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

Policy Gradient Theorem: *Point of Departure*

Policy Gradient Theorem

For a policy network with non-zero probabilities, the gradient of the average trajectory return is always given by

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

Policy gradient theorem is the base of

Policy Gradient Methods $\equiv$ *PGM*

It gives a feasible approach for training a policy network; however, depending on how we use it we can end up with various PGMs

PGMs in Nutshell

Policy Gradient Theorem

For a policy network with non-zero probabilities, the gradient of the average trajectory return is always given by

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

PGMs can roughly divided into three classes

1. *Vanilla PGM estimates* $q_{\pi_{\boldsymbol{\theta}}}(S, A)$ *and* $\nabla \log \pi_{\boldsymbol{\theta}}(A|S)$ *via sampling*
2. *Baseline PGM that reduces estimation variance by temporal unbiasing trick*
3. *Trust region PGM enables reuse of older samples to improve efficiency*
   - ↳ *This is what we learn in the next section of this chapter*

We are going through them in the same order!

Vanilla PGM: *Basic SGD*

Vanilla PGM is pretty straightforward: *sample environment with a trajectory and train policy network via SGD using result of policy gradient theorem*

- *Use SGD to update $\boldsymbol{\theta}$ in each time, i.e., update as $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla \mathcal{J}(\pi_{\boldsymbol{\theta}})$*
- *Compute gradient using policy gradient theorem*

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

- *Estimate the gradient via individual samples, i.e.,*

$$\hat{\nabla} \mathcal{J}(\pi_{\boldsymbol{\theta}}) = Q_t \nabla \log \pi_{\boldsymbol{\theta}}(A_t|S_t)$$

where Q_t is an estimator of action-value at pair (S_t, A_t) in sample

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

Vanilla PGM: *Generic Form*

```
VanillaPGM():
```

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

Use a ***Q-estimator*** `QEst()`

while *interacting* ***do***

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

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

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

        Set $Q_t = \mathtt{QEst}(S_t, A_t)$

        Update policy network $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha Q_t \nabla \log \pi_{\boldsymbol{\theta}} \left(A_t | S_t \right)$

    end for

end while

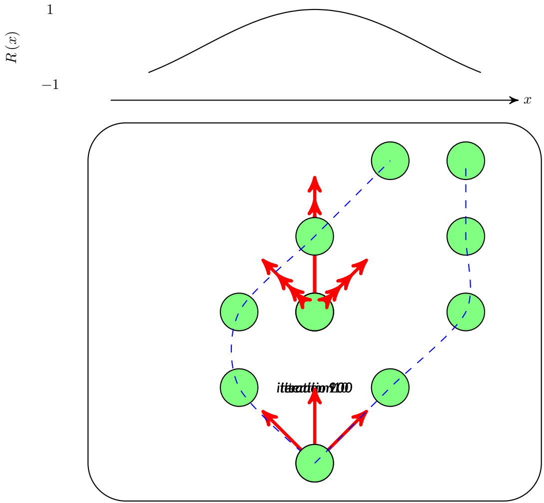
Revisiting REINFORCE

It is easy to see that `REINFORCE()` is a vanilla PGM: *here, we set estimator of action-value to be*

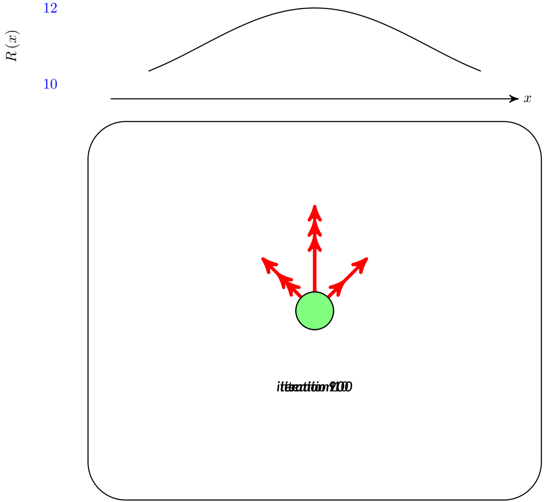
$$Q_t = G_t = \sum_{i=t}^{T-1} \gamma^i R_{i+1}$$

+ *But in our initial derivation, we saw derived G_0 instead of G_t!*
– *Well! If we replace in the policy gradient theorem, we could see that it would be still an estimator if we replace G_t with G_0*
+ *So, we have many estimators! How can we choose among them?!*
– *This is what we do in baseline PGM*
+ *What about using TD to estimate action-values?*
– *We could do it! But there will be a bit of complications. We will see it soon!*

Example: *Controlling Moving Particle – Case I*



Example: *Controlling Moving Particle – Case II*



Vanilla PGM: *Bias Issue*

This is a crucial observation: *with a simple shift in value, vanilla PGM slows significantly in convergence!*

`VanillaPGM():`

1: *Initiate with $\boldsymbol{\theta}$ and learning rate α*
2: *Use a* **Q-estimator** `QEst()`
3: ***while** interacting **do***
4: *Sample the environment with policy $\pi_{\boldsymbol{\theta}}$*

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

5: ***for** $t = 0 : T-1$ **do***
6: *Set* $Q_t = \texttt{QEst}(S_t, A_t)$
7: *Update as* $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \boxed{Q_t} \nabla \log \pi_{\boldsymbol{\theta}} (A_t | S_t)$ $\leftarrow$ here is the trouble
8: ***end for***
9: ***end while***

Vanilla PGM: *Bias Issue*

This is a crucial observation: *with a simple shift in value, vanilla PGM slows significantly in convergence!*

Let's look at this update rule

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha Q_t \nabla \log \pi_{\boldsymbol{\theta}} \left(A_t | S_t \right)$$

With larger values, Q_t becomes larger, hence

- *if $\nabla \log \pi_{\boldsymbol{\theta}} \left(A_t | S_t \right)$ becomes a small positive*
  - ↳ *$\boldsymbol{\theta}$ increases largely*
- *if $\nabla \log \pi_{\boldsymbol{\theta}} \left(A_t | S_t \right)$ becomes a small negative*
  - ↳ *$\boldsymbol{\theta}$ drops largely*

We need to have Q_t concentrated around zero

Vanilla PGM: *Bias Issue*

We need to have Q_t concentrated around zero

+ *But, wait a moment! We derived this expression from policy gradient theorem! If we change Q_t to something else, we are deviating from it!*
– Well! This is not necessarily true!

Let's try an experiment: in the gradient term given by policy gradient algorithm, we replace the action-value term with a shifted one, i.e., replace $q_{\pi_{\boldsymbol{\theta}}}(S, A)$ with

$$q'_{\pi_{\boldsymbol{\theta}}}(S, A) = q_{\pi_{\boldsymbol{\theta}}}(S, A) - u(S)$$

The term $u(S)$ can change with state, but it is fixed in terms of actions

Unbiasing Policy Gradient

Replacing $q'_{\pi_{\boldsymbol{\theta}}}(S, A)$ *into the gradient expression, we have*

$$\begin{aligned}\mathcal{E}(\pi_{\boldsymbol{\theta}}) &= \mathbb{E}_{S\sim d_{\pi_{\boldsymbol{\theta}}}, A|S\sim\pi_{\boldsymbol{\theta}}}\left\{q'_{\pi_{\boldsymbol{\theta}}}(S, A)\nabla\log\pi_{\boldsymbol{\theta}}(A|S)\right\} \\ &= \mathbb{E}_{S\sim d_{\pi_{\boldsymbol{\theta}}}, A|S\sim\pi_{\boldsymbol{\theta}}}\left\{(q_{\pi_{\boldsymbol{\theta}}}(S, A) - u(S))\nabla\log\pi_{\boldsymbol{\theta}}(A|S)\right\} \\ &= \mathbb{E}_{S\sim d_{\pi_{\boldsymbol{\theta}}}}\left\{\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left\{(q_{\pi_{\boldsymbol{\theta}}}(S, A) - u(S))\nabla\log\pi_{\boldsymbol{\theta}}(A|S)\,|S\right\}\right\} \\ &= \underbrace{\mathbb{E}_{S\sim d_{\pi_{\boldsymbol{\theta}}}}\left\{\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left\{q_{\pi_{\boldsymbol{\theta}}}(S, A)\nabla\log\pi_{\boldsymbol{\theta}}(A|S)\,|S\right\}\right\}}_{\nabla\mathcal{J}(\pi_{\boldsymbol{\theta}})} \\ &\qquad - \underbrace{\mathbb{E}_{S\sim d_{\pi_{\boldsymbol{\theta}}}}\left\{u(S)\,\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left\{\nabla\log\pi_{\boldsymbol{\theta}}(A|S)\,|S\right\}\right\}}_{?}\end{aligned}$$

So, we have

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

To compute the second term, we can use a simple trick

Gradient Averaging Trick

Assume $X \sim p_{\boldsymbol{\theta}}(x)$: X *is distributed by a distribution that is parameterized by some* $\boldsymbol{\theta}$*. We can then write*

$$\mathbb{E}_{p_{\boldsymbol{\theta}}}\left\{\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(X)\right\} = \mathbb{E}_{p_{\boldsymbol{\theta}}}\left\{\frac{\nabla p_{\boldsymbol{\theta}}(X)}{p_{\boldsymbol{\theta}}(X)}\right\} = \int_x \frac{\nabla p_{\boldsymbol{\theta}}(x)}{p_{\boldsymbol{\theta}}(x)} p_{\boldsymbol{\theta}}(x)$$

$$= \int_x \nabla p_{\boldsymbol{\theta}}(x) = \nabla \underbrace{\int_x p_{\boldsymbol{\theta}}(x)}_{1} = \nabla 1 = 0$$

Lemma: *Gradient Averaging*

For any parameterized distribution $p_{\boldsymbol{\theta}}(x)$*, we have*

$$\mathbb{E}_{p_{\boldsymbol{\theta}}}\left\{\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(X)\right\} = 0$$

Unbiasing Policy Gradient

This concludes that

$$
\begin{aligned}
\mathcal{E}\left(\pi_{\boldsymbol{\theta}}\right) &= \nabla \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right) - \mathbb{E}_{S \sim d_{\pi_{\boldsymbol{\theta}}}} \left\{ u\left(S\right) \underbrace{\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left\{\nabla \log \pi_{\boldsymbol{\theta}}\left(A|S\right)|S\right\}}_{0} \right\} \\
&= \nabla \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right)
\end{aligned}
$$

In other words: we can add any term that is fixed in terms of actions to our value estimator without any harm to policy gradient theorem

- *This fixed term is often called baseline*
- *It can improve convergence behavior*
- *It's something to be engineered in general*
  - ↳ *But no worries! We will see an obvious choice shortly* ☺

Policy Gradient Theorem with *Baseline*

Policy Gradient with Baseline

For a policy network with non-zero probabilities, the gradient of the average trajectory return is always given by

$$\nabla \mathcal{J}(\pi_{\boldsymbol{\theta}}) = \mathbb{E}_{S \sim d_{\pi_{\boldsymbol{\theta}}}, A|S \sim \pi_{\boldsymbol{\theta}}} \{ (\underbrace{q_{\pi_{\boldsymbol{\theta}}}}_{\text{Policy evaluation}} (S, A) - \overbrace{u}^{\text{Baseline}} (S)) \nabla \log \pi_{\boldsymbol{\theta}} (A|S) \}$$

for any ***baseline*** $u(\cdot)$

Baseline PGM: *General Form*

`BaselinePGM()`:

Initiate with $\boldsymbol{\theta}$ and learning rate α
Use a ***Q-estimator*** `QEst()`
while *interacting* ***do***
    Sample the environment with policy $\pi_{\boldsymbol{\theta}}$

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

    for $t = 0 : T - 1$ ***do***
        Set $Q_t = \texttt{QEst}(S_t, A_t)$
        Compute baseline estimator $B_t = u(S_t)$
        Update policy network $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha (Q_t - B_t) \nabla \log \pi_{\boldsymbol{\theta}} (A_t | S_t)$
    end for
end while

Value Function: *Obvious Choice of Baseline*

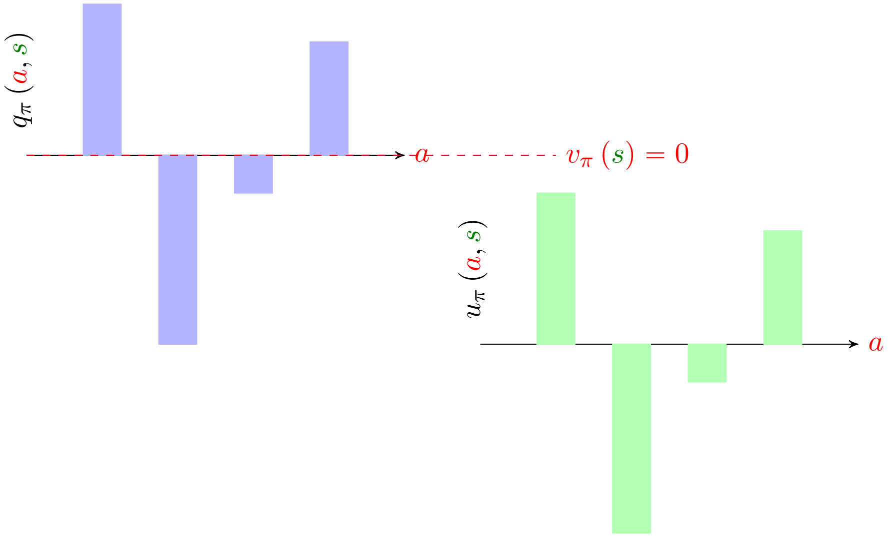
+ *What is the obvious choice of baseline?!*
– *Value function $v_{\pi_{\boldsymbol{\theta}}}(s)$!*
+ *How is it obvious?!*
– *In this case, shifted action-value represents the co-called advantage*

Advantage

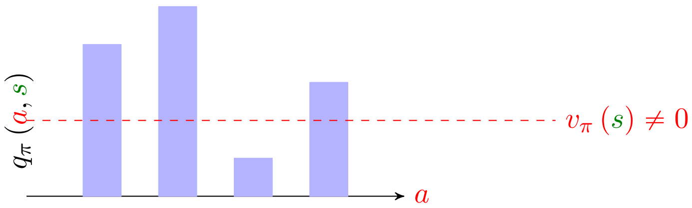
Given policy π, the advantage of action a at state s is defined as

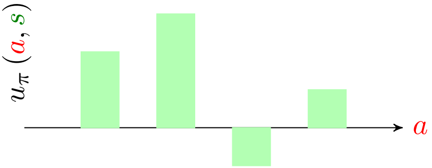
$$u_\pi(a, s) = q_\pi(a, s) - v_\pi(s)$$

Advantage: *Visualization*



Advantage: *Visualization*





Advantage at any state concentrates around zero

Baseline PGM: *Advantage Optimization*

Policy Gradient with Advantage

For a policy network with non-zero probabilities, the gradient of the average trajectory return is also given by

$$
\begin{aligned}
\nabla \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right) &= \mathbb{E}_{S \sim d_{\pi_{\boldsymbol{\theta}}}, A|S \sim \pi_{\boldsymbol{\theta}}} \left\{ u_{\pi_{\boldsymbol{\theta}}}\left(S, A\right) \nabla \log \pi_{\boldsymbol{\theta}}\left(A|S\right) \right\} \\
&= \mathbb{E}_{S \sim d_{\pi_{\boldsymbol{\theta}}}, A|S \sim \pi_{\boldsymbol{\theta}}} \left\{ \left(q_{\pi_{\boldsymbol{\theta}}}\left(S, A\right) - v_{\pi_{\boldsymbol{\theta}}}\left(S\right) \right) \nabla \log \pi_{\boldsymbol{\theta}}\left(A|S\right) \right\}
\end{aligned}
$$

+ *But how can we find an estimator for advantage?*
- *If we can estimate action-values, we can obviously use*

$$
v_{\pi_{\boldsymbol{\theta}}}\left(s\right) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left\{ q_{\pi_{\boldsymbol{\theta}}}\left(s, A\right) \right\} = \int_a q_{\pi_{\boldsymbol{\theta}}}\left(s, a\right) \pi_{\boldsymbol{\theta}}\left(a|s\right)
$$

Baseline PGM: *Advantage Optimization*

`AdvantagePGM():`

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

Use a ***Q-estimator*** `QEst()`

while *interacting* ***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***

    Set $Q_t = \texttt{QEst}(S_t, A_t)$

    Compute value $V_t = \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left\{ Q_t | S_t \right\}$ *and sample advantage* $U_t = Q_t - V_t$

    Update policy network $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha U_t \nabla \log \pi_{\boldsymbol{\theta}} \left(A_t | S_t \right)$

  end for

end while

PGM with Temporal Difference Estimate

+ *We have only considered Monte Carlo approach to estimate values! Why don't we use TD?!*
– Well! If we only work with a policy network, it could be challenging

---

Say we have a particular sample trajectory that looks at time t as

$$S_t, A_t \xrightarrow{R_{t+1}} S_{t+1}$$

If we use TD-0 to estimate $q_{\pi_{\boldsymbol{\theta}}}(S_t, A_t)$, we would write

$$\hat{q}_{\pi_{\boldsymbol{\theta}}}(S_t, A_t) = R_{t+1} + \gamma \hat{v}_{\pi_{\boldsymbol{\theta}}}(S_{t+1})$$

where do we get this estimate?!

Estimating via TD in PGM: *Main Challenge*

In value-based RL, we gradually find an estimate for values

- *In tabular RL, we make a Q-table and update it*
- *In deep RL, we train a value network, e.g., a DQN*

In pure PGM, we have neither of them!

+ *Then what can we do? We cannot always use Monte Carlo! What if the environment is not episodic?*
– *Well there are three solutions with only one of them working!*

Estimating via TD in PGM: *Possible Solutions*

1. *We may stick to Monte Carlo approach*
   - ↳ *It has high variance and is hard to use in non-episodic environments*
2. *We may try to evaluate the policy in each iteration*
   - ↳ *This is in practice computationally infeasible*
3. *We may train a value network in addition to the policy network*
   - ↳ *This describe the class of actor-critic methods*
     - ↳ *An actor who plays with the policy network and update it via PGM*
     - ↳ *A critic who evaluates the policy with a value network and update it with DQL*
   - ↳ *We will get to these methods in the next chapter*

For now, let's make an idealistic assumption: we assume that we can really evaluate a policy, i.e., given $\pi_{\boldsymbol{\theta}}$ for any $\boldsymbol{\theta}$

we can compute $v_{\pi_{\boldsymbol{\theta}}}(s)$ and $q_{\pi_{\boldsymbol{\theta}}}(s, A)$

We will later get rid of this idealistic assumption by the help of value networks

Estimating via TD in PGM: *Idealistic Case*

With this assumption, we can rewrite our algorithms in an online form, e.g.,

```
AdvantagePGM():
```

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

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

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

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

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

5: *Compute $q_{\pi_{\boldsymbol{\theta}}}(S_t, A_t)$ and $v_{\pi_{\boldsymbol{\theta}}}(S_t) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\{q_{\pi_{\boldsymbol{\theta}}}(S_t, A_t) | S_t\}$*

6: *Compute sample advantage $U_t = q_{\pi_{\boldsymbol{\theta}}}(S_t, A_t) - v_{\pi_{\boldsymbol{\theta}}}(S_t)$*

7: ***end for***

8: *Update policy network*

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \sum_{t=0}^{T-1} U_t \nabla \log \pi_{\boldsymbol{\theta}}(A_t | S_t)$$

9: ***end while***

TD Error as Advantage Estimator

$$S_t, A_t \xrightarrow{R_{t+1}} S_{t+1}$$

Let's look at the sample advantage: we have

$$U_t = q_{\pi_{\boldsymbol{\theta}}} \left(S_t, A_t \right) - v_{\pi_{\boldsymbol{\theta}}} \left(S_t \right)$$

Using Bellman's equation, we can write

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

which we can be estimated by

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

This is the TD-0 error: TD error is an estimator of advantage!

Advantage PGM: *Online via TD Estimate*

`AdvantagePGM():`

1: *Initiate with $\boldsymbol{\theta}$ and learning rate α*
2: ***while** interacting **do***
3: *Sample the environment with policy $\pi_{\boldsymbol{\theta}}$*

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

4: ***for** $t = 0 : T-1$ **do***
5: *Compute sample advantage $U_t = R_{t+1} + \gamma v_{\pi_{\boldsymbol{\theta}}}(S_{t+1}) - v_{\pi_{\boldsymbol{\theta}}}(S_t)$*
6: ***end for***
7: *Update policy network*

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \sum_{t=0}^{T-1} U_t \nabla \log \pi_{\boldsymbol{\theta}}(A_t | S_t)$$

8: ***end while***

+ *And, we do not need action-values!*
– Right! *Value function* is *enough*

Advantage PGM: *Online via TD Estimate*

Obviously, we can find a more robust estimator via TD-n

`AdvantagePGM`(n):

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

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

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

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

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

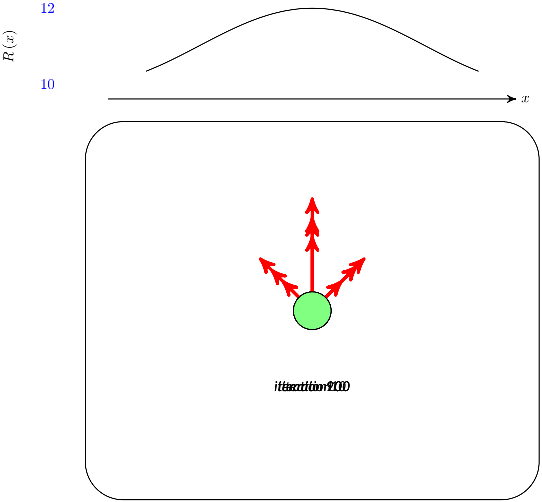
5: *Compute* $U_t = \sum_{i=0}^{n} \gamma^i R_{t+i+1} + \gamma^{n+1} v_{\pi_{\boldsymbol{\theta}}}\left(S_{t+n+1}\right) - v_{\pi_{\boldsymbol{\theta}}}\left(S_t\right)$

6: ***end for***

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \sum_{t=0}^{T-n-1} U_t \nabla \log \pi_{\boldsymbol{\theta}}\left(A_t | S_t\right)$$

7: ***end while***

Example: *Controlling Moving Particle – Case II*



Crucial Challenge in PGM: *Sample Inefficiency*

After implementing `AdvantagePGM()`, we see: *though using baseline, the variance reduces, it still needs long time to converge*

the main reason is that PGM is sample inefficient

In all above algorithms

- *We sample* $S_0, A_0 \xrightarrow{R_1} \cdots \xrightarrow{R_T} S_T$ *and use it for update*
- *We never get back to this sample*

This is generally a big challenge in PGMs!

+ *Can't we do what we did in DQL?!*
– *You mean experience reply?!*
+ *Right! Just keep previous samples in a buffer and reuse them again*
– *Well! The issue is that those samples were collected by other policies, i.e.,* $\pi_{\boldsymbol{\theta}}$ *for old* $\boldsymbol{\theta}$*. Through time, we have gone far away from them!*

Solution: *Trust Region PGM*

+ *So was it with DQL! How did we get rid of that?!*
– *We were playing off-policy, so we did not need to have sample with the target policy*
+ *So, isn't there any way to improve the sample efficiency?*
– *There is one and we do know it!*

The solution to this challenge is to use the idea of importance sampling: recall that if $X \sim p(x)$ we could write

$$\mathbb{E}_q\{X\} = \int_x xq(x) = \int_x x\frac{q(x)}{p(x)}p(x) = \mathbb{E}_p\left\{X\frac{q(X)}{p(X)}\right\}$$

This leads to PGMs with trust region that we will learn next!