Fraunhofer IIS

Fraunhofer-Institut für Integrierte Schaltungen IIS

Reinforcement Learning

—

Exercise 6: Policy Approximation

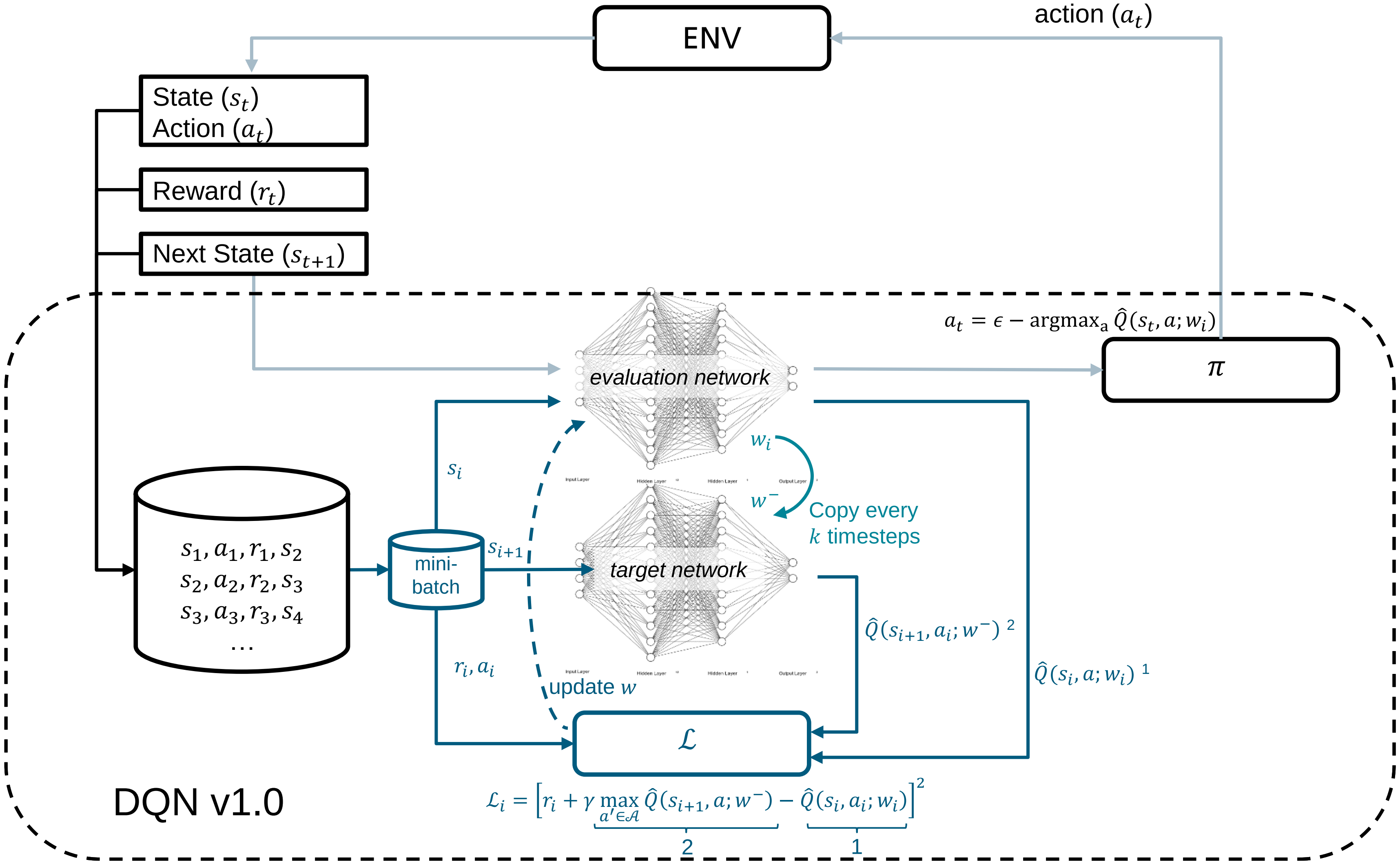
11.06.2024

Alexander Mattick

Exercise Sheet 6

Value Function Approximation

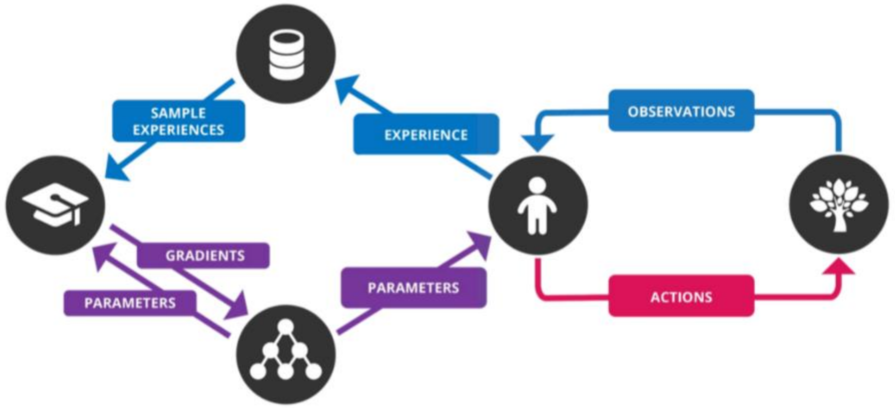
Deep Q-Networks (DQNs)



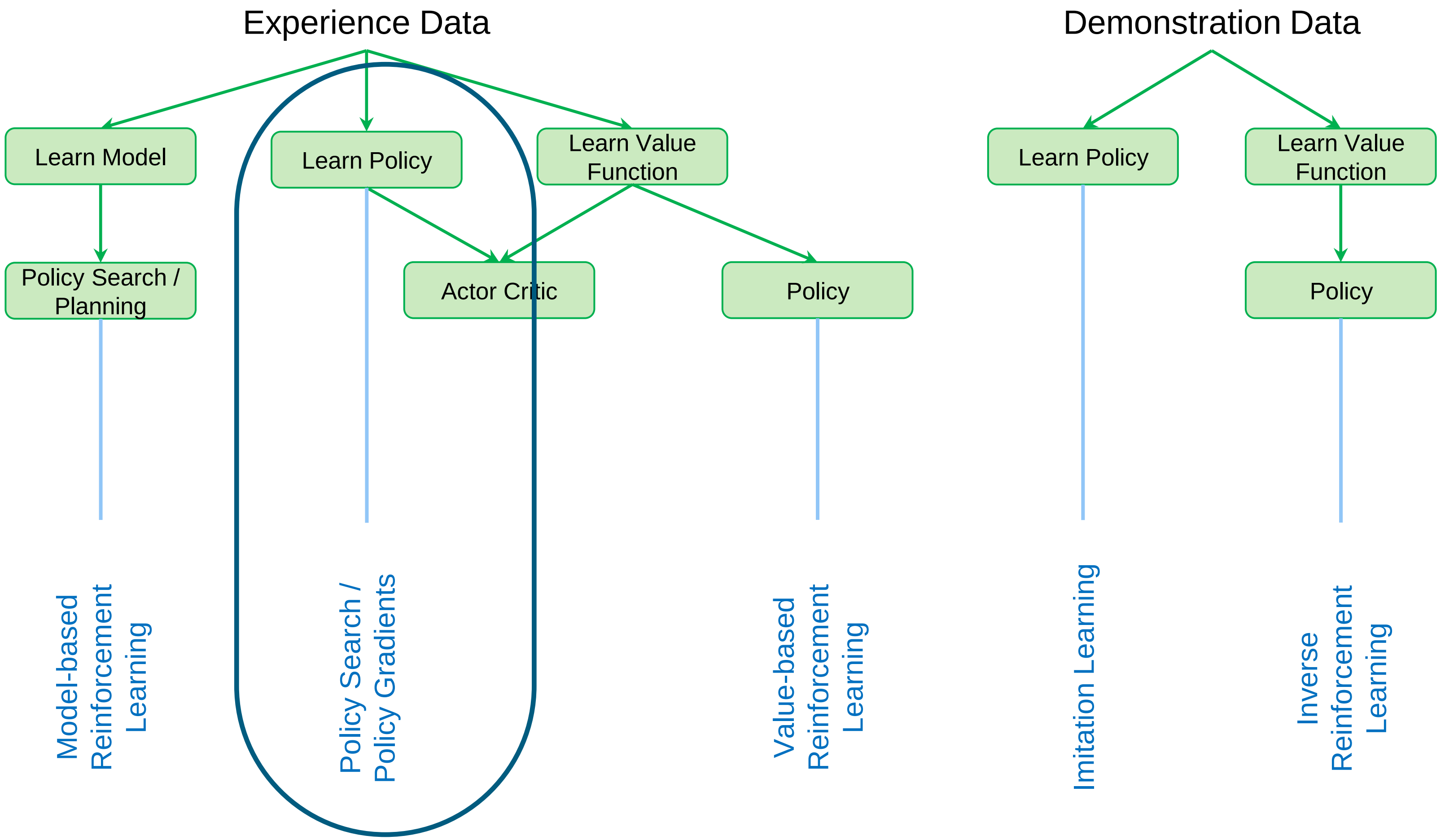
Deep Q Networks

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode $= 1, M$ **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \operatorname{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For



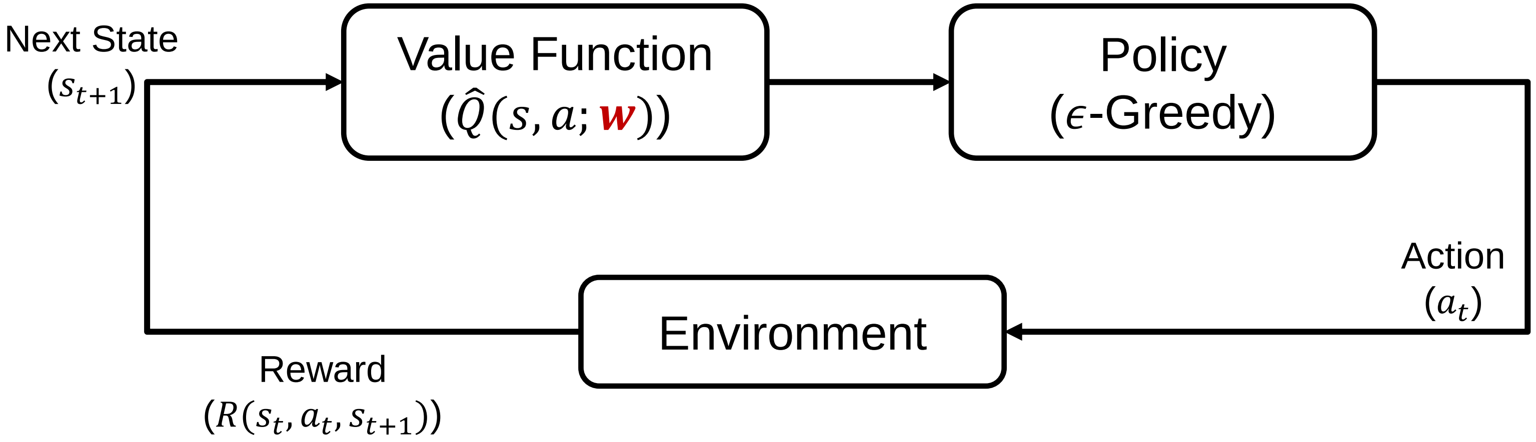
Policy-based Reinforcement Learning



Policy-based Reinforcement Learning

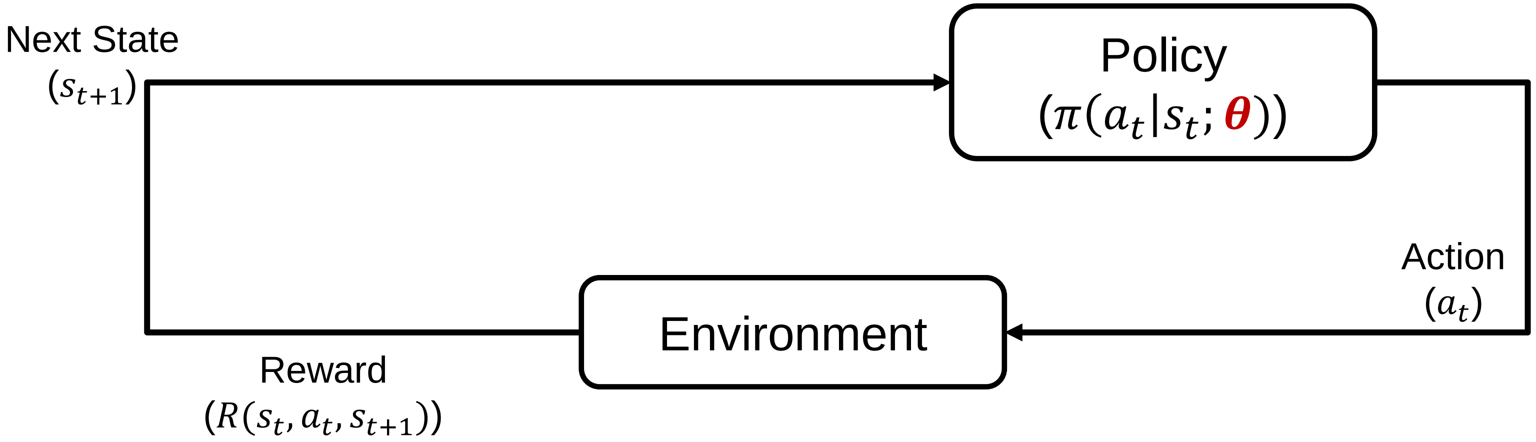
Change of pipeline

So far:



Goal: find w that approximates the true Q-function

Now:



Goal: find θ that maximizes long term reward

Policy-based Reinforcement Learning

Advantages and Disadvantages

Advantages:

- **Good convergence properties**
- Easily extended to high-dimensional or continuous state and action spaces
- Can learn ***stochastic* policies**
- Sometimes policies are simple while values and models are complex
  - e.g., rich domain, but optimal is always to go left

Disadvantages:

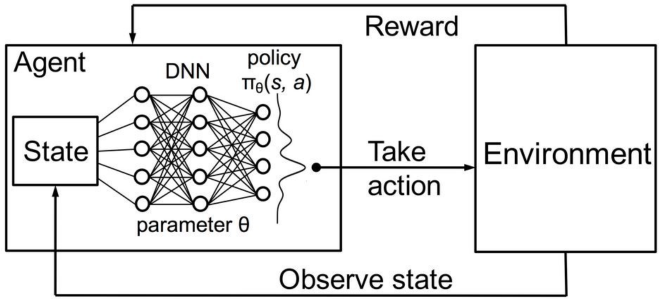
- Susceptible to local optima (especially with non-linear FA)
- Obtained knowledge is specific, does not always generalize well
- Ignores a lot of information in the data (when used in isolation)

Policy-based Reinforcement Learning

Stochastic policies

We have seen deterministic policies like this:

State gives $Q(s, a; w)$ and we selected $\pi(a|s)$ by $\operatorname{argmax}_a Q(s, a; w)$

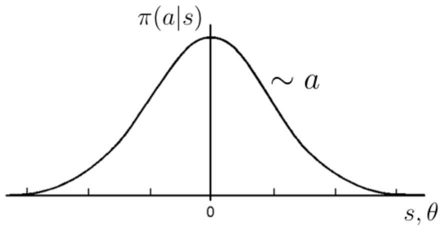


https://towardsdatascience.com/self-learning-ai-agents-iv-stochastic-policy-gradients-b53f088fce20

Instead, stochastic policies do something like this:

$$\pi(a|s) = \mathbb{P}[a|s; \theta]$$

(policy is represented as a probability distribution)



- optimal policy might be stochastic

Policy-based Reinforcement Learning

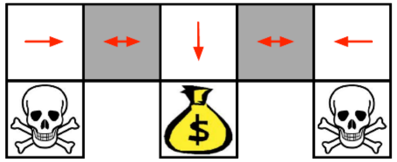
Stochastic policies

Instead, stochastic policies do something like this:

$$\pi(a|s) = \mathbb{P}[a|s;\theta]$$

Side note: Sometimes a stochastic policy is better than a deterministic one, even if the environment is deterministic
Stochastic Policies also allow for nicer exploration (later).

Downside is that they often are less interpretable.



- An optimal stochastic policy will randomly move E or W in grey states

$$\pi_\theta(\text{wall to N and S, move E}) = 0.5$$
$$\pi_\theta(\text{wall to N and S, move W}) = 0.5$$

- It will reach the goal state in a few steps with high probability
- Policy-based RL can learn the optimal stochastic policy

Source: https://omkar-ranadive.github.io/posts/rl-l7-ds

Policy-based Reinforcement Learning

Optimizing via gradient ascent

- Our goal is to maximize the expected reward:

$$G(\tau) := \sum_{t=0}^{T-1} \gamma^t R(s_t, a_t)$$

$$\max_\theta \mathbb{E}_{\pi_\theta} G(\tau)$$

(where π_θ is a parameterized policy, e.g., a neural network)

- But how do we maximize this?

  → Gradient Ascent! *Suppose* we know how to calculate the gradient w.r.t. the parameters:

$$\nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta} G(\tau)$$

Policy Gradient

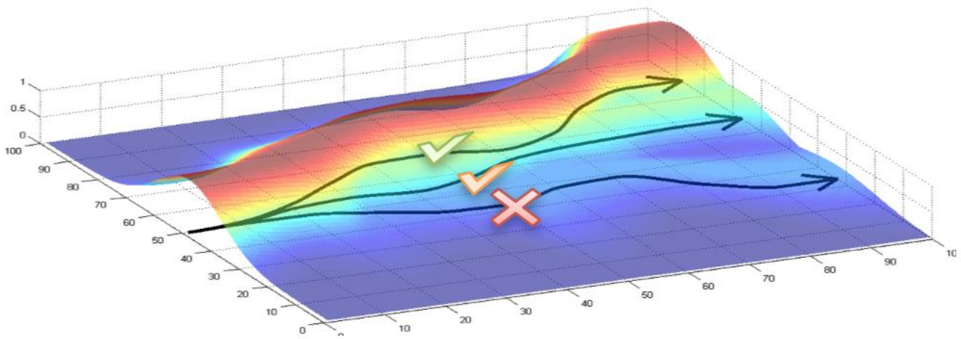
often in literature referred to as $\nabla_\theta J(\pi_\theta)$

- Then we can update our parameters θ in the direction of the gradient:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta} G(\tau)$$

Policy-based Reinforcement Learning

REINFORCE



Pieter Abbeel. DeepRL Bootcamp 4A Policy Gradients.

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Intuition: The gradient tries to

- Increase probability of paths with positive G
- Decrease probability of paths with negative G

Exercise Sheet 7.1

Vanilla Policy Gradient (VPG)

Actor Critic Approaches

Introduce critic that estimates Q

- The policy gradient we used so far (without baseline to begin with):

$$\nabla_\theta \mathbb{E}_{\pi_\theta} G(\tau) \approx \frac{1}{L}\sum_{\tau}\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\, G(\tau)$$

$$\approx \frac{1}{L}\sum_{\tau}\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \underbrace{\sum_{t'=t}^{T} \gamma^{t'-t} R(s_{t'}, a_{t'})}_{= \, Q^\pi(s_t, a_t)}$$

➢ Use e.g. a neural network to approximate Q!

- In practice: estimate $v^\pi(s_t;\phi)$ explicitly, and then sample

$$q^\pi(s_t, a_t) \approx G_t^{(n)}$$

i.e. $\hat{G}_t^{(1)} = R_t + \gamma v^\pi(s_{t+1};\phi)$

Actor Critic Approaches

Advantage Actor Critic (A2C)

- Introduce a baseline:

$$\nabla_\theta \mathbb{E}_{\pi_\theta} G(\tau) \approx \frac{1}{L}\sum_\tau \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \underbrace{\sum_{t'=t}^{T} \gamma^{t'-t} R(s_{t'}, a_{t'})}^{= Q^\pi(s_t,a_t)} - b(s_t)$$

$$= \frac{1}{L}\sum_\tau \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \underbrace{\boxed{(\hat{G}_t - b(s_t))}}_{:= A^\pi(s_t,a_t)}$$

- Calculate via TD error:

$$\begin{aligned} A^\pi(s_t,a_t) &= Q^\pi(s_t,a_t) - V^\pi(s_t) \\ &= r + \gamma \cdot v^\pi - v^\pi(s_t) \end{aligned}$$

- Or multi-step TD error: "Generalized Advantage Estimation (GAE)"

Exercise Sheet 7.2

Advantage Actor Critic (A2C)

Fraunhofer IIS

Thank you for your attention!

Fraunhofer IIS

Fraunhofer-Institut für Integrierte Schaltungen IIS