Deep Reinforcement Learning

Overview of main articles
Part 2. Policy gradient algorithms

Sergey Ivanov

February 24, 2019

MSU

Table of contents i

Basic policy gradient methods

Direct optimization

Recall RL goal:

$$\mathbb{E}_{\pi(\theta)}\mathbb{E}_{\mathcal{T}} R \to \max_{\theta}$$

Direct optimization

Recall RL goal:

$$\mathbb{E}_{\pi(\theta)}\mathbb{E}_{\mathcal{T}} R \to \max_{\theta}$$

Let's optimize our goal directly!

$$\nabla_{\theta}\mathbb{E}_{\pi(\theta)}\mathbb{E}_{\mathcal{T}} R \quad \text{— ?}$$

Direct optimization

Recall RL goal:

$$\mathbb{E}_{\pi(\theta)}\mathbb{E}_{\mathcal{T}} R \to \max_{\theta}$$

Let's optimize our goal directly!

$$\nabla_{\theta}\mathbb{E}_{\pi(\theta)}\mathbb{E}_{\mathcal{T}} R \quad \text{— ?}$$

Direct optimization

Recall RL goal:

$$\mathbb{E}_{\pi(\theta)}\mathbb{E}_{\mathcal{T}} R \to \max_{\theta}$$

Let's optimize our goal directly!

$$\nabla_{\theta}\mathbb{E}_{\pi(\theta)}\mathbb{E}_{\mathcal{T}} R \quad \text{— ?}$$

Options:

* Metaheurisics
* Log-derivative trick[1].

[1] aka REINFORCE

Stochastic estimators optimization via log-derivative trick

$$\nabla_\theta \mathbb{E}_{x\sim\pi(x,\theta)} f(x) = \nabla_\theta \int \pi(x,\theta) f(x) dx$$

Stochastic estimators optimization via log-derivative trick

$$\nabla_\theta \mathbb{E}_{x\sim\pi(x,\theta)} f(x) = \nabla_\theta \int \pi(x,\theta) f(x) dx = \Big\{$$

$$\Big\} = \int \nabla_\theta \pi(x,\theta) f(x) dx$$

Stochastic estimators optimization via log-derivative trick

$$\nabla_\theta \mathbb{E}_{x\sim\pi(x,\theta)} f(x) = \nabla_\theta \int \pi(x,\theta) f(x) dx = \Big\{$$  $$\Big\} = \int \nabla_\theta \pi(x,\theta) f(x) dx$$

Problem: and what?

Stochastic estimators optimization via log-derivative trick

$$\nabla_\theta \mathbb{E}_{x\sim\pi(x,\theta)} f(x) = \nabla_\theta \int \pi(x,\theta) f(x) dx = \Big\{$$

$$\Big\} = \int \nabla_\theta \pi(x,\theta) f(x) dx$$

Problem: and what?

Log-derivative trick

$$\nabla_\theta \pi(\theta) = \pi(\theta) \nabla_\theta \log \pi(\theta)$$

$$\nabla_\theta \mathbb{E}_{x \sim \pi(x,\theta)} f(x) = \nabla_\theta \int \pi(x,\theta) f(x) dx = \Big\{$$

$$\Big\} = \int \nabla_\theta \pi(x,\theta) f(x) dx =$$

Problem: and what?

Log-derivative trick

$$\nabla_\theta \pi(\theta) = \pi(\theta) \nabla_\theta \log \pi(\theta)$$

$$= \int \pi(x,\theta) \nabla_\theta \log \pi(x,\theta) f(x) dx$$

Stochastic estimators optimization via log-derivative trick

$$\nabla_\theta \mathbb{E}_{x\sim\pi(x,\theta)} f(x) = \nabla_\theta \int \pi(x,\theta) f(x) dx = \Big\{$$

$$\Big\} = \int \nabla_\theta \pi(x,\theta) f(x) dx =$$

Problem: and what?

Log-derivative trick

$$\nabla_\theta \pi(\theta) = \pi(\theta) \nabla_\theta \log \pi(\theta)$$

$$= \int \pi(x,\theta) \nabla_\theta \log \pi(x,\theta) f(x) dx = \mathbb{E}_{x\sim\pi(x,\theta)} \nabla_\theta \log \pi(x,\theta) f(x)$$

From importance sampling point of view

Recall Importance Sampling. For arbitrary distribution $\phi(x)$:

$$\mathbb{E}_{x \sim \pi(x,\theta)} f(x) = \mathbb{E}_{x \sim \phi(x)} \frac{\pi(x,\theta)}{\phi(x)} f(x)$$

From importance sampling point of view

Recall Importance Sampling. For arbitrary distribution $\phi(x)$:

$$\mathbb{E}_{x\sim\pi(x,\theta)} f(x) = \mathbb{E}_{x\sim\phi(x)} \frac{\pi(x,\theta)}{\phi(x)} f(x)$$

From importance sampling point of view

Recall Importance Sampling. For arbitrary distribution $\phi(x)$:

$$\mathbb{E}_{x\sim\pi(x,\theta)} f(x) = \mathbb{E}_{x\sim\phi(x)} \frac{\pi(x,\theta)}{\phi(x)} f(x)$$

Let's set $\phi(x) \equiv \pi(x,\theta)$:

From importance sampling point of view

Recall Importance Sampling. For arbitrary distribution $\phi(x)$:

$$\mathbb{E}_{x\sim\pi(x,\theta)} f(x) = \mathbb{E}_{x\sim\phi(x)} \frac{\pi(x,\theta)}{\phi(x)} f(x)$$

Let's set $\phi(x) \equiv \pi(x,\theta)$:

$$\nabla_\theta \mathbb{E}_{x\sim\phi(x)} \frac{\pi(x,\theta)}{\phi(x)} f(x) = \mathbb{E}_{x\sim\phi(x)} \frac{\nabla_\theta \pi(x,\theta)}{\phi(x)} f(x)$$

Note: that is the same gradient as with log-derivative trick[2].

[2] really? Could it even happen otherwise?

REINFORCE

Let's apply log-derivative trick to our goal!

$$\nabla_\theta \mathbb{E}_{\pi(\theta)} \mathbb{E}_{\mathcal{T}} R = \mathbb{E}_{\pi(\theta)} \nabla_\theta \log \pi(\theta) \mathbb{E}_{\mathcal{T}} R$$

REINFORCE

Let's apply log-derivative trick to our goal!

$$\nabla_\theta \mathbb{E}_{\pi(\theta)} \mathbb{E}_{\mathcal{T}} R = \mathbb{E}_{\pi(\theta)} \nabla_\theta \log \pi(\theta) \mathbb{E}_{\mathcal{T}} R \approx$$

We can estimate this gradient using Monte-Carlo by playing, let's say, one game:

$$\approx \sum_t^T \nabla_\theta \log \pi(a_t \mid s_t, \theta) R$$

Problems of REINFORCE

- $\times$ Doesn't work.

Problems of REINFORCE

× Doesn't work.
  - **Reason:** *high variance* of Monte-Carlo gradient estimation.

Problems of REINFORCE

× Doesn't work.
  - **Reason:** *high variance* of Monte-Carlo gradient estimation.
  - you can play more than one game for one gradient step, but that doesn't help much.

Baseline

Proposition

For arbitrary distribution $\pi(\theta)$:

$$\mathbb{E}\nabla_\theta \log \pi(\theta) = \int \nabla_\theta \pi(\theta) = \nabla_\theta \int \pi(\theta) = \nabla_\theta 1 = 0$$

Baseline

Proposition

For arbitrary distribution $\pi(\theta)$:

$$\mathbb{E}\nabla_\theta \log \pi(\theta) = \int \nabla_\theta \pi(\theta) = \nabla_\theta \int \pi(\theta) = \nabla_\theta 1 = 0$$

Adding $\mathbb{E}\nabla_\theta \log \pi(\theta) b$ for some b to gradient estimate will not lead to bias, but may change variance.

Lowest variance baseline

Theorem

$$b = \frac{\mathbb{E}(\nabla_\theta \log \pi(\theta))^2 R}{\mathbb{E}(\nabla_\theta \log \pi(\theta))^2}$$

is the baseline which leads to the lowest variance.

Lowest variance baseline

Theorem

$$b = \frac{\mathbb{E}(\nabla_\theta \log \pi(\theta))^2 R}{\mathbb{E}(\nabla_\theta \log \pi(\theta))^2}$$

is the baseline which leads to the lowest variance.

* similar to average reward, which is also a good baseline.

Careful REINFORCE

Strange thing: our gradient estimate depends on R, which includes reward in the first state $r(s_0)$, where we haven't performed any actions.[3]

[3] did we make any mistake?

Careful REINFORCE

Strange thing: our gradient estimate depends on R, which includes reward in the first state $r(s_0)$, where we haven't performed any actions.[3]

Let's untangle our goal:

$$\nabla_\theta \mathbb{E}_{p(s_1)} \left(r(s_1) + \mathbb{E}_{a_1 \sim \pi(s_1, \theta)} \mathbb{E}_{p(s_2 | s_1, a)} \left[r(s_2) + \dots \right] \right)$$

[3] did we make any mistake?

Careful REINFORCE

Strange thing: our gradient estimate depends on R, which includes reward in the first state $r(s_0)$, where we haven't performed any actions.[3]

Let's untangle our goal:

$$\nabla_\theta \mathbb{E}_{p(s_1)} \left(r(s_1) + \mathbb{E}_{a_1 \sim \pi(s_1, \theta)} \mathbb{E}_{p(s_2 | s_1, a)} \left[r(s_2) + \ldots \right] \right) =$$

After carefully applying log-derivative trick:

$$= \mathbb{E} \sum_t^T \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(\sum_{t' = t+1}^T r(s_{t'}) \right)$$

[3] did we make any mistake?

Careful REINFORCE

Strange thing: our gradient estimate depends on R, which includes reward in the first state $r(s_0)$, where we haven't performed any actions.[3]

Let's untangle our goal:

$$\nabla_\theta \mathbb{E}_{p(s_1)} \left(r(s_1) + \mathbb{E}_{a_1 \sim \pi(s_1, \theta)} \mathbb{E}_{p(s_2 \mid s_1, a)} \left[r(s_2) + \dots \right] \right) =$$

After carefully applying log-derivative trick:

$$= \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(\sum_{t'=t+1}^{T} r(s_{t'}) \right)$$

✓ that's much better!

[3] did we make any mistake?

Actor-critic

Note: $\sum_{t'=t+1}^{T} r(s_{t'})$ is estimation of $Q^{\pi}(s_t, a_t)$!

$$\nabla = \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(\sum_{t'=t+1}^{T} r(s_{t'}) \right)$$

Actor-critic

Note: $\sum_{t'=t+1}^{T} r(s_{t'})$ is estimation of $Q^{\pi}(s_t, a_t)$!

$$\begin{aligned}\nabla = \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(\sum_{t'=t+1}^{T} r(s_{t'}) \right) = \\ = \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) Q^{\pi}(s_t, a_t)\end{aligned}$$

Actor-critic

Note: $\sum_{t'=t+1}^{T} r(s_{t'})$ is estimation of $Q^{\pi}(s_t, a_t)$!

$$\nabla = \mathbb{E} \sum_{t}^{T} \nabla_{\theta} \log \pi(a_t \mid s_t, \theta) \left(\sum_{t'=t+1}^{T} r(s_{t'}) \right) =$$

$$= \mathbb{E} \sum_{t}^{T} \nabla_{\theta} \log \pi(a_t \mid s_t, \theta) Q^{\pi}(s_t, a_t)$$

Better estimation of $Q^{\pi}(s, a)$
should lead to lower variance.

Actor-critic

Note: $\sum_{t'=t+1}^{T} r(s_{t'})$ is estimation of $Q^\pi(s_t, a_t)$!

$$\nabla = \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(\sum_{t'=t+1}^{T} r(s_{t'}) \right) =$$

$$= \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) Q^\pi(s_t, a_t)$$

Better estimation of $Q^\pi(s, a)$
should lead to lower variance.

* π is an *actor*
* estimate of $Q^\pi(s, a)$ is a *critic*

Advantage Actor Critic

Let's insert some baseline:

$$\nabla = \mathbb{E}\sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) Q^\pi(s_t, a_t)$$

Advantage Actor Critic

Let's insert some baseline:

$$\nabla = \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(Q^\pi(s_t, a_t) - b \right)$$

Advantage Actor Critic

Let's insert some baseline:

$$\nabla = \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(Q^\pi(s_t, a_t) - b \right)$$

* Recall average $Q^\pi(s_t, a_t)$ is a good baseline.

Advantage Actor Critic

Let's insert some baseline:

$$\nabla = \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(Q^\pi(s_t, a_t) - b \right)$$

* Recall average $Q^\pi(s_t, a_t)$ is a good baseline.
* Recall $\mathbb{E} Q^\pi(s_t, a_t) = V^\pi(s_t)$

Advantage Actor Critic

Let's insert some baseline:

$$\nabla = \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(Q^\pi(s_t, a_t) - b\right)$$

* Recall average $Q^\pi(s_t, a_t)$ is a good baseline.
* Recall $\mathbb{E}Q^\pi(s_t, a_t) = V^\pi(s_t)$
* Recall definition $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$

Advantage Actor Critic

Let's insert some baseline:

$$\nabla = \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) A^\pi(s_t, a_t)$$

* Recall average $Q^\pi(s_t, a_t)$ is a good baseline.
* Recall $\mathbb{E} Q^\pi(s_t, a_t) = V^\pi(s_t)$
* Recall definition $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$

Critic set up

Critic can be a second neural net!

Critic set up

Critic can be a second neural net!

Options:

* approximate $A^{\pi}(s, a)$

Critic set up

Critic can be a second neural net!

Options:

* approximate $A^{\pi}(s, a)$
* approximate $Q^{\pi}(s, a)$[4]

[4] can we just use Q-learning for this?

Critic set up

Critic can be a second neural net!

Options:

* approximate $A^{\pi}(s, a)$
* approximate $Q^{\pi}(s, a)$[4]
* approximate $V^{\pi}(s)$:

[4] can we just use Q-learning for this?

Critic set up

Critic can be a second neural net!

Options:

* approximate $A^\pi(s, a)$
* approximate $Q^\pi(s, a)$[4]
* approximate $V^\pi(s)$:

$$Q^\pi(s_t, a_t) - V^\pi(s_t) \approx r(s_{t+1}) + V^\pi(s_{t+1}) - V^\pi(s_t)$$

[4] can we just use Q-learning for this?

Critic set up

Critic can be a second neural net!

Options:

* approximate $A^{\pi}(s, a)$
* approximate $Q^{\pi}(s, a)$[4]
* approximate $V^{\pi}(s)$:

  $$Q^{\pi}(s_t, a_t) - V^{\pi}(s_t) \approx r(s_{t+1}) + V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$$

  ✓ the least complex one! [5]

[4] can we just use Q-learning for this?
[5] why?

Critic learning

For given state s we can calculate a target $y = V^{\pi}(s) \approx \sum_{t'=t+1}^{T} r(s_{t'})$.

At the end of the game, make a step of gradient descent to teach critic.

Critic learning

For given state s we can calculate a target $y = V^{\pi}(s) \approx \sum_{t'=t+1}^{T} r(s_{t'})$.

At the end of the game, make a step of gradient descent to teach critic.

Problem: the batch is highly correlated.

Critic learning

For given state s we can calculate a target $y = V^\pi(s) \approx \sum_{t'=t+1}^{T} r(s_{t'})$.

At the end of the game, make a step of gradient descent to teach critic.

Problem: the batch is highly correlated.

* well, play more games.

Critic learning

For given state s we can calculate a target $y = V^{\pi}(s) \approx \sum_{t'=t+1}^{T} r(s_{t'})$.

At the end of the game, make a step of gradient descent to teach critic.

Problem: the batch is highly correlated.

* well, play more games.
  - :(for one step of gradient descent, yeah...

Critic learning

For given state s we can calculate a target $y = V^{\pi}(s) \approx \sum_{t'=t+1}^{T} r(s_{t'})$.

At the end of the game, make a step of gradient descent to teach critic.

Problem: the batch is highly correlated.

* well, play more games.

  :(for one step of gradient descent, yeah...

Alternative: $y = V^{\pi}(s) \approx r(s') + V^{\pi}(s')$

A2C: Complete algorithm

Advantage Actor-Critic (A2C) Algorithm:

- get (s, a, r, s')

A2C: Complete algorithm

Advantage Actor-Critic (A2C) Algorithm:

- get (s, a, r, s')
- update critic $\hat{V}(s)$ using target $r + \hat{V}(s')$

A2C: Complete algorithm

Advantage Actor-Critic (A2C) Algorithm:

- get (s, a, r, s')
- update critic $\hat{V}(s)$ using target $r + \hat{V}(s')$
- evaluate $\hat{A}(s, a) = r + \hat{V}(s') - \hat{V}(s)$

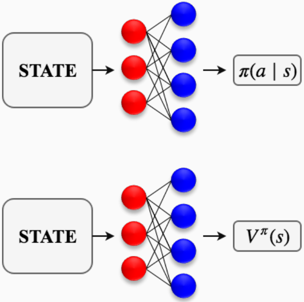
A2C: Complete algorithm

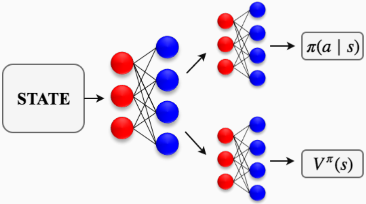
Advantage Actor-Critic (A2C) Algorithm:

- get (s, a, r, s')
- update critic $\hat{V}(s)$ using target $r + \hat{V}(s')$
- evaluate $\hat{A}(s, a) = r + \hat{V}(s') - \hat{V}(s)$
- update policy using estimate of gradient $\nabla_\theta \log \pi(a \mid s, \theta)\hat{A}(s, a)$

Dealing with two networks

Option 1: just two neural nets



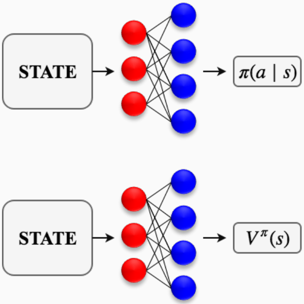


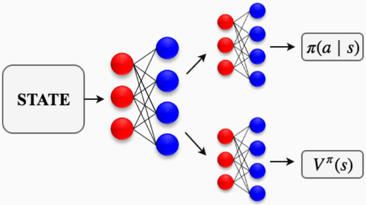
Option 2: shared feature extractor

Dealing with two networks

Option 1: just two neural nets

× obviously redundant





Option 2: shared feature extractor

× may be unstable

A2C: Resume

✓ lower variance!

A2C: Resume

- ✓ lower variance!
- $\times$ yet policy gradient estimates are not unbiased anymore![6]

[6] why?

A2C: Resume

- ✓ lower variance!
- $\times$ yet policy gradient estimates are not unbiased anymore![6]
- $\times$ batch_size $= 1$

[6] why?

A2C: Resume

- ✓ lower variance!
- × yet policy gradient estimates are not unbiased anymore![6]
- × batch_size $= 1$
  - ✓ do gradient descent step every N game steps.

[6] why?

A2C: Resume

- ✓ lower variance!
- × yet policy gradient estimates are not unbiased anymore![6]
- × batch_size $= 1$
  - ✓ do gradient descent step every N game steps.
  - ✓ play several games in parallel.

[6] why?

A2C: Resume

- ✓ lower variance!
- × yet policy gradient estimates are not unbiased anymore!
- × batch_size $= 1$
  - ✓ do gradient descent step every N game steps.
  - ✓ play several games in parallel.

Check out this comic about A2C!

Generalized Advantage Estimation (GAE) (2018)

Playing with Q and V...

$$\nabla = \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(Q^\pi(s_t, a_t) - V^\pi(s_t) \right)$$

In practice we may use separate approximations for $Q^\pi(s_t, a_t)$ and baseline $b = V^\pi(s_t)$ and play with different ways to do that:

Playing with Q and V...

$$\nabla = \mathbb{E} \sum_t^T \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(Q^\pi(s_t, a_t) - V^\pi(s_t) \right)$$

In practice we may use separate approximations for $Q^\pi(s_t, a_t)$ and baseline $b = V^\pi(s_t)$ and play with different ways to do that:

$$\nabla = \mathbb{E} \sum_t^T \nabla_\theta \log \pi(a_t \mid s_t, \theta) \Psi_t$$

Ψ_t	**bias**	**variance**
$\sum_t^T r(s_t)$	0	very high
$Q^\pi(s_t, a_t)$	tolerant	high
$A^\pi(s_t, a_t)$	tolerant	low enough
$\sum_t^T r(s_t) - V^\pi(s_t)$	0	low

Eligibility trace

We may use critic only for baseline:

$$\nabla = \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(\sum_{t'=t+1}^{T} r(s_{t'}) - V^\pi(s_t) \right)$$

Eligibility trace

We may use critic only for baseline:

$$\nabla = \mathbb{E} \sum_t^T \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(\sum_{t'=t+1}^T r(s_{t'}) - V^\pi(s_t) \right)$$

✓ unbiased gradient

Eligibility trace

We may use critic only for baseline:

$$\nabla = \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(\sum_{t'=t+1}^{T} r(s_{t'}) - V^\pi(s_t) \right)$$

✓ unbiased gradient

× higher variance

Eligibility trace

We may use critic only for baseline:

$$\nabla = \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(\sum_{t'=t+1}^{T} r(s_{t'}) - V^\pi(s_t) \right)$$

✓ unbiased gradient

× higher variance

Or use a compromise (for simplicity $\gamma = 1$):

$$\nabla = \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(\sum_{t'=t+1}^{t+N} r(s_{t'}) + V^\pi(s_{t+N}) - V^\pi(s_t) \right)$$

Eligibility trace

We may use critic only for baseline:

$$\nabla = \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(\sum_{t'=t+1}^{T} r(s_{t'}) - V^\pi(s_t) \right)$$

✓ unbiased gradient

× higher variance

Or use a compromise (for simplicity $\gamma = 1$):

$$\nabla = \mathbb{E} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left(\sum_{t'=t+1}^{t+N} r(s_{t'}) + V^\pi(s_{t+N}) - V^\pi(s_t) \right)$$

× new hyperparameter N

✓ regulates trade-off between variance and bias

GAE

So, for different N we have different advantage estimators.

So, for different N we have different advantage estimators.

Generalized Advantage Estimaton (2018):

> Create an ensemble out of them!

GAE

So, for different N we have different advantage estimators.

Generalized Advantage Estimaton (2018):

Create an ensemble out of them!

Let $A^{\pi}_{(N)}(s_t, a_t)$ be a N-step advantage estimator:

$$A^{\pi}_{(N)} = \sum_{t'=t+1}^{t+N} r(s_{t'}) + V^{\pi}(s_{t+N}) - V^{\pi}(s_t)$$

GAE

So, for different N we have different advantage estimators.

Generalized Advantage Estimaton (2018):

Create an ensemble out of them!

Let $A^{\pi}_{(N)}(s_t, a_t)$ be a N-step advantage estimator:

$$A^{\pi}_{(N)} = \sum_{t'=t+1}^{t+N} r(s_{t'}) + V^{\pi}(s_{t+N}) - V^{\pi}(s_t)$$

Let's take exponentially-weighted average:

$$A^{\pi}_{(\mathrm{GAE})}(s_t, a_t) = (1 - \lambda)(A^{\pi}_{(1)} + \lambda A^{\pi}_{(2)} + \lambda^2 A^{\pi}_{(3)} + \dots)$$

GAE in practice

Move convenient formula:

$$A^{\pi}_{(\mathrm{GAE})}(s_t, a_t) = \sum_{i=0}^{\infty} (\lambda\gamma)^i (r(s_{t+i}) + \gamma V^{\pi}(s_{t+i+1}) - V^{\pi}(s_{t+i}))$$

GAE in practice

Move convenient formula:

$$A^{\pi}_{(\mathrm{GAE})}(s_t, a_t) = \sum_{i=0}^{\infty} (\lambda\gamma)^i \left(r(s_{t+i}) + \gamma V^{\pi}(s_{t+i+1}) - V^{\pi}(s_{t+i})\right)$$

* $\lambda = 0$: A2C algorithm
* $\lambda = 1$: infinite eligibility trace algorithm

GAE in practice

Move convenient formula:

$$A^{\pi}_{(\mathrm{GAE})}(s_t, a_t) = \sum_{i=0}^{\infty} (\lambda\gamma)^i (r(s_{t+i}) + \gamma V^{\pi}(s_{t+i+1}) - V^{\pi}(s_{t+i}))$$

* $\lambda = 0$: A2C algorithm
* $\lambda = 1$: infinite eligibility trace algorithm
* the balance is in between... 

NEXT: TRPO

Trust Region Policy Optimization (TRPO) (2017)

Utilizing data

Problem: Actor-Critic algorithm is *on-policy*.

- $\times$ we throw away obtained data after one optimization step.

Utilizing data

Problem: Actor-Critic algorithm is *on-policy*.

× we throw away obtained data after one optimization step.

! but we have to do this!

$$\nabla(\theta) = \mathbb{E}_{\mathcal{T} \sim \pi(\theta)} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) A^\pi(s_t, a_t)$$

Utilizing data

Problem: Actor-Critic algorithm is *on-policy*.

× we throw away obtained data after one optimization step.

! but we have to do this!

$$\nabla(\theta) = \mathbb{E}_{\mathcal{T} \sim \pi(\theta)} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) A^\pi(s_t, a_t)$$

Use importance sampling!

Off-policy Actor-Critic

Let denote $P(\mathcal{T} \mid \pi)$ a probability of trajectory under policy π:

$$P(\mathcal{T} \mid \pi) = p(s_0) \prod_{t=0} \left[\pi(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)\right]$$

Off-policy Actor-Critic

Let denote $P(\mathcal{T} \mid \pi)$ a probability of trajectory under policy π:

$$P(\mathcal{T} \mid \pi) = p(s_0) \prod_{t=0} \left[\pi(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)\right]$$

Then off-policy actor-critic gradient estimation can be obtained:

$$\nabla(\theta) = \mathbb{E}_{\mathcal{T} \sim \tilde{\pi}} \left[\frac{P(\mathcal{T} \mid \pi)}{P(\mathcal{T} \mid \tilde{\pi})} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) A^\pi(s_t, a_t) \right]$$

Off-policy Actor-Critic

Let denote $P(\mathcal{T} \mid \pi)$ a probability of trajectory under policy π:

$$P(\mathcal{T} \mid \pi) = p(s_0) \prod_{t=0} [\pi(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)]$$

Then off-policy actor-critic gradient estimation can be obtained:

$$\nabla(\theta) = \mathbb{E}_{\mathcal{T} \sim \tilde{\pi}} \left[\frac{P(\mathcal{T} \mid \pi)}{P(\mathcal{T} \mid \tilde{\pi})} \sum_{t}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) A^\pi(s_t, a_t) \right]$$

$\times$ though transition probability reduce, this *importance sampling weight* tends to be very close to 0.

TRPO foundations

May be if π is close to $\tilde{\pi}$, this
weight is practically acceptable

TRPO foundations

May be if π is close to $\tilde{\pi}$, this weight is practically acceptable

Trust-Region Policy Optimization (2017) hints:

- a lot of theory on relative performance of two close policies
- attempt to build policy optimization procedure with guarantees of optimizing the objective.[6]
- practical application of natural policy gradients.

[6] what is an obvious drawback of procedure with such property?

TRPO foundations

> May be if π is close to $\tilde{\pi}$, this weight is practically acceptable

Trust-Region Policy Optimization (2017) hints:

- a lot of theory on relative performance of two close policies
- attempt to build policy optimization procedure with guarantees of optimizing the objective.[6]
- practical application of natural policy gradients.

× doesn't provide enthusiastic results on practice...

[6] what is an obvious drawback of procedure with such property?

Relative Policy Performance Identity

Let's denote $J(\pi)$ a performance of policy π, i.e. our objective:

$$J(\pi) \stackrel{\text{def}}{=} \mathbb{E}_{\mathcal{T} \sim \pi} \sum_{t=0} \gamma^t r(s_t) = \mathbb{E}_{s_0} V^\pi(s_0)$$

Relative Policy Performance Identity

Let's denote $J(\pi)$ a performance of policy π, i.e. our objective:

$$J(\pi) \stackrel{\text{def}}{=} \mathbb{E}_{\mathcal{T} \sim \pi} \sum_{t=0} \gamma^t r(s_t) = \mathbb{E}_{s_0} V^\pi(s_0)$$

Theorem (Kakade & Langford, 2002):

$$J(\tilde{\pi}) - J(\pi) = \mathbb{E}_{\mathcal{T} \sim \tilde{\pi}} \sum_{t=0} \gamma^t A^\pi(s_t, a_t)$$

Relative Policy Performance Identity: Proof

$$J(\tilde{\pi}) - J(\pi) = \mathbb{E}_{\mathcal{T} \sim \tilde{\pi}} \sum_{t=0} \gamma^t r(s_t) - J(\pi) =$$

Relative Policy Performance Identity: Proof

$$
\begin{aligned}
J(\tilde{\pi}) - J(\pi) &= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \sum_{t=0} \gamma^t r(s_t) - J(\pi) = \\
&= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \sum_{t=0} \gamma^t r(s_t) - \mathbb{E}_{s_0} V^{\pi}(s_0) =
\end{aligned}
$$

Relative Policy Performance Identity: Proof

$$
\begin{aligned}
J(\tilde{\pi}) - J(\pi) &= \mathbb{E}_{\mathcal{T} \sim \tilde{\pi}} \sum_{t=0} \gamma^t r(s_t) - J(\pi) = \\
&= \mathbb{E}_{\mathcal{T} \sim \tilde{\pi}} \sum_{t=0} \gamma^t r(s_t) - \mathbb{E}_{s_0} V^\pi(s_0) = \\
&= \mathbb{E}_{\mathcal{T} \sim \tilde{\pi}} \left[\sum_{t=0} \gamma^t r(s_t) - V^\pi(s_0) \right] =
\end{aligned}
$$

Relative Policy Performance Identity: Proof

$$
\begin{aligned}
J(\tilde{\pi}) - J(\pi) &= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \sum_{t=0} \gamma^t r(s_t) - J(\pi) = \\
&= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \sum_{t=0} \gamma^t r(s_t) - \mathbb{E}_{s_0} V^\pi(s_0) = \\
&= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \left[\sum_{t=0} \gamma^t r(s_t) - V^\pi(s_0) \right] = \\
&= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \left[\sum_{t=0} \gamma^t r(s_t) + \sum_{t=0} \left[\gamma^{t+1} V^\pi(s_{t+1}) - \gamma^t V^\pi(s_t) \right] \right] =
\end{aligned}
$$

Relative Policy Performance Identity: Proof

$$
\begin{aligned}
J(\tilde{\pi}) - J(\pi) &= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \sum_{t=0} \gamma^t r(s_t) - J(\pi) = \\
&= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \sum_{t=0} \gamma^t r(s_t) - \mathbb{E}_{s_0} V^\pi(s_0) = \\
&= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \left[\sum_{t=0} \gamma^t r(s_t) - V^\pi(s_0) \right] = \\
&= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \left[\sum_{t=0} \gamma^t r(s_t) + \sum_{t=0} \left[\gamma^{t+1} V^\pi(s_{t+1}) - \gamma^t V^\pi(s_t) \right] \right] = \\
&= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \sum_{t=0} \gamma^t \left(r(s_t) + \gamma V^\pi(s_{t+1}) - V^\pi(s_t) \right) =
\end{aligned}
$$

Relative Policy Performance Identity: Proof

$$
\begin{aligned}
J(\tilde{\pi}) - J(\pi) &= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \sum_{t=0} \gamma^t r(s_t) - J(\pi) = \\
&= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \sum_{t=0} \gamma^t r(s_t) - \mathbb{E}_{s_0} V^\pi(s_0) = \\
&= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \left[\sum_{t=0} \gamma^t r(s_t) - V^\pi(s_0) \right] = \\
&= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \left[\sum_{t=0} \gamma^t r(s_t) + \sum_{t=0} \left[\gamma^{t+1} V^\pi(s_{t+1}) - \gamma^t V^\pi(s_t) \right] \right] = \\
&= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \sum_{t=0} \gamma^t \left(r(s_t) + \gamma V^\pi(s_{t+1}) - V^\pi(s_t) \right) = \\
&= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \sum_{t=0} \gamma^t A^\pi(s_t, a_t)
\end{aligned}
$$

Applying importance sampling

Denote $d_\pi(s)$ a *discounted state-visitation probability* for policy π:

$$d_\pi(s) = (1-\gamma)\sum_{t=0}\gamma^t \mathcal{P}(s_t = s)$$

Applying importance sampling

Denote $d_\pi(s)$ a *discounted state-visitation probability* for policy π:

$$d_\pi(s) = (1-\gamma)\sum_{t=0}\gamma^t \mathcal{P}(s_t = s)$$

Let's separate the expectation over trajectory to the expectation over policy choices and over state transitions:

$$J(\tilde{\pi}) - J(\pi) = \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}}\sum_{t=0}\gamma^t A^\pi(s_t, a_t) =$$

Applying importance sampling

Denote $d_\pi(s)$ a *discounted state-visitation probability* for policy π:

$$d_\pi(s) = (1-\gamma)\sum_{t=0}\gamma^t \mathcal{P}(s_t = s)$$

Let's separate the expectation over trajectory to the expectation over policy choices and over state transitions:

$$\begin{aligned} J(\tilde{\pi}) - J(\pi) &= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}}\sum_{t=0}\gamma^t A^\pi(s_t, a_t) = \\ &= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\tilde{\pi}}}\mathbb{E}_{a\sim\tilde{\pi}}A^\pi(s_t, a_t) = \end{aligned}$$

Applying importance sampling

Denote $d_\pi(s)$ a *discounted state-visitation probability* for policy π:

$$d_\pi(s) = (1-\gamma)\sum_{t=0}\gamma^t \mathcal{P}(s_t = s)$$

Let's separate the expectation over trajectory to the expectation over policy choices and over state transitions:

$$\begin{aligned} J(\tilde{\pi}) - J(\pi) &= \mathbb{E}_{\mathcal{T}\sim\tilde{\pi}} \sum_{t=0}\gamma^t A^\pi(s_t, a_t) = \\ &= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\tilde{\pi}}}\mathbb{E}_{a\sim\tilde{\pi}} A^\pi(s_t, a_t) = \\ \{\text{importance sampling}\} &= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\tilde{\pi}}}\mathbb{E}_{a\sim\pi} \frac{\tilde{\pi}(a_t \mid s_t)}{\pi(a_t \mid s_t)} A^\pi(s_t, a_t) \end{aligned}$$

Approximation

Suppose π is current policy and $\tilde{\pi}$ is a policy after one optimization step.

To make this step, we can't sample from $d_{\tilde{\pi}}$.

Approximation

Suppose π is current policy and $\tilde{\pi}$ is a policy after one optimization step. To make this step, we can't sample from $d_{\tilde{\pi}}$.

Available approximation:

$$
\begin{aligned}
&\frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\tilde{\pi}}}\mathbb{E}_{a\sim\pi}\frac{\tilde{\pi}(a_t \mid s_t)}{\pi(a_t \mid s_t)}A^{\pi}(s_t, a_t) \approx \\
\approx &\frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\pi}}\mathbb{E}_{a\sim\pi}\frac{\tilde{\pi}(a_t \mid s_t)}{\pi(a_t \mid s_t)}A^{\pi}(s_t, a_t) =
\end{aligned}
$$

Approximation

Suppose π is current policy and $\tilde{\pi}$ is a policy after one optimization step. To make this step, we can't sample from $d_{\tilde{\pi}}$.

Available approximation:

$$\begin{aligned}
&\frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\tilde{\pi}}}\mathbb{E}_{a\sim\pi}\frac{\tilde{\pi}(a_t \mid s_t)}{\pi(a_t \mid s_t)}A^{\pi}(s_t, a_t) \approx \\
\approx&\frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\pi}}\mathbb{E}_{a\sim\pi}\frac{\tilde{\pi}(a_t \mid s_t)}{\pi(a_t \mid s_t)}A^{\pi}(s_t, a_t) = \\
=&\mathbb{E}_{\mathcal{T}\sim\pi}\frac{\tilde{\pi}(a_t \mid s_t)}{\pi(a_t \mid s_t)}A^{\pi}(s_t, a_t) \stackrel{\text{def}}{=} L(\tilde{\pi})
\end{aligned}$$

Approximation

Suppose π is current policy and $\tilde{\pi}$ is a policy after one optimization step. To make this step, we can't sample from $d_{\tilde{\pi}}$.

Available approximation:

$$
\begin{aligned}
&\frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\tilde{\pi}}}\mathbb{E}_{a\sim\pi}\frac{\tilde{\pi}(a_t \mid s_t)}{\pi(a_t \mid s_t)}A^{\pi}(s_t, a_t) \approx \\
\approx&\frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\pi}}\mathbb{E}_{a\sim\pi}\frac{\tilde{\pi}(a_t \mid s_t)}{\pi(a_t \mid s_t)}A^{\pi}(s_t, a_t) = \\
=&\mathbb{E}_{\mathcal{T}\sim\pi}\frac{\tilde{\pi}(a_t \mid s_t)}{\pi(a_t \mid s_t)}A^{\pi}(s_t, a_t) \stackrel{\text{def}}{=} L(\tilde{\pi})
\end{aligned}
$$

Theorem:

If ε is the approximation error:

$$|\varepsilon| \leq \text{Const}\, KL^{\max}(\tilde{\pi} \parallel \pi)$$

The familiar gradients...

Let π's parameters be θ_k (fixed), $\tilde{\pi}$'s parameters be θ.
To optimize θ, let's find $\nabla_\theta L(\tilde{\pi}(\theta))|_{\theta_k}$:

$$\nabla_\theta L(\tilde{\pi}(\theta))|_{\theta_k} = \nabla_\theta \left[\mathbb{E}_{\mathcal{T}\sim\pi} \frac{\tilde{\pi}_\theta(a_t \mid s_t)}{\pi(a_t \mid s_t)} A^\pi(s_t, a_t) \right] \Bigg|_{\theta_k} =$$

The familiar gradients...

Let π's parameters be θ_k (fixed), $\tilde{\pi}$'s parameters be θ.
To optimize θ, let's find $\nabla_\theta L(\tilde{\pi}(\theta))|_{\theta_k}$:

$$\begin{aligned}
\nabla_\theta L(\tilde{\pi}(\theta))|_{\theta_k} = \nabla_\theta \left[\mathbb{E}_{\mathcal{T}\sim\pi} \frac{\tilde{\pi}_\theta(a_t \mid s_t)}{\pi(a_t \mid s_t)} A^\pi(s_t, a_t) \right] \Bigg|_{\theta_k} = \\
= \mathbb{E}_{\mathcal{T}\sim\pi} \frac{\nabla_\theta \tilde{\pi}_\theta(a_t \mid s_t)|_{\theta_k}}{\pi(a_t \mid s_t)} A^\pi(s_t, a_t) =
\end{aligned}$$

The familiar gradients...

Let π's parameters be θ_k (fixed), $\tilde{\pi}$'s parameters be θ.
To optimize θ, let's find $\nabla_\theta L(\tilde{\pi}(\theta))|_{\theta_k}$:

$$\nabla_\theta L(\tilde{\pi}(\theta))|_{\theta_k} = \nabla_\theta \left[\mathbb{E}_{\mathcal{T}\sim\pi} \frac{\tilde{\pi}_\theta(a_t \mid s_t)}{\pi(a_t \mid s_t)} A^\pi(s_t, a_t) \right] \Bigg|_{\theta_k} =$$

$$= \mathbb{E}_{\mathcal{T}\sim\pi} \frac{\nabla_\theta \tilde{\pi}_\theta(a_t \mid s_t)|_{\theta_k}}{\pi(a_t \mid s_t)} A^\pi(s_t, a_t) =$$

$$\{\pi \equiv \tilde{\pi}(\theta_k)\} = \mathbb{E}_{\mathcal{T}\sim\pi} \nabla_\theta \log \tilde{\pi}_\theta(a_t \mid s_t)|_{\theta_k} A^\pi(s_t, a_t)$$

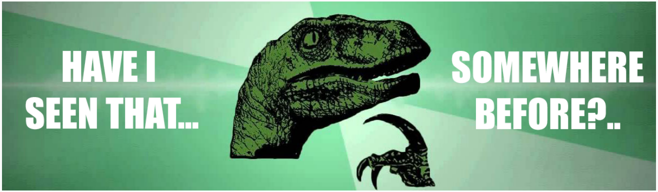
The familiar gradients...

Let π's parameters be θ_k (fixed), $\tilde{\pi}$'s parameters be θ.
To optimize θ, let's find $\nabla_\theta L(\tilde{\pi}(\theta))|_{\theta_k}$:

$$\nabla_\theta L(\tilde{\pi}(\theta))|_{\theta_k} = \nabla_\theta \left[\mathbb{E}_{\mathcal{T}\sim\pi} \frac{\tilde{\pi}_\theta(a_t \mid s_t)}{\pi(a_t \mid s_t)} A^\pi(s_t, a_t) \right] \Bigg|_{\theta_k} =$$

$$= \mathbb{E}_{\mathcal{T}\sim\pi} \frac{\nabla_\theta \tilde{\pi}_\theta(a_t \mid s_t)|_{\theta_k}}{\pi(a_t \mid s_t)} A^\pi(s_t, a_t) =$$

$$\{\pi \equiv \tilde{\pi}(\theta_k)\} = \mathbb{E}_{\mathcal{T}\sim\pi} \nabla_\theta \log \tilde{\pi}_\theta(a_t \mid s_t)|_{\theta_k} A^\pi(s_t, a_t)$$



Improvement guarantees

From approximation error bound follows:

$$J(\tilde{\pi}) - J(\pi) \geq L(\tilde{\pi}) - C \, KL^{\max}(\tilde{\pi} \parallel \pi)$$

Improvement guarantees

From approximation error bound follows:

$$J(\tilde{\pi}) - J(\pi) \geq L(\tilde{\pi}) - C\, KL^{\max}(\tilde{\pi} \parallel \pi)$$

Consider the lower bound optimization procedure:

$$\pi_{k+1} = \underset{\tilde{\pi}}{argmax} \left[L(\tilde{\pi}) - C\, KL^{\max}(\tilde{\pi} \parallel \pi_k) \right]$$

Improvement guarantees

From approximation error bound follows:

$$J(\tilde{\pi}) - J(\pi) \geq L(\tilde{\pi}) - C\, KL^{\max}(\tilde{\pi} \parallel \pi)$$

Consider the lower bound optimization procedure:

$$\pi_{k+1} = \underset{\tilde{\pi}}{argmax} \left[L(\tilde{\pi}) - C\, KL^{\max}(\tilde{\pi} \parallel \pi_k) \right]$$

Then:

$$\begin{aligned} J(\pi_{k+1}) - J(\pi_k) \geq L(\pi_{k+1}) - C\, KL^{\max}(\pi_{k+1} \parallel \pi_k) \geq \\ \geq L(\pi_k) - C\, KL^{\max}(\pi_k \parallel \pi_k) = 0 - 0 = 0 \end{aligned}$$

Improvement guarantees

From approximation error bound follows:

$$J(\tilde{\pi}) - J(\pi) \geq L(\tilde{\pi}) - C\,KL^{\max}(\tilde{\pi} \parallel \pi)$$

Consider the lower bound optimization procedure:

$$\pi_{k+1} = \underset{\tilde{\pi}}{argmax}\,[L(\tilde{\pi}) - C\,KL^{\max}(\tilde{\pi} \parallel \pi_k)]$$

Then:

$$\begin{aligned} J(\pi_{k+1}) - J(\pi_k) \geq L(\pi_{k+1}) - C\,KL^{\max}(\pi_{k+1} \parallel \pi_k) \geq \\ \geq L(\pi_k) - C\,KL^{\max}(\pi_k \parallel \pi_k) = 0 - 0 = 0 \end{aligned}$$

✓ procedure guarantees to improve $J(\pi)$!

Harsh reality

$$\pi_{k+1} = \underset{\tilde{\pi}}{argmax} \left[\mathbb{E}_{\mathcal{T} \sim \pi_k} \frac{\tilde{\pi}(a_t \mid s_t)}{\pi_k(a_t \mid s_t)} A^{\pi_k}(s_t, a_t) - C \, KL^{\max}(\tilde{\pi} \parallel \pi_k) \right]$$

- × A^π is never precise.
- × expectations estimations are never precise.

Harsh reality

$$\pi_{k+1} = \underset{\tilde{\pi}}{argmax} \left[\mathbb{E}_{\mathcal{T} \sim \pi_k} \frac{\tilde{\pi}(a_t \mid s_t)}{\pi_k(a_t \mid s_t)} A^{\pi_k}(s_t, a_t) - C\, KL^{\max}(\tilde{\pi} \parallel \pi_k) \right]$$

- × A^{π} is never precise.
- × expectations estimations are never precise.
- × we can't calculate maximal divergence between two policies over *all* states.

Harsh reality

$$\pi_{k+1} = \underset{\tilde{\pi}}{argmax} \left[\mathbb{E}_{\mathcal{T} \sim \pi_k} \frac{\tilde{\pi}(a_t \mid s_t)}{\pi_k(a_t \mid s_t)} A^{\pi_k}(s_t, a_t) - C\, KL^{\max}(\tilde{\pi} \parallel \pi_k) \right]$$

× A^{π} is never precise.

× expectations estimations are never precise.

× we can't calculate maximal divergence between two policies over *all* states.

 TRPO: $KL^{\max}(\tilde{\pi} \parallel \pi_k) \approx \mathbb{E}_{s \sim d_{\pi_k}} KL(\tilde{\pi} \parallel \pi_k)[s]$

Harsh reality

$$\pi_{k+1} = \underset{\tilde{\pi}}{argmax} \left[\mathbb{E}_{\mathcal{T} \sim \pi_k} \frac{\tilde{\pi}(a_t \mid s_t)}{\pi_k(a_t \mid s_t)} A^{\pi_k}(s_t, a_t) - C\, KL^{\max}(\tilde{\pi} \parallel \pi_k) \right]$$

- × A^π is never precise.
- × expectations estimations are never precise.
- × we can't calculate maximal divergence between two policies over *all* states.

 TRPO: $KL^{\max}(\tilde{\pi} \parallel \pi_k) \approx \mathbb{E}_{s \sim d_{\pi_k}} KL(\tilde{\pi} \parallel \pi_k)[s]$

- × the constant over here is huge when γ is close to 1 and depends on MDP characteristics.

Harsh reality

$$\pi_{k+1} = \underset{\tilde{\pi}}{argmax} \left[\mathbb{E}_{\mathcal{T} \sim \pi_k} \frac{\tilde{\pi}(a_t \mid s_t)}{\pi_k(a_t \mid s_t)} A^{\pi_k}(s_t, a_t) - C\, KL^{\max}(\tilde{\pi} \parallel \pi_k) \right]$$

× A^π is never precise.

× expectations estimations are never precise.

× we can't calculate maximal divergence between two policies over *all* states.

 TRPO: $KL^{\max}(\tilde{\pi} \parallel \pi_k) \approx \mathbb{E}_{s \sim d_{\pi_k}} KL(\tilde{\pi} \parallel \pi_k)[s]$

× the constant over here is huge when γ is close to 1 and depends on MDP characteristics.

 TRPO: Trust-Region optimization scheme!

Trust-Region optimization

$$\begin{cases} \pi_{k+1} = \mathbb{E}_{\mathcal{T} \sim \pi_k} \frac{\tilde{\pi}(a_t|s_t)}{\pi_k(a_t|s_t)} A^{\pi_k}(s_t, a_t) \to \max\limits_{\tilde{\pi}} \\ \text{s.t.} \quad \mathbb{E}_{s \sim d_{\pi_k}} KL(\tilde{\pi} \parallel \pi_k)[s] \le \delta \end{cases}$$

Trust-Region optimization

$$\begin{cases} \pi_{k+1} = \mathbb{E}_{\mathcal{T} \sim \pi_k} \frac{\tilde{\pi}(a_t | s_t)}{\pi_k(a_t | s_t)} A^{\pi_k}(s_t, a_t) \to \max\limits_{\tilde{\pi}} \\ \text{s.t.} \quad \mathbb{E}_{s \sim d_{\pi_k}} KL(\tilde{\pi} \parallel \pi_k)[s] \le \delta \end{cases}$$

× δ is a hyperparameter. 

Trust-Region optimization

$$\begin{cases} \pi_{k+1} = \mathbb{E}_{\mathcal{T} \sim \pi_k} \frac{\tilde{\pi}(a_t|s_t)}{\pi_k(a_t|s_t)} A^{\pi_k}(s_t, a_t) \to \max\limits_{\tilde{\pi}} \\ \text{s.t.} \quad \mathbb{E}_{s \sim d_{\pi_k}} KL(\tilde{\pi} \parallel \pi_k)[s] \leq \delta \end{cases}$$

× δ is a hyperparameter. 

✓ respects distance in policy space!

- also known in theory as *natural gradient*. In previous policy gradient methods we implicitly used the constrain

$$\|\tilde{\theta} - \theta_k\|_2^2 \leq \alpha$$

where α was learning rate of optimizer.

Natural Policy Gradient

Metric in most general form may depend from current coordinates:

$$\rho(x, x+d) = d^T G(x) d$$

- $G(x)$ is called *metric tensor*.

Natural Policy Gradient

Metric in most general form may depend from current coordinates:

$$\rho(x, x+d) = d^T G(x) d$$

- $G(x)$ is called *metric tensor*.

Theorem:

For space of policies, *Fisher information matrix* is metric tensor:

$$H(\theta) = \mathbb{E}_{a \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a \mid s) \log \pi_\theta(a \mid s)^T \right]$$

Natural Policy Gradient

Metric in most general form may depend from current coordinates:

$$\rho(x, x+d) = d^T G(x) d$$

- $G(x)$ is called *metric tensor*.

Theorem:

For space of policies, *Fisher information matrix* is metric tensor:

$$H(\theta) = \mathbb{E}_{a \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a \mid s) \log \pi_\theta(a \mid s)^T \right]$$

Main natural gradient property (parametrization invariance)

For any parametrization π_θ

$$H^{-1} \nabla_\theta \pi_\theta$$

is the same vector in policies space.

Practical application

Recalling standard optimization methods to solve constraint task:

$$\begin{cases} L(\theta) \to \max\limits_{\theta} \\ \text{s.t.} \quad \mathbb{E}_{s \sim d_{\pi(\theta_k)}} KL(\pi(\theta) \parallel \pi(\theta_k))[s] \leq \delta \end{cases}$$

Practical application

Recalling standard optimization methods to solve constraint task:

$$\begin{cases} L(\theta) \to \max\limits_{\theta} \\ \text{s.t.} \quad \mathbb{E}_{s \sim d_{\pi(\theta_k)}} KL(\pi(\theta) \parallel \pi(\theta_k))[s] \leq \delta \end{cases}$$

Linear approximation of optimized objective:

$$L(\theta) \approx L(\theta_k) + g^T(\theta - \theta_k) \qquad \text{where} \quad g = \nabla_\theta L(\theta)|_{\theta_k}$$

Practical application

Recalling standard optimization methods to solve constraint task:

$$\begin{cases} L(\theta) \to \max\limits_{\theta} \\ \text{s.t.} \quad \mathbb{E}_{s \sim d_{\pi(\theta_k)}} KL(\pi(\theta) \parallel \pi(\theta_k))[s] \leq \delta \end{cases}$$

Linear approximation of optimized objective:

$$L(\theta) \approx L(\theta_k) + g^T(\theta - \theta_k) \qquad \text{where} \quad g = \nabla_\theta L(\theta)|_{\theta_k}$$

Quadratic approximation of constraint [7]:

$$\mathbb{E}_s \, KL(\pi(\theta) \parallel \pi(\theta_k))[s] \approx (\theta - \theta_k)^T H(\theta - \theta_k)$$
$$\text{where} \quad H = \mathbb{E}_s \, \nabla^2_\theta \, KL(\pi(\theta) \parallel \pi(\theta_k))[s]\Big|_{\theta_k}$$

[7] where is linear term?

Trust-Region optimization procedure

Theorem:

$\nabla^2_\theta KL(\pi(\theta) \parallel \pi(\theta_k))[s]\Big|_{\theta_k}$ is Fisher information matrix.

✓ that's why solving this task is equivalent to gradient ascent with natural policy gradient!

Trust-Region optimization procedure

Theorem:

$\nabla^2_\theta KL(\pi(\theta) \parallel \pi(\theta_k))[s]\big|_{\theta_k}$ is Fisher information matrix.

✓ that's why solving this task is equivalent to gradient ascent with natural policy gradient!

Solution (derived with K.K.T. theorem):

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g_k^T H_k^{-1} g_k}} H_k^{-1} g_k$$

Trust-Region optimization procedure

Theorem:

$\nabla^2_\theta KL(\pi(\theta) \parallel \pi(\theta_k))[s]\big|_{\theta_k}$ is Fisher information matrix.

- ✓ that's why solving this task is equivalent to gradient ascent with natural policy gradient!

Solution (derived with K.K.T. theorem):

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g_k^T H_k^{-1} g_k}} H_k^{-1} g_k$$

- ✓ δ substitutes learning rate.
- × g_k, H_k can only be estimated via samples.

Trust-Region optimization procedure

Theorem:

$\nabla^2_\theta KL(\pi(\theta) \parallel \pi(\theta_k))[s]\Big|_{\theta_k}$ is Fisher information matrix.

✓ that's why solving this task is equivalent to gradient ascent with natural policy gradient!

Solution (derived with K.K.T. theorem):

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g_k^T H_k^{-1} g_k}} H_k^{-1} g_k$$

✓ δ substitutes learning rate.

× g_k, H_k can only be estimated via samples. 

× **Problem:** how to compute H_k^{-1} on practice? For neural networks with N parameters inversion complexity is $\mathcal{O}(N^3)$!..

Conjugate Gradients saving the day

Remembering CG algorithm:

- solves system of linear equations $H_k x = g_k$.

Conjugate Gradients saving the day

Remembering CG algorithm:

- solves system of linear equations $H_k x = g_k$.
- ✓ after j iterations returns sub-optimal solution (approximation of $H^{-1}g$, optimal in Krylov subspace, $\mathcal{L}(g, Hg, H^2 g \dots H^{j-1} g)$)

Conjugate Gradients saving the day

Remembering CG algorithm:

- solves system of linear equations $H_k x = g_k$.

✓ after j iterations returns sub-optimal solution (approximation of $H^{-1}g$, optimal in Krylov subspace, $\mathcal{L}(g, Hg, H^2 g \ldots H^{j-1}g)$)

✓ only $f(v) = H_k v$ is required.

  $=$ can be implemented on PyTorch!

Conjugate Gradients saving the day

Remembering CG algorithm:

- solves system of linear equations $H_k x = g_k$.
- ✓ after j iterations returns sub-optimal solution (approximation of $H^{-1}g$, optimal in Krylov subspace, $\mathcal{L}(g, Hg, H^2g \ldots H^{j-1}g)$)
- ✓ only $f(v) = H_k v$ is required.
  - $=$ can be implemented on PyTorch!

Results

With all these approximations no theoretical guarantees remain, of course.

Results

With all these approximations no theoretical guarantees remain, of course.

What is suggested in different papers?

- NPG (2005): just use $H^{-1}g$ as gradient, computed somehow, without looking for improvement guarantees.

Results

With all these approximations no theoretical guarantees remain, of course.

What is suggested in different papers?

- NPG (2005): just use $H^{-1}g$ as gradient, computed somehow, without looking for improvement guarantees.
- TRPO (2017): use line search with basic backtracking to guarantee $L(\pi) \geq 0$ and $KL(\pi \parallel \pi_k) < \delta$.

Results

With all these approximations no theoretical guarantees remain, of course.

What is suggested in different papers?

- NPG (2005): just use $H^{-1}g$ as gradient, computed somehow, without looking for improvement guarantees.
- TRPO (2017): use line search with basic backtracking to guarantee $L(\pi) \geq 0$ and $KL(\pi \parallel \pi_k) < \delta$.
- PPO (2017): coming soon.
- ACKTR (2017): coming soon.