システム制御理論と統計的機械学習

第 9 章：強化学習

加嶋 健司

October 10, 2025

京都大学 情報学研究科

本章の流れ

9.1 最適制御と強化学習
強化学習の概要

9.2 モデルベースト設計
方策反復法と方策勾配法

9.3 モデルフリー設計
行動価値関数をデータから推定

最適制御と強化学習

最適制御問題

基本的には以下の無限時間最適制御問題を解くことが目標

Problem (時不変無限区間最適制御問題)

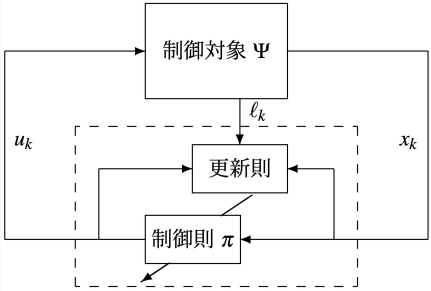
遷移確率密度関数 Ψ と初期状態 $x_0 \in \mathrm{rv}(\mathbb{X})$ をもつ確率システム，
非負のステージコスト $\ell : \mathbb{X} \times \mathbb{U} \to \mathbb{R}_+$，集合値関数 $U(\mathrm{x}) \subset \mathbb{U}$，$\mathrm{x} \in \mathbb{X}$ および $\beta \in (0, 1)$
$\pi(x_k) \in U(x_k)$ を満たし，

$$J(\pi) := \mathbb{E}^{\pi}\left[\sum_{k=0}^{\infty} \beta^k \ell(x_k, u_k)\right] \tag{9.1}$$

を最小化する制御則 π を求めよ．

モデルフリー設計

- モデルベースト設計：遷移確率密度関数 Ψ を利用する設計法（たとえば Ψ で定まるベルマン方程式を解く）
- **強化学習**（reinforcement learning）：Ψ は未知であるとし（モデルフリー設計），(x_k, u_k, ℓ_k) のデータを用いて制御則 π を更新して制御器を設計すること
- 制御則を π^* に収束させる更新則を設計
  更新に有用な情報を得るための入力設計



強化学習制御器の構造

モデルベースト設計

ベルマン方程式 (1) 価値関数

作用素

$$\mathscr{A}_x[V](\mathrm{x},\mathrm{u}) := \int_{\mathbb{X}} V(\mathrm{x}')\Psi(\mathrm{x}'|\mathrm{x},\mathrm{u})\mathrm{d}\mathrm{x}' = \mathbb{E}[V(x_{k+1})|x_k = \mathrm{x},\ u_k = \mathrm{u}] \tag{9.2}$$

$$\mathscr{B}[V](\mathrm{x}) := \inf_{\mathrm{u}\in U(\mathrm{x})} (\ell + \beta\mathscr{A}_x[V])(\mathrm{x},\mathrm{u}) \tag{9.3}$$

$$\mathscr{B}^\pi[V](\mathrm{x}) := \int_{U(\mathrm{x})} (\ell + \beta\mathscr{A}_x[V])(\mathrm{x},\mathrm{u})\pi(\mathrm{u}|\mathrm{x})\mathrm{d}\mathrm{u} \tag{9.4}$$

状態価値関数の定義

$$\mathsf{V}^\pi(\mathrm{x}) := \mathbb{E}^\pi_{x_0=\mathrm{x}}\left[\sum_{k=0}^{\infty}\beta^k\ell_k\right] \tag{9.5}$$

$$\mathsf{V}^*(\mathrm{x}) := \inf_\pi \mathsf{V}^\pi(\mathrm{x}) \tag{9.6}$$

行動価値関数の定義

$$\mathsf{Q}^\pi(\mathrm{x},\mathrm{u}) := \mathbb{E}^\pi_{x_0=\mathrm{x},\ u_0=\mathrm{u}}\left[\sum_{k=0}^{\infty}\beta^k\ell_k\right] \tag{9.7}$$

$$\mathsf{Q}^*(\mathrm{x},\mathrm{u}) := \inf_\pi \mathsf{Q}^\pi(\mathrm{x},\mathrm{u}) \tag{9.8}$$

ベルマン方程式 (2) 再掲

$$\mathsf{Q}^*(\mathrm{x},\mathrm{u}) = (\ell + \beta\mathscr{A}_x[\mathsf{V}^*])(\mathrm{x},\mathrm{u})$$

$$\mathsf{V}^*(\mathrm{x}) = \mathcal{B}[\mathsf{V}^*](\mathrm{x}) = \inf_{\mathrm{u}\in U(\mathrm{x})} \mathsf{Q}^*(\mathrm{x},\mathrm{u})$$

$$\mathsf{Q}^\pi(\mathrm{x},\mathrm{u}) = (\ell + \beta\mathscr{A}_x[\mathsf{V}^\pi])(\mathrm{x},\mathrm{u})$$

$$\mathsf{V}^\pi(\mathrm{x}) = \mathscr{B}^\pi[\mathsf{V}^\pi](\mathrm{x}) = \int_{U(\mathrm{x})} \mathsf{Q}^\pi(\mathrm{x},\mathrm{u})\pi(\mathrm{u}|\mathrm{x})\mathrm{du}$$

$$\pi^*(\mathrm{x}) = \arg\min_{\mathrm{u}\in U(\mathrm{x})} \mathsf{Q}^*(\mathrm{x},\mathrm{u})$$

状態価値関数・行動価値関数の入力

	x_0	u_0	$u_k,\ k \geq 1$
$\mathsf{V}^*(\mathrm{x})$	x	$\pi^*(\mathrm{x})$	$\pi^*(x_k)$
$\mathsf{Q}^*(\mathrm{x},\mathrm{u})$	x	u	$\pi^*(x_k)$
$\mathsf{V}^\pi(\mathrm{x})$	x	$\pi(\mathrm{x})$	$\pi(x_k)$
$\mathsf{Q}^\pi(\mathrm{x},\mathrm{u})$	x	u	$\pi(x_k)$

ベルマン方程式 (3) 価値反復法

補題 9.2.1 – $\mathscr{B}^\pi$ の単調性

関数 $V_1,\ V_2 : \mathbb{X} \to \mathbb{R}$ が $V_1 \le V_2$ を満たすならば，$\mathscr{B}^\pi[V_1] \le \mathscr{B}^\pi[V_2]$ である．

補題 9.2.2 – 終端コストと状態価値関数

$$(\mathscr{B}^\pi)^{\bar{k}}[V](\mathrm{x}) = \mathbb{E}^\pi_{x_0=\mathrm{x}}\left[\sum_{k=0}^{\bar{k}-1} \beta^k \ell(x_k, u_k) + \beta^{\bar{k}} V(x_{\bar{k}})\right] \tag{9.14}$$

定理 9.2.3 – V^π に対する価値反復法（$\ell,\ V$ は有界）

$$\lim_{k\to\infty} \|(\mathscr{B}^\pi)^k[V] - \mathsf{V}^\pi\|_\infty = 0 \tag{9.15}$$

ベルマン方程式 (4) 方策反復法

方策反復法（policy iteration）：$\pi_+(\mathrm{x}) := \underset{\mathrm{u}\in U(\mathrm{x})}{\arg\min}\ \mathsf{Q}^\pi(\mathrm{x},\mathrm{u})$ により制御則を更新

定理 9.2.4 – 方策改善定理（policy improvement theorem)

ℓ は有界な関数，π を任意の状態フィードバック制御則とする．このとき，

$$\mathsf{Q}^\pi(\mathrm{x},\pi_+(\mathrm{x})) \le \mathsf{V}^\pi(\mathrm{x}),\ \forall \mathrm{x}\in\mathbb{X} \tag{9.17}$$

を満たす任意の π_+ に対して，$\mathsf{V}^{\pi_+} \le \mathsf{V}^\pi$ が成り立つ．逆に，V^π が有界かつ

$$\mathsf{Q}^\pi(\mathrm{x},\mathrm{u}) \ge \mathsf{V}^\pi(\mathrm{x}) - \varepsilon(1-\beta),\ \forall \mathrm{x}\in\mathbb{X},\ \mathrm{u}\in U(\mathrm{x}) \tag{9.18}$$

が成り立つとき，$\mathsf{V}^\pi \le \mathsf{V}^* + \varepsilon$ である．

- 初期時刻以外は π で固定したままで，初期時刻の入力のみ変更して性能が改善しなければ，π は既に最適である（初期時刻以外を同時に探索しても改善しない）

ベルマン方程式 (5) LQR に対する方策反復法

LQR に対する方策反復法は，リカッチ方程式の反復解法（Hewer，1971）

$$x_{k+1} = Ax_k + Bu_k + v_k,\ \ell_k = \begin{bmatrix} x_k^\top & u_k^\top \end{bmatrix} \begin{bmatrix} Q & S \\ S^\top & R \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \tag{9.21}$$

1. 制御則 π を $u_k = -Kx_k$ としたとき，状態価値関数 $\mathrm{V}^\pi(\mathrm{x}) = \mathrm{x}^\top \Pi \mathrm{x} + \gamma$（$\because$ 定理 6.2.2）

$$\beta(A - BK)^\top \Pi (A - BK) + \begin{bmatrix} I & -K^\top \end{bmatrix} \begin{bmatrix} Q & S \\ S^\top & R \end{bmatrix} \begin{bmatrix} I \\ -K \end{bmatrix} = \Pi \tag{9.22}$$

2. 行動価値関数は，（$\because \mathrm{Q}^\pi(\mathrm{x}, \mathrm{u}) = (\ell + \beta \mathscr{A}_x[\mathrm{V}^\pi])(\mathrm{x}, \mathrm{u})$）

$$\mathrm{Q}^\pi(\mathrm{x}, \mathrm{u}) = \begin{bmatrix} \mathrm{x}^\top & \mathrm{u}^\top \end{bmatrix} \Upsilon^\pi \begin{bmatrix} \mathrm{x} \\ \mathrm{u} \end{bmatrix} + \gamma_\pi,\ \Upsilon^\pi := \begin{bmatrix} * & S + \beta A^\top \Pi B \\ * & R + \beta B^\top \Pi B \end{bmatrix} \tag{9.23}$$

3. 更新則は，

$$\underset{\mathrm{u} \in \mathbb{R}^{n_u}}{\arg\min}\ \mathrm{Q}^\pi(\mathrm{x}, \mathrm{u}) = -K_+ \mathrm{x} := -(\Upsilon^\pi)_{22}^{-1} (\Upsilon^\pi)_{12}^\top \mathrm{x} \tag{9.24}$$

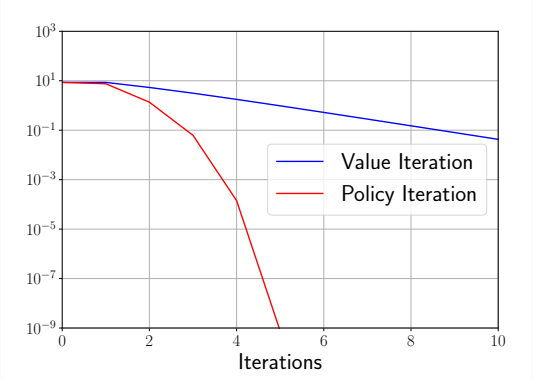
方策勾配法 (6) LQR に対する反復法の例

- 価値反復法であるリカッチイタレーションでは，特別な初期化は不要であるが，反復中に得られる制御則は制御区間を有限で打ち切った問題の最適制御性能しか考慮しておらず，十分収束するまでは安定性も保証されない
- 方策反復法は安定化制御則で初期化する必要があるが，反復ごとに無限時間の制御性能が改善される

例 9.2.5 – 方策反復法

方策反復法では各反復においてリアプノフ方程式を解く必要があるが，誤差の収束は価値反復法よりも速い．

- 価値反復法 $\Pi_{i+1} = \mathrm{Ric}(\Pi_i)$



方策勾配法 (1) 勾配の計算

方策勾配法（policy gradient method）：

制御則をパラメータ $\theta \in \mathbb{R}^{n_p}$ で規定し，コストの θ に関する勾配を用いて最適化

- ある θ に対する Q^{π_θ} が既知でも，勾配計算は自明でない
- 確定方策 $\pi_\theta(\mathrm{x})$ のとき，

$$J(\pi_\theta) = \mathbb{E}\left[\mathrm{V}^{\pi_\theta}(x_0)\right] = \mathbb{E}\left[\mathrm{Q}^{\pi_\theta}(x_0, \pi_\theta(x_0))\right] \tag{9.25}$$

の θ に関する勾配をとればよいが，π_θ はこちらが自由に設定できる関数であるものの，Q^{π_θ} の関数形の θ への依存性は自明ではない．

方策勾配法 (2) 方策勾配定理

定理 9.2.6 – 方策勾配定理

確率方策 $\pi_\theta(\mathrm{u}|\mathrm{x})$

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}^{\pi_\theta}\left[\sum_{k=0}^{\infty}\beta^k \int_{U(x_k)} (\nabla_\theta \pi_\theta(\mathrm{u}|x_k))\mathrm{Q}^{\pi_\theta}(x_k,\mathrm{u})\mathrm{du}\right] \tag{9.26}$$

確定方策 $\pi_\theta(\mathrm{x})$

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}^{\pi_\theta}\left[\sum_{k=0}^{\infty}\beta^k \nabla_\theta \pi_\theta(x_k)\nabla_u \mathrm{Q}^{\pi_\theta}(x_k,\pi_\theta(x_k))\right] \tag{9.27}$$

- Q^{π_θ} の θ に関する勾配 $\nabla_\theta \mathrm{Q}^{\pi_\theta}$ が現れていない
- 期待値が φ_{x_0} ではなく $\sum_{k=0}^{\infty}\beta^k\varphi_{x_k}^{\pi_\theta}$ により重み付けされた積分

方策勾配法 (3) 行動価値関数の必要情報

注意 9.2.7 – 各手法に必要な Q^{π_θ} の情報

- 方策反復法の更新則

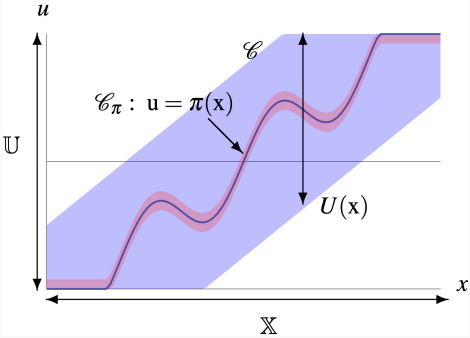
$$\pi_+(\mathrm{x}) := \operatorname*{argmin}_{\mathrm{u}\in U(\mathrm{x})} \mathrm{Q}^{\pi}(\mathrm{x},\mathrm{u})$$

の計算には，$\mathcal{C} := \{(\mathrm{x},\mathrm{u}) : \mathrm{x}\in\mathbb{X},\ \mathrm{u}\in U(\mathrm{x})\}$ 全体

- 方策勾配法の勾配計算

$$\mathbb{E}^{\pi_\theta}\left[\sum_{k=0}^{\infty}\beta^k \nabla_\theta \pi_\theta(x_k)\nabla_u \mathrm{Q}^{\pi_\theta}(x_k,\pi_\theta(x_k))\right]$$

には，$\mathcal{C}_{\pi_\theta} := \{(\mathrm{x},\pi_\theta(\mathrm{x})) : \mathrm{x}\in\mathbb{X}\}$ 近傍



行動価値関数を求める領域

モデルフリー設計

行動価値関数の推定 (1)

遷移確率密度関数 Ψ を用いずに方策反復法や方策勾配法を実装

問題 9.3.1 – 行動価値関数の推定

遷移確率密度関数 Ψ および状態フィードバック制御則 π に対して，$x_{:}$, $u_{:}$, $\ell_{:}$ の実現値系列を用いて，行動価値関数 Q^{π} を推定せよ．

アルゴリズム 4 方策反復法

input π
output π_{+}
1: Q^{π} を求める
2: **for all** x in $\mathbb{X}$ **do**
3: $\quad \pi_{+}(\mathrm{x}) \leftarrow \underset{\mathrm{u}\in U(\mathrm{x})}{\arg\min}\ \mathrm{Q}^{\pi}(\mathrm{x},\mathrm{u})$

アルゴリズム 6 方策勾配法

input θ, a
output θ_{+}
1: Q^{π_θ} を求める
2: $\nabla_\theta J(\pi_\theta)$ を求める
3: $\theta_{+} \leftarrow \theta - a\nabla_\theta J(\pi_\theta)$

行動価値関数の推定 (2) TD 誤差

TD 誤差（temporal difference error）

$$\delta_k := \ell_k + \beta\widehat{\mathrm{Q}}^{\pi}(x_{k+1}, \pi(x_{k+1})) - \widehat{\mathrm{Q}}^{\pi}(x_k, u_k) \tag{9.28}$$

- $\mathrm{Q}^{\pi}(\mathrm{x}_k, \mathrm{u}_k)$ を $\ell(\mathrm{x}_k, \mathrm{u}_k) + \beta\mathrm{Q}^{\pi}(\mathrm{x}_{k+1}, \pi(\mathrm{x}_{k+1}))$ の近似とする際の誤差
- ベルマン方程式 $\mathrm{Q}^{\pi}(\mathrm{x}, \mathrm{u}) = (\ell + \beta\mathscr{A}_x[\mathrm{V}^{\pi}])(\mathrm{x}, \mathrm{u})$ より，$\widehat{\mathrm{Q}}^{\pi} = \mathrm{Q}^{\pi}$ ならば

$$\mathbb{E}[\delta_k | \mathrm{x}_{:k}, \mathrm{u}_{:k}, l_{:k-1}] = 0, \ \forall k, \ \mathrm{x}_{:k}, \ \mathrm{u}_{:k}, \ \boldsymbol{l}_{:k-1} \tag{9.29}$$

δ_k が 0 に近い値をとる $\widehat{\mathrm{Q}}^{\pi}$ を Q^{π} の推定結果とするという立場をとる

行動価値関数の推定 (3) TD 学習

$$\delta_k := \ell_k + \beta \widehat{\mathrm{Q}}^{\pi}(x_{k+1}, \pi(x_{k+1})) - \widehat{\mathrm{Q}}^{\pi}(x_k, u_k)$$

各組 $(\mathrm{x}_k, \mathrm{u}_k, l_k, \mathrm{x}_{k+1})$ に対する一段先予測誤差の和の最小化

$$\sum_k \left(l_k + \beta \widehat{\mathrm{Q}}^{\pi}(\mathrm{x}_{k+1}, \pi(\mathrm{x}_{k+1})) - \widehat{\mathrm{Q}}^{\pi}(\mathrm{x}_k, \mathrm{u}_k) \right)^2 \tag{9.30}$$

- $\pi(\mathrm{x}_{k+1})$ を u_{k+1} としたり，u_k を $\pi(\mathrm{x}_k)$ としたりしないように注意
- $u_k = \pi(x_k)$ のもとでの閉ループデータである必要はない（過去に π とは無関係に取得・蓄積されたデータも利用可能）

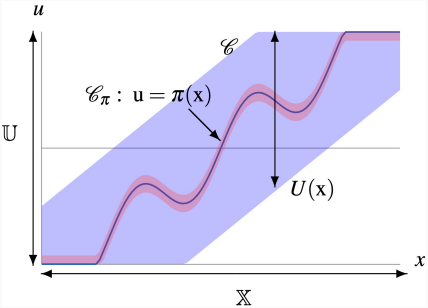
行動価値関数の推定 (6) TD 学習の例

確定的な状態フィードバック $u_k = \pi(x_k)$ のもとでのデータのみでは，$\mathcal{C}_\pi$ 付近の行動価値関数しか推定できない

- 雑音 z_k を付加して

$$u_k = \pi(x_k) + z_k \tag{9.32}$$

- $\varepsilon > 0$ に対して，u_k を確率 $(1-\varepsilon)$ で $\pi(x_k)$，確率 ε で一様乱数とする（ε-greedy）
- $T > 0$ に対して，$\pi(\cdot|\mathrm{x}) \propto \exp(-Q(\mathrm{x},\cdot)/T)$ とする（ボルツマン選択則）



行動価値関数を求める領域

行動価値関数の推定 (5) RLS フィルタの利用

特徴写像 $\phi : \mathbb{X} \times \mathbb{U} \to \mathbb{R}^n_{\mathrm{f}}$ を用いて，

$$\widehat{\mathrm{Q}}^{\pi}(\mathrm{x}, \mathrm{u}; \hat{\mathrm{p}}) := \hat{\mathrm{p}}^{\top} \phi(\mathrm{x}, \mathrm{u}),\ \hat{\mathrm{p}} \in \mathbb{R}^{n_{\mathrm{f}}} \tag{9.33}$$

- RLS フィルタを用いて最小化

$$\sum_k \left(l_k - \hat{\mathrm{p}}^{\top} \left\{ \phi(\mathrm{x}_k, \mathrm{u}_k) - \beta \phi(\mathrm{x}_{k+1}, \pi(\mathrm{x}_{k+1})) \right\} \right)^2 \tag{9.34}$$

- 制御対象が線形のとき，Q^{π} は $\begin{bmatrix} \mathrm{x}^{\top} & \mathrm{u}^{\top} \end{bmatrix}^{\top}$ の二次形式
  x, u の要素の二次単項式および定数 1 を並べたベクトル $\phi(\mathrm{x}, \mathrm{u})$ を用いて，

$$\mathrm{Q}^{\pi}(\mathrm{x}, \mathrm{u}) = \begin{bmatrix} \mathrm{x}^{\top} & \mathrm{u}^{\top} \end{bmatrix} \Upsilon^{\pi} \begin{bmatrix} \mathrm{x} \\ \mathrm{u} \end{bmatrix} + \gamma_{\pi} = \mathrm{p}^{\top} \phi(\mathrm{x}, \mathrm{u}) \tag{9.35}$$

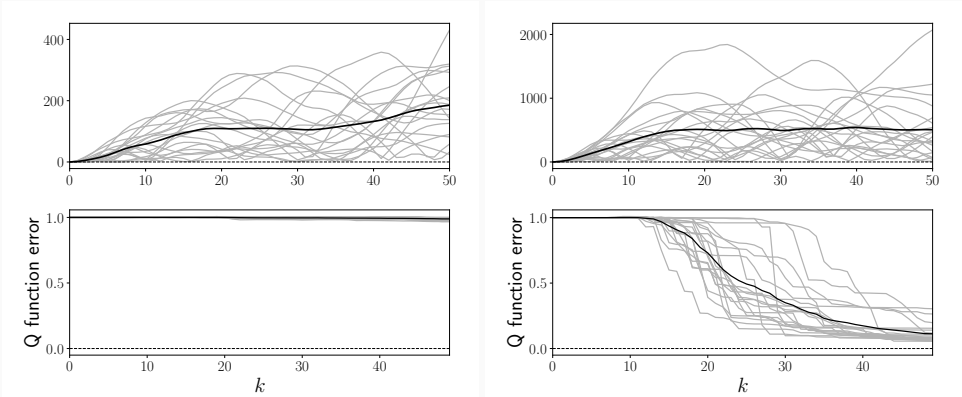
  p は Υ^{π} の対称性から $(n_x + n_u)(n_x + u_u + 1)/2 + 1$ 次元

行動価値関数の推定 (6) TD 学習の例

例 9.3.3 – TD 学習における探索と利用のトレードオフ

線形システムに対して $u_k = -K_0 x_k + z_k$（$\rho(A - BK_0) = 0.88$）で制御された際の軌道

上段：状態の変動，下段：学習誤差，左：$z_k \sim \mathcal{N}(0, 2^2)$，右：$z_k \sim \mathcal{N}(0, 10^2)$



アルゴリズム例 (1) TD 学習にもとづく方策反復法

ボルツマン選択則を用いる方策反復法と TD 学習の組み合わせ

アルゴリズム 7 TD 学習を用いた方策反復法

input k, $\tilde{\pi}$, l, x, $\mathscr{D}$ ▷ k, π_{k-1}, l_{k-1}, x_k

output π, u, $\mathscr{D}$ ▷ π_k, u_k

if $k \,\%\, k_{\mathrm{update}} = 0$ **then** ▷ 時刻 k_{update} ごとに制御則を更新

  $\mathscr{D} \cup \{(k, l, \mathrm{x})\}$ に対して式 (9.30) を最小化する $\widehat{Q}^{\pi}$ を求める ▷ $Q^{\tilde{\pi}}$ の推定

  for all x' in $\mathbb{X}$ **do**

    $\pi(\cdot|\mathrm{x}') \leftarrow c\exp(-\widehat{Q}^{\pi}(\mathrm{x}', \cdot)/T)$ ▷ 式 (9.16) とボルツマン選択則

  Initialize: $\mathscr{D}$ ▷ 必要に応じてをデータを初期化

else

  $\pi \leftarrow \tilde{\pi}$

$\mathrm{u} \sim \pi(\cdot|\mathrm{x})$ ▷ 制御入力を計算

$\mathscr{D} \leftarrow \mathscr{D} \cup \{(k, l, \mathrm{x}, \mathrm{u})\}$ ▷ データを保存

k_{update} ステップの学習ごとに制御則を更新

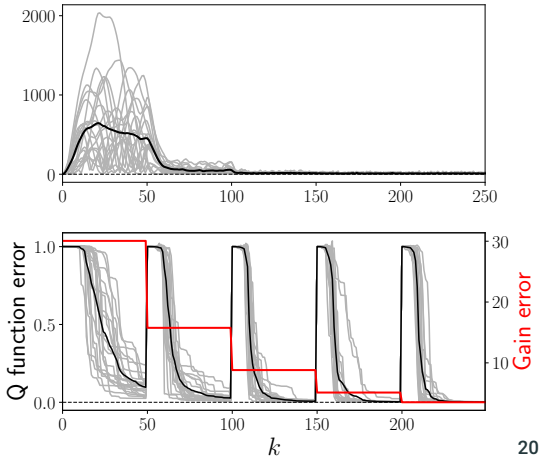
アルゴリズム例 (2) TD 学習にもとづく方策反復法の例

例 9.3.4 – TD 学習と方策反復法にもとづく強化学習

- $z_k \sim \mathcal{N}(0, 10^2)$，$k_{\text{update}} = 50$
- 制御則の更新ごとに初期化

$$\hat{\mathrm{p}}_k = 0,\ \Sigma_k = 10^4 \times I$$

上段：状態の変動
下段：相対近似誤差
Q 関数 $\|\hat{\Upsilon}_k - \Upsilon^{\pi_i}\|/\|\Upsilon^{\pi_i}\|$
ゲイン $\|K_k - K^*\|/\|K^*\|$



アルゴリズム例 (3) 制御則の更新間隔

k_{update} が長いと，行動価値関数の推定精度は向上するが，その間の制御性能は妥協

$$\sum_k \left(l_k + \beta \widehat{\mathrm{Q}}^\pi(\mathrm{x}_{k+1}, \pi(\mathrm{x}_{k+1})) - \widehat{\mathrm{Q}}^\pi(\mathrm{x}_k, \mathrm{u}_k) \right)^2 \tag{9.30}$$

- $\mathbb{X}$, $\mathbb{U}$ が有限集合であるマルコフ決定過程の場合，特定の点における $\widehat{\mathrm{Q}}^\pi$ の関数値を容易に更新できる（$a_k \in [0,1]$ はステップ幅）

$$\widehat{\mathrm{Q}}^\pi(\mathrm{x}_k, \mathrm{u}_k) \leftarrow \widehat{\mathrm{Q}}^\pi(\mathrm{x}_k, \mathrm{u}_k) + a_k \left(l_k + \beta \widehat{\mathrm{Q}}^\pi(\mathrm{x}_{k+1}, \pi(\mathrm{x}_{k+1})) - \widehat{\mathrm{Q}}^\pi(\mathrm{x}_k, \mathrm{u}_k) \right)$$

- ベルマン方程式 $\mathrm{Q}^*(\mathrm{x}, \mathrm{u}) = (\ell + \beta \mathscr{A}_x[\mathsf{V}^*])(\mathrm{x}, \mathrm{u})$ の解 Q^* を求める場合は，

$$\widehat{\mathrm{Q}}^*(\mathrm{x}_k, \mathrm{u}_k) \leftarrow \widehat{\mathrm{Q}}^*(\mathrm{x}_k, \mathrm{u}_k) + a_k \left(l_k + \beta \inf_{\mathrm{u} \in U(\mathrm{x}_{k+1})} \widehat{\mathrm{Q}}^*(\mathrm{x}_{k+1}, \mathrm{u}) - \widehat{\mathrm{Q}}^*(\mathrm{x}_k, \mathrm{u}_k) \right)$$

- このように推定値をもとに別の推定値を構築することを**ブートストラップ**（bootstrap）とよぶ．

アルゴリズム例 (4) SARSA と Q 学習

- **SARSA 法**：Q^{π} を推定し，制御則は $\pi(\mathrm{x}) = \underset{\mathrm{u}\in U(\mathrm{x})}{\arg\min}\ \widehat{\mathrm{Q}}^{\pi}(\mathrm{x},\mathrm{u})$．方策オン型．
- **Q 学習**（Q learning）：Q^{*} を推定し，制御則は $\pi^{*}(\mathrm{x}) = \underset{\mathrm{u}\in U(\mathrm{x})}{\arg\min}\ \widehat{\mathrm{Q}}^{*}(\mathrm{x},\mathrm{u})$．方策オフ型．
- Q 学習の方が楽観的．

アルゴリズム 9 SARSA と Q 学習

input $\tilde{\mathrm{x}}$, $\tilde{\mathrm{u}}$, l, x, a, $\widehat{Q}$ ▷ $(\mathrm{x}_{k-1}, \mathrm{u}_{k-1}, l_{k-1}, \mathrm{x}_k)$, ステップ幅 $a_k > 0$

output x, u, $\widehat{Q}$ ▷ x_k, u_k

if type = SARSA **then**

  $\mathrm{u}^{\star}$ を確率 $(1-\varepsilon)$ で $\underset{\mathrm{u}\in U(\mathrm{x})}{\arg\min}\ \widehat{Q}(\mathrm{x},\mathrm{u})$，確率 ε で一様乱数 ▷ $\mathrm{u}^{\star} \sim \pi(x_k)$

else if type = Q-Learning **then**

  $\mathrm{u}^{\star} \leftarrow \underset{\mathrm{u}\in U(\mathrm{x})}{\arg\min}\ \widehat{Q}(\mathrm{x},\mathrm{u})$ ▷ $\mathrm{u}^{\star} := \underset{\mathrm{u}\in U(\mathrm{x}_k)}{\arg\min}\ \widehat{Q}(\mathrm{x}_k,\mathrm{u})$

$\widehat{Q}(\tilde{\mathrm{x}},\tilde{\mathrm{u}}) \leftarrow \widehat{Q}(\tilde{\mathrm{x}},\tilde{\mathrm{u}}) + a(l + \beta\widehat{Q}(\mathrm{x},\mathrm{u}^{\star}) - \widehat{Q}(\tilde{\mathrm{x}},\tilde{\mathrm{u}}))$ ▷ 式 (9.36) または式 (9.37)

u を確率 $(1-\varepsilon)$ で $\underset{\mathrm{u}\in U(\mathrm{x})}{\arg\min}\ \widehat{Q}(\mathrm{x},\mathrm{u})$，確率 ε で一様乱数 ▷ ε-greedy に制御入力を計算

アルゴリズム例 (5) TD 学習にもとづく方策勾配法

actor-critic 法

1. 制御則 π_θ に従って，$\bar{k}$ ステップの制御を $\bar{s}$ 巡行った標本 $(\mathrm{x}_k^s, \mathrm{u}_k^s)_{k=0,\dots,\bar{k}}^{s=1,\dots,\bar{s}}$ を取得
2. 制御則評価の役割を担う行動価値関数 (critic ともよばれる) をもとに，

$$\nabla_\theta J(\pi_\theta) \simeq \frac{1}{\bar{s}} \sum_{s=1}^{\bar{s}} \sum_{k=0}^{\bar{k}} \beta^k (\nabla_\theta \log(\pi_\theta(\mathrm{u}_k^s|\mathrm{x}_k^s))) \left(\widehat{\mathrm{Q}}^{\pi_\theta}(\mathrm{x}_k^s, \mathrm{u}_k^s) - b(\mathrm{x}_k^s) \right) \tag{9.38}$$

   から標本近似で勾配を計算

3. 勾配法により制御則 (actor ともよばれる) を改善

- 制御則 π_θ は設計者が決定できる関数であるため，その勾配も既知
- **ベースライン** b は期待値には影響せず，分散を減らすことができ，b を状態価値関数とした $\mathrm{Q}^\pi - \mathrm{V}^\pi$ はアドバンテージ関数とよばれる．

第 9 章

1. 最適制御と強化学習

2. モデルベースト設計
    - ベルマン方程式
    - 方策勾配法

3. モデルフリー設計
    - 行動価値関数の推定
    - アルゴリズム例