POLICY SEARCH METHODS 2

Herke van Hoof

Big picture: How to learn policies

$$D = \{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)\}_{i=1 \ldots N}$$

Learn value function
$V(\mathbf{s}), Q(\mathbf{s}, \mathbf{a})$

Optimize policy
$\pi(\mathbf{a}|\mathbf{s})$

Learn dynamics model
$p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ $r(\mathbf{s}, \mathbf{a})$

Policy
$\pi(\mathbf{a}|\mathbf{s})$

Learn value or policy
$V(\mathbf{s}), Q(\mathbf{s}, \mathbf{a}), \pi(\mathbf{a}|\mathbf{s})$

Policy
$\pi(\mathbf{a}|\mathbf{s})$

Value-based

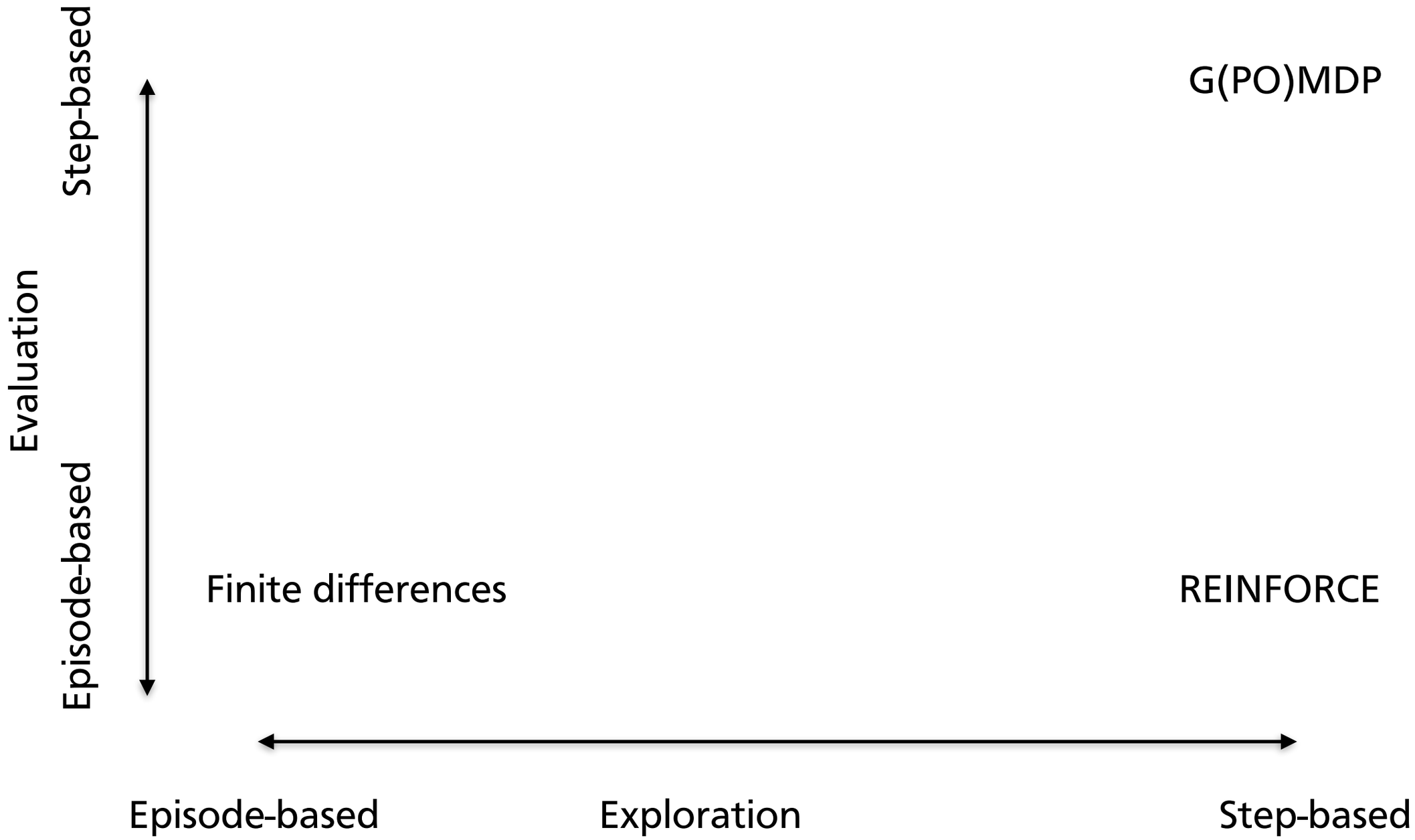
Policy-based

Model-free RL

Model-based RL

Thanks to Jan Peters

Big picture: How to learn policies



Policy Gradient Theorem

By shuffling the terms around, we can re-formulate Reinforce / G(PO)MDP

$$\mathbb{E}_\tau \left[\sum_{t=1}^{T} r_t \sum_{t'=1}^{t} \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) \right]$$

$$= \mathbb{E}_\tau \left[\sum_{t'=1}^{T} \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) \sum_{t=t'}^{T} r_t \right]$$

	a_1	a_2	a_3
r_1	→		
r_2	→	→	
r_3	→	→	→

	a_1	a_2	a_3
r_1	↓		
r_2	↓	↓	
r_3	↓	↓	↓

Policy Gradient Theorem

By shuffling the terms around, we can re-formulate Reinforce / G(PO)MDP

$$\mathbb{E}_\tau \left[\sum_{t=1}^{T} r_t \sum_{t'=1}^{t} \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) \right]$$

$$= \mathbb{E}_\tau \left[\sum_{t'=1}^{T} \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) \sum_{t=t'}^{T} r_t \right]$$

Replace return by its expected value

$$= \mathbb{E}_\tau \left[\sum_{t'=1}^{T} \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) q_\pi(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \right]$$

	a_1	a_2	a_3
r_1	→		
r_2	→	→	
r_3	→	→	→
			3
r_3	↓	↓	↓

value func q_pie is defined as (or should)
= Expectation of return.

sum of imm reward = true value
q_pi(s,a)=estimated value func, it brings some error

Policy Gradient Theorem

$$\nabla J = \mathbb{E}_\tau \left[\sum_{t'=1}^{T} \nabla \log \pi(\mathbf{a}_{t'} | \mathbf{s}_{t'}) q_\pi(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \right]$$

State-action pairs at each time step will contribute equally to the total gradient. So this is equivalent to an expectation over the on-pol

what is the advantage that we replace true reward as estimated reward: it is good for continuing case: when the episode is really long, when the episode does not terminate.

$$\nabla J = \mathbb{E}_{\mu(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})} \nabla \log \pi(\mathbf{a}|\mathbf{s}) q_\pi(\mathbf{s}, \mathbf{a})$$

Formal proof, also for the continuing case:

Sutton et al., Policy Gradient Methods for Reinforcement Learning with Function Approximation

PGT Actor Critic

has bias
reduce the variance
compared with actor only policy based method

$$\nabla J = \mathbb{E}_{\mu(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})} \nabla \log \pi(\mathbf{a}|\mathbf{s}) q_\pi(\mathbf{s}, \mathbf{a})$$

Maybe we can replace q_π by an estimate (q_w or $v_w(s_{t+1}) + r_{t+1}$)

Plug in samples to estimate the expected value

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \left(R_{t+1} + \gamma \hat{v} \left(S_{t+1}, \mathbf{w} \right) \right) \nabla \log \pi \left(\mathbf{a}_t \mid \mathbf{s}_t, \boldsymbol{\theta}_t \right)$$

Advantage: break the high variance of Monte-Carlo returns

Disadvantage: possibly include bias?

PGT Actor Critic

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \left(R_{t+1} + \gamma \hat{v} \left(S_{t+1}, \mathbf{w} \right) \right) \nabla \log \pi \left(\mathbf{a}_t \mid \mathbf{s}_t, \boldsymbol{\theta}_t \right)$$

This algorithm has a parametrised policy *and* a parametrised action-value function, with parameters θ and *w,* respectively.

The policy is often called an actor while the value function is called critic. A method that uses both is an *actor-critic* method.

PGT Actor Critic

it is biased in this case. because it always converge to the target. the target is an estimated return, not true return, so it is always biased. but it has low variance. because we dont do MC

Note the difference between a critic and a baseline

Reinforce w baseline $$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \left(G_t - \hat{v}\right) (\mathbf{s}_t, \mathbf{w}) \nabla \log \pi \left(\mathbf{a}_t \mid \mathbf{s}_t, \boldsymbol{\theta}_t\right)$$

(Removes variance by centering targets, always unbiased)

Actor-Critic $$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \left(R_{t+1} + \gamma \hat{v} \left(\mathbf{s}_{t+1}, \mathbf{w}\right)\right) \nabla \log \pi \left(\mathbf{a}_t \mid \mathbf{s}_t, \boldsymbol{\theta}_t\right)$$

(Removes variance by removing long-term dependencies. Bias?)

Actor-Critic + baseline

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \left(R_{t+1} + \gamma \hat{v} \left(\mathbf{s}_{t+1}, \mathbf{w}\right) - \hat{v} \left(\mathbf{s}_t, \mathbf{w}\right)\right) \nabla \log \pi \left(\mathbf{a}_t \mid \mathbf{s}_t, \boldsymbol{\theta}_t\right)$$
$$= \delta$$

TD error = delta

PGT Actor critic

One-step Actor–Critic (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
  Initialize S (first state of episode)
  $I \leftarrow 1$
  Loop while S is not terminal (for each time step):
    $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S, \boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

With baseline!

If we discount, future time steps less important

Compatible function approximation thm.

Disadvantage of actor-critic: possibly include bias?

Question: are there functions q_π that give an unbiased estimate?

$$\nabla J = \mathbb{E}_{\mu(\mathbf{s}),a}\left[\nabla \log \pi(\mathbf{a}_{t'}|\mathbf{s}_{t'}) q_\pi(\mathbf{s}_{t'}, \mathbf{a}_{t'})\right] \stackrel{?}{=} \mathbb{E}_{\mu(\mathbf{s}),a}\left[\nabla \log \pi(\mathbf{a}_{t'}|\mathbf{s}_{t'}) \hat{q}_{\mathbf{w}}(\mathbf{s}_{t'}, \mathbf{a}_{t'})\right]$$

[Sutton 2000]

Compatible function approximation thm.

Disadvantage of actor-critic: possibly include bias?

Question: are there functions q_π that give an unbiased estimate?

$$\nabla J = \mathbb{E}_{\mu(\mathbf{s}),a}\left[\nabla \log \pi(\mathbf{a}_{t'}|\mathbf{s}_{t'}) q_\pi(\mathbf{s}_{t'}, \mathbf{a}_{t'})\right] \stackrel{?}{=} \mathbb{E}_{\mu(\mathbf{s}),a}\left[\nabla \log \pi(\mathbf{a}_{t'}|\mathbf{s}_{t'}) \hat{q}_{\mathbf{w}}(\mathbf{s}_{t'}, \mathbf{a}_{t'})\right]$$

Answer: Yes, if value function is 'compatible':

$$\nabla_{\mathbf{w}} \hat{q}_{\mathbf{w}} = \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) \quad \text{e.g. } \hat{q}_{\mathbf{w}} = \mathbf{w}^T \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$$

and

$$\mathbb{E}_{\mathbf{s}\sim\mu, \mathbf{a}\sim\pi(\cdot,\mathbf{s})}\left[\left(q_\pi(\mathbf{s}, \mathbf{a}) - \hat{q}_{\mathbf{w}}(\mathbf{s}, \mathbf{a})\right) \nabla_{\mathbf{w}} \hat{q}_{\mathbf{w}}(\mathbf{s}, \mathbf{a})\right] = 0$$

$$\left(\text{e.g. } \mathbf{w} = \arg\min_{\mathbf{w}} \mathbb{E}_{\mathbf{s}\sim\mu, \mathbf{a}\sim\pi(\cdot,\mathbf{s})}\left(q_\pi(\mathbf{s}, \mathbf{a}) - \hat{q}_{\mathbf{w}}(\mathbf{s}, \mathbf{a})\right)^2\right)$$

Compatible function approximation

Proof:

1) $\nabla_{\mathbf{w}} \hat{q}_{\mathbf{w}} = \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$

2) $\mathbb{E}_{\mathbf{s} \sim \mu, \mathbf{a} \sim \pi(\cdot, \mathbf{s})} \left[(q_\pi(\mathbf{s}, \mathbf{a}) - \hat{q}_{\mathbf{w}}(\mathbf{s}, \mathbf{a})) \nabla_{\mathbf{w}} \hat{q}_{\mathbf{w}}(\mathbf{s}, \mathbf{a}) \right] = 0$

Substitute (1) into (2)

$\mathbb{E}_{\mu(\mathbf{s}), \mathbf{a}} \left[(q_\pi(\mathbf{s}, \mathbf{a}) - \hat{q}_{\mathbf{w}}(\mathbf{s}, \mathbf{a})) \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{a}|\mathbf{s}) \right] = 0$

[Sutton 2000]

Compatible function approximation

Proof:

1) $\nabla_{\mathbf{w}} \hat{q}_{\mathbf{w}} = \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$

2) $\mathbb{E}_{\mathbf{s}\sim\mu, \mathbf{a}\sim\pi(\cdot,\mathbf{s})} \left[(q_\pi(\mathbf{s}, \mathbf{a}) - \hat{q}_{\mathbf{w}}(\mathbf{s}, \mathbf{a})) \nabla_{\mathbf{w}} \hat{q}_{\mathbf{w}}(\mathbf{s}, \mathbf{a}) \right] = 0$

Substitute (1) into (2)

$$\mathbb{E}_{\mu(\mathbf{s}),\mathbf{a}} \left[(q_\pi(\mathbf{s}, \mathbf{a}) - \hat{q}_{\mathbf{w}}(\mathbf{s}, \mathbf{a})) \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{a}|\mathbf{s}) \right] = 0$$

$$\mathbb{E}_{\mu(\mathbf{s}),\mathbf{a}} \left[q_\pi(\mathbf{s}, \mathbf{a}) \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{a}|\mathbf{s}) \right] = \mathbb{E}_{\mu(\mathbf{s}),\mathbf{a}} \left[\hat{q}_{\mathbf{w}}(\mathbf{s}, \mathbf{a}) \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{a}|\mathbf{s}) \right]$$

$$= \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Again yielding a **unbiased** and **consistent** estimate

This can again be combined with a **baseline**

Compatible function approximation

Compatibility:

$$\nabla_{\mathbf{w}} \hat{q}_{\mathbf{w}} = \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) \quad \text{e.g. } \hat{q}_{\mathbf{w}} = \mathbf{w}^T \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$$

Example: Softmax policy $\pi(a|s) = \dfrac{e^{\theta^T \phi_{sa}}}{\sum_b e^{\theta^T \phi_{sb}}}$

Compatible function approximation

Compatibility:

$$\nabla_{\mathbf{w}} \hat{q}_{\mathbf{w}} = \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) \quad \text{e.g. } \hat{q}_{\mathbf{w}} = \mathbf{w}^T \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$$

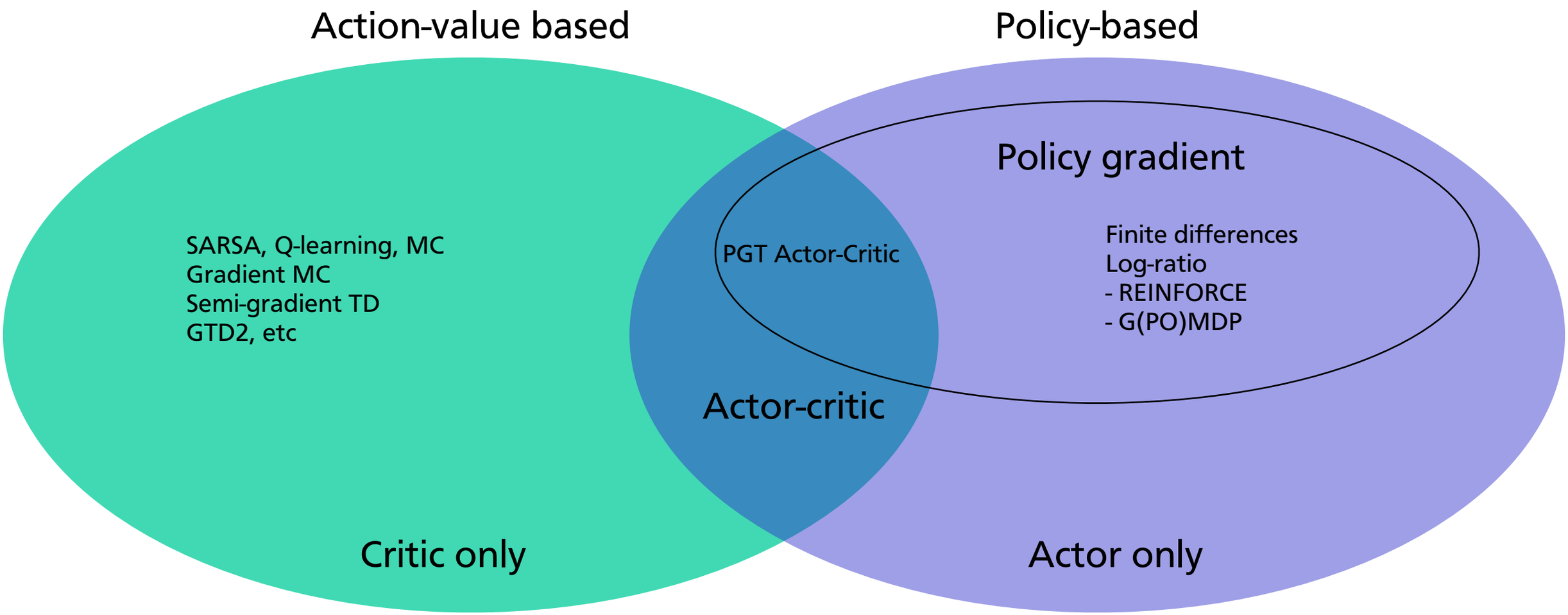
Example: Softmax policy $\pi(a|s) = \dfrac{e^{\theta^T \phi_{sa}}}{\sum_b e^{\theta^T \phi_{sb}}}$

$$\nabla \log \frac{e^{\theta^T \phi_{sa}}}{\sum_b e^{\theta^T \phi_{sb}}} = \nabla \theta^T \phi_{sa} - \nabla \log \sum_b e^{\theta^T \phi_{sb}}$$

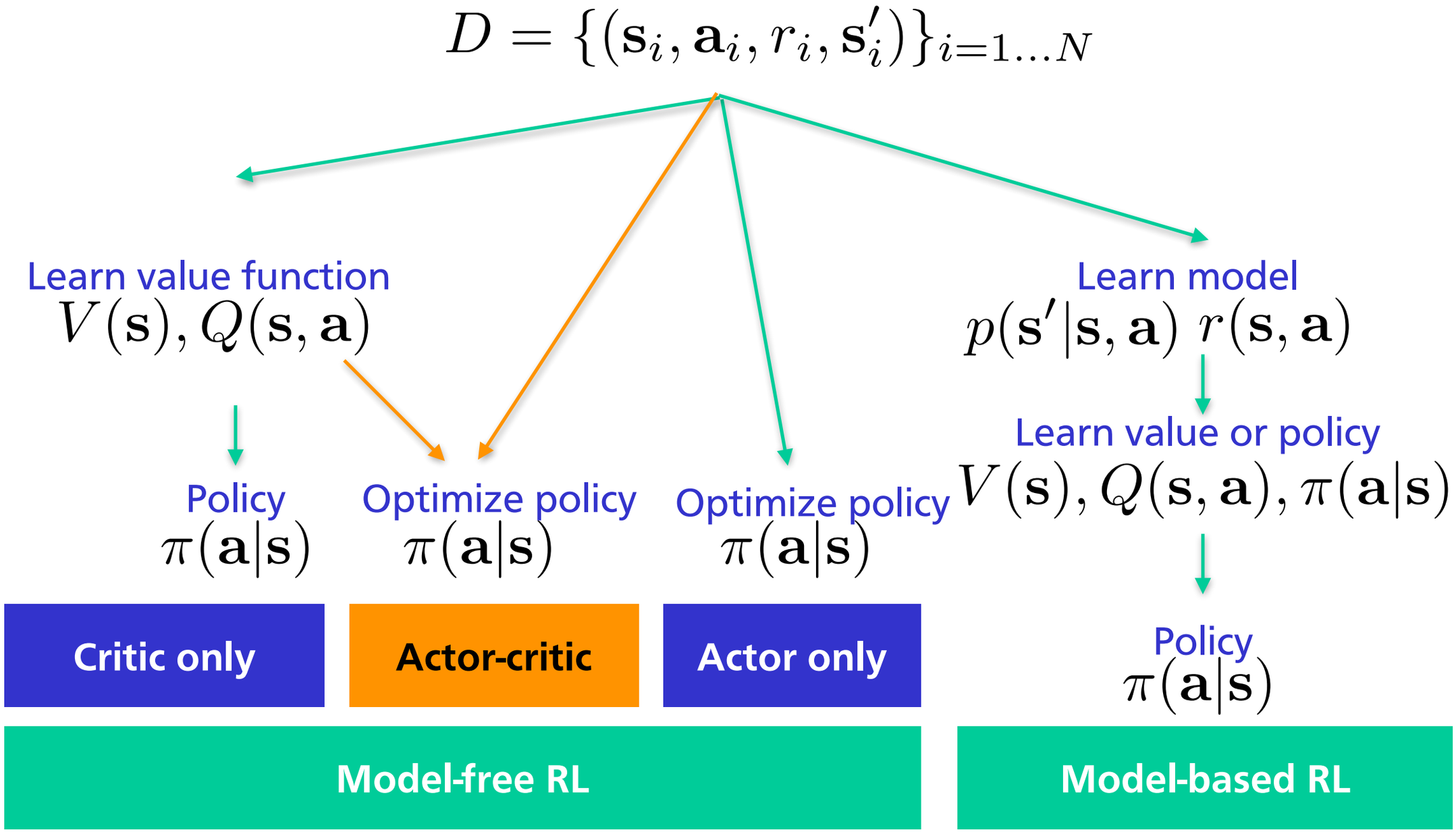
$$= \phi_{sa} - \frac{1}{\sum_b e^{\theta^T \phi_{sb}}} \sum_b e^{\theta^T \phi_{sb}} \phi_{sb} \quad = \phi_{sa} - \sum_b \underbrace{\frac{e^{\theta^T \phi_{sb}}}{\sum_b e^{\theta^T \phi_{sb}}}}_{=\pi(b|s)} \phi_{sb}$$

$$\hat{q}_{\mathbf{w}}(s, a) = \mathbf{w}^T \left(\phi_{sa} - \sum_b \pi(b|s) \phi_{sb} \right)$$

Policies and action-values

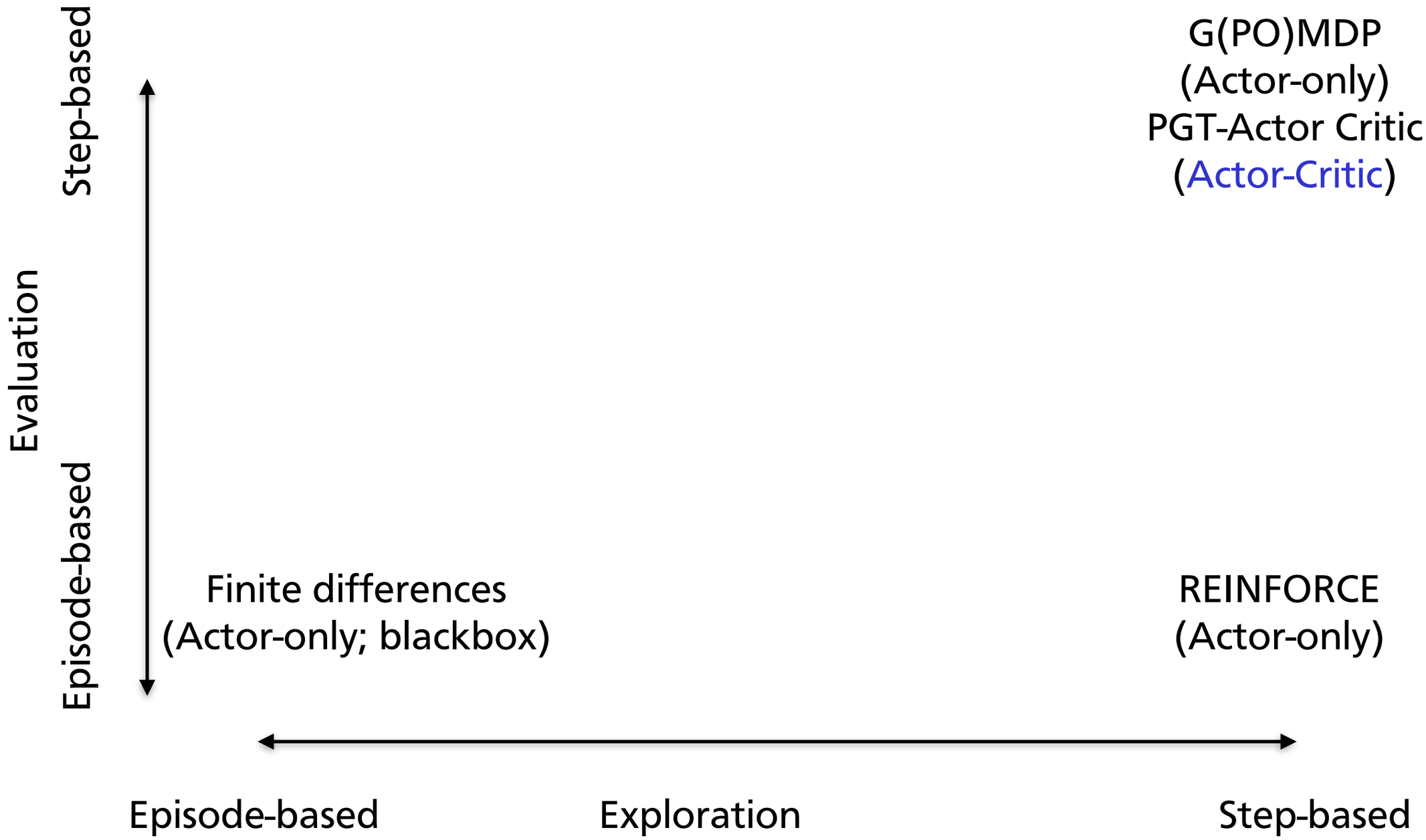


Big picture: How to learn policies



Thanks to Jan Peters

Comparison


Step-based
Evaluation
Episode-based
G(PO)MDP
(Actor-only)
PGT-Actor Critic
(Actor-Critic)
Finite differences
(Actor-only; blackbox)
REINFORCE
(Actor-only)
Episode-based
Exploration
Step-based

Before we look at more advanced policy search methods, let’s look at when we can use policy search methods

Advantages of policy-based methods

Policy search methods typically preferred in any of the below cases:

Problems with continuous actions

If we need to learn stochastic policies

If we have prior knowledge about the type of policy

If it is important to have ‘small’ policy updates between subsequent time steps

Advantages of policy-based methods

Policy search methods typically preferred in any of the below cases:

Problems with continuous actions

If we need to learn stochastic policies

If we have prior knowledge about the type of policy

If it is important to have ‘small’ policy updates between subsequent time steps

Many of these aspects present with physical systems like robots…

Actor critic addresses some weaknesses

Actor-only methods have high variance from Monte-Carlo

- Actor-critic lowers variance using critic

A lot of the methods we discussed are specific to episodic setting

- Actor-critic can be formulated for continuing setting

Actor-critic can be ‘fiddly’, many moving parts

Requires stochastic policies, what if deterministic is optimal?

- If amount of randomness is learned, can get close to deterministic
- We will also see a policy gradient method to learn deterministic policies

A CLOSER LOOK AT TARGET VALUES

- The target values are the part that differs most between policy gradient methods
- 'Vanilla' policy gradients can be written as:

$$g = \mathbb{E}\left[\sum_{t=0}^{\infty} \Psi_t \nabla_\theta \log \pi_\theta \left(a_t | s_t\right)\right]$$

- With different choices for $\boldsymbol{\Psi}$, e.g:
  - G(PO)MPD: $\Psi_t = \sum_{t'=t}^{\infty} r_{t'}$
  - PGT-AC: $\Psi_t = q_\pi(\mathbf{s}_t, \mathbf{a}_t)$

A CLOSER LOOK AT TARGET VALUES

- The target values are the part that differs most between policy gradient methods
- 'Vanilla' policy gradients can be written as:

$$g = \mathbb{E}\left[\sum_{t=0}^{\infty} \Psi_t \nabla_\theta \log \pi_\theta\left(a_t|s_t\right)\right]$$

- With different choices for $\boldsymbol{\Psi}$, e.g:
  - G(PO)MPD: $\Psi_t = \sum_{t'=t}^{\infty} r_{t'}$
  - PGT-AC: $\Psi_t = q_\pi(\mathbf{s}_t, \mathbf{a}_t)$

Less variance with value function baseline:

$$\Psi_t = \sum_{t'=t}^{\infty} r_{t'} - V(\mathbf{s}_t)$$

$$\Psi_t = q_\pi(\mathbf{s}_t, \mathbf{a}_t) - V(\mathbf{s}_t)$$

A CLOSER LOOK AT TARGET VALUES

- Different target values have their own problems:
- $$\Psi_t = \sum_{t'=t}^{\infty} r_{t'} - V(\mathbf{s}_t)$$

  Although the baseline reduces variance somewhat, REINFORCE still has a very high variance
- $$\Psi_t = q_\pi(\mathbf{s}_t, \mathbf{a}_t) - V(\mathbf{s}_t)$$

  An approximation of the q-function might introduce bias (except when using compatible functions and Q has converged)

- What does this remind you of?

REMEMBER N-STEP METHODS?

- We can also use n-step returns!
- Let's first define the target value as the **advantage**

$$A(\mathbf{s}_t, \mathbf{a}_t) = Q(\mathbf{s}_t, \mathbf{a}_t) - V(\mathbf{s}_t)$$

advanced estimator of target reward
= sum over TD errors
=substract baseline + TD(n) estimator

- Now an n-step advantage estimator is given by

$$\hat{A}_t^n = \sum_{l=0}^{n-1} \delta_{t+l} = -V(\mathbf{s_t}) + r_t + \gamma r_{t+1} + \cdots + \gamma^{n-1} r_{r+n-1} + \gamma^n V(s_{t+n})$$

if n is large, then we update more states in the past. but variance is higher
if n is small, then reward propagate slower. but lower variance

????? 1h 29min

- Again (like with n-
- High-n: update more states +*variance*
- Low-n: only 1 step back-up, bad when V is bad +*bias*

REMEMBER N-STEP METHODS?

- Rather than choosing 1 value of n, we can combine many:

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} := (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda\hat{A}_t^{(2)} + \lambda^2\hat{A}_t^{(3)} + \ldots\right)$$

Why 1-λ?

REMEMBER N-STEP METHODS?

- Rather than choosing 1 value of n, we can combine many:

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} := (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \ldots\right)$$

$$=$$

$$=$$

$$=$$

$$=$$

REMEMBER N-STEP METHODS?

- Rather than choosing 1 value of n, we can combine many:

$$
\begin{aligned}
\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} &:= (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \ldots\right) \\
&= (1-\lambda)\left(\delta_t^V + \lambda\left(\delta_t^V + \gamma\delta_{t+1}^V\right) + \lambda^2\left(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V\right) + \ldots\right) \\
&= (1-\lambda)\left(\delta_t^V\left(1 + \lambda + \lambda^2 + \ldots\right) + \gamma\delta_{t+1}^V\left(\lambda + \lambda^2 + \lambda^3 + \ldots\right)\right. \\
&\quad \left. + \gamma^2\delta_{t+2}^V\left(\lambda^2 + \lambda^3 + \lambda^4 + \ldots\right) + \ldots\right) \\
&= (1-\lambda)\left(\delta_t^V\left(\frac{1}{1-\lambda}\right) + \gamma\delta_{t+1}^V\left(\frac{\lambda}{1-\lambda}\right) + \gamma^2\delta_{t+2}^V\left(\frac{\lambda^2}{1-\lambda}\right) + \cdots\right) \\
&= \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V
\end{aligned}
$$

skip推导过程。重要的是结果：sum of discounted TD errors
gamma: discounting factor

(Generalized Advantage Estimages, Schulman [2016])

REMEMBER N-STEP METHODS?

- Sidenote:
- A similar trick can also be used to learn value functions
- In that case, the trick is called TD(λ)
- 1-step TD is equal to TD(λ) when choosing $\lambda=0$
  (in that case longer returns have a weight of 0).
  1-step TD is also called TD(0)
- TD(λ) can be implemented without storing the whole episode by using a trick called “eligibility traces”.
  (won’t be covered in this course…)

this trace is not covered in this course

REMEMBER N-STEP METHODS?

- 2 special cases:

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} := (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \ldots\right) = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V$$

- $\mathrm{GAE}(\gamma, 0)$:

- $\mathrm{GAE}(\gamma, 1)$:

REMEMBER N-STEP METHODS?

- 2 special cases:

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} := (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda\hat{A}_t^{(2)} + \lambda^2\hat{A}_t^{(3)} + \ldots\right) = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^V$$

- $\mathrm{GAE}(\gamma, 0): \quad \hat{A}_t := \delta_t \quad = r_t + \gamma V(s_{t+1}) - V(s_t)$

$\mathrm{GAE}(\gamma, 1): \quad \hat{A}_t := \sum_{l=0}^{\infty}\gamma^l\delta_{t+l} = \sum_{l=0}^{\infty}\gamma^l r_{t+l} - V(s_t)$

GENERALIZED ADVANTAGE ESTIMATION

- Notice that we were writing GAE as a function of λ *and* γ?
- Normally γ is specified by the problem, and it defines how much we really care about getting a reward now or later
- But we can plan with a smaller γ, treating it as a hyperparameter
  (we thus have a 'problem γ' and a 'solution γ')
- What do you think is the effect on bias? On variance?

GENERALIZED ADVANTAGE ESTIMATION

Initialize policy parameter θ_0 and value function parameter ϕ_0.
for $i = 0, 1, 2, \ldots$ **do**
  Simulate current policy π_{θ_i} until N timesteps are obtained.
  Compute δ_t^V at all timesteps $t \in \{1, 2, \ldots, N\}$, using $V = V_{\phi_i}$.
  Compute $\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V$ at all timesteps.
  Compute θ_{i+1} with TRPO update, Equation (31).
  Compute ϕ_{i+1} with Equation (30).
end for

could use any value learning algorithm, here a new method that uses trust regions (not covered)

could use any actor-critic algorithm, here: TRPO (see next week!)

This implementation requires full episodes (disadvantages?)

Figure from Schulman [2016]

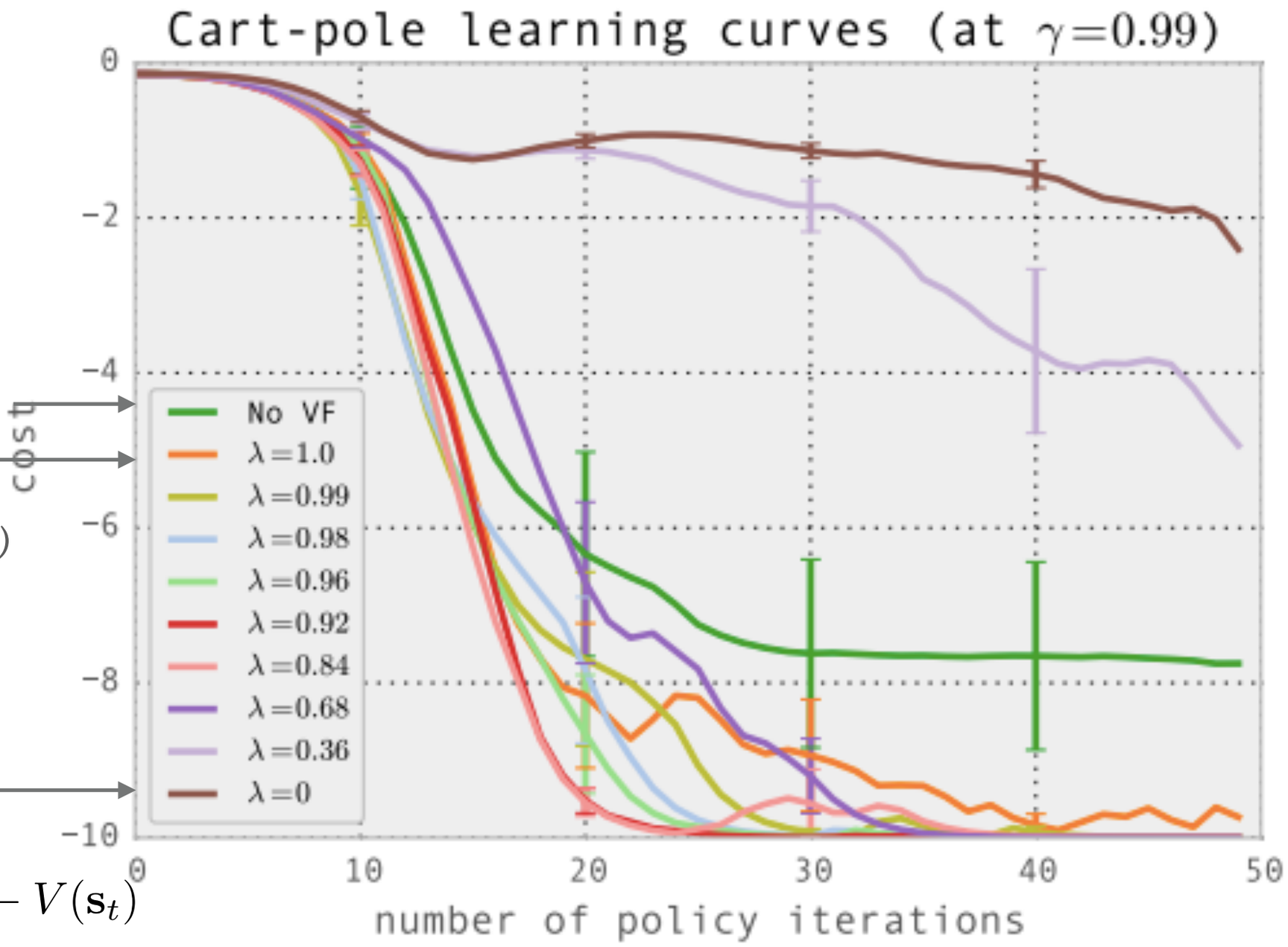
RESULTS: DIFFERENT λ

Target:

MC target + avg baseline

MC target + V baseline

(MC = G(PO)MDP target)

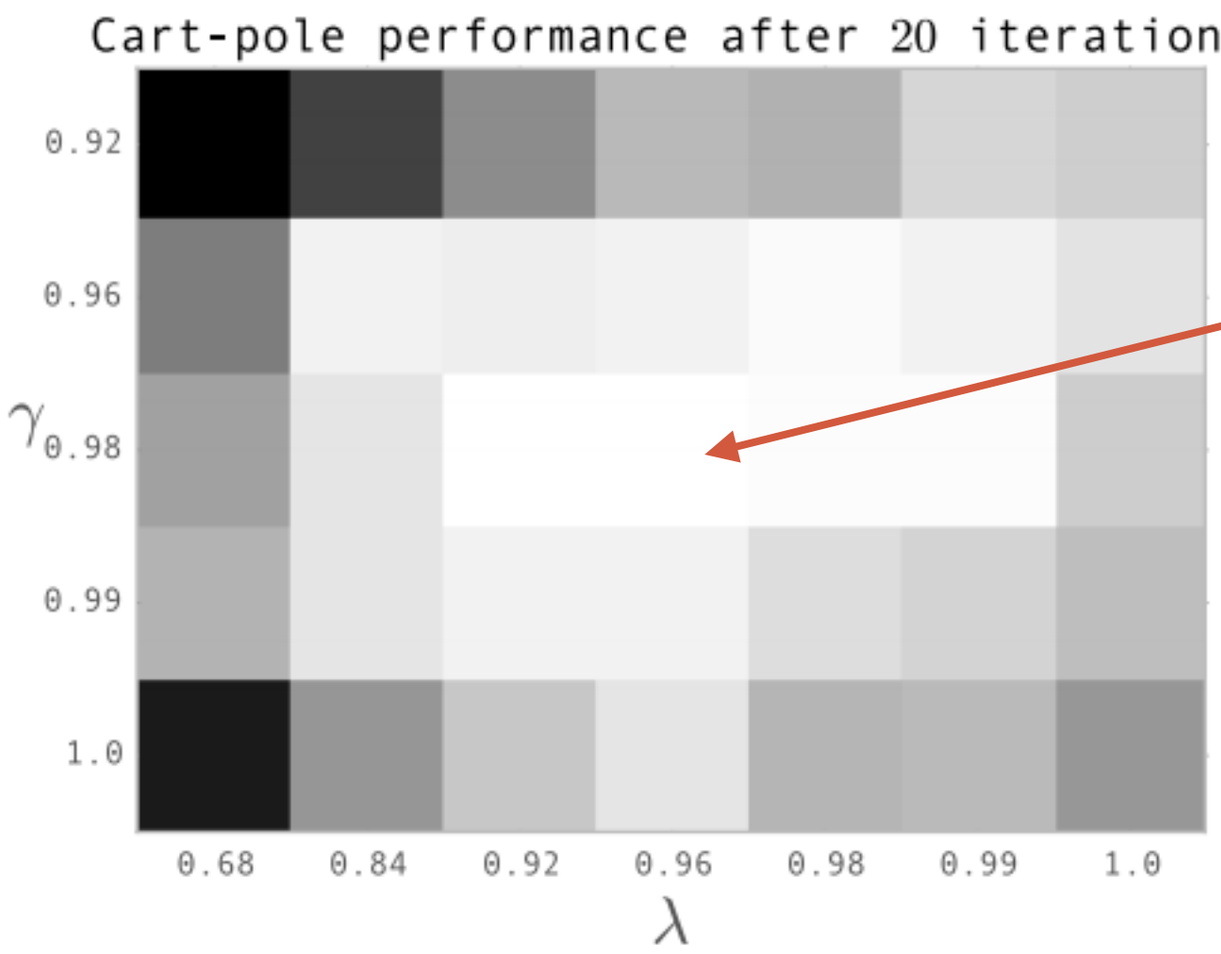


Actor-critic with baseline

$$\Psi_t = \gamma V(\mathbf{s}_{t+1}) + r_t - V(\mathbf{s}_t)$$

Figure from Schulman [2016]

RESULTS: DIFFERENT λ AND γ



best performance
(on undiscounted problem)

More bootstrapping

MC target

GENERALIZED ADVANTAGE ESTIMATION

- Generalized advantage estimation (GAE) can be used with many policy gradient methods
- E.g. well-known deep-RL method `A3C' [Mnih, 2016] uses an 'advantage actor critic' with policy update proportional to:

$$\theta_{t+1} \leftarrow \theta_t + \alpha A_t \nabla_\theta \log \pi_\theta \left(a_t | s_t\right)$$

- This is often used with GAE

CONCLUSIONS ABOUT GAE

- GAE can interpolate between:
  - MC (∞-step return) used by REINFORCE
  - TD (1-step return) used by PGT-AC
- n-step methods are comparable, but GAE averages over many different n
- Can be tuned to problem at hand using λ and γ
- Can yield huge improvement compared to either TD or MC
- Can be used with many policy-based algorithms

Thanks for your attention!

Feedback?

h.c.vanhoof@uva.nl

REFERENCES

- [Schulman 2016] Schulman, J., Moritz, P., Levine, S., Jordan, M.I. and Abbeel, P. High-dimensional continuous control using generalised advantage estimates. In ICLR.