Lecture 11: Stochastic Policy Gradient Methods

Oliver Wallscheid

PADERBORN UNIVERSITY

Preface (1)

Shift from (indirect) value-based approaches

$$\hat{q}(\boldsymbol{x}, u, \boldsymbol{w}) \approx q(\boldsymbol{x}, u) \tag{11.1}$$

to (direct) policy-based solutions:

$$\boldsymbol{\pi}(\boldsymbol{u}|\boldsymbol{x}) = \mathbb{P}\left[\boldsymbol{U} = \boldsymbol{u}|\boldsymbol{X} = \boldsymbol{x}\right] \approx \boldsymbol{\pi}(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta})\,. \tag{11.2}$$

- Above, $\boldsymbol{\theta} \in \mathbb{R}^d$ is the policy parameter vector.
- Note, that $\boldsymbol{u}$ is now vectorial and might contain multiple continuous quantities.

Goal of today's lecture

- Introduce an algorithm class based on a parameterizable policy $\pi(\boldsymbol{\theta})$.
- Extend the action space to continuous actions.
- Combine the policy-based and value-based approach.

Preface (2)

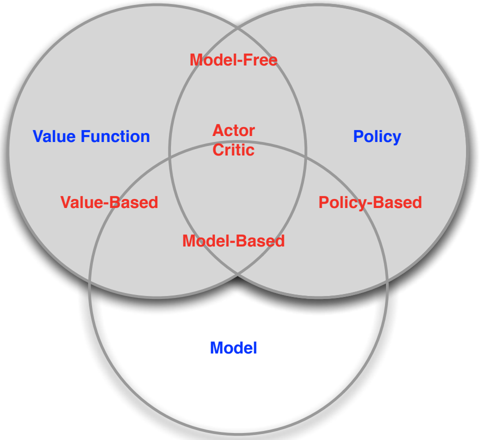

Fig. 11.1: Main categories of reinforcement learning algorithms (source: D. Silver, Reinforcement learning, 2015. CC BY-NC 4.0)

Table of contents

Motivating example: strategic gaming

Task: Two-player game of extended rock-paper-scissors

- A deterministic policy (i.e., value-based with given feature representation) can be easily exploited by the opponent.
- Conversely, a uniform random policy would be unpredictable.

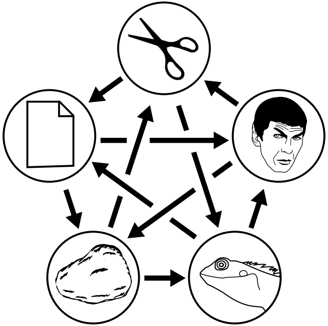

Fig. 11.2: Rock paper scissors lizard Spock game mechanics (source: www.wikipedia.org, by Diriector Doc CC BY-SA 4.0)

Example policy function: discrete action space

Assumption:

- Action space is discrete and compact.

A typical policy function is:

- Soft-max in action preferences

$$\pi(u|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{e^{h(\boldsymbol{x},u,\boldsymbol{\theta})}}{\sum_i e^{h(\boldsymbol{x},i,\boldsymbol{\theta})}} \tag{11.3}$$

  with $h(\boldsymbol{x}, u, \boldsymbol{\theta}) : \mathcal{X} \times \mathcal{U} \times \mathbb{R}^d \rightarrow \mathbb{R}$ being the numerical preference per state-action pair.
- Denominator of (11.3) sums up action probabilities to one per state.
- Is designed as a stochastic policy but can approach deterministic behavior in the limit.
- The preference is parametrized via a function approximator, e.g., linear in features

$$h(\boldsymbol{x}, u, \boldsymbol{\theta}) = \boldsymbol{\theta}^{\mathsf{T}} \tilde{\boldsymbol{x}}(\boldsymbol{x}, u). \tag{11.4}$$

Example policy function: continuous action space (1)

Assumption:

- Action space is continuous and there is only one scalar action $u \in \mathbb{R}$.

A typical policy function is:

- Gaussian probability density

$$\pi(u|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{\sigma(\boldsymbol{x}, \boldsymbol{\theta})\sqrt{2\pi}} \exp\left(-\frac{(u - \mu(\boldsymbol{x}, \boldsymbol{\theta}))^2}{2\sigma(\boldsymbol{x}, \boldsymbol{\theta})^2}\right) \tag{11.5}$$

  with mean $\mu(\boldsymbol{x}, \boldsymbol{\theta}) : \mathcal{X} \times \mathbb{R}^d \rightarrow \mathbb{R}$ and standard deviation $\sigma(\boldsymbol{x}, \boldsymbol{\theta}) : \mathcal{X} \times \mathbb{R}^d \rightarrow \mathbb{R}$ given by parametric function approximation.
- Variants regarding function μ and σ:
  1. Both share a mutual parameter set $\boldsymbol{\theta}$ (e.g., artificial neural network with multiple outputs).
  2. Both are parametrized independently $\boldsymbol{\theta} = \begin{bmatrix}\boldsymbol{\theta}_\mu & \boldsymbol{\theta}_\sigma\end{bmatrix}^\mathsf{T}$ (e.g., by two linear regression functions).
  3. Only $\mu(\boldsymbol{x}, \boldsymbol{\theta})$ is parametrized while σ is scheduled externally.

- Output of the functions $\mu_k = (\boldsymbol{x}_k, \boldsymbol{\theta}_k)$ and $\sigma_k = (\boldsymbol{x}_k, \boldsymbol{\theta}_k)$ can change in every time step.
- Depending on σ exploration is an inherent part of the (stochastic) policy.

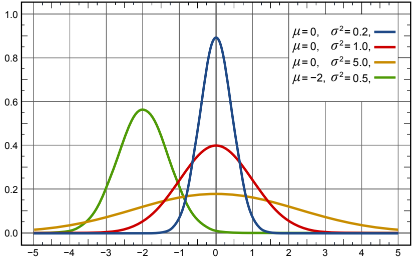

Fig. 11.3: Exemplary univariate Gaussian probability density functions (source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018, CC BY-NC-ND 2.0)

Example policy function: continuous action space (3)

Assumption:

- Action space is continuous and there are multiple actions $\boldsymbol{u} \in \mathbb{R}^m$.

A typical policy function is:

- Multivariate Gaussian probability density

$$\pi(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^m \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\boldsymbol{u} - \boldsymbol{\mu})^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\boldsymbol{u} - \boldsymbol{\mu})\right) \tag{11.6}$$

  with mean $\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\theta}) : \mathcal{X} \times \mathbb{R}^d \rightarrow \mathbb{R}^m$ and covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{x}, \boldsymbol{\theta}) : \mathcal{X} \times \mathbb{R}^d \rightarrow \mathbb{R}^{m \times m}$ given by parametric function approximation.
- Same parametrization variants apply to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ as in the scalar action case.
- In addition, $\boldsymbol{\Sigma}$ can be considered a diagonal matrix and clipped to reduce complexity as well as ensure nonsingularity.

Example policy function: continuous action space (4)

- Below we find an example for

$$\boldsymbol{\mu} = \begin{bmatrix}-0.4 & 0.3\end{bmatrix}^\mathsf{T} \quad \text{and} \quad \boldsymbol{\Sigma} = \begin{bmatrix}0.04 & 0\\ 0 & 0.02\end{bmatrix}.$$

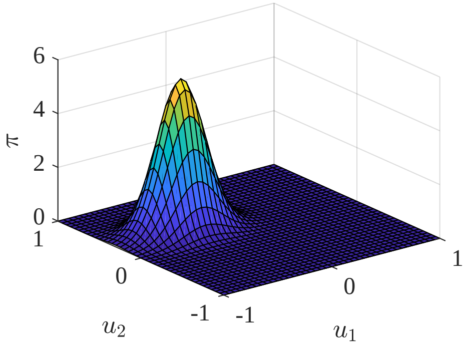

Fig. 11.4: Exemplary bivariate Gaussian probability density function

Policy objective function

- Goal: find optimal $\boldsymbol{\theta}^*$ given the policy $\boldsymbol{\pi}(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta})$.
- Problem: which measure of optimality should we use?

Possible optimality metrics:

- Start state value (in episodic tasks):

$$J(\boldsymbol{\theta}) = v_{\pi_{\boldsymbol{\theta}}}(\boldsymbol{x}_0) = \mathbb{E}\left[v|\boldsymbol{X} = \boldsymbol{x}_0, \boldsymbol{\theta}\right] \tag{11.7}$$

- Average reward (in continuing tasks):

$$J(\boldsymbol{\theta}) = \overline{r}_{\pi_{\boldsymbol{\theta}}} = \int_{\mathcal{X}} \mu_\pi(\boldsymbol{x}) \int_{\mathcal{U}} \pi(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta}) \int_{\mathcal{X},\mathcal{R}} p(\boldsymbol{x}', r|\boldsymbol{x}, \boldsymbol{u}) r \tag{11.8}$$

  - Above, $\mu_\pi(\boldsymbol{x})$ is again the steady-state distribution $\mu_\pi(\boldsymbol{x}) = \lim_{k\to\infty} \mathbb{P}\left[\boldsymbol{X}_k = \boldsymbol{x}|\boldsymbol{U}_{0:k-1} \sim \pi\right]$.

Policy optimization

- In essence, policy-based RL is an optimization problem.
- Depending on the policy function and task, finding $\boldsymbol{\theta}^*$ might be a
  - non-linear,
  - multidimensional and
  - non-stationary problem.
- Hence, we might consider global optimization techniques[1] like
  - Simple heuristics: random search, grid search,...
  - Meta-heuristics: evolutionary algorithms, particle swarm,....
  - Surrogate-model-based optimization: Bayes opt.,...
  - Gradient-based techniques with multi-start initialization.

[1] Recommended reading: J. Stork et al., *A new Taxonomy of Continuous Global Optimization Algorithms*, https://arxiv.org/abs/1808.08818, 2020

Policy gradient

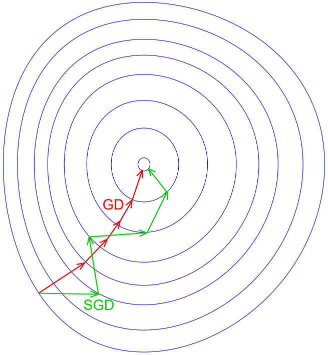

Fig. 11.5: Exemplary optimization paths for (stochastic) gradient ascent (derivative work of www.wikipedia.org, CC0 1.0)

- We will focus on gradient-based methods (policy gradient).
- Hence, we will assume that the gradient

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial J}{\partial \theta_1} & \cdots & \frac{\partial J}{\partial \theta_d} \end{bmatrix}^{\mathsf{T}}$$

required for gradient ascent optimization always exists:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}).$$

- True gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is usually approximated, e.g., by stochastic gradient descent (SGD) or derived variants.

Policy gradient theorem

Theorem 11.1: Policy Gradient

Given a metric $J(\boldsymbol{\theta})$ for the undiscounted episodic (11.7) or continuing tasks (11.8) and a parameterizable policy $\pi(\boldsymbol{u}|\boldsymbol{x},\boldsymbol{\theta})$ the policy gradient is

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_\pi\left[q_\pi(\boldsymbol{x},\boldsymbol{u})\frac{\nabla_{\boldsymbol{\theta}}\pi(\boldsymbol{u}|\boldsymbol{x},\boldsymbol{\theta})}{\pi(\boldsymbol{u}|\boldsymbol{x},\boldsymbol{\theta})}\right]. \tag{11.9}$$

- Having samples $\langle \boldsymbol{x}_i, \boldsymbol{u}_i\rangle$, an estimate of q_π and the policy function $\pi(\boldsymbol{\theta})$ available we receive an analytical solution for the policy gradient!
- Using identity $\nabla \ln a = \frac{\nabla a}{a}$ we can re-write to

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_\pi\left[q_\pi(\boldsymbol{x},\boldsymbol{u})\nabla_{\boldsymbol{\theta}}\ln\pi(\boldsymbol{u}|\boldsymbol{x},\boldsymbol{\theta})\right] \tag{11.10}$$

  with $\nabla_{\boldsymbol{\theta}}\ln\pi(\boldsymbol{u}|\boldsymbol{x},\boldsymbol{\theta})$ also called the score function.
- Derivation available in chapter 13.2 / 13.6 in the lecture book of Barto and Sutton.

Intuitive interpretation of policy parameter update

- Inserting the policy gradient theorem into gradient ascent approach:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \mathbb{E}_\pi \left[q_\pi(\boldsymbol{x}, \boldsymbol{u}) \frac{\nabla_{\boldsymbol{\theta}} \pi(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta})}{\pi(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta})} \right].$$

- Move in the direction that favor actions that yield an increased value.
- Scale the update step size inversely to the action probability to compensate that some actions are selected more frequently.

Also note:

- The policy gradient is not depending on the state distribution!
- Hence, we do not need any knowledge of the environment and receive a model-free RL approach!

Simple score function examples

Soft-max policy with linear function approximation:

$$\pi(u|\boldsymbol{x},\boldsymbol{\theta}) = \frac{e^{\boldsymbol{\theta}^\mathsf{T}\tilde{\boldsymbol{x}}(\boldsymbol{x},u)}}{\sum_i e^{\boldsymbol{\theta}^\mathsf{T}\tilde{\boldsymbol{x}}(\boldsymbol{x},i)}}$$

$$\Leftrightarrow \quad \nabla_{\boldsymbol{\theta}} \ln \pi(u|\boldsymbol{x},\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \left(\boldsymbol{\theta}^\mathsf{T}\tilde{\boldsymbol{x}}(\boldsymbol{x},u) - \ln\left(\sum_i e^{\boldsymbol{\theta}^\mathsf{T}\tilde{\boldsymbol{x}}(\boldsymbol{x},i)}\right)\right)$$

$$= \tilde{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{u}) - \mathbb{E}_\pi\left[\tilde{\boldsymbol{x}}(\boldsymbol{x},\cdot)\right]$$

Univariate Gaussian policy with linear function approximation and given σ:

$$\pi(u|\boldsymbol{x},\boldsymbol{\theta}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(u-\boldsymbol{\theta}^\mathsf{T}\tilde{\boldsymbol{x}}(\boldsymbol{x},u))^2}{2\sigma^2}\right)$$

$$\Leftrightarrow \quad \nabla_{\boldsymbol{\theta}} \ln \pi(u|\boldsymbol{x},\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \left(\ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{(u-\boldsymbol{\theta}^\mathsf{T}\tilde{\boldsymbol{x}}(\boldsymbol{x},u))^2}{2\sigma^2}\right)$$

$$= \frac{\left(u-\boldsymbol{\theta}^\mathsf{T}\tilde{\boldsymbol{x}}(\boldsymbol{x},u)\right)\tilde{\boldsymbol{x}}(\boldsymbol{x},u)}{\sigma^2}$$

Pro and cons: policy vs. value-based approaches

Pro value-based solutions (e.g., Q-learning):

- Estimated value is an intuitive performance metric.
- Considered sample-efficient (cf. replay buffer or bootstrapping).

Pro policy-based solutions (e.g., using policy gradient):

- Seamless integration of stochastic and dynamic policies.
- Straightforward applicable to large/continuous action spaces. In contrast, value-based approaches would require explicit optimization

$$\boldsymbol{u}^* = \arg\max_{\boldsymbol{u}} q(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}).$$

Mutual hassle:

- Gradient-based optimization with (non-linear) function approximation is likely to deliver only suboptimal and local policy optima.

Table of contents

Basic concept

Initial situation:

- Score function $\nabla_{\boldsymbol{\theta}} \ln \pi(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta})$ can be calculated analytically using suitable policy and chain rule (e.g., by algorithmic differentiation).
- Open question: how to retrieve $q_\pi(\boldsymbol{x}, \boldsymbol{u})$ in

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[q_\pi(\boldsymbol{x}, \boldsymbol{u}) \nabla_{\boldsymbol{\theta}} \ln \pi(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta})\right] \ ?$$

Monte Carlo policy gradient:

- Use sampled episodic return g_k to approximate $q_\pi(\boldsymbol{x}, \boldsymbol{u})$:

$$q_\pi(\boldsymbol{x}, \boldsymbol{u}) \approx g_k$$
$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \gamma^k g_k \nabla_{\boldsymbol{\theta}} \ln \pi(\boldsymbol{u}_k|\boldsymbol{x}_k, \boldsymbol{\theta}_k).$$

- The discounting of the policy gradient is introduced as an extension to Theo. 11.1 (which assumed an undiscounted episodic task).
- Also known as *REINFORCE* approach.

Algorithmic implementation: Monte Carlo policy gradient (REINFORCE)

- Usual technical convergence requirements regarding α apply.
- Use sampled return as unbiased estimate of q.
- Recall previous MC-based methods: high variance, slow learning.

input: a differentiable policy function $\pi(\boldsymbol{u}|\boldsymbol{x},\boldsymbol{\theta})$
parameter: step size $\alpha \in \{\mathbb{R}|0 < \alpha < 1\}$
init: parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$ arbitrarily
for $j = 1, 2, \ldots,$ *episodes* **do**
    generate an episode following $\pi(\cdot|\cdot,\boldsymbol{\theta})$: $\boldsymbol{x}_0, \boldsymbol{u}_0, r_1, \ldots, \boldsymbol{x}_T$;
    for $k = 0, 1, \ldots, T-1$ *time steps* **do**
        $g \leftarrow \sum_{i=k+1}^{T} \gamma^{i-k-1} r_i$;
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\gamma^k g \nabla_{\boldsymbol{\theta}} \ln \pi(\boldsymbol{u}_k|\boldsymbol{x}_k,\boldsymbol{\theta})$;

Algo. 11.1: Monte Carlo policy gradient (output: parameter vector $\boldsymbol{\theta}^*$ for $\pi^*(\boldsymbol{u}|\boldsymbol{x},\boldsymbol{\theta}^*)$)

REINFORCE example: short-corridor problem (1)

- Gridworld style problem with two actions: left (l), right (r)
- Second-left state's action execution is reversed
- Feature representation: $\tilde{x}(x, u = \mathsf{r}) = \begin{bmatrix}1 & 0\end{bmatrix}^\mathsf{T}$, $\tilde{x}(x, u = \mathsf{l}) = \begin{bmatrix}0 & 1\end{bmatrix}^\mathsf{T}$
- A policy-based approach searches for the optimal probability split

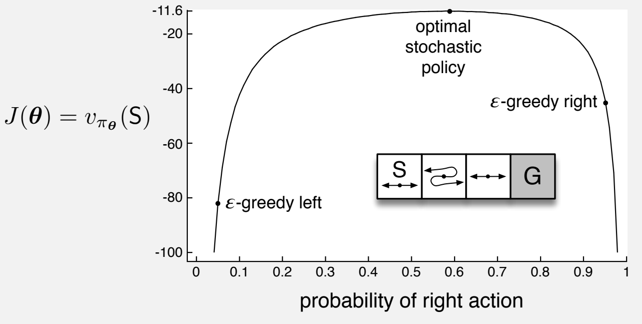

Fig. 11.6: Short-corridor problem with $\varepsilon = 0.1$ (source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018, CC BY-NC-ND 2.0)

REINFORCE example: short-corridor problem (2)

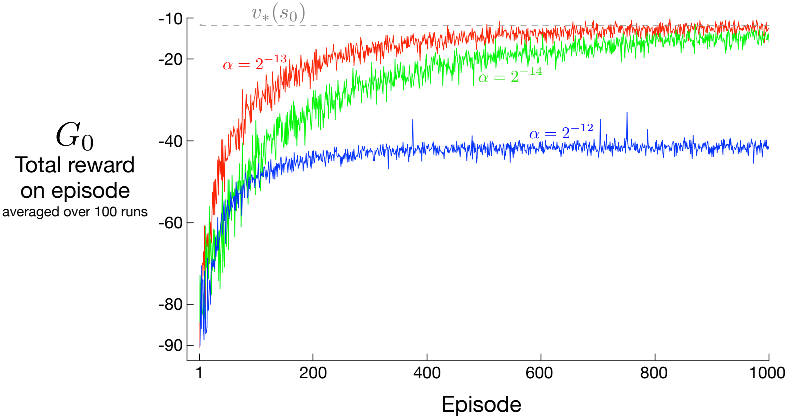

Fig. 11.7: Comparison of Monte Carlo policy gradient approach on short-corridor problem from Fig. 11.6 for different learning rates (source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018, CC BY-NC-ND 2.0)

Baseline

- Motivation: add a comparison term to the policy gradient to reduce variance while not affecting its expectation.
- Introduce the baseline $b(\boldsymbol{x})$:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[\left(q_\pi(\boldsymbol{x}, \boldsymbol{u}) - b(\boldsymbol{x}) \right) \nabla_{\boldsymbol{\theta}} \ln \pi(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta}) \right]. \tag{11.11}$$

- Since $b(\boldsymbol{x})$ is only depending on the state but not on the actions/policy we did not change the policy gradient in expectation:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[q_\pi(\boldsymbol{x}, \boldsymbol{u}) \nabla_{\boldsymbol{\theta}} \ln \pi(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta}) \right] - \underbrace{\mathbb{E}_\pi \left[b(\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \ln \pi(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta}) \right]}_{=0}.$$

- Consequently, the Monte Carlo policy parameter update yields:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \gamma^k \left(g_k - b(\boldsymbol{x}_k) \right) \nabla_{\boldsymbol{\theta}} \ln \pi(\boldsymbol{u}_k|\boldsymbol{x}_k, \boldsymbol{\theta}_k).$$

Advantage function

- Intuitive choice of the baseline is the state value $b(\boldsymbol{x}) = v_\pi(\boldsymbol{x})$.
- The resulting policy gradient becomes

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[\left(q_\pi(\boldsymbol{x}, \boldsymbol{u}) - v_\pi(\boldsymbol{x}) \right) \nabla_{\boldsymbol{\theta}} \ln \pi(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta}) \right]. \tag{11.12}$$

- Here, the difference between action and state value is the advantage function

$$a_\pi(\boldsymbol{x}, \boldsymbol{u}) = q_\pi(\boldsymbol{x}, \boldsymbol{u}) - v_\pi(\boldsymbol{x}). \tag{11.13}$$

- Interpretation: value difference taking (arbitrary) action $\boldsymbol{u}$ and thereafter following policy π compared to the state value following same policy (i.e., choosing $\boldsymbol{u} \sim \pi$) given the state.
- Hence, we might rewrite to:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[a_\pi(\boldsymbol{x}, \boldsymbol{u}) \nabla_{\boldsymbol{\theta}} \ln \pi(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta}) \right]. \tag{11.14}$$

Algo. implementation: MC policy gradient with baseline

- Implementation requires approximation $b(\boldsymbol{x}) \approx \hat{v}(\boldsymbol{x}, \boldsymbol{w})$.
- Hence, we are learning two parameter sets $\boldsymbol{\theta}$ and $\boldsymbol{w}$.
- Keep using sampled return as action-value estimate: $q_\pi(\boldsymbol{x}, \boldsymbol{u}) \approx g_k$.

input: a differentiable policy function $\pi(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta})$ and state-value function $\hat{v}(\boldsymbol{x}, \boldsymbol{w})$
parameter: step sizes $\{\alpha_w, \alpha_\theta\} \in \{\mathbb{R}|0 < \alpha < 1\}$
init: parameter vectors $\boldsymbol{w} \in \mathbb{R}^\zeta$ and $\boldsymbol{\theta} \in \mathbb{R}^d$ arbitrarily
for $j = 1, 2, \ldots,$ *episodes* **do**
  generate an episode following $\pi(\cdot|\cdot, \boldsymbol{\theta})$: $\boldsymbol{x}_0, \boldsymbol{u}_0, r_1, \ldots, \boldsymbol{x}_T$;
  for $k = 0, 1, \ldots, T-1$ *time steps* **do**
    $g \leftarrow \sum_{i=k+1}^{T} \gamma^{i-k-1} r_i$;
    $\delta \leftarrow g - \hat{v}(\boldsymbol{x}_k, \boldsymbol{w})$;
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha_w \delta \nabla_{\boldsymbol{w}} \hat{v}(\boldsymbol{x}_k, \boldsymbol{w})$;
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_\theta \gamma^k \delta \nabla_{\boldsymbol{\theta}} \ln \pi(\boldsymbol{u}_k|\boldsymbol{x}_k, \boldsymbol{\theta})$;

Algo. 11.2: Monte Carlo policy gradient with baseline (output: parameter vector $\boldsymbol{\theta}^*$ for $\pi^*(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta}^*)$) and $\boldsymbol{w}^*$ for $\hat{v}^*(\boldsymbol{x}, \boldsymbol{w}^*)$)

REINFORCE comparison w/o baseline

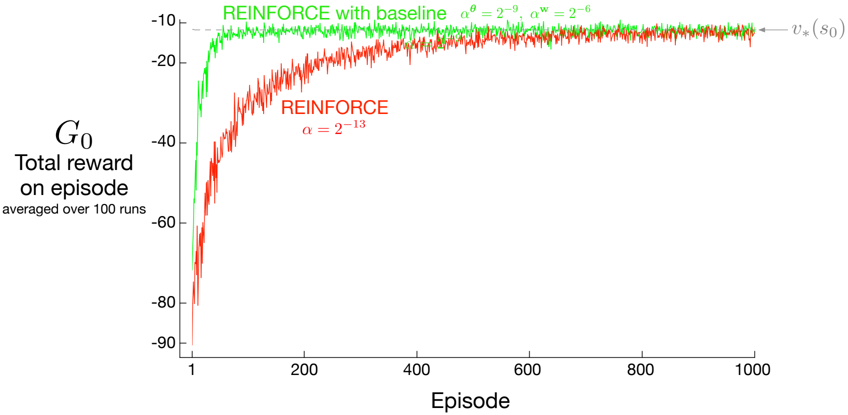

Fig. 11.8: Comparison of Monte Carlo policy gradient on short-corridor problem from Fig. 11.6 where both algorithms' learning rates have been tuned (source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018, CC BY-NC-ND 2.0)

Table of contents

General actor-critic idea

Conclusion of Monte Carlo policy gradient with baseline:

- Will learn an unbiased policy gradient.
- As the other MC-based methods: learns slowly due to high variance.
- Updates only available after full episodes.

Alternative: use an additional function approximator, the so-called critic, to estimate q_π (i.e., approximate policy gradient):

$$
\begin{aligned}
v(\boldsymbol{x}) &\approx \hat{v}(\boldsymbol{x}, \boldsymbol{w}_v),\\
q(\boldsymbol{x}, \boldsymbol{u}) &\approx \hat{q}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}_q),\\
a(\boldsymbol{x}, \boldsymbol{u}) &\approx \hat{q}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}_q) - \hat{v}(\boldsymbol{x}, \boldsymbol{w}_v).
\end{aligned}
$$

- Realization: any prediction tool discussed so far (TD(0), LSTD,...).
- Potential: we can use online step-by-step updates to estimate $\hat{q}$.
- Disadvantage: we would train two value estimates by $\boldsymbol{w}_v$ and $\boldsymbol{w}_q$.

Integrating the advantage function

- The TD error is

$$\delta_\pi = r + \gamma v_\pi(\boldsymbol{x}') - v_\pi(\boldsymbol{x}). \tag{11.15}$$

- In expectation the TD error is equivalent to the advantage function

$$\begin{aligned}\mathbb{E}_\pi\left[\delta_\pi|\boldsymbol{x},\boldsymbol{u}\right] &= \mathbb{E}_\pi\left[r + \gamma v_\pi(\boldsymbol{x}')|\boldsymbol{x},\boldsymbol{u}\right] - v_\pi(\boldsymbol{x})\\ &= q_\pi(\boldsymbol{x},\boldsymbol{u}) - v_\pi(\boldsymbol{x})\\ &= a_\pi(\boldsymbol{x},\boldsymbol{u}).\end{aligned} \tag{11.16}$$

- Hence, the TD error can be used to calculate the policy gradient:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_\pi\left[\delta_\pi \nabla_{\boldsymbol{\theta}} \ln \pi(\boldsymbol{u}|\boldsymbol{x},\boldsymbol{\theta})\right]. \tag{11.17}$$

- This results in requiring only one function parameter set:

$$\delta_\pi \approx r + \gamma \hat{v}_\pi(\boldsymbol{x}',\boldsymbol{w}) - \hat{v}_\pi(\boldsymbol{x},\boldsymbol{w}). \tag{11.18}$$

Actor-critic structure

- Critic (policy evaluation) and actor (policy improvement) can be considered another form of generalized policy iteration (GPI).
- Online and on-policy algorithm for discrete and continuous action spaces with built-in exploration by stochastic policy functions.

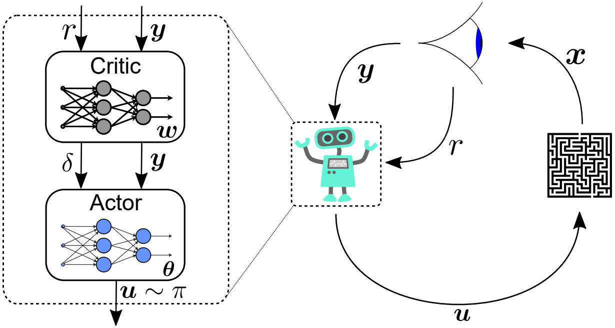

Fig. 11.9: Simplified flow diagram of actor-critic-based RL

Algo. implementation: actor-critic with TD(0) targets

- Analog to MC-based policy gradient optional discounting on the gradient updates is introduced.

input: a differentiable policy function $\pi(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta})$ and state-value function $\hat{v}(\boldsymbol{x}, \boldsymbol{w})$
parameter: step sizes $\{\alpha_w, \alpha_\theta\} \in \{\mathbb{R}|0 < \alpha < 1\}$
init: parameter vectors $\boldsymbol{w} \in \mathbb{R}^\zeta$ and $\boldsymbol{\theta} \in \mathbb{R}^d$ arbitrarily
for $j = 1, 2, \ldots,$ *episodes* **do**
    initialize $\boldsymbol{x}_0$;
    for $k = 0, 1, \ldots, T-1$ *time steps* **do**
        apply $\boldsymbol{u}_k \sim \pi(\cdot|\boldsymbol{x}_k, \boldsymbol{\theta})$ and observe $\boldsymbol{x}_{k+1}$ and r_{k+1};
        $\delta \leftarrow r_{k+1} + \gamma \hat{v}(\boldsymbol{x}_{k+1}, \boldsymbol{w}) - \hat{v}(\boldsymbol{x}_k, \boldsymbol{w})$;
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha_w \delta \nabla_{\boldsymbol{w}} \hat{v}(\boldsymbol{x}_k, \boldsymbol{w})$;
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_\theta \gamma^k \delta \nabla_{\boldsymbol{\theta}} \ln \pi(\boldsymbol{u}_k|\boldsymbol{x}_k, \boldsymbol{\theta})$;

Algo. 11.3: Actor-critic for episodic tasks using TD(0) targets (output: parameter vector $\boldsymbol{\theta}^*$ for $\pi^*(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta}^*)$) and $\boldsymbol{w}^*$ for $\hat{v}^*(\boldsymbol{x}, \boldsymbol{w}^*)$)

Actor-critic generalization

- Using the TD(0) error as the target to train the critic is convenient.
- However, the usual alternatives can be applied to train $\hat{v}(\boldsymbol{x}, \boldsymbol{w})$.
- n-step bootstrapping:

$$v(\boldsymbol{x}_k) \approx r_{k+1} + \gamma r_{k+2} + \cdots + \gamma^{n-1} r_{k+n} + \gamma^n \hat{v}_{k+n-1}(\boldsymbol{x}_{k+n}, \boldsymbol{w}).$$

- λ-return (forward view):

$$v(\boldsymbol{x}_k) \approx (1-\lambda) \sum_{n=1}^{T-k-1} \lambda^{(n-1)} g_{k:k+n} + \lambda^{T-k-1} g_k.$$

- TD(λ) using eligibility traces (backward view):

$$\begin{aligned}
\boldsymbol{z}_k &= \gamma\lambda \boldsymbol{z}_{k-1} + \nabla_{\boldsymbol{w}} \hat{v}(\boldsymbol{x}_k, \boldsymbol{w}_k),\\
\delta_k &= r_{k+1} + \gamma \hat{v}(\boldsymbol{x}_{k+1}, \boldsymbol{w}_k) - \hat{v}(\boldsymbol{x}_k, \boldsymbol{w}_k).
\end{aligned}$$

Algo. implementation: actor-critic with TD(λ) targets

input: a differentiable policy function $\pi(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta})$
input: a differentiable state-value function $\hat{v}(\boldsymbol{x}, \boldsymbol{w})$
parameter: $\{\alpha_w, \alpha_\theta\} \in \{\mathbb{R}|0 < \alpha < 1\}$, $\{\lambda_w, \lambda_\theta\} \in \{\mathbb{R}|0 \leq \lambda \leq 1\}$
init: parameter vectors $\boldsymbol{w} \in \mathbb{R}^\zeta$ and $\boldsymbol{\theta} \in \mathbb{R}^d$ arbitrarily
for $j = 1, 2, \ldots,$ *episodes* **do**
    initialize $\boldsymbol{x}_0$, $\boldsymbol{z}_w = 0$, $\boldsymbol{z}_\theta = 0$;
    for $k = 0, 1, \ldots, T-1$ *time steps* **do**
        apply $\boldsymbol{u}_k \sim \pi(\cdot|\boldsymbol{x}_k, \boldsymbol{\theta})$ and observe $\boldsymbol{x}_{k+1}$ and r_{k+1};
        $\delta \leftarrow r_{k+1} + \gamma \hat{v}(\boldsymbol{x}_{k+1}, \boldsymbol{w}) - \hat{v}(\boldsymbol{x}_k, \boldsymbol{w})$;
        $\boldsymbol{z}_w \leftarrow \gamma \lambda_w \boldsymbol{z}_w + \nabla_{\boldsymbol{w}} \hat{v}(\boldsymbol{x}_k, \boldsymbol{w})$;
        $\boldsymbol{z}_\theta \leftarrow \gamma \lambda_d \boldsymbol{z}_\theta + \gamma^k \nabla_{\boldsymbol{\theta}} \ln \pi(\boldsymbol{u}_k|\boldsymbol{x}_k, \boldsymbol{\theta})$;
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha_w \delta \boldsymbol{z}_w$;
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_\theta \delta \boldsymbol{z}_\theta$;

Algo. 11.4: Actor-critic for episodic tasks using TD(λ) targets (output: parameter vector $\boldsymbol{\theta}^*$ for $\pi^*(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\theta}^*)$) and $\boldsymbol{w}^*$ for $\hat{v}^*(\boldsymbol{x}, \boldsymbol{w}^*)$)

Summary: what you've learned today

- Policy-based methods are a new class within the RL toolbox.
  - Instead of learning a policy indirectly from a value the policy is directly parametrized.
  - The policy function allows discrete and continuous actions with inherent stochastic exploration.
- Solving the underlying optimization task is complex. However, the policy gradient theorem provides a suitable theoretical baseline for gradient-based optimization.
- Anyhow, to calculate policy gradients we require a value estimate.
  - Monte Carlo prediction is straightforward, but comes with high variance and slow learning.
  - Adding a state-dependent baseline comparison does not change the policy gradient in expectation but enables decreasing the variance.
- Extending this idea naturally leads to integrating a critic network, i.e., an additional function approximation to estimate the value.
- The critic can be fed by the usual targets (TD(0), TD(λ),...).