Reinforcement Learning

Chapter 5: RL via Policy Gradient

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

University of Toronto

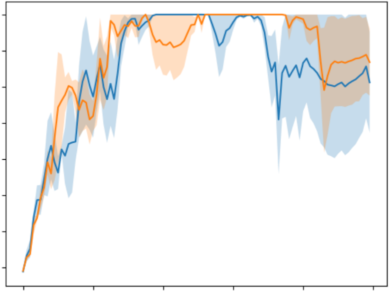
Fall 2025

Observing Basic PGM

Let's break the problem down: *even by using baseline, we still observe instability while we use PGM*

If we plot the average reward we collect through time

- *We might see it getting improved up to some point*
- *It then could drop drastically at some other point*



Main Reason: *Estimate Variance*

The main reason *for this behavior is high variance of the gradient estimator: recall that we estimate the gradient of average value by*

$$\hat{\nabla} \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right) = \sum_{t=0}^{T-1} U_t \nabla \log \pi_{\boldsymbol{\theta}}\left(A_t | S_t\right)$$

This is true that

$$\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left\{\hat{\nabla} \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right)\right\} = \nabla \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right)$$

However, its variance, i.e.,

$$\mathbb{V}\mathrm{ar}\left\{\hat{\nabla} \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right)\right\} = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left\{\left(\hat{\nabla} \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right) - \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right)\right)^2\right\}$$

could be very large: one given sample take us far away from the true direction!

Reducing Variance: *Using Mini-Batches*

+ *But, isn't that always the case in SGD?! We assume that those errors cancel each other out! Right?!*
– That's right! *But apparently, it's not working here!*

To find out an explanation to this behavior, let's try to reduce the variance by using larger mini-batches

- *Collect B sample trajectories by policy $\pi_{\boldsymbol{\theta}}$*
- *Compute the gradient estimator for each trajectory*

$$\hat{\nabla}_b \mathcal{J}(\pi_{\boldsymbol{\theta}}) = \sum_{t=0}^{T-1} U_t \nabla \log \pi_{\boldsymbol{\theta}}(A_t | S_t)$$

- *Average the estimators to get a better estimator*

$$\hat{\nabla} \mathcal{J}(\pi_{\boldsymbol{\theta}}) = \frac{1}{B} \sum_{b=1}^{B} \hat{\nabla}_b \mathcal{J}(\pi_{\boldsymbol{\theta}})$$

Advantage PGM: *Mini-Batch Version*

```
AdvantagePGM():
```

Initiate with $\boldsymbol{\theta}$ and learning rate α

while *interacting* ***do***

  for *mini-batch* $b = 1 : B$ ***do***

    Sample the environment with policy $\pi_{\boldsymbol{\theta}}$

$$S_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T$$

    for $t = 0 : T - 1$ ***do***

      Compute sample advantage $U_t = R_{t+1} + \gamma v_{\pi_{\boldsymbol{\theta}}}(S_{t+1}) - v_{\pi_{\boldsymbol{\theta}}}(S_t)$

    end for

    Compute sample gradient $\hat{\nabla}_b = \sum_{t=0}^{T-1} U_t \nabla \log \pi_{\boldsymbol{\theta}}(A_t | S_t)$

  end for

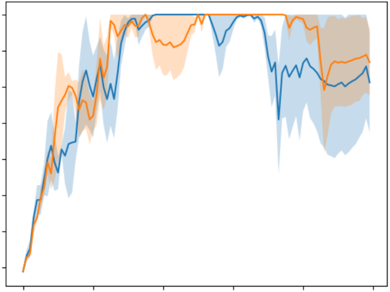
  Update policy network

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \frac{\alpha}{B} \sum_{b=1}^{B} \hat{\nabla}_b$$

end while

Observing Mini-Batch

After trying mini-batch PGM: *we see that the variance of the curve slightly improves; however, we still see that problem*



- + *What does this say then?*
- – *It says that the problem is simply coming from high variance*

Alternative Look at *Advantage Optimization*

In the latest version of PGM: *we saw that the gradient of the average value can be computed as*

$$\nabla \mathcal{J}(\pi_{\boldsymbol{\theta}}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left\{ u_{\pi_{\boldsymbol{\theta}}}(S, A) \nabla \log \pi_{\boldsymbol{\theta}}(A|S) \right\}$$

Let's assume that we can compute it exactly: then, we will muse gradient descent to find optimal $\boldsymbol{\theta}$, i.e.,

```
AdvantageGD():
```

1: *Start with some initial $\boldsymbol{\theta}_0$*
2: **for** $k = 1 : K$ **do**
3: *Compute the exact gradient*

$$\nabla \mathcal{J}(\pi_{\boldsymbol{\theta}_k}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}_k}} \left\{ u_{\pi_{\boldsymbol{\theta}_k}}(S, A) \nabla \log \pi_{\boldsymbol{\theta}}(A|S) |_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} \right\}$$

4: *Update the parameters as $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \nabla \mathcal{J}(\pi_{\boldsymbol{\theta}_k})$*
5: ***end for***

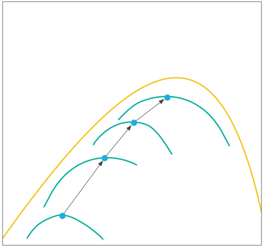
Alternative Look: *Surrogate Function*

Let us now define the following surrogate function at point $\boldsymbol{\theta}_k$

$$\mathcal{L}_k\left(\pi_{\boldsymbol{\theta}}\right)=\mathbb{E}_{\pi_{\boldsymbol{\theta}_k}}\left\{u_{\pi_{\boldsymbol{\theta}_k}}\left(S, A\right)\frac{\pi_{\boldsymbol{\theta}}\left(A|S\right)}{\pi_{\boldsymbol{\theta}_k}\left(A|S\right)}\right\}$$

We should pay attention that

1. *We have a sequence of surrogate functions*
   - ↳ *Each function is defined locally at point $\boldsymbol{\theta}_k$*
2. $\pi_{\boldsymbol{\theta}}\left(A|S\right)$ *is the only term that belongs to $\boldsymbol{\theta}$*
   - ↳ *Everything else depends on $\boldsymbol{\theta}_k$ which is a constant*



Alternative Look: *Surrogate Function*

$$\boxed{\mathcal{L}_k\left(\pi_{\boldsymbol{\theta}}\right) = \mathbb{E}_{\pi_{\boldsymbol{\theta}_k}}\left\{u_{\pi_{\boldsymbol{\theta}_k}}\left(S, A\right)\frac{\pi_{\boldsymbol{\theta}}\left(A|S\right)}{\pi_{\boldsymbol{\theta}_k}\left(A|S\right)}\right\}}$$

Let's compute the gradient of this surrogate function at $\boldsymbol{\theta} = \boldsymbol{\theta}_k$

$$\begin{aligned}\nabla\mathcal{L}_k\left(\pi_{\boldsymbol{\theta}_k}\right) &= \mathbb{E}_{\pi_{\boldsymbol{\theta}_k}}\left\{u_{\pi_{\boldsymbol{\theta}_k}}\left(S, A\right)\frac{\nabla\pi_{\boldsymbol{\theta}}\left(A|S\right)}{\pi_{\boldsymbol{\theta}_k}\left(A|S\right)}\right\} \\ &= \mathbb{E}_{\pi_{\boldsymbol{\theta}_k}}\left\{u_{\pi_{\boldsymbol{\theta}_k}}\left(S, A\right)\nabla\log\pi_{\boldsymbol{\theta}}\left(A|S\right)|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}\right\} \\ &= \nabla\mathcal{J}\left(\pi_{\boldsymbol{\theta}_k}\right)\end{aligned}$$

This is exactly the gradient that we update our policy with!

Alternative Look: *GD with Surrogate Function*

So, we could re-write the gradient descent loop as

```
AdvantageGD():
```
1: *Start with some initial* $\boldsymbol{\theta}_0$
2: **for** $k = 1 : K$ **do**
3: *Compute the exact gradient* $\nabla \mathcal{L}_k \left(\pi_{\boldsymbol{\theta}_k}\right)$
4: *Update the parameters as* $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \nabla \mathcal{L}_k \left(\pi_{\boldsymbol{\theta}_k}\right)$
5: ***end for***

+ *Now, what's the point?! It's still same problem!*
– Sure! But, let's see what we are doing now

In each iteration we add gradient scaled with α *to the previous parameters*

– *Why we do that?*
+ *We want to increase* $\mathcal{L}_k \left(\pi_{\boldsymbol{\theta}}\right)$ *maximally*
– *Exactly! So, why don't we simply replace* $\boldsymbol{\theta}_{k+1}$ *with its maximizer?!*

Alternative Look: *GD with Surrogate Function*

We re-write the gradient descent loop as follows

```
AdvantageGD():
```
1: *Start with some initial* $\boldsymbol{\theta}_0$
2: **for** $k = 1 : K$ **do**
3: *Compute the surrogate function* $\mathcal{L}_k(\pi_{\boldsymbol{\theta}})$
4: *Update the parameters as* $\boldsymbol{\theta}_{k+1} = \operatorname{argmax}_{\boldsymbol{\theta}} \mathcal{L}_k(\pi_{\boldsymbol{\theta}})$
5: ***end for***

This algorithm algorithm does the learning rate tuning by itself

It gets us rid of specifying the learning rate α

+ *Nice job! But, how are we su[supposed] to find the surrogate function?*
– *You could guess! By sampling!*

Understanding Surrogate Function

Let's get back to the definition of the surrogate function:*it is easy to interprete it as importance sampling*

$$\mathcal{L}_k\left(\pi_{\boldsymbol{\theta}}\right) = \mathbb{E}_{\pi_{\boldsymbol{\theta}_k}}\left\{u_{\pi_{\boldsymbol{\theta}_k}}\left(S, A\right)\frac{\pi_{\boldsymbol{\theta}}\left(A|S\right)}{\pi_{\boldsymbol{\theta}_k}\left(A|S\right)}\right\}$$

We sample trajectories by policy $\pi_{\boldsymbol{\theta}_k}$

- *We collect advantage samples* $u_{\pi_{\boldsymbol{\theta}_k}}\left(S, A\right)$
- *We do know the policy samples* $\pi_{\boldsymbol{\theta}_k}\left(A|S\right)$

But we compute the average with respect to $\pi_{\boldsymbol{\theta}}$

This gives us a very good explanation of why PGM is unstable

Review: *Importance Sampling Trade-off*

Consider the basic setting in importance sampling:

we have samples from $X \sim p(x)$ *but we want to estimate* $X \sim q(x)$

If we could sample from $q(x)$

$$\mu = \mathbb{E}_q \{X\}$$

The variance of the estimate is

$$\begin{aligned} \mathbb{Var}\{X\} &= \mathbb{E}_q \left\{ (X - \mu)^2 \right\} \\ &= \mathbb{E}_q \left\{ X^2 \right\} - \mu^2 = \sigma^2 \end{aligned}$$

Review: *Importance Sampling Trade-off*

Consider the basic setting in importance sampling:

we have samples from $X \sim p(x)$ *but we want to estimate* $X \sim q(x)$

Now that we sample $p(x)$

$$\bar{\mu} = \mathbb{E}_p \left\{ X \frac{q(X)}{p(X)} \right\} = \mathbb{E}_q \{X\} = \mu$$

The variance of this estimate is

$$\mathbb{Var}\{X\} = \mathbb{E}_p \left\{ \left(X \frac{q(X)}{p(X)} \right)^2 \right\} - \mu^2 = \int_x x^2 \frac{q^2(x)}{p^2(x)} p(x) - \mu^2$$

$$= \int_x x^2 \frac{q(x)}{p(x)} q(x) - \mu^2 = \mathbb{E}_q \left\{ X^2 \frac{q(X)}{p(X)} \right\} - \mu^2 \neq \sigma^2$$

Review: *Importance Sampling Trade-off*

Consider the basic setting in importance sampling:

we have samples from $X \sim p(x)$ but we want to estimate $X \sim q(x)$

If we could sample from $q(x)$

$$\mu = \mathbb{E}_q\{X\}$$

and the estimate variance is

$$\sigma^2 = \mathbb{E}_q\{X^2\} - \mu^2$$

Now that we sample $p(x)$

$$\mu = \mathbb{E}_q\{X\}$$

and the estimate variance is

$$\bar{\sigma}^2 = \mathbb{E}_q\left\{X^2 \frac{q(X)}{p(X)}\right\} - \mu^2$$

With importance sampling, variance scales with ratio of the distributions

Back to Surrogate: *Root of High Variance*

Gradient descent based on surrogate functions optimizes

$$\mathcal{L}_k\left(\pi_{\boldsymbol{\theta}}\right) = \mathbb{E}_{\pi_{\boldsymbol{\theta}_k}}\left\{ u_{\pi_{\boldsymbol{\theta}_k}}\left(S, A\right) \frac{\pi_{\boldsymbol{\theta}}\left(A|S\right)}{\pi_{\boldsymbol{\theta}_k}\left(A|S\right)} \right\}$$

We can assume that by sampling the environment, we are estimating this surrogate function by importance sampling from samples of policy $\pi_{\boldsymbol{\theta}_k}$:let $\hat{\mathcal{L}}_k\left(\pi_{\boldsymbol{\theta}}\right)$ be our estimate; then we could say

$$\mathbb{V}\mathrm{ar}\left\{ \hat{\mathcal{L}}_k\left(\pi_{\boldsymbol{\theta}}\right) \right\} \propto \frac{\pi_{\boldsymbol{\theta}}\left(A|S\right)}{\pi_{\boldsymbol{\theta}_k}\left(A|S\right)}$$

- *This explains why we see unreliable estimates in PGM!*
+ *How exactly?!*
- *Let's break it down!*

Back to Surrogate: *Root of High Variance*

$$\mathbb{Var}\left\{\hat{\mathcal{L}}_k\left(\pi_{\boldsymbol{\theta}}\right)\right\} \propto \frac{\pi_{\boldsymbol{\theta}}\left(A|S\right)}{\pi_{\boldsymbol{\theta}_k}\left(A|S\right)}$$

Looking at this variance, we could say: we should not naively update $\boldsymbol{\theta}_k$ by maximizing its surrogate function. We should update it such that

1. *surrogate function is maximized, and*
2. *the policy specified by $\pi_{\boldsymbol{\theta}_{k+1}}$ is rather close to $\pi_{\boldsymbol{\theta}_k}$*

\+ *But aren't we doing that?! We just change the policy parameters slightly, i.e., $\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2$ is typically small*

– *Well! That doesn't say anything about difference between $\pi_{\boldsymbol{\theta}_{k+1}}$ and $\pi_{\boldsymbol{\theta}_k}$*

Observation: *Sensitivity of Policy Network*

Sensitivity of Policy

Even if we change parameters $\boldsymbol{\theta}$ slightly, $\pi_{\boldsymbol{\theta}}$ can change hugely!

Given this observation, we could modify our gradient descent loop as

```
AdvantageGD():
```

1: Start with some initial $\boldsymbol{\theta}_0$

2: **for** $k = 1 : K$ **do**

3: Compute the surrogate function $\mathcal{L}_k(\pi_{\boldsymbol{\theta}})$

4: Update the parameters as

$$\boldsymbol{\theta}_{k+1} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \mathcal{L}_k(\pi_{\boldsymbol{\theta}}) \quad \text{subject to} \quad \pi_{\boldsymbol{\theta}} \text{ and } \pi_{\boldsymbol{\theta}_k} \text{ are close}$$

5: **end for**

+ *How can we quantify "$\pi_{\boldsymbol{\theta}}$ and $\pi_{\boldsymbol{\theta}_k}$ being close"?*

Review: *Kullback-Leibler Divergence*

KL Divergence

Kullback-Leibler divergence between two distributions p and q is defined as

$$D_{\mathrm{KL}}\left(p \| q\right) = \mathbb{E}_p \left\{ \log \left(\frac{p\left(X\right)}{q\left(X\right)} \right) \right\} = \int_x \log \left(\frac{p\left(x\right)}{q\left(x\right)} \right) p\left(x\right)$$

We can use this definition to find the divergence between $\pi_{\boldsymbol{\theta}}$ and $\pi_{\boldsymbol{\theta}_k}$

$$\bar{D}_{\mathrm{KL}}\left(\pi_{\boldsymbol{\theta}} \| \pi_{\boldsymbol{\theta}_K}\right) = \mathbb{E}_{S \sim d_{\pi_{\boldsymbol{\theta}_k}}} \left\{ \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left\{ \log \left(\frac{\pi_{\boldsymbol{\theta}}\left(A | S\right)}{\pi_{\boldsymbol{\theta}_k}\left(A | S\right)} \right) \right\} \right\}$$

Review: *Properties of KL Divergence*

KL divergence shows interesting properties

- *It is zero when distributions are the same*

$$D_{\mathrm{KL}}\left(p\|p\right) = 0$$

and increases when they get more different

- *It is always non-negative, i.e., for any p and q*

$$D_{\mathrm{KL}}\left(p\|q\right) \geqslant 0$$

↳ *This property is often called Gibbs' inequality*

But remember that KL divergence is asymmetric, i.e.,

$$D_{\mathrm{KL}}\left(p\|q\right) \neq D_{\mathrm{KL}}\left(q\|p\right)$$

Trust Region Policy Gradient Method

Back to our modified gradient descent loop, we could write

`AdvantageGD():`

1: *Start with some initial* $\boldsymbol{\theta}_0$

2: **for** $k = 1 : K$ **do**

3: *Compute the surrogate function* $\mathcal{L}_k\left(\pi_{\boldsymbol{\theta}}\right)$

4: *Update the parameters as*

$$\boldsymbol{\theta}_{k+1} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathcal{L}_k\left(\pi_{\boldsymbol{\theta}}\right) \quad \text{subject to} \quad \bar{D}_{\mathrm{KL}}\left(\pi_{\boldsymbol{\theta}} \| \pi_{\boldsymbol{\theta}_k}\right) \leqslant d_{\max}$$

5: ***end for***

This modified approach is called

Trust Region PGM

since it computes the best policy gradient within a trusted resion

Surrogate Optimization: *Exact Solution*

+ *How can we solve the optimization in the loop then?*
– *As you could guess, we are going to find a way around it!*

The concrete way to solve it is to use regularization: we want to solve

$$\boldsymbol{\theta}_{k+1} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathcal{L}_k\left(\pi_{\boldsymbol{\theta}}\right) \quad \textit{subject to} \quad \bar{D}_{\mathrm{KL}}\left(\pi_{\boldsymbol{\theta}} \| \pi_{\boldsymbol{\theta}_k}\right) \leqslant d_{\max}$$

We solve instead

$$\boldsymbol{\theta}_{k+1} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathcal{L}_k\left(\pi_{\boldsymbol{\theta}}\right) - \beta\Big(\bar{D}_{\mathrm{KL}}\left(\pi_{\boldsymbol{\theta}} \| \pi_{\boldsymbol{\theta}_k}\right) - d_{\max}\Big)$$

for some β that potentially minimizes the regularized objective

This is going to be computationally very expensive!

Surrogate Optimization: *Natural Policy Gradient*

We instead use Taylor expansion to approximate both surrogate and constraint

Taylor Expansion

An analytic function $f(x)$ can be expanded around point x_0 as

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(x_0)(x - x_0)^2 + \cdots$$

Let's start with the surrogate function

$$\mathcal{L}_k(\pi_{\boldsymbol{\theta}}) = \mathcal{L}_k(\pi_{\boldsymbol{\theta}_k}) + \nabla \mathcal{L}_k(\pi_{\boldsymbol{\theta}_k})^{\mathsf{T}} (\boldsymbol{\theta} - \boldsymbol{\theta}_k) + \varepsilon$$

In Assignment 3, we show $\mathcal{L}_k(\pi_{\boldsymbol{\theta}_k}) = 0$: so, setting$\nabla_k = \nabla \mathcal{L}_k(\pi_{\boldsymbol{\theta}_k})$ we get

$$\mathcal{L}_k(\pi_{\boldsymbol{\theta}}) \approx \nabla_k^{\mathsf{T}} (\boldsymbol{\theta} - \boldsymbol{\theta}_k)$$

Surrogate Optimization: *Natural Policy Gradient*

Next, we go for constraint term

$$\bar{D}_{\mathrm{KL}}\left(\pi_{\boldsymbol{\theta}}\|\pi_{\boldsymbol{\theta}_k}\right) = \bar{D}_{\mathrm{KL}}\left(\pi_{\boldsymbol{\theta}_k}\|\pi_{\boldsymbol{\theta}_k}\right) + \nabla \bar{D}_{\mathrm{KL}}\left(\pi_{\boldsymbol{\theta}}\|\pi_{\boldsymbol{\theta}_k}\right)|_{\boldsymbol{\theta}_k}^{\mathsf{T}}\left(\boldsymbol{\theta}-\boldsymbol{\theta}_k\right) + \frac{1}{2}\left(\boldsymbol{\theta}-\boldsymbol{\theta}_k\right)^{\mathsf{T}}\nabla^2 \bar{D}_{\mathrm{KL}}\left(\pi_{\boldsymbol{\theta}}\|\pi_{\boldsymbol{\theta}_k}\right)|_{\boldsymbol{\theta}_k}\left(\boldsymbol{\theta}-\boldsymbol{\theta}_k\right) + \varepsilon$$

In Assignment 3, we show that

$$\bar{D}_{\mathrm{KL}}\left(\pi_{\boldsymbol{\theta}_k}\|\pi_{\boldsymbol{\theta}_k}\right) = 0$$

$$\nabla \bar{D}_{\mathrm{KL}}\left(\pi_{\boldsymbol{\theta}}\|\pi_{\boldsymbol{\theta}_k}\right)|_{\boldsymbol{\theta}_k} = \mathbf{0}$$

So, by defining $\mathbf{H}_k = \nabla^2 \bar{D}_{\mathrm{KL}}\left(\pi_{\boldsymbol{\theta}}\|\pi_{\boldsymbol{\theta}_k}\right)|_{\boldsymbol{\theta}_k}$ *we have*

$$\bar{D}_{\mathrm{KL}}\left(\pi_{\boldsymbol{\theta}}\|\pi_{\boldsymbol{\theta}_k}\right) \approx \frac{1}{2}\left(\boldsymbol{\theta}-\boldsymbol{\theta}_k\right)^{\mathsf{T}}\mathbf{H}_k\left(\boldsymbol{\theta}-\boldsymbol{\theta}_k\right)$$

Surrogate Optimization: *Natural Policy Gradient*

Now, let's replace these approximations

$$\mathcal{L}_k\left(\pi_{\boldsymbol{\theta}}\right) \approx \nabla_k^{\mathsf{T}}\left(\boldsymbol{\theta}-\boldsymbol{\theta}_k\right)$$
$$\bar{D}_{\mathrm{KL}}\left(\pi_{\boldsymbol{\theta}} \| \pi_{\boldsymbol{\theta}_k}\right) \approx \frac{1}{2}\left(\boldsymbol{\theta}-\boldsymbol{\theta}_k\right)^{\mathsf{T}} \mathbf{H}_k\left(\boldsymbol{\theta}-\boldsymbol{\theta}_k\right)$$

into the optimization problem

$$\boldsymbol{\theta}_{k+1}=\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \nabla_k^{\mathsf{T}}\left(\boldsymbol{\theta}-\boldsymbol{\theta}_k\right) \quad \textit{subject to} \quad \frac{1}{2}\left(\boldsymbol{\theta}-\boldsymbol{\theta}_k\right)^{\mathsf{T}} \mathbf{H}_k\left(\boldsymbol{\theta}-\boldsymbol{\theta}_k\right) \leqslant d_{\max }$$

This is a classic linear programming whose solution is given by

$$\boldsymbol{\theta}_{k+1}=\boldsymbol{\theta}_k+\sqrt{\frac{2 d_{\max }}{\nabla_k^{\mathsf{T}} \mathbf{H}_k^{-1} \nabla_k}} \mathbf{H}_k^{-1} \nabla_k$$

Surrogate Optimization: *Natural Policy Gradient*

This is like classic update with linear correction and tuned learning rate

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \sqrt{\frac{2d_{\max}}{\nabla_k^\mathsf{T} \mathbf{H}_k^{-1} \nabla_k}} \mathbf{H}_k^{-1} \nabla_k$$

This is often referred to as

natural policy gradient

It gives a better direction for update; however,

- *It could still not increase surrogate or deviate constraint*
  - ↳ *This is due to the inaccuracy of approximations*
- *It requires estimate of $\mathbf{H}_k$ which is computationally expensive*
- *It also needs to invert estimate of $\mathbf{H}_k$ which is again expensive*

Natural PGM

`NaturalPGM():`

1: *Start with some initial* $\boldsymbol{\theta}$
2: ***while*** *interacting* ***do***
3: ***for*** *mini-batch* $b = 1 : B$ ***do***
4: *Sample the environment with policy* $\pi_{\boldsymbol{\theta}}$

$$S_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T$$

5: ***end for***
6: *Estimate* $\hat{\nabla}$ *and* $\hat{\mathbf{H}}$ *from samples*
7: *Update the parameters as*

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \sqrt{\frac{2d_{\max}}{\hat{\nabla}^{\mathsf{T}} \hat{\mathbf{H}} \hat{\nabla}}} \hat{\mathbf{H}}^{-1} \hat{\nabla}$$

8: ***end while***

TRPO and PPO Algorithms

There are two sets of solutions to overcome the challenges in natural PGM

- *Trust Region Policy Optimization*
  - ↳ *Regularize learning rate via backtracking line*
    - ↳ *This guarantees the satisfaction of constraint*
  - ↳ *Use sampling to find the estimate $\hat{\mathbf{H}}$*
  - ↳ *Use gradient conjugate to compute the inverse*
- *Proximal Policy Optimization*
  - ↳ *Skip all these steps by computationally-efficient clipping*
    - ↳ *The clipping guarantees the satisfaction of constraint*
  - ↳ *We do not need to find estimate $\hat{\mathbf{H}}$ anymore*