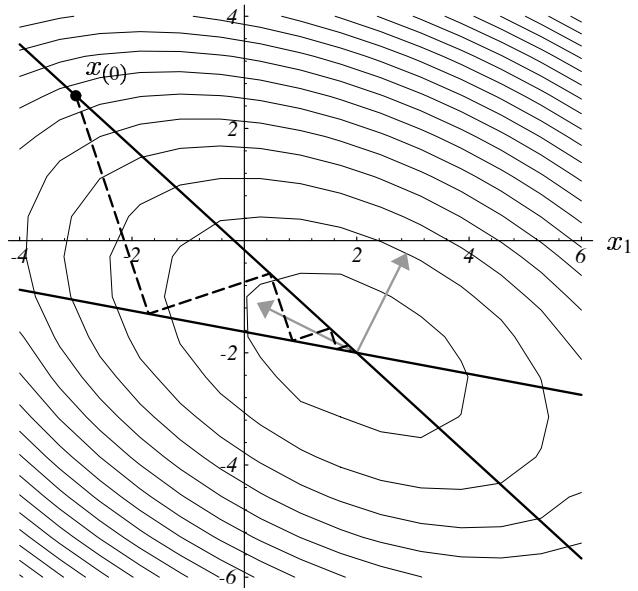
There is something wrong with the policy gradient. Example:

- $r(s_t, a_t) = -s_t^2 - a_t^2$
- $\log \pi_\theta(a_t|s_t) = -\frac{1}{2\sigma^2}(ks_t - a_t)^2 + c$, with $\theta = (k, \sigma)$



Peters et al.: Natural Actor-Critic. 2018

poor conditioning



Shewchuk: An Introduction to the Conjugate Gradient Method without the Agonizing Pain. Edition 1 ¼.

* Example take from Sergey Levine's CS285: Advanced Policy Gradients

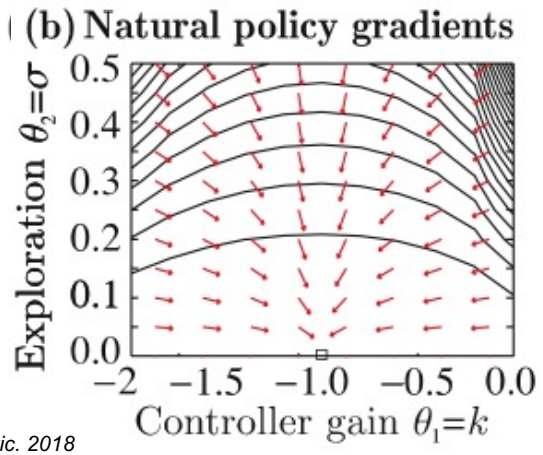
Primer: Natural Policy Gradient

- Given:
  - $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ and $\pi_\theta(a_t|s_t)$
- Some parameters change probabilities a lot more than other!
  → a single value for α is not sufficient
- What we essentially do (optimization perspective on 1st order gradient descent):
  - $\theta' \leftarrow \arg\max_{\theta'}(\theta' - \theta)^T \nabla_\theta J(\theta)$, subject to $\|\theta' - \theta\|^2 \leq \epsilon$
  - $\theta' \leftarrow \arg\max_{\theta'}(\theta' - \theta)^T \nabla_\theta J(\theta)$, subject to $D(\pi_{\theta'}, \pi_\theta) \leq \epsilon$

"trust region"

But how to rescale???

→ For instance, KL-divergence



Peters et al.: Natural Actor-Critic. 2018

Better with 2nd order information

→ $\theta' \leftarrow \arg\max_{\theta'}(\theta' - \theta)^T \nabla_\theta J(\theta)$, s.t. $\|\theta' - \theta\|_{\mathbf{F}}^2 \leq \epsilon$

→ $\theta \leftarrow \theta + \alpha \mathbf{F}^{-1} \nabla_\theta J(\theta)$

Trust-Region Policy Optimization (TRPO)

“Simple” Idea

Regularize updates to the policy parameters,
such that the policy does not change too much.

Primer: Trust-Region Methods

Optimization in Machine Learning

- It is common to formulate ML problems as optimization problems:
  - Minimize the squared error
  - Minimize the cross entropy
  - Maximize the log-likelihood
  - Maximize the discounted sum of rewards
  - Minimize the sum of control cost

Primer: Trust-Region Methods

Optimization in Machine Learning: two classes

1. Line Search, e.g., gradient descent
   - find a (some) direction of improvement
   - (cleverly) select a step length

2. Trust-Region Methods
   - select a trust region (analog to max step length)
   - find a point of improvement in that region

Primer: Trust-Region Methods

- **Idea:**
  - Approximate the real objective f with something simpler, i.e., $\tilde{f}$
  - Solve $\tilde{x}^* = \arg\min_x \tilde{f}(x)$
- **Problem:**
  - The optimum $\tilde{x}^*$ might be in a region where $\tilde{f}$ poorly approximates f
  - $\tilde{x}^*$ might be far from optimal
- **Solution:**
  - Restrict the search to a region tr where we trust $\tilde{f}$ to approximate f well
  - Solve $\tilde{x}^* = \arg\min_{x \in tr} \tilde{f}(x)$

Primer: Trust-Region Methods

2nd order Taylor approximation

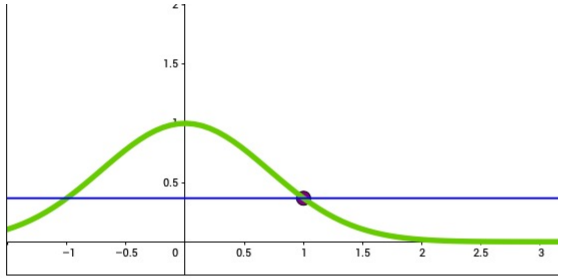
- Example: chose $\tilde{f}$ to be quadratic approximation of f:

$$f(x) \approx \tilde{f}(x) = f(c) + \nabla f(c)^T (x - c) + \frac{1}{2!}(x - c)^T H(c)(x - c),$$

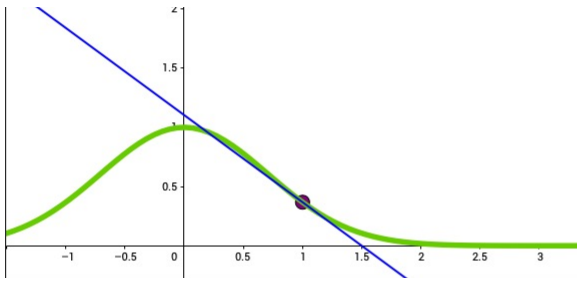
where ∇f is the gradient and H is the Hessian

→ $p_n(x) = \sum_{j=0}^{n} \frac{1}{j!} f^{(j)}(a)(x - a)^j$

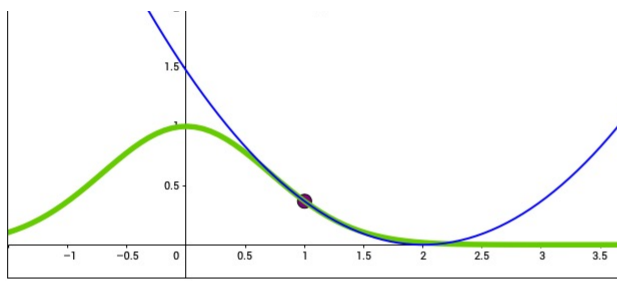
Example: nth order Taylor approximation of $f(x) = e^{-x^2}, c = 1$



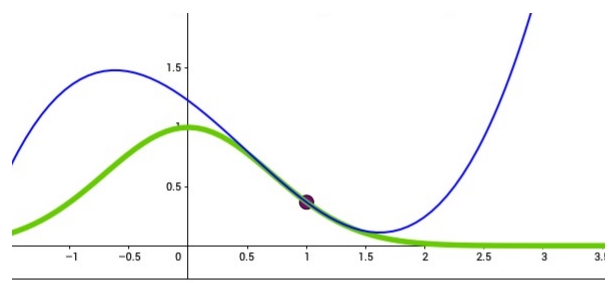
$p_0(x) = 0.37$



$p_1(x) = p_0(x) - 0.74 \cdot (x - 1)$



$p_2(x) = p_1(x) + 0.74 \cdot \frac{(x-1)^2}{2!}$



$p_3(x) = p_2(x) + 1.47 \cdot \frac{(x-1)^3}{3!}$

https://mathinsight.org/applet/taylor_polynomial

Primer: Trust-Region Methods

2nd order Taylor approximation

- Example: chose $\tilde{f}$ to be quadratic approximation of f:

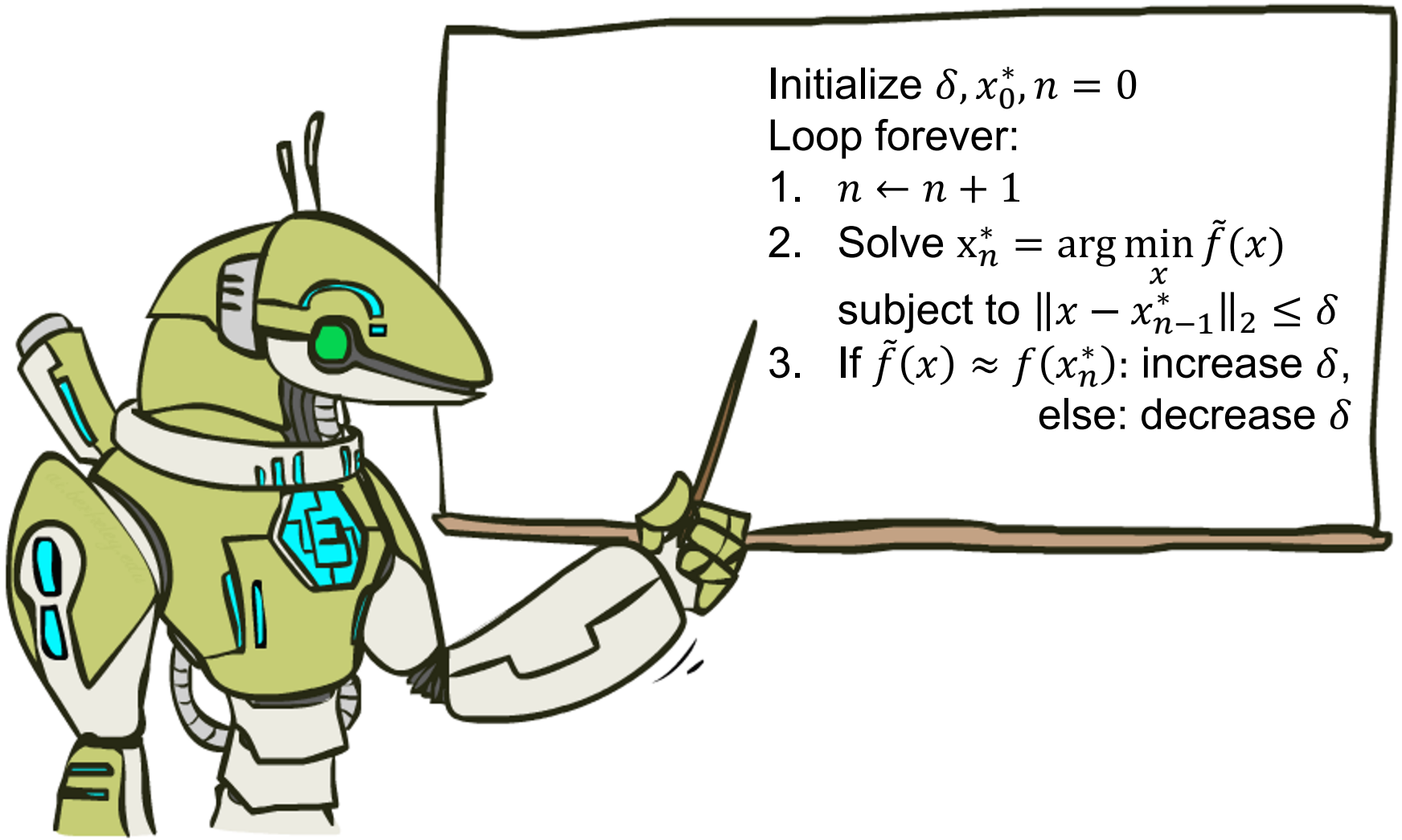
$$f(x) \approx \tilde{f}(x) = f(c) + \nabla f(c)^T (x - c) + \frac{1}{2!}(x - c)^T H(c)(x - c),$$

where ∇f is the gradient and H is the Hessian

- The trust region is often chosen to be a hypersphere:

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

Primer: Trust-Region Methods



Primer: Trust-Region Methods

2nd order Taylor approximation

- $\tilde{f}$ often chosen to be quadratic approximation of f:

$$\min_x f(c) + \nabla f(c)^T (x - c) + \frac{1}{2!}(x - c)^T H(c)(x - c),$$

subject to $\|x - c\|_2 \leq \delta$.

- When H is positive semi-definite
  - Convex optimization
  - Simple and globally optimal solution
  - e.g.: conjugate gradient

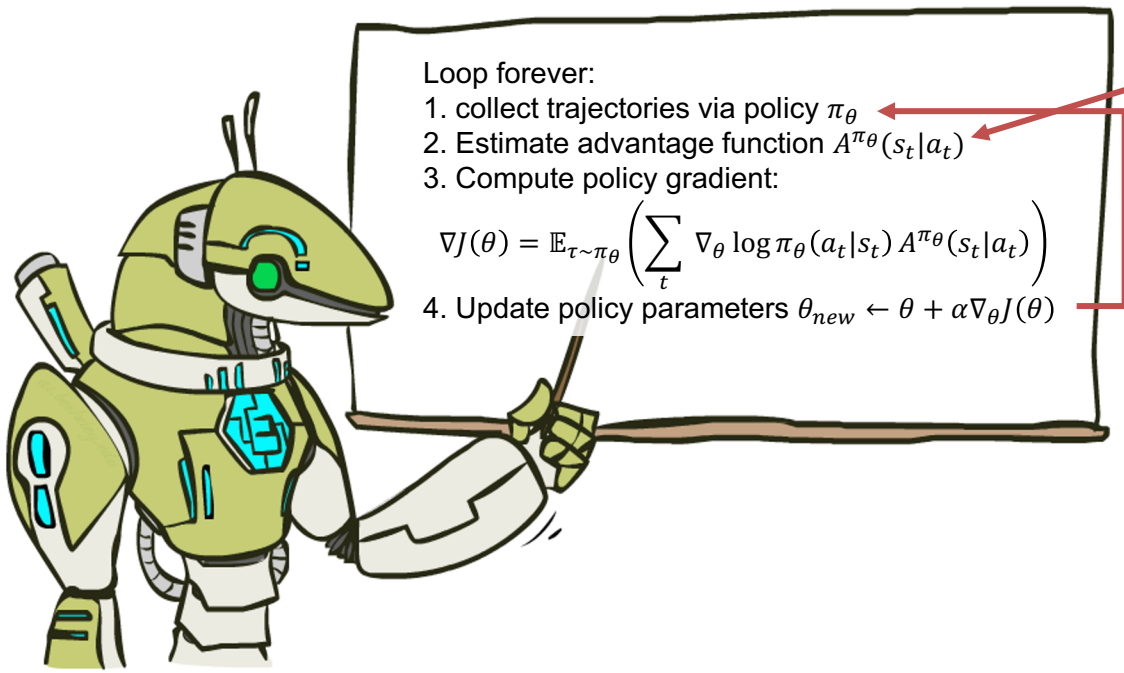
- When H is not positive semi-definite
  - Non-convex optimization
  - Simple heuristics that guarantee improvement

Trust-Region Policy Optimization (TRPO)

So back to what we actually do…

The problem(s) of the Policy Gradient (PG) is that

- PG keeps old and new policy close in parameter space, while
- small changes can lead to large differences in performance, and
- "large" step-sizes hurt performance (whatever "large" means…)



random at beginning

Non-stationary input data due to changing policy and reward distribution change

Update carefully

→ We want improvement and not degradation

Trust-Region Policy Optimization (TRPO)

- We want to optimize $\eta(\pi)$, i.e., the expected return of policy π:

$$\eta(\pi) = \mathbb{E}_{s_0 \sim \rho_0, a^t \sim \pi_{old}(\cdot|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

- We collect data with π_{old} and optimize to get a new policy π_{new}
- In fact, so far…
  - we did not really *optimize* anything, as
  - we just favored those $(a_t|s_t)$ that had a higher advantage
- Question:
  - How can we write down an optimization problem that allows to do small updates on a policy π based on data sampled from π (on-policy data)?

[1] *Kakade et al.: Approximately Optimal Approximate Reinforcement Learning. ICML 2002.*

Trust-Region Policy Optimization (TRPO)

- We want to optimize $\eta(\pi)$, i.e., the expected return of policy π:

$$\eta(\pi) = \mathbb{E}_{s_0 \sim \rho_0, a^t \sim \pi_{old}(\cdot|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

- We collect data with π_{old} and optimize to get a new policy π_{new}
- Let's express $\eta(\pi_{new})$ in terms of advantage over the original policy[1]:

$$\eta(\pi_{new}) = \eta(\pi_{old}) + \mathbb{E}_{\tau \sim \pi_{new}} \left[\sum_{t=0}^{\infty} \gamma^t A_{\pi_{old}}(s_t, a_t) \right]$$

Expected return of the new policy

Expected return of the old policy

Sample from new policy

[1] *Kakade et al.: Approximately Optimal Approximate Reinforcement Learning. ICML 2002.*

Trust-Region Policy Optimization (TRPO)

- We want to optimize $\eta(\pi)$, i.e., the expected return of policy π:

$$\eta(\pi) = \mathbb{E}_{s_0 \sim \rho_0, a^t \sim \pi_{old}(\cdot|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

- We collect data with π_{old} and optimize to get a new policy π_{new}
- Let's express $\eta(\pi_{new})$ in terms of advantage over the original policy[1]:

$$\eta(\pi_{new}) = \eta(\pi_{old}) + \mathbb{E}_{\tau \sim \pi_{new}} \left[\sum_{t=0}^{\infty} \gamma^t A_{\pi_{old}}(s_t, a_t) \right]$$

$$= \eta(\pi_{old}) + \sum_s \rho_{\pi_{new}}(s) \sum_a \pi_{new}(a|s) A_{\pi_{old}}(s, a)$$

Discounted visitation frequency according to **new** policy:

$$\rho_{\pi_{new}}(s) = P(s_0 = s) + \gamma P(s_1 = s) + \gamma^2 P(s_2 = s) + \cdots$$

[1] *Schulman et al.: Trust-Region Policy Optimization. ICML 2015.*

Trust-Region Policy Optimization (TRPO)

- We want to optimize $\eta(\pi)$, i.e., the expected return of policy π:

$$\eta(\pi) = \mathbb{E}_{s_0 \sim \rho_0, a^t \sim \pi_{old}(\cdot|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

- We collect data with π_{old} and optimize to get a new policy π_{new}
- Let's express $\eta(\pi_{new})$ in terms of advantage over the original policy:

$$\eta(\pi_{new}) = \eta(\pi_{old}) + \mathbb{E}_{\tau \sim \pi_{new}} \left[\sum_{t=0}^{\infty} \gamma^t A_{\pi_{old}}(s_t, a_t) \right]$$

$$= \eta(\pi_{old}) + \sum_s \rho_{\pi_{new}}(s) \sum_a \underbrace{\pi_{new}(a|s) A_{\pi_{old}}(s,a)}_{> 0}$$

new expected return $>$ old expected return

If we can guarantee this…

→ New objective guarantees improvement from $\pi_{old} \rightarrow \pi_{new}$

Trust-Region Policy Optimization (TRPO)

- We want to optimize $\eta(\pi)$, i.e., the expected return of policy π:

$$\eta(\pi) = \mathbb{E}_{s_0 \sim \rho_0, a^t \sim \pi_{old}(\cdot|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

- We collect data with π_{old} and optimize to get a new policy π_{new}
- Let's express $\eta(\pi_{new})$ in terms of advantage over the original policy:

$$\eta(\pi_{new}) = \eta(\pi_{old}) + \mathbb{E}_{\tau \sim \pi_{new}} \left[\sum_{t=0}^{\infty} \gamma^t A_{\pi_{old}}(s_t, a_t) \right]$$

$$= \eta(\pi_{old}) + \sum_{s} \rho_{\pi_{new}}(s) \sum_{a} \pi_{new}(a|s) A_{\pi_{old}}(s, a)$$

However, this cannot be easily estimated. The state visitations that we sampled so far are coming from the old policy!

→ **we cannot optimize this in the current form!**

Trust-Region Policy Optimization (TRPO)

$$\eta(\pi_{new}) = \eta(\pi_{old}) + \sum_s \rho_{\pi_{new}}(s) \sum_a \pi_{new}(a|s) A_{\pi_{old}}(s,a)$$

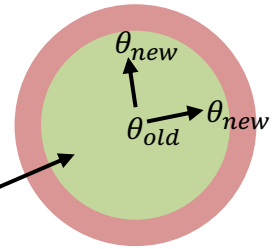
approximate locally

$$\approx$$

$$L(\pi_{new}) = \eta(\pi_{old}) + \sum_s \rho_{\pi_{old}}(s) \sum_a \pi_{new}(a|s) A_{\pi_{old}}(s,a)$$

This we already sampled
→ We already have this!

- The approximation is accurate within step size δ (trust region)
  - δ needs to be chosen based on a lower-bound approximation error
- Monotonic improvement guaranteed
  - (within the green region!)



Trust region
→ $\pi_{\theta_{new}}(s,a)$ does not change too much

Trust-Region Policy Optimization (TRPO)

- If we want to optimize $L(\theta_{new})$ instead of $\eta(\theta_{new})$ …
  with a guarantee of monotonic improvement on $\eta(\theta_{new})$, …
  … we need a bound on $L(\theta_{new})$.

- It can be proven that there exists the following bound[1,2]:

$$\eta(\pi_{new}) \geq L(\pi_{new}) - C \cdot D_{KL}^{max}(\pi_{old}, \pi_{new}), \text{ where } C = \frac{4\varepsilon\gamma}{(1-\gamma)^2}$$

[1] *Schulman et al.: Trust-Region Policy Optimization. ICML 2015.*
[2] *Kakade et al.: Approximately Optimal Approximate Reinforcement Learning. ICML 2002.*

Primer: KL-Divergence

- A measure of distance between two probability distributions P and Q:

$$D_{KL}(P\|Q) = \mathbb{E}_{x\sim P}\left[\log\frac{P(x)}{Q(x)}\right] = \mathbb{E}_{x\sim P}[\log P(x) - \log Q(x)]$$

- Intuition stems from theory of channel coding:
  - The amount of information I (in *bits* or *nats*) that is transmitted from a sender to a receiver depends on the actual probability of the actual incident/event

$$I(x) = \log_a\left(\frac{1}{p_x}\right) = \log_a 1 - \log_a p_x = -\log_a p(x)$$

  - *Example #1: two events A and B have a probability of 0.75 and 0.25; then the information that A has happened is $-log_2(0.75) = 0.41$ (and for B it is 2)*
  - *Example #2: A and B are equally likely, i.e., 0.5 each; then the information about which of them has happened is $-log_2(0.5) = 1$*
  - If we use base 2 it tells us how much uncertainty is cut off by half!
- For simplification we use nats as $\log_2 x = \log x/\log 2$ ($\log 2$ is constant)

Primer: KL-Divergence

- The entropy (of a probability distribution) is given by:

$$H(p) = -\sum_i p_i \log(p_i)$$

  - It measures the average amount of information from one sample drawn from the underlying probability distribution p
    → it measures how unpredictable p is
    → defines the lower bound of the optimal bit-encoding

- As usual, we can also write $H(p)$ in its expectation and draw x from p:

$$H(p) = \mathbb{E}_{x\sim p}[-\log p(x)]$$

Primer: KL-Divergence

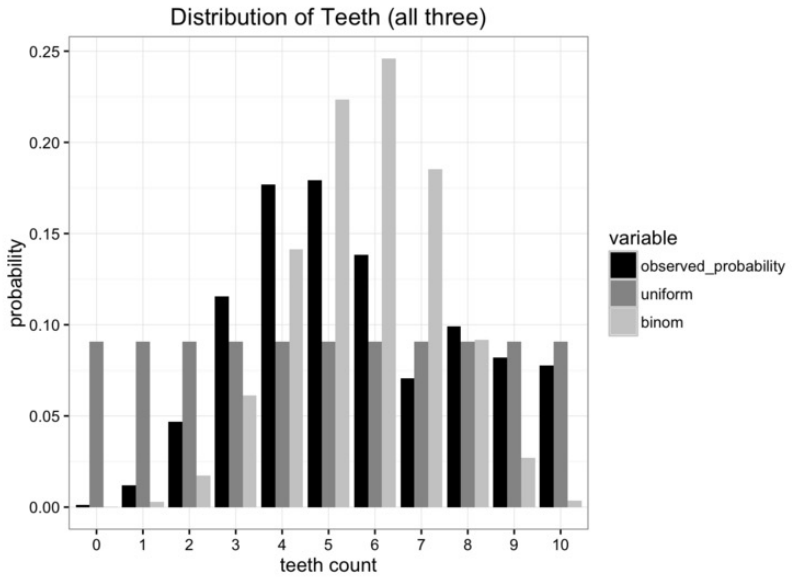
- The cross-entropy defines the average number of bits to identify a sampled event if it is encoded using q rather than p:

$$H(p, q) = -\sum_i p_i \log_2(q_i)$$

  - tells us the difference about the true probability distribution p and the predicted probability distribution q (given the encoding)
  - If p and q are equal, then the cross-entropy and the entropy are equal

- As before, we can write $H(p, q)$ in its expectation:

$$H(p, q) = \mathbb{E}_{x \sim p}[-\log q(x)]$$



from https://www.countbayesie.com/blog/2017/5/9/kullback-leibler-divergence-explained

Primer: KL-Divergence

- KL-Divergence tells us how much far away the cross-entropy is from the entropy (note that $H(p,q) \geq H(p)$) :

$$D_{KL}(p||q) = H(p,q) - H(p)$$

- Measure of dissimilarity between two probability distributions P and Q:

$$\begin{aligned} D_{KL}(p\|q) &= \mathbb{E}_{x\sim p}[-\log q(x)] - \mathbb{E}_{x\sim p}[-\log p(x)] \\ &= \mathbb{E}_{x\sim p}[-\log q(x) - (-\log p(x))] \\ &= \mathbb{E}_{x\sim p}[-\log q(x) + \log p(x)] \\ &= \mathbb{E}_{x\sim p}[\log p(x) - \log q(x)] = \mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right] \end{aligned}$$

$D_{KL}(p||q) = \sum_i p(i) \log \frac{p(i)}{q(i)}$ and $D_{KL}(p||q) = \int P(x) \log \frac{p(x)}{q(x)} dx$

- Note: KL-Divergence is asymmetric and hence no distance metric!

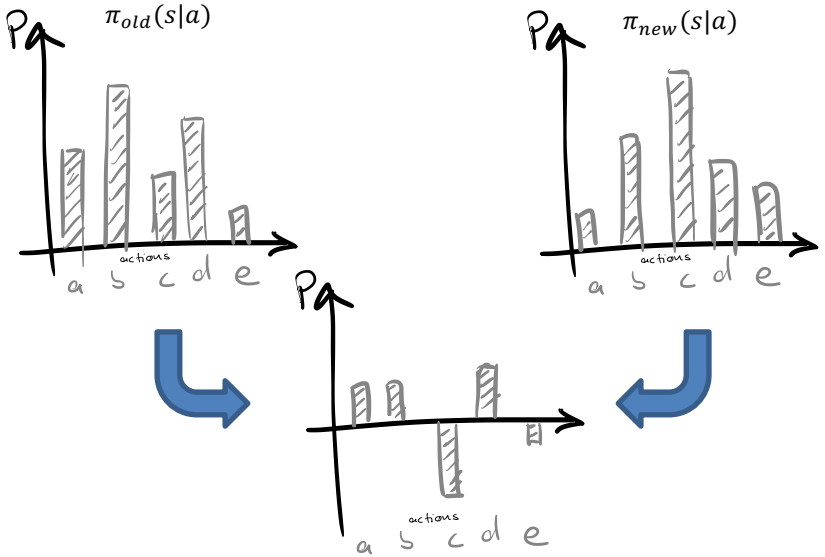
for more information see also https://www.youtube.com/watch?v=ErfnhcEV1O8 and https://medium.com/activating-robotic-minds/demystifying-kl-divergence-7ebe4317ee68

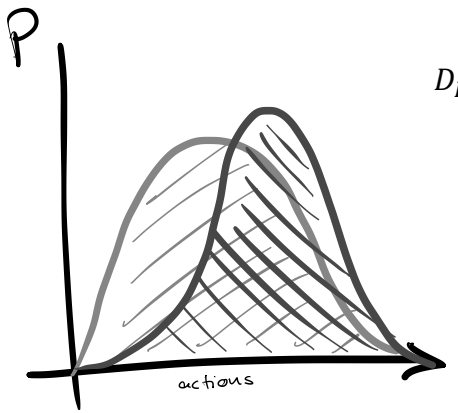
→ back to where we came from…

Trust-Region Policy Optimization (TRPO)

- If we want to optimize $L(\theta_{new})$ instead of $\eta(\theta_{new})$ …
  with a guarantee of monotonic improvement on $\eta(\theta_{new})$, …
  … we need a bound on $L(\theta_{new})$.
- It can be proven that the following bound holds[1,2]:

$$\eta(\pi_{new}) \geq L(\pi_{new}) - C \cdot D_{KL}^{max}(\pi_{old}, \pi_{new}), \text{ where } C = \frac{4\varepsilon\gamma}{(1-\gamma)^2}$$





$$D_{KL}(\pi_{old} \| \pi_{new}) = \pi(a) \log \frac{\pi_{old}(a)}{\pi_{new}(a)}$$

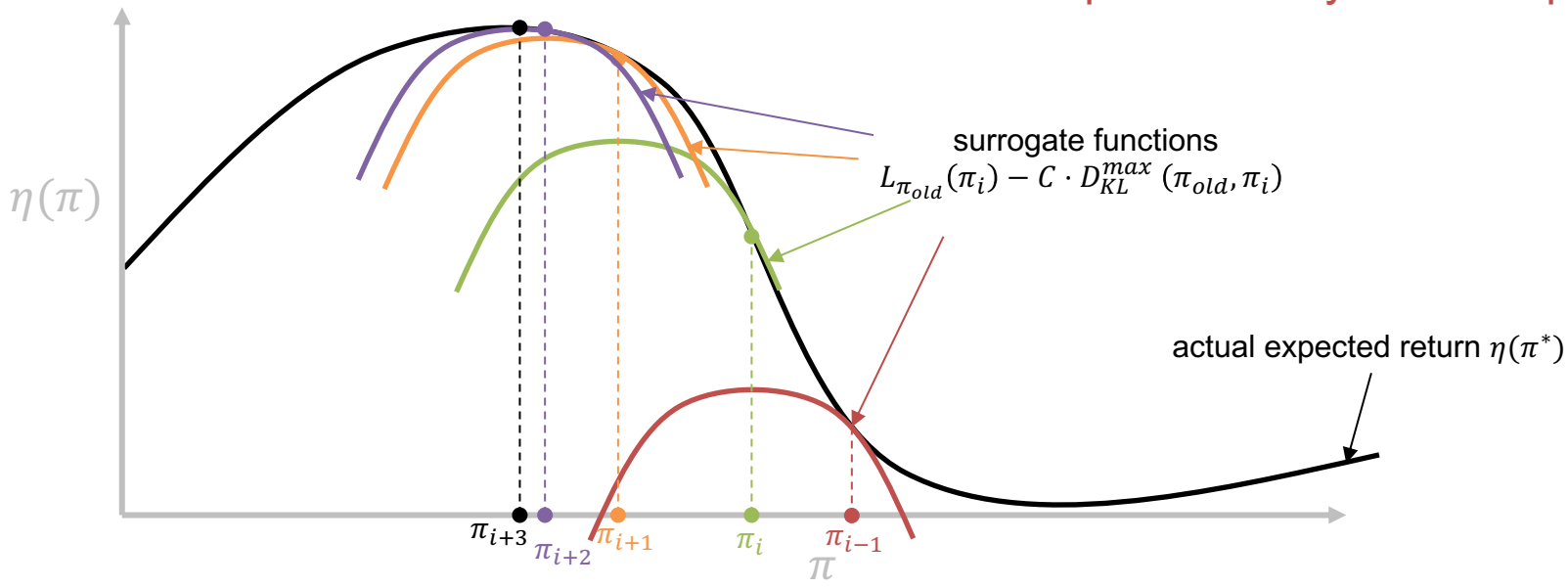
[1] *Schulman et al.: Trust-Region Policy Optimization. ICML 2015.*
[2] *Kakade et al.: Approximately Optimal Approximate Reinforcement Learning. ICML 2002.*

Trust-Region Policy Optimization (TRPO)

- A monotonically increasing policy can be defined by (minorization-maximization algorithm):

$$\pi = \arg\max_{\pi}[L(\pi_{new}) - C \cdot D_{KL}^{max}(\pi_{old}, \pi_{new})], \text{ where } C = \frac{4\varepsilon\gamma}{(1-\gamma)^2}$$

Hard to tune
→ in practice only small steps



Trust-Region Policy Optimization (TRPO)

- A monotonically increasing policy can be defined by (minorization-maximization algorithm):

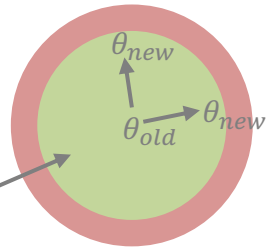
$$\pi = \arg\max_{\pi}[L(\pi_{new}) - C \cdot D_{KL}^{max}(\pi_{old}, \pi_{new})], \text{ where } C = \frac{4\varepsilon\gamma}{(1-\gamma)^2}$$

Side-note:

- A constraint on the KL-divergence between new and old policy (i.e., a trust region constraint) allows larger step sizes while being mathematically equivalent:

$$\pi = \arg\max_{\pi} L_{\pi_{old}}, \text{ such that } D_{KL}^{max}(\pi_{old}, \pi) \leq \delta$$

- Approximation with L is accurate within δ
  → here, monotonic improvement guaranteed



Trust region
→ $\pi_{\theta_{new}}(s, a)$ does not change too much

Trust-Region Policy Optimization (TRPO)

- Solving the KL-penalized problem (btw: directly over θ not π_θ)

$$\arg\max_\theta \left[L\left(\tilde{\theta}\right) - C \cdot D_{KL}^{max}(\theta, \theta_{old}) \right]$$

- Use mean KL-divergence instead of max:

$$\arg\max_\theta \left[L\left(\tilde{\theta}\right) - C \cdot \overline{D_{KL}}(\theta, \theta_{old}) \right]$$

linear approximation of L — quadratic approximation of KL

$$\arg\max_\theta \left[\frac{\partial}{\partial\theta} L(\theta)\Big|_{\theta=\theta_{old}} \cdot (\theta - \theta_{old}) - \frac{C}{2}(\theta - \theta_{old})^T \cdot \frac{\partial^2}{\partial^2\theta} \overline{D_{KL}}(\theta, \theta_{old})\Big|_{\theta=\theta_{old}} \cdot (\theta - \theta_{old}) \right]$$

$$\arg\max_\theta \left[g \cdot (\theta - \theta_{old}) - \frac{C}{2}(\theta - \theta_{old})^T F (\theta - \theta_{old}) \right]$$

$$g = \frac{\partial}{\partial\theta} L(\theta)\Big|_{\theta=\theta_{old}} \qquad F = \frac{\partial^2}{\partial^2\theta} \overline{D_{KL}}(\theta, \theta_{old})\Big|_{\theta=\theta_{old}}$$

Trust-Region Policy Optimization (TRPO)

- Derive with respect to θ and set to 0:

$$
\begin{aligned}
0 &= \frac{\partial}{\partial\theta}\left[g \cdot (\theta - \theta_{old}) - \frac{c}{2}(\theta - \theta_{old})^T F (\theta - \theta_{old})\right] \\
0 &= 1 \cdot \left[g - \frac{c}{2}(\theta - \theta_{old})^T F\right] + (\theta - \theta_{old})\left(-\frac{c}{2}F\right) \\
\mathrm{g} - \frac{c}{2}(\theta - \theta_{old})^T F &= \frac{c}{2}(\theta - \theta_{old})F \\
F^{-1}g - \frac{c}{2}(\theta - \theta_{old})^T &= \frac{c}{2}(\theta - \theta_{old}) \\
F^{-1}g &= c(\theta - \theta_{old}) \\
\frac{1}{c}F^{-1}g &= \theta - \theta_{old}
\end{aligned}
$$

- Solution: iterative optimization with Newton‘s method:

$$\theta_{new} = \theta_{old} + \frac{1}{c}H^{-1}g$$

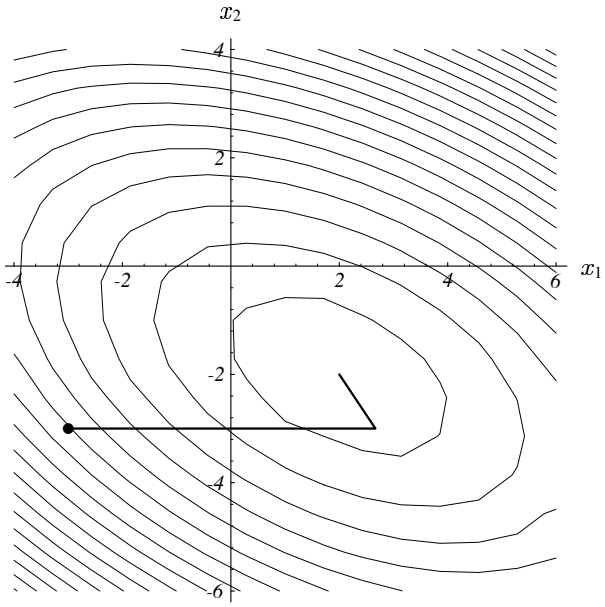
Trust-Region Policy Optimization (TRPO)

- The update step

$$\theta_{new} = \theta_{old} + \frac{1}{c} H^{-1} g$$

involves computing the inverse of the Hessian
→ too expensive for large $|\theta|$

- Conjugate Gradient (CG) computes $F^{-1}g$ approximately without forming F explicitely
  - CG solves $x = A^{-1}b$ without explicitely forming A
  - After k iterations, CG has minimized $\frac{1}{2}x^T Ax - bx$



Jonathan Richard Shewchuk: An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. Edition 1 ¼. 1994.

Trust-Region Policy Optimization (TRPO)

- Unconstrained problem: $\arg\max_\theta L(\theta) - C \cdot \overline{D_{KL}}(\theta_{old}, \theta)$
- Constrained problem: $\arg\max_\theta L(\theta)$ subject to $C \cdot \overline{D_{KL}}(\theta, \theta_{old}) \leq \delta$
  - δ is a hyper-parameter, remains fixed over the whole learning process

- Solve constrained quadratic problem: compute $F^{-1}g$ and then rescale step to get correct KL:
  1. $\max_\theta g \cdot (\theta - \theta_{old})$ subject to $\frac{1}{2}(\theta - \theta_{old})^T F)(\theta - \theta_{old}) \leq \delta$
  2. Lagrangian: $\mathcal{L}(\theta, \lambda) = g \cdot (\theta - \theta_{old}) - \frac{c}{2}[(\theta - \theta_{old})^T F(\theta - \theta_{old}) - \delta]$
  3. Differentiate with respect to θ and get $\theta - \theta_{old} = \frac{1}{c}F^{-1}g$
  4. We want $\frac{1}{2}s^T Fs = \delta$
  5. Given candidate step $s_{unscaled}$ rescale $s = \sqrt{\frac{2\delta}{s_{unscaled} \cdot F \cdot s_{unscaled}}} \cdot s_{unscaled}$

Trust-Region Policy Optimization (TRPO)

Algorithm 1 Trust Region Policy Optimization

Input: initial policy parameters θ_0, initial value function parameters ϕ_0

Hyperparameters: KL-divergence limit δ, backtracking coefficient α, maximum number of backtracking steps K

for $k = 0, 1, 2, \ldots$ **do**

Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

Compute rewards-to-go $\hat{R}_t$.

Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.

Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta_k} \hat{A}_t.$$

Use the conjugate gradient algorithm to compute

$$\hat{x}_k \approx \hat{H}_k^{-1} \hat{g}_k,$$

where $\hat{H}_k$ is the Hessian of the sample average KL-divergence.

Update the policy by backtracking line search with

$$\theta_{k+1} = \theta_k + \alpha^j \sqrt{\frac{2\delta}{\hat{x}_k^T \hat{H}_k \hat{x}_k}} \hat{x}_k,$$

where $j \in \{0, 1, 2, \ldots K\}$ is the smallest value which improves the sample loss and satisfies the sample KL-divergence constraint.

Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

end for

You have seen this very often so far

This is an approximation to the *natural policy gradient (2nd order)*

You have seen this very often so far

Trust-Region Policy Optimization (TRPO)

Intuition/Summary

- We approximate the expected return function locally around the current policy.
  - The accuracy decreases when the new policy and the current policy diverge from each other.

- But we can establish an upper bound for the error.
  - Therefore, we can guarantee a policy improvement if we optimize the local approximation within a trusted region.
  - Outside this region, the bet is off.

- Even it may have a better-calculated value, its range of error fails the improvement guarantee. With such a guarantee inside the trust region, we can locate the optimal policy iteratively. So even it takes a while to prove it mathematically, the reasoning is simple.

Trust-Region Policy Optimization (TRPO)

- Shortcomings:
  - TRPO minimizes a quadratic equation to approximate the inverse of the Fisher Information Matrix (FIM), i.e., the Hessian
  - To do this for every policy is expensive
  - It requires a large batch of rollouts to approximate it correctly
  - Less sample-efficient to other PG methods when trained with first-order methods such as Adam

→ These become a significant issue for large/deep networks!

- **And: Yes, TRPO is a complex algorithm that is hard to understand & implement**

References

- Pascal Poupart: CS885 Lecture 14c: Trust Region Methods
- Sergey Levine: CS285: Advanced Policy Gradients

Further reading:

- TRPO (Trust Region Policy Optimization) : In depth Research Paper Review: https://www.youtube.com/watch?v=CKaN5PgkSBc