Reinforcement Learning

Chapter 6: Actor Critic Methods

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

University of Toronto

Fall 2025

Extension to Other PGMs

In practice, all PGMs we studies in Chapter 5 can be implemented via the actor-critic idea through the following general framework

Loop over the following three steps

1. *Specify policy and value networks*
   ↳ *They could be either separate DNNs or DNNs with shared layers*
2. *Set the policy and sample a batch of trajectories*
3. *Go over the batch for multiple epochs*
   ↳ *After each mini-batch estimate the policy and value gradient*
   ↳ *Update both policy and value networks after each mini-batch*

Let's now look at the TRPO and PPO algorithms in actor-critic framework

TRPO: *Actor-Critic*

```
TRPO():
```

Initiate with $\boldsymbol{\theta}$ *and* $\mathbf{w}$, *as well as factor* $\alpha < 1$ *and learning rate* β
while *interacting* **do**
    Sample a *batch of trajectories* $S_0, A_0 \xrightarrow{R_1} \cdots \xrightarrow{R_T} S_T$ by policy $\pi_{\boldsymbol{\theta}}$
    for multiple epochs **do**
        for samples in each *mini-batch* **do**
            Compute *sample advantage* using $v_{\mathbf{w}}(\cdot)$
            Update policy *gradient* $\hat{\nabla}_{\boldsymbol{\theta}}$ and *value gradient* $\hat{\nabla}_{\mathbf{w}}$
        end for
        Compute a *Hessian estimator* $\hat{\mathbf{H}}$ and solve $\hat{\mathbf{H}}\mathbf{y} = \hat{\nabla}$ for $\mathbf{y}$
        Backtrack on a line to find minimum i satisfying $\bar{D}_{\mathrm{KL}}(\pi_{\boldsymbol{\theta}'} \| \pi_{\boldsymbol{\theta}}) \leqslant d_{\max}$

$$\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} + \alpha^{i} \sqrt{\frac{2 d_{\max}}{\mathbf{y}^{\mathsf{T}} \hat{\mathbf{H}} \mathbf{y}}}\, \mathbf{y}$$

        Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}'$ and $\mathbf{w} \leftarrow \mathbf{w} + \beta \hat{\nabla}_{\mathbf{w}}$
    end for
end while

Recall: *Use Importance Sampling*

Attention: *Importance Sampling*

It is important to remember that in each iteration of PGM

we estimate the policy gradient via importance sampling

Let's denote the policy of current mini-batch with $\pi_{\boldsymbol{\theta}}$: in next mini-batch we compute the policy gradient as

$$\hat{\nabla}_{\boldsymbol{\theta}} \leftarrow \hat{\nabla}_{\boldsymbol{\theta}} + \sum_{t=0}^{T} U_t \frac{\nabla \pi_{\mathbf{x}} \left(A_t | S_t \right) |_{\mathbf{x}=\boldsymbol{\theta}}}{\pi_{\boldsymbol{\theta}} \left(A_t | S_t \right)}$$

with U_t being the sample advantage of policy $\pi_{\boldsymbol{\theta}}$

Recall: *Use Importance Sampling*

+ *You mentioned this before! What is new about this?!*
– *Well! We should also consider it in our value estimation!*

Denote the policy that we sampled with in line 3 with $\pi_{\boldsymbol{\theta}_{\rm old}}$: after multiple mini-batches we have an updated policy gradient $\pi_{\boldsymbol{\theta}}$

⚠ *To update the value network in this mini-batch, we consider the Bellman equation which says*

$$v_{\pi_{\boldsymbol{\theta}}}\left(S_t\right) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left\{R_{t+1} + \gamma v_{\pi_{\boldsymbol{\theta}}}\left(S_{t+1}\right)\right\}$$

and set the labels for value network training as

$$\hat{v}_{\pi_{\boldsymbol{\theta}}}\left(S_t\right) = R_{t+1} + \gamma v_{\pi_{\boldsymbol{\theta}}}\left(S_{t+1}\right)$$

But this is only valid if we had sampled the trajectory by $\pi_{\boldsymbol{\theta}}$!

Recall: *Use Importance Sampling*

+ *Shall we use importance sampling here as well?!*
– *Sure!*

By importance sampling we could say

$$
\begin{aligned}
v_{\pi_{\boldsymbol{\theta}}}(S_t) &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left\{R_{t+1} + \gamma v_{\pi_{\boldsymbol{\theta}}}(S_{t+1})\right\} \\
&= \mathbb{E}_{\pi_{\boldsymbol{\theta}_{\text{old}}}}\left\{\left(R_{t+1} + \gamma v_{\pi_{\boldsymbol{\theta}}}(S_{t+1})\right)\frac{\pi_{\boldsymbol{\theta}}(A_t|S_t)}{\pi_{\boldsymbol{\theta}_{\text{old}}}(A_t|S_t)}\right\}
\end{aligned}
$$

and now we can compute value estimators from our sample trajectories as

$$
\hat{v}_{\pi_{\boldsymbol{\theta}}}(S_t) = \left(R_{t+1} + \gamma v_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(S_{t+1})\right)\frac{\pi_{\boldsymbol{\theta}}(A_t|S_t)}{\pi_{\boldsymbol{\theta}_{\text{old}}}(A_t|S_t)}
$$

Recall: *Use Importance Sampling*

This means that the advantage should be computed as

$$
\begin{aligned}
U_t &= \underbrace{\left(R_{t+1} + \gamma v_{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}}\left(S_{t+1}\right)\right)}_{\text{samples in batch}} \underbrace{\frac{\pi_{\boldsymbol{\theta}}\left(A_t | S_t\right)}{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}\left(A_t | S_t\right)}}_{\text{importance sampling}} - v_{\mathbf{w}}\left(S_t\right) \\
&\approx \left(R_{t+1} + \gamma v_{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}}\left(S_{t+1}\right) - v_{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}}\left(S_t\right)\right) \frac{\pi_{\boldsymbol{\theta}}\left(A_t | S_t\right)}{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}\left(A_t | S_t\right)} \\
&= U_t^{\mathrm{old}} \frac{\pi_{\boldsymbol{\theta}}\left(A_t | S_t\right)}{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}\left(A_t | S_t\right)}
\end{aligned}
$$

This is the correct way of estimating advantage: other implementations that ignore this will have bias especially with too much epochs

Recall: *Use Importance Sampling*

Consequently, the value gradient is updated as

$$
\begin{aligned}
\hat{\nabla}_{\mathbf{w}} &\leftarrow \hat{\nabla}_{\mathbf{w}} + \frac{1}{T} \sum_{t=0}^{T-1} U_t \nabla v_{\mathbf{w}} \left(S_t \right) \\
&\leftarrow \hat{\nabla}_{\mathbf{w}} + U_t^{\text{old}} \frac{\pi_{\boldsymbol{\theta}} \left(A_t | S_t \right)}{\pi_{\boldsymbol{\theta}_{\text{old}}} \left(A_t | S_t \right)} \nabla v_{\mathbf{w}} \left(S_t \right)
\end{aligned}
$$

at the end of each trajectory

+ *Shall we also consider this when we update the policy?*
- *Of course!*

Recall: *Use Importance Sampling*

In a given mini-batch we update the policy gradient as

$$
\begin{aligned}
\hat{\nabla}_{\boldsymbol{\theta}} &\leftarrow \hat{\nabla}_{\boldsymbol{\theta}} + \sum_{t=0}^{T} U_t \frac{\nabla \pi_{\mathbf{x}} \left(A_t | S_t \right) |_{\mathbf{x}=\boldsymbol{\theta}}}{\pi_{\boldsymbol{\theta}} \left(A_t | S_t \right)} \\
&\leftarrow \hat{\nabla}_{\boldsymbol{\theta}} + \sum_{t=0}^{T} U_t^{\mathrm{old}} \frac{\pi_{\boldsymbol{\theta}} \left(A_t | S_t \right)}{\pi_{\boldsymbol{\theta}_{\mathrm{old}}} \left(A_t | S_t \right)} \frac{\nabla \pi_{\mathbf{x}} \left(A_t | S_t \right) |_{\mathbf{x}=\boldsymbol{\theta}}}{\pi_{\boldsymbol{\theta}} \left(A_t | S_t \right)} \\
&\leftarrow \hat{\nabla}_{\boldsymbol{\theta}} + \sum_{t=0}^{T} U_t^{\mathrm{old}} \frac{\nabla \pi_{\mathbf{x}} \left(A_t | S_t \right) |_{\mathbf{x}=\boldsymbol{\theta}}}{\pi_{\boldsymbol{\theta}_{\mathrm{old}}} \left(A_t | S_t \right)}
\end{aligned}
$$

which is now consistent with importance sampling

TRPO: *Actor-Critic*

```
TRPO_AC():
```

1: *Initiate with* $\boldsymbol{\theta} = \boldsymbol{\theta}_{\rm old}$ *and* $\mathbf{w}$, *as well as factor* $\alpha < 1$ *and learning rate* β

2: ***while*** *interacting* ***do***

3: *Sample a batch of trajectories* $S_0, A_0 \xrightarrow{R_1} \cdots \xrightarrow{R_T} S_T$ *by policy* $\pi_{\boldsymbol{\theta}_{\rm old}}$

4: *Compute sample advantage* $U_t^{\rm old}$ *using* $v_{\mathbf{w}}(\cdot)$

5: ***for*** *multiple epochs in each mini-batch* ***do***

6: *Compute gradient estimators* $(\hat{\nabla}_{\boldsymbol{\theta}}, \hat{\nabla}_{\mathbf{w}})$ *from* $U_t^{\rm old}$ *via importance sampling*

7: *Compute a Hessian estimator* $\hat{\mathbf{H}}$ *and solve* $\hat{\mathbf{H}}\mathbf{y} = \hat{\nabla}_{\boldsymbol{\theta}}$ *for* $\mathbf{y}$

8: *Backtrack on a line to find minimum* i *satisfying* $\bar{D}_{\rm KL}(\pi_{\boldsymbol{\theta}'} \| \pi_{\boldsymbol{\theta}}) \leqslant d_{\max}$

$$\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} + \alpha^i \sqrt{\frac{2 d_{\max}}{\mathbf{y}^{\mathsf{T}} \hat{\mathbf{H}} \mathbf{y}}} \mathbf{y}$$

9: *Update* $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}'$ *and* $\mathbf{w} \leftarrow \mathbf{w} + \beta \hat{\nabla}_{\mathbf{w}}$

10: ***end for***

11: *Update* $\pi_{\boldsymbol{\theta}} \leftarrow \pi_{\boldsymbol{\theta}_{\rm old}}$

12: ***end while***

PPO: *Actor-Critic*

In PPO, we maximize in each iteration the restricted surrogate

$$\tilde{\mathcal{L}}(\pi_{\mathbf{x}}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left\{\min\left\{U_t \frac{\pi_{\mathbf{x}}(A_t|S_t)}{\pi_{\boldsymbol{\theta}}(A_t|S_t)}, \ell_\varepsilon(U_t)\right\}\right\}$$

Similar to TRPO, we can estimate restricted surrogate via importance sampling

$$\begin{aligned}\tilde{\mathcal{L}}(\pi_{\mathbf{x}}) &= \mathtt{mean}\left[\sum_t \min\left\{U_t \frac{\pi_{\mathbf{x}}(A_t|S_t)}{\pi_{\boldsymbol{\theta}}(A_t|S_t)}, \ell_\varepsilon(U_t)\right\}\right] \\ &= \mathtt{mean}\left[\sum_t \min\left\{U_t^{\mathrm{old}} \frac{\pi_{\mathbf{x}}(A_t|S_t)}{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}(A_t|S_t)}, \ell_\varepsilon(U_t)\right\}\right]\end{aligned}$$

where the mean is computed over the sample trajectories of a mini-batch

PPO Algorithm: *Actor-Critic*

`PPO_AC():`

Initiate with $\boldsymbol{\theta} = \boldsymbol{\theta}_{\mathrm{old}}$ *and* $\mathbf{w}$*, as well as learning rates* α *and* β

while *interacting* ***do***

  Sample a batch of trajectories $S_0, A_0 \xrightarrow{R_1} \cdots \xrightarrow{R_T} S_T$ *by policy* $\pi_{\boldsymbol{\theta}_{\mathrm{old}}}$

  Compute sample advantage U_t^{old} *using* $v_{\mathbf{w}}(\cdot)$

  for *multiple epochs in each mini-batch* ***do***

    Compute value gradient estimator $\hat{\nabla}_{\mathbf{w}}$ *from* U_t^{old} *via importance sampling*

    Compute the restricted surrogate

$$\tilde{\mathcal{L}}(\pi_{\mathbf{x}}) = \mathtt{mean}\left[\sum_t \min\left\{U_t^{\mathrm{old}} \frac{\pi_{\mathbf{x}}(A_t|S_t)}{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}(A_t|S_t)}, \ell_{\varepsilon}(U_t)\right\}\right]$$

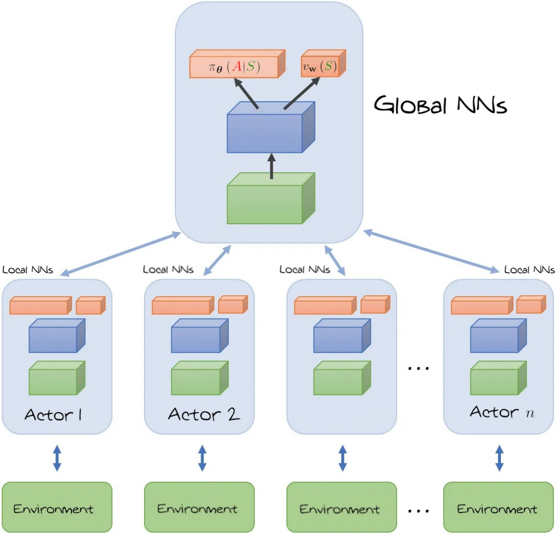
    Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla \tilde{\mathcal{L}}(\pi_{\mathbf{x}})\,|_{\mathbf{x}=\boldsymbol{\theta}}$ *and* $\mathbf{w} \leftarrow \mathbf{w} + \beta \hat{\nabla}_{\mathbf{w}}$

  end for

  Update $\pi_{\boldsymbol{\theta}} \leftarrow \pi_{\boldsymbol{\theta}_{\mathrm{old}}}$

end while

Distributed Actor-Critic



Distributed Setting: *Asynchronous vs Synchronous*

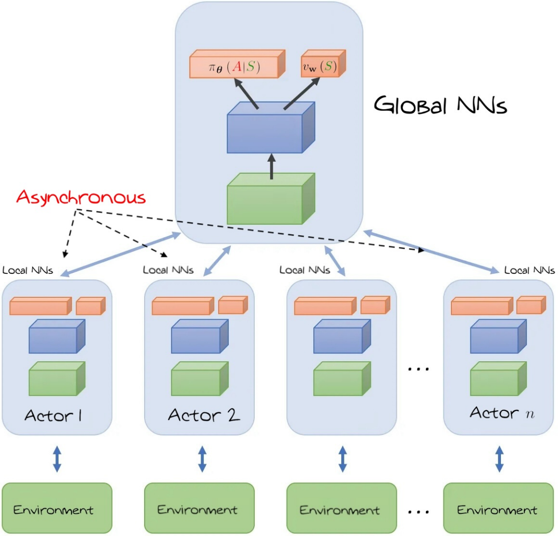
We can implement actor-critic approaches in a distributed fashion

- *Multiple actors few samples with local policy and value network*
  - ↳ *Each actor focuses on a specific part of environment*
- *They share their gradient estimators with the server*
- *Server treats the collected estimators as of a large mini-batch*
  - ↳ *It updates its networks on this large mini-batch*
- *All actors update their local networks every time server shares its networks*

We can implement this setting

- *Synchronous*
  - ↳ *This is basically the same as what we do in A2C, TRPO or PPO*
- *Asynchronous*
  - ↳ *Server does not wait for all actors to send their estimators*
  - ↳ *It uses what it has every couple of rounds and remaining in next rounds*

A3C: *Asynchronous A2C*



Some Final Remarks

It turns out that asynchronous update can negatively impact convergence

- *A3C is hence not really extended to other PGMs*
- *In practice we usually implement actor-critic approaches in synchronous distributed form*

+ *Is that it? Are we free to go now?!*
– *Pretty much Yes! Just we may further take a look at deterministic policy gradient approaches as well*
+ *Is it a new set of approaches?!*
– *No! It's a specific form of actor-critic methods that are better compatible with continuous actions*