Reinforcement Learning
Lecture 9: Policy gradients

Lecturer: Prof. Dr. Mathias Niepert

Institute for Artificial Intelligence
Machine Learning and Simulation Lab

University of Stuttgart
Germany

imprs-is

June 27, 2024

Outline

Policy Optimization

Policy–based reinforcement learning

- So far we approximated the action–value function and generated a policy from it
- Approximation of the action–value function

$$\hat{q}(s, a, \boldsymbol{w}) \approx q_\pi(s, a)$$

- Generation of policy by, e.g., ϵ–greedy

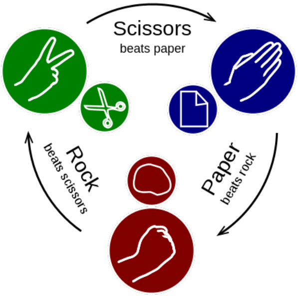
$$\hat{q}(s, a, \boldsymbol{w}) \xrightarrow{\epsilon\text{-greedy}} \pi$$

- Now we directly *parameterize* the policy π

Policy–based reinforcement learning

- Advantages:
  - can converge to a deterministic policy (as opposed to ϵ-greedy)
  - effective in high-dimensional or *continuous* action spaces
  - can learn *stochastic* policies
- Disadvantages:
  - typically converge to a *local* rather than *global* optimum
  - learning a policy can be challenging due to *high variance* of the learning signal

Stochastic policies



- Consider the *iterated* version of rock–paper–scissors
- What happens if you play a deterministic policy?
- Which policy is best?

Policy optimization

- Policy optimization:

$$\pi_* = \pi(a \mid s, \boldsymbol{\theta}_*)$$

  with

$$\boldsymbol{\theta}_* = \arg\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

  where J is some *performance measure*
- Discounted return: $G_0 = \sum_{k=0}^{T-1} \gamma^k R_{k+1}$

$$\pi_* = \arg\max_{\pi} \mathbb{E}_\pi[G_0] = \arg\max_{\pi} \mathbb{E}_\pi[v_\pi(s_0) \mid S_0 = s_0]$$

- Undiscounted return: $G_0 = \sum_{k=0}^{T-1} R_{k+1}$, i.e., $\gamma = 1$

$$\pi_* = \arg\max_{\pi} \mathbb{E}_\pi[R_0 + R_1 + \ldots + R_{T-1} \mid S_0 = s_0]$$

Parameterized policies

- Policies parameterized by parameter $\boldsymbol{\theta} \in \mathbb{R}^d$:
  - deterministic: $a = \pi(s, \boldsymbol{\theta})$
  - stochastic: $a \sim \pi(a \mid s, \boldsymbol{\theta}) = \Pr\{A_t = a \mid S_t = s, \boldsymbol{\theta}\}$
- Objective becomes $\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \max_{\boldsymbol{\theta}} v_{\pi_\theta}(s_0) = \max_{\boldsymbol{\theta}} \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[G_0]$
- We can parameterize π in any way, as long as it is *differentiable* wrt to $\boldsymbol{\theta}$
- In general we require $\pi(a \mid s, \boldsymbol{\theta}) \in [0, 1]$ for all s, a
- If the action space is discrete (and not too large): **softmax policy**

$$\pi(a \mid s, \boldsymbol{\theta}) = \frac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_b e^{h(s,b,\boldsymbol{\theta})}}$$

  where $h(\cdot)$ is the *action preference* function
- Can this approach the deterministic policy?

Policy gradient methods

- Problem:

$$\pi_* = \pi(a \mid s, \boldsymbol{\theta}_*)$$

with

$$\boldsymbol{\theta}_* = \arg\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[G_0]$$

Intuition: collect a bunch of trajectories, and ...

1. Make the good (high return) trajectories more probable by
2. Making the actions of these good trajectories more probable by
3. Pushing the policy toward generating these good actions

- Policy gradient methods:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \widehat{\nabla J(\boldsymbol{\theta})}$$

  - where $\nabla J(\boldsymbol{\theta})$ is the *policy gradient*:

$$\nabla J(\boldsymbol{\theta}) = \left(\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_0}, \ldots, \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_d} \right)^T$$

Score function

- We now compute the policy gradient analytically
- Assume policy $\pi_{\boldsymbol{\theta}}$ is differentiable whenever it is non-zero
- We have the following useful identity

$$\begin{aligned}\boldsymbol{\nabla}_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a \mid s) &= \pi_{\boldsymbol{\theta}}(a \mid s) \frac{\boldsymbol{\nabla}_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a \mid s)}{\pi_{\boldsymbol{\theta}}(a \mid s)} \\ &= \pi_{\boldsymbol{\theta}}(a \mid s) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a \mid s)\end{aligned}$$

- The **score function** is $\boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a \mid s)$

Gradient of expectation $\rightarrow$ expectation of gradient

- Consider $\mathbb{E}_{x \sim p(x|\boldsymbol{\theta})}[f(x)]$ for some function f
- We want to compute the gradient wrt $\boldsymbol{\theta}$

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathbb{E}_x[f(x)] &= \nabla_{\boldsymbol{\theta}} \int f(x) p(x \mid \boldsymbol{\theta}) dx \\
&= \int f(x) \nabla_{\boldsymbol{\theta}} p(x \mid \boldsymbol{\theta}) dx \\
&= \int f(x) \frac{\nabla_{\boldsymbol{\theta}} p(x \mid \boldsymbol{\theta})}{p(x \mid \boldsymbol{\theta})} p(x \mid \boldsymbol{\theta}) dx \\
&= \int \left[f(x) \nabla_{\boldsymbol{\theta}} \log p(x \mid \boldsymbol{\theta}) \right] p(x \mid \boldsymbol{\theta}) dx \\
&= \mathbb{E}_x[f(x) \underbrace{\nabla_{\boldsymbol{\theta}} \log p(x \mid \boldsymbol{\theta})}_{\text{score}}]
\end{aligned}
$$

Score: Softmax Policy

- Weight actions using linear combination of features $h(s,a) = \phi(s,a)^\top \boldsymbol{\theta}$
- Probability of action is proportional to exponentiated weight

$$\pi_{\boldsymbol{\theta}} = \frac{\exp\left(\phi(s,a)^\top \boldsymbol{\theta}\right)}{\sum_a \exp(\phi(s,a)^\top \boldsymbol{\theta})}$$

- The score function is

$$\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a \mid s) = \phi(s,a) - \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\phi(s,\cdot)]$$

Score: Gaussian Policy

- In continuous action spaces, a Gaussian policy is natural
- Mean is a linear combination of state features $\mu(s) = \phi(s)^\top \boldsymbol{\theta}$
- Variance may be fixed σ^2, or can also parametrized
- Policy is Gaussian, $a \sim \mathcal{N}(\mu(s), \sigma^2)$
- The score function is

$$\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a \mid s) = \frac{(a - \mu(s))\, \phi(s)}{\sigma^2}$$

One-Step MDPs

- Consider a simple class of one-step MDPs
  - Starting in state $s \sim \mu(s)$
  - Terminating after one time-step (taking action a) with reward $r = R_{s,a}$
- Use score function to compute the policy gradient

$$\begin{aligned}
J(\boldsymbol{\theta}) &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[r] \\
&= \sum_{s \in \mathcal{S}} \mu(s) \sum_{a \in \mathcal{A}} \pi_{\boldsymbol{\theta}}(a \mid s) R_{s,a} \\
\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \sum_{s \in \mathcal{S}} \mu(s) \sum_{a \in \mathcal{A}} \pi_{\boldsymbol{\theta}}(a \mid s) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a \mid s) R_{s,a} \\
&= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[r \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a \mid s)]
\end{aligned}$$

Policy Gradient Theorem

- The policy gradient theorem generalises the previous derivation to multi-step MDPs
- Replaces instantaneous reward r with long-term value $q_\pi(s, a)$
- Policy gradient theorem applies to start state objective, average reward and average value objective

Theorem (Policy-gradient)

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[q_\pi(s, a) \nabla_{\boldsymbol{\theta}} \log \pi(a \mid s, \boldsymbol{\theta})\right]$$

Update rule: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G_t \nabla_{\boldsymbol{\theta}} \log \pi(A_t \mid S_t, \boldsymbol{\theta})$

Policy Gradient Methods

Monte–Carlo policy gradient (REINFORCE)

- Update parameters $\boldsymbol{\theta}$ by stochastic gradient ascent
  - using the policy gradient theorem
  - using return G_t as an unbiased sample of $q_\pi(S_t, A_t)$

Initialize a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Initialize $\boldsymbol{\theta} \in \mathbb{R}^{d'}$
for all episodes **do**
    Generate an episode $\tau = (S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T)$ following $\pi(.|., \boldsymbol{\theta})$
    for $t = 0, 1, \ldots, T-1$ **do**
        $G_t \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ // return at time t
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G_t \nabla_{\boldsymbol{\theta}} \log \pi(A_t \mid S_t, \boldsymbol{\theta})$ // update rule
    end for
end for

REINFORCE with baseline

- Monte–Carlo policy gradient still suffers from high variance

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{S}} \mu(s) \sum_{a \in \mathcal{A}} \pi_{\boldsymbol{\theta}}(a \mid s) \left(q_\pi(s,a) - b(s)\right) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a \mid s)$$

Theorem (Policy-gradient with baseline)

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\left(q_\pi(s,a) - b(s)\right) \nabla_{\boldsymbol{\theta}} \log \pi(a \mid s, \boldsymbol{\theta})\right]$$

Update rule: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t \left(G_t - b(S_t)\right) \nabla_{\boldsymbol{\theta}} \log \pi(A_t \mid S_t, \boldsymbol{\theta})$

REINFORCE with baseline

- We use an approximation of the value function $\hat{v}(s, \boldsymbol{w}) \approx v_\pi(s)$ as a *baseline*
- Policy parameters: $\boldsymbol{\theta}$
- Value estimator parameters: $\boldsymbol{w}$
- Update rule:

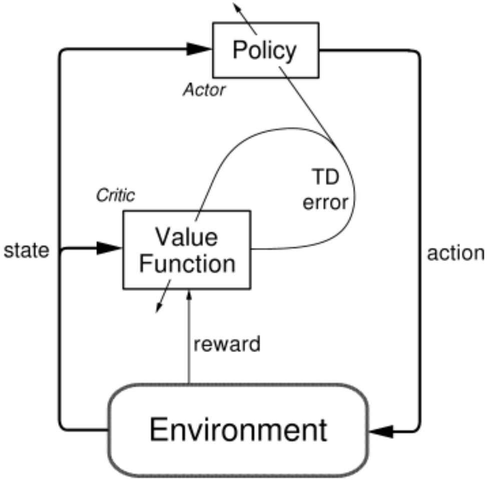
$$
\begin{aligned}
\delta &\leftarrow G - \hat{v}(S_t, \boldsymbol{w}) \\
\boldsymbol{w} &\leftarrow \boldsymbol{w} + \alpha^{\boldsymbol{w}} \gamma^t \delta \nabla_{\boldsymbol{w}} \hat{v}(S_t, \boldsymbol{w}) \\
\boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla_{\boldsymbol{\theta}} \log \pi(A_t \mid S_t, \boldsymbol{\theta})
\end{aligned}
$$

- Interpretation:
  - $\uparrow$ log–prob of action A_t proportionally to how much G_t is better than expected
  - baseline accounts for and removes the effect of past actions

Estimating the action–value function

- We are solving the prediction problem: policy evaluation
- How good is policy $\pi_{\boldsymbol{\theta}}$ with current parameters $\boldsymbol{\theta}$?
- Familiar toolset for *fitting* the baseline:
  - Monte–Carlo policy evaluation
  - TD-learning
  - TD(λ)
  - LSPI

Actor-critic concept



Actor-critic vs. baseline

- Delivers trade off between *variance reduction* of policy gradients with *bias introduction* from value function methods
- **Critic:** updates value–function parameters $\boldsymbol{w}$
- **Actor:** updates policy parameters $\boldsymbol{\theta}$ using critic
- REINFORCE with baseline uses value–function as *baseline* not as a *critic*
  - not used for *bootstrapping*
- One–step actor-critic update:

$$
\begin{aligned}
\delta &\leftarrow R_{t+1} + \gamma \hat{v}(S_{t+1}, \boldsymbol{w}) - \hat{v}(S_t, \boldsymbol{w}) \\
\boldsymbol{w} &\leftarrow \boldsymbol{w} + \alpha^{\boldsymbol{w}} \gamma^t \delta \nabla_{\boldsymbol{w}} \hat{v}(S_t, \boldsymbol{w}) \\
\boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla_{\boldsymbol{\theta}} \log \pi(A_t \mid S_t, \boldsymbol{\theta})
\end{aligned}
$$

Bias in Actor–Critic algorithms

- Approximating (bootstrapping) the policy gradient introduces bias
- Biased policy gradient might not find the right solution
- But reduces variance and makes learning substantially more efficient
- *Compatible function approximation* avoids this problem:

What about continuous action spaces?

- Softmax works for (not too large) discrete action–spaces
- In continuous action spaces, a **Gaussian policy** is a common choice
- Gaussian policy:

$$A_t \sim \mathcal{N}(\mu(S_t, \boldsymbol{\theta}), \sigma^2(S_t, \boldsymbol{\theta}))$$

- Variance may also be constant across the state space
- E.g., neural network outputs the mean of each action dimension as a function of the state

Summary

- *Action–value methods*: learn values, select action accordingly
- *Policy–gradient methods:* learn directly a parameterized Policy
- Advantages:
  - learn specific probabilities for taking the actions
  - learn appropriate levels of exploration, or ...
  - approach deterministic policies
  - can handle continuous action spaces
  - some policies are simpler to represent than value function
  - *policy gradient theorem* (exact expression how performance is affected)
- REINFORCE with baseline reduces variance without adding bias
- Value–functions for bootstrapping (Actor–Critic) introduces bias, ... but substantially reduces variance

Policy gradient theorem (notes 1)

- Recall: $\nabla_{\boldsymbol{\theta}} \mathbb{E}_x[f(x)] = \mathbb{E}_x[f(x) \nabla_{\boldsymbol{\theta}} \log p(x \mid \boldsymbol{\theta})]$
- Let us consider a random trajectory τ in place of x:
  $\tau = (S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T, S_T)$ and G_0 instead of f, thus we have:

$$p(\tau \mid \boldsymbol{\theta}) = \Pr\{S_0\} \prod_{t=0}^{T-1} \pi(A_t \mid S_t, \boldsymbol{\theta}) p(S_{t+1} \mid S_t, A_t) \qquad \text{// trajectory likelihood}$$

$$\log p(\tau \mid \boldsymbol{\theta}) = \log \Pr\{S_0\} + \sum_{t=0}^{T-1} \log \pi(A_t \mid S_t, \boldsymbol{\theta}) + \log p(S_{t+1} \mid S_t, A_t) \qquad \text{// log-likelihood}$$

$$\nabla_{\boldsymbol{\theta}} \log p(\tau \mid \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \sum_{t=0}^{T-1} \log \pi(A_t \mid S_t, \boldsymbol{\theta}) \qquad \text{// gradient of log-likelihood (score function)}$$

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_\tau[G_0] = \mathbb{E}_\tau \left[G_0 \nabla_{\boldsymbol{\theta}} \sum_{t=0}^{T-1} \log \pi(A_t \mid S_t, \boldsymbol{\theta}) \right] \qquad \text{// gradient of expected return}$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_\tau[G_0] = \mathbb{E}_\tau[G_0 \nabla_{\boldsymbol{\theta}} \log p(\tau \mid \boldsymbol{\theta})] \qquad \text{// gradient of performance}$$

Policy gradient theorem (notes 2)

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \mathbb{E}_\tau[G_0] &= \mathbb{E}_\tau\left[G_0 \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi(A_t \mid S_t, \boldsymbol{\theta})\right] \\ &= \mathbb{E}_\tau\left[\left(\sum_{t=0}^{T-1} R_{t+1}\right) \nabla_{\boldsymbol{\theta}} \log \sum_{t=0}^{T-1} \pi(A_t \mid S_t, \boldsymbol{\theta})\right]\end{aligned}$$

- For a single reward R_k we have:

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_\tau[R_k] = \mathbb{E}_\tau\left[R_k \sum_{t=0}^{k-1} \nabla_{\boldsymbol{\theta}} \log \pi(A_t \mid S_t, \boldsymbol{\theta})\right]$$

- Summing over all k, we get:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \mathbb{E}_\tau[G_0] &= \mathbb{E}_\tau\left[\sum_{k=1}^{T} R_k \sum_{t=0}^{k-1} \nabla_{\boldsymbol{\theta}} \log \pi(A_t \mid S_t, \boldsymbol{\theta})\right] \\ &= \mathbb{E}_\tau\left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi(A_t \mid S_t, \boldsymbol{\theta}) \sum_{k=t+1}^{T} R_k\right]\end{aligned}$$

Policy gradient theorem (notes 3)

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\tau}[G_0] &= \mathbb{E}_{\tau}\left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi(A_t \mid S_t, \boldsymbol{\theta}) \sum_{k=t+1}^{T} R_k\right] \\ &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[q_\pi(S_t, A_t) \nabla \log \pi(A_t \mid S_t, \boldsymbol{\theta})\right]\end{aligned}$$

Theorem

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[q_\pi(S_t, A_t) \nabla \log \pi(A_t \mid S_t, \boldsymbol{\theta})\right]$$