# Natural PG, TRPO, PPO

**A review**

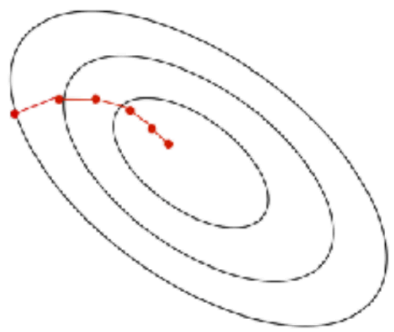
Yuning Wu, 4/2/2021

Choosing a proper stepsize in policy gradient is critical.

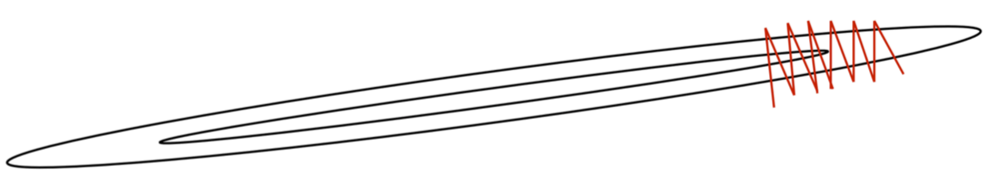
- Too big, can’t recover from bad moves.
- Too small, low sample efficiency.

## The spirit of natural policy gradient, is to take a **constrained** GD step.

- Euclidean distance in parameter space as constraint is unfavorable.
- The threshold is hard to pick.
- Perform badly in ill-conditioned curvature.



normal case



ill conditioned surface

Bounce around in high curvature direction. Make slow progress in low curvature direction.

# Natural PG

The spirit of natural policy gradient, is to take a **constrained** GD step.

- Use KL-divergence in distribution space.
- The threshold is easy to pick.
- Solves the curvature issue.

$$\theta_{new} = \theta_{old} + d*$$
$$\theta' = \theta + d*$$
$$d* = arg \max_{KL\left[\pi_\theta || \pi_{\theta+d}\right] \leq \epsilon} U(\theta + d)$$

## Two questions.

1. What is the natural policy gradient? (Direction)
2. What is the stepsize? (Scale)

$$d* = arg \max_d U(\theta + d) - \lambda \left( KL \left[ \pi_\theta || \pi_{\theta+d} \right] - \epsilon \right)$$

**Answer to 1st question, three ingredients.**

1. Turning a constrained objective into unconstrained penalized objective.
2. Taylor expansion.
3. KL-divergence Hessian / Fisher Information matrix.

$$d^* \approx arg\max_{d} U(\theta_{old}) + \nabla_\theta U(\theta)\,|_{\theta=\theta_{old}} \cdot d - \frac{1}{2}\lambda\left(d^\top \cdot \nabla^2_\theta KL\left[\pi_{\theta_{old}} \| \pi_\theta\right]\Big|_{\theta=\theta_{old}} \cdot d\right) + \lambda\epsilon$$

## Answer to 1st question, three ingredients.

1. Turning a constrained objective into unconstrained penalized objective.
2. Taylor expansion.
   - Please refer to lecture slides for thorough expansion.
   - If you have hesitation about Taylor expansion, please come to OH.
3. KL-divergence Hessian / Fisher Information matrix.

$$d^* \approx arg\max_{d} U(\theta_{old}) + \nabla_\theta U(\theta)\,|_{\theta=\theta_{old}} \cdot d - \frac{1}{2}\lambda \left( d^\top \cdot \nabla^2_\theta KL\left[\pi_{\theta_{old}} || \pi_\theta\right] \Big|_{\theta=\theta_{old}} \cdot d \right) + \lambda\epsilon$$

## Answer to 1st question, three ingredients.

1. Turning a constrained objective into unconstrained penalized objective.
2. Taylor expansion.
3. KL-divergence Hessian / Fisher Information matrix.
    - Just plug in the definition of KL-divergence.

$$\nabla^2_\theta KL\left[\pi_{\theta_{old}} \| \pi_\theta\right]\Big|_{\theta=\theta_{old}} = \mathbb{E}_{x \sim \pi_{\theta_{old}}}\left[\nabla_\theta \log \pi_\theta(x)\,|_{\theta=\theta_{old}} \cdot \nabla_\theta \log \pi_\theta(x)\,|_{\theta=\theta_{old}}^\top\right]$$

## Answer to 1st question, three ingredients.

1. Turning a constrained objective into unconstrained penalized objective.
2. Taylor expansion.
3. KL-divergence Hessian / Fisher Information matrix.
   - Just plug in the definition of KL-divergence.

$$\mathbb{E}_{x \sim \pi_{\theta_{old}}}\left[ \nabla_\theta \log \pi_\theta(x) \,|_{\theta=\theta_{old}} \cdot \nabla_\theta \log \pi_\theta(x) \,|_{\theta=\theta_{old}}^\top \right] = \boldsymbol{F}$$

## Answer to 1st question, three ingredients.

1. Turning a constrained objective into unconstrained penalized objective.
2. Taylor expansion.
3. KL-divergence Hessian / Fisher Information matrix.
    - Just plug in the definition of KL-divergence.
    - This is just the Fisher Information matrix! [post link]

$$d* \approx arg\max_d U(\theta_{old}) + \nabla_\theta U(\theta)|_{\theta=\theta_{old}} \cdot d - \frac{1}{2}\lambda\left(d^\top \cdot \boldsymbol{F} \cdot d\right) + \lambda\epsilon$$

## Answer to 1st question, three ingredients.

1. Turning a constrained objective into unconstrained penalized objective.
2. Taylor expansion.
3. KL-divergence Hessian / Fisher Information matrix.
   - Just plug in the definition of KL-divergence.
   - This is just the Fisher Information matrix! [post link]

$$d^* \approx arg\max_d U(\theta_{old}) + \nabla_\theta U(\theta)\,|_{\theta=\theta_{old}} \cdot d - \frac{1}{2}\lambda\left(d^\top \cdot \boldsymbol{F} \cdot d\right) + \lambda\epsilon$$

$$d^* \approx arg\max_d \nabla_\theta U(\theta)\,|_{\theta=\theta_{old}} \cdot d - \frac{1}{2}\lambda\left(d^\top \cdot \boldsymbol{F} \cdot d\right)$$

**Answer to 1st question, a standard optimization problem now.**

1. Take gradient.
2. Set to zero.

$$d^* = \frac{2}{\lambda} \cdot \boldsymbol{F}^{-1} \nabla_\theta U(\theta) |_{\theta=\theta_{old}}$$

$$d^* = \frac{2}{\lambda} \cdot g_N$$

## Answer to 1st question, solved.

- $g_N$ is the natural gradient we are looking for.

$$KL\left[\pi_{\theta_{old}} \| \pi_{\theta}\right] \approx \frac{1}{2}(\alpha g_N)^\top \boldsymbol{F}(\alpha g_N) \approx \epsilon$$

## Answer to 2nd question, the stepsize $\alpha$.

- Assumption: we want the KL-divergence between old and new policy to be at most $\epsilon$.
- Use second-order Taylor expansion again.

$$\alpha = \sqrt{\frac{2\epsilon}{g_N^\top \boldsymbol{F}^{-1} g_N}}$$

## Answer to 2nd question, the stepsize $\alpha$, solved.

- Assumption: we want the KL-divergence between old and new policy to be at most $\epsilon$.
- Use second-order Taylor expansion again.

**Algorithm 1** Natural Policy Gradient

Input: initial policy parameters $\theta_0$

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

Collect set of trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$

Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm

Form sample estimates for

- policy gradient $\hat{g}_k$ (using advantage estimates)
- and KL-divergence Hessian / Fisher Information Matrix $\hat{H}_k$

Compute Natural Policy Gradient update:

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{\hat{g}_k^T \hat{H}_k^{-1} \hat{g}_k}} \hat{H}_k^{-1} \hat{g}_k$$

**end for**

# TRPO

## Natural PG (NPG) is a major ingredient of TRPO.

- NPG
- Monotonic improvement theorem. [post link]
- Line search of parameter.

- Pseudocode:

**for** iteration$=1, 2, \ldots$ **do**
  Run policy for $T$ timesteps or $N$ trajectories
  Estimate advantage function at all timesteps

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

**end for**

# NPG is a major ingredient of TRPO.

- NPG
- Monotonic improvement theorem. [post link]
- Line search of parameter.

**Algorithm 3** Trust Region Policy Optimization

Input: initial policy parameters $\theta_0$
**for** $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Form sample estimates for
  - policy gradient $\hat{g}_k$ (using advantage estimates) NPG
  - and KL-divergence Hessian-vector product function $f(v) = \hat{H}_k v$

  Use CG with $n_{cg}$ iterations to obtain $x_k \approx \hat{H}_k^{-1}\hat{g}_k$
  Estimate proposed step $\Delta_k \approx \sqrt{\frac{2\epsilon}{x_k^T \hat{H}_k x_k}} x_k$
  Perform backtracking line search with exponential decay to obtain final update Line search

$$\theta_{k+1} = \theta_k + \alpha^j \Delta_k$$

**end for**

## Issues with TRPO.

- Calculation of fisher information matrix (second order) is expensive!
- Is there a cheaper first order method that could achieve similar performance?

# PPO

## Two types of PPO.

- Adaptive KL-penalty
- Clipped objective

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t \left[ \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)} \hat{A}_t - \beta KL \left[ \pi_{\theta_{old}} \| \pi_\theta \right] \right]$$

Objective

$$d = \hat{\mathbb{E}}_t \left[ KL \left[ \pi_{\theta_{old}} \| \pi_\theta \right] \right]$$

- If $d < d_{target}/1.5, \beta \leftarrow \beta/2$
- If $d < d_{target} \times 1.5, \beta \leftarrow \beta \times 2$

Need to update $\beta$ to enforce KL constraint

## Two types of PPO.

- Adaptive KL-penalty
- Clipped objective

$$\boxed{L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[ \min \left( \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)} \hat{A}_t, clip \left( \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t \right) \right]}$$

Objective

## The clipping version vs. the KL penalty version.

- Clipping version is easier to implement.
- Generally speaking, clipping is working almost as well as KL penalty version in practice.
- Strongly recommend reading the original PPO [paper].