制御システム論分野研究会資料

2020 年 10 月 8 日

M2 竹内 維吹

進捗報告 10.8

1 前提

入力アフィン系のセルフトリガー制御を考える.

$$\dot{s} = f(s) + g(s)a \tag{1}$$

1.1 倒立振子による実験

倒立時の振子の角度を $\theta = 0$ とし, 加えられる入力が $A = [-10\mathrm{N}\cdot\mathrm{m}, 10\mathrm{N}\cdot\mathrm{m}]$ と制限されるような倒立振子を考える. この倒立振子のダイナミクスは, 以下のように与えられる.

$$\frac{\mathrm{d}}{\mathrm{dt}}\begin{pmatrix}\theta\\ \dot{\theta}\end{pmatrix} = \begin{pmatrix}\dot{\theta}\\ \frac{3g}{2l}\sin\theta + \frac{3}{ml^2}a\end{pmatrix} \tag{2}$$

コンピュータで強化学習を行う場合, これを離散化したシステムについて計算を行う必要がある. 上記の状態方程式を離散化すると以下のようになる.

$$\theta_{t+1} = \theta_t + \dot{\theta}_t\delta_t + \frac{3g}{2l}\sin\theta_t\delta_t^2 + \frac{3}{ml^2}a\delta_t^2 \tag{3a}$$

$$\dot{\theta}_{t+1} = \dot{\theta}_t + \frac{3g}{2l}\sin\theta_t\delta_t + \frac{3}{ml^2}a\delta_t \tag{3b}$$

ただし, δ_t は離散化定数である.

2 現状確認

2.1 セルフトリガー制御

図 1 のような制御系を考える.

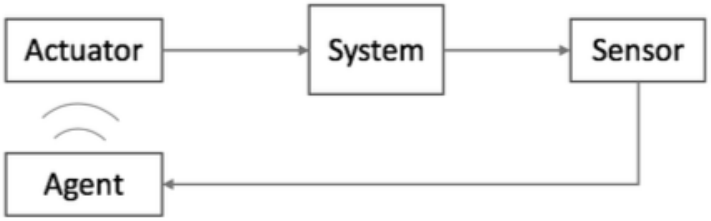

図 1　制御系

これに対するフィードバック制御を考える. 状態変数 s を観測してアクチュエータに入力信号を送信することを「インタラクション」と呼ぶと, セルフトリガー制御では, 連続的なインタラクションは行わずに, 次のイ

ンタラクションを何秒後に行うかをエージェントが決定する．それを数式上で表すため, エージェントの制御則 $\pi(s)$ は 2 つの要素からなるベクトル値関数であるとし, 1 つ目の要素はアクチュエータに送信する入力 a, 2 つ目の要素は次にインタラクションを行うまでの時間間隔 τ (s : 秒) を表すものとする. また, 次のインタラクションを行う時刻までは 1 つ前のインタラクションで送信した入力 a を加え続けるものとする (ZOH 制御).

2.2 目標点の確認

研究を通しての目標は「安全性を確保しながら, 最適セルフトリガー制御則 π^* の強化学習を実現させること」である。ここで

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, J(\pi) \tag{4}$$

$$J(\pi) = \mathbb{E}_{s_0 \in d_0}[V^\pi(s_0)] \tag{5}$$

$$V^\pi(s_0) = \sum_{i=0}^{\infty} \gamma^i C_i^\pi \tag{6}$$

$$C_i^\pi = -\int_{T_i}^{T_{i+1}} s(t)^\top Q s(t) \mathrm{d}t + \tau_i a_i^\top R a_i + \lambda \tau_i,\ T_i = \sum_{l=0}^{i} \tau_l \tag{7}$$

であり, π_1, π_2 は π の第 1, 第 2 成分である．また, i はインタラクションの回数を示し, a_i, τ_i はそれぞれ i 回目のインタラクションでの方策 π の出力である．

さて, 一般的に強化学習では, 1 ステップ 1 ステップの行動の良し悪しを評価して方策を更新していく．インタラクションとインタラクションの間の区間を「インターバル」と呼ぶと, 式 (6) より, この問題は各インターバルを 1 ステップとした強化学習問題であると考えることができる．

以下では方策 π を θ でパラメトライズし, $\theta^* = \underset{\theta}{\operatorname{argmax}}\, J(\pi_\theta)$ を解くことによって $\pi^* = \underset{\pi}{\operatorname{argmax}}\, J(\pi)$ を得るものとする．方策勾配を用いた強化学習では $\nabla_\theta J(\pi)$ を用いて θ^* を求める．その際方策勾配 $\nabla_\theta J(\pi)$ の近似のため, 実環境とのインタラクションによって得られたデータ組 $\{s, a, r, s'\}$ を用いる．「学習中の安全」という言葉を, 「このデータ組の収集を決められた安全領域 $\mathcal{C}$ の内部でのみ行うこと」と定義する．

2.3 実現可能性の検証:サンプル値系での実験

上記の目標を達成する見込みがあるのかを検証するために, サンプル値系での実験を行う．サンプル値系では, セルフトリガー制御と同様に連続的なインタラクションは行わない．セルフトリガー制御との違いは, インタラクションの間隔がエージェントによって状態 s 依存で決定するのではなく, 制御問題の設定として定数 t_{int} で与えられる点である. したがってサンプル値系での制御方策 π_{sample} は, アクチュエータに送信する入力信号 a のみを出力する関数として与える．

サンプル値系での実験により, $t_{\mathrm{int}} = 0.001(s)$ のサンプル値系での最適方策

$$\begin{cases} \pi_{\mathrm{sample},1} = -\underset{\pi_{\mathrm{sample}}}{\operatorname{argmax}} \displaystyle\sum_{i=0}^{\infty} \int_{it_{\mathrm{int}}}^{(i+1)t_{\mathrm{int}}} s(t)^\top Q s(t) \mathrm{d}t + t_{\mathrm{int}} a_i^\top R a_i \\ a_i = \pi_{\mathrm{sample}}(s(it_{\mathrm{int}})) \end{cases} \tag{8}$$

を初期方策として, $t_{\mathrm{int}} = 0.002(s)$ のサンプル値系での最適方策

$$\begin{cases} \pi_{\mathrm{sample},2} = -\underset{\pi_{\mathrm{sample}}}{\operatorname{argmax}} \displaystyle\sum_{i=0}^{\infty} \int_{it_{\mathrm{int}}}^{(i+1)t_{\mathrm{int}}} s(t)^\top Q s(t) \mathrm{d}t + t_{\mathrm{int}} a_i^\top R a_i \\ a_i = \pi_{\mathrm{sample}}(s(it_{\mathrm{int}})) \end{cases} \tag{9}$$

を学習中の安全性を満たしながら学習できるかを検証する．

2.4　セルフトリガー制御への発展

前節での検証によって, インタラクション間隔を大きくしても安全強化学習を行うことが可能であることを確認できたとする. サンプル値系での制御則は入力信号 a のみを出力する関数であったので, 入力信号 a とインタラクション間隔 τ の二つの要素を出力する必要があるセルフトリガー制御の初期方策として方策 $\pi_{\mathrm{sample},1}$ をそのまま用いることはできない.

そこで代替策として,

$$\begin{cases} \pi_1(s) = \pi_{\mathrm{sample},1}(s) \\ \pi_2(s) = 0.001 \end{cases} \tag{10}$$

とする方策 π_{init} をセルフトリガー制御の強化学習のための初期方策として用いる.

3　安全性の定義

3.1　インタラクション間隔 τ の安全性

ECBF(後から書きます)

3.2　入力信号 a の安全性

強化学習ではデータの収集に環境とのインタラクションを行う必要がある. DDPG と呼ばれるアルゴリズムは方策オン型の強化学習とよばれ, データの収集方策に学習中の暫定最適方策を用いる. したがって, 学習初期の方策では安全性が保証されないことがしばしばある. この課題を解決するために, 制御バリア関数を用いる.

関数 $h(s)$ が以下の条件を満たす時, システム (1) に対する制御バリア関数であるという.

$$\sup_{a \in A} \left\{ \frac{\partial h}{\partial s}(f(s) + g(s)a) + K(h(s)) \right\} \geq 0 \tag{11}$$

ただし, $K(s)$ はクラス K 関数である.

さて, 2.2 節にて登場した安全領域 $\mathcal{C}$ を

$$\mathcal{C} = \{ s \in S \mid h(s) \geq 0 \} \tag{12}$$

として与える. このとき $h(s)$ が制御バリア関数であるならば, 状態 $s \in C$ を初期状態とした時, それ以降の全時刻において, 状態 s が $s \in C$ を満たすようにする入力が存在することを保証する. そのような入力集合は現時刻での状態 s に依存し,

$$U(s) = \left\{ a \in A \mid \frac{\partial h}{\partial s}(f(s) + g(s)a) + K(h(s)) \geq 0 \right\}, \forall s \in C \tag{13}$$

としてその集合を与える.

学習中の安全性を確保するために, 図 2 のようにエージェントの出力を $U(s)$ の要素に射影するレイヤーを設ける.

当面は, エージェントの出力 a_π に最も近い $U(s)$ の元への射影を考える.

4　今後の方針

4.1　シミュレーション環境の構築

さて, 制御バリア関数による安全性保証は連続システムに対して行われるものである. また, セルフトリガー制御則の第 2 成分 τ の出力ロジックを勾配によって更新できるように, τ を連続値として扱う必要がある. しかし, コンピュータ上でシミュレーションを行うには (1) を時間に関して離散化を行わなくてはならない.

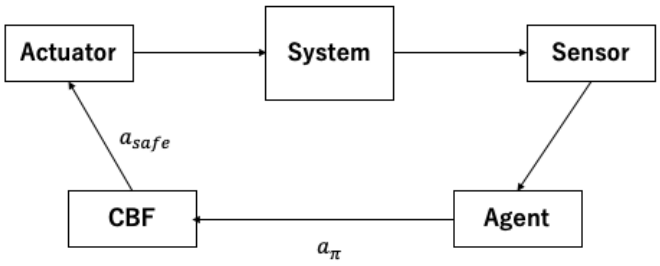

図 2　制御系

そこで, インタラクション間隔 τ を整数個に等間隔に分割し, その時間幅を用いて離散化を行う. この時, 離散化幅は $0.001(s) \sim 0.005(s)$ になるように分割数を調整する. δ_t を上から抑える理由は離散化誤差を抑えるためである. (τ を) 下から抑えるのは, コンピュータ上で $\frac{1}{\infty} = 0$ となってしまうからで, それを回避するためである.

ここで, 式 (7) のインターバル報酬 C_i^π が定積分を用いて表されているため, これをシミュレーション環境で近似する手法を考える. ダイナミクス (1) の離散化幅 δ_t の離散近似システムが

$$s_{t+1} = f_d(s_t, \delta_t) + g_d(s_t, \delta_t)a_t \tag{14}$$

と書かれているとする. インタラクション間隔 τ を N 分割した時, $\delta_t = \frac{\tau}{N}$ を用いて

$$C_i^\pi \approx -\delta_t \sum_{k=0}^{N} s_{n_i+k}^\top Q s_{n_i+k} + \tau_i a_i^\top R a_i + \lambda \tau_i \tag{15}$$

と近似する. ここで s_{n_i} は i 回目のインタラクションを行った瞬間の状態変数 $s(T_i)$ と同じ値が代入されるものとする.

4.2　サンプル値系での安全性確保ロジックの構築

2.3 節で記述した通り, t_{int} が $0.001(s)$ の最適方策 $\pi_{\text{sample},1}$ を初期値として, $0.002(s)$ の最適方策 $\pi_{\text{sample},2}$ を安全強化学習できるのか検証する. 本節ではその安全性の確保方法についてもう少し掘り下げて議論する.

ECBF を用いることによって, 入力 a に対するインタラクション間隔 τ の限界を与えることができる. その値を $\tau_{\max}(a)$ と書く. もし $\tau_{\max}(a) < 0.002$ であるなら, CBF を用いて $U(s)$ の元 a_{safe} を選び $\tau_{\max}(a_{\text{safe}}) \geq 0.002$ となれば, 次のインタラクションまで a_{safe} を加え続けても状態変数が $\mathcal{C}$ を出ていくことはない. しかし $\forall a \in U(s)$ に対して $\tau_{\max}(a) < 0.002$ であるなら, 次のインタラクションを $0.001(s)$ 後に行うことで, 安全性を確保する必要がある. ($0.001 \leq \tau_{\max}(a) < 0.002$ を仮定)

ここまでの議論を整理すると, 「サンプル値系における安全性保証」とは「1. 入力信号の安全性, 2. サンプル間隔の安全性」を保証することになる. これらが行われることを回避する学習方法については今後検討する.

5　セルフトリガー制御の強化学習

5.1　問題設定の再確認

式 (10) で表される方策を初期方策として, 式 (4) で表される最適方策を学習したい. その際 CBF や ECBF を用いて安全性を確保する. そのために以下のものが Given であると仮定する.

- 初期方策 π_{init}
- 安全領域 $\mathcal{C}$, システム (1) 指数制御バリア関数 ECBF: $h(\mathcal{C}) \geq 0$

5.2 安全確保手法

学習中 (データ収集中), $\pi(s) = \begin{bmatrix} a & \tau \end{bmatrix}$ と出力された場合を考える. ECBF を用いると, 状態 s が安全領域 $\mathcal{C}$ を出ていかないための, 入力 a を加え続けられる最大の時間 $\tau_{\max}(a)$ を与えることができる. 次のインタラクションまでの時間間隔を $\min\{\tau, \tau_{\max}(a)\}$ とすることで, 安全性を確保する.

5.3 生じると予想される課題点

5.3.1 $\mathcal{C}$ 内での経験の均一性

方策 π の正しい改善には critic ネットワークの出力である Q^π の近似精度が必要不可欠である. また, DDPG は off-policy アルゴリズムであり, 過去のデータを保存した experience replay から作成したミニバッチを用いて critic の更新に用いる TD 誤差の計算を行う. 安全強化学習においては, $\mathcal{C}$ の縁での挙動が重要であるため, 作成するミニバッチには原点付近の (s, a) に偏らずに, 状態変数に対して満遍なく含まれることが望ましい. 現状は experience replay から等確率でミニバッチを作成しているので, $\mathcal{C}$ の縁での経験が少なく, 原点付近ばかりを経験していると, $\mathcal{C}$ の縁付近での Q^π は近似精度は高くないことが予想される.

そこで, experience replay を原点付近の s と, $\mathcal{C}$ の縁付近とで分けて, 均等にミニバッチを作成するのはどうかと考える.

5.3.2 シミュレーションの離散化による影響

要検討

6 セルフトリガー制御の強化学習特有の特徴に関する調査

6.1 準備: critic の学習速度の確認

actor ネットワークのパラメータは, 以下の方策勾配を用いて更新される.

$$\nabla_{\theta^\mu} J(\theta^\mu) = \mathbb{E}_{s\sim\rho^{\theta^\mu}}[\nabla_{\theta^\mu}\mu(s|\theta^\mu)\nabla_a Q^{\theta^\mu}(s,a)|_{a=\mu(s|\theta^\mu)}] \tag{16}$$

ここで, $\rho^{\theta^\mu}(s) = \int_S \sum_{t=0}^{\infty} \gamma^t d_0(s_0)\Pr(s_0 \to s, k, \theta^\mu)\mathrm{d}s_0$ は, 割引分布という (時間割引された, 全時刻における状態 s の出現割合みたいなもの (?)).

式 (16) から, Q^{θ^μ} が既知でなくては, 正しい方策勾配が計算できないことがわかる. actor-critic は policy iteration アルゴリズムの 1 つであり, 方策 μ の更新と Q^μ を交互に更新し続ける. DDPG では actor と critic を 1 ステップにつき 1 回のみ更新する (つまり, critic の収束を待たずして次のステップに進む). そこで, actor を固定したときに, 実際には critic の学習にはどの程度のステップ数が必要なのかを知っておきたく, その実験を行った.

ここでは, 3 つの方策 μ_i に対する Q^μ の学習を実験する. ただし

$$\pi_1(s) = \begin{bmatrix} \mathrm{lqr}_{0.001}(s) \\ 0.001 \end{bmatrix} \tag{17}$$

$$\pi_2(s) = \begin{bmatrix} \mathrm{lqr}_{0.005}(s) \\ 0.1 \end{bmatrix} \tag{18}$$

$$\pi_3(s) = \begin{bmatrix} \mathrm{a}(s) \\ \tau(s) \end{bmatrix} \tag{19}$$

$$\tag{20}$$

とする. (π_3 に関しては, 完全にランダムに初期化をした NN で表現する, という意味で記載した.)

学習は $\theta \in [-\pi, \pi], \dot{\theta} \in [-\pi, \pi]$ を 50000 点発生させて, $\{s, \mu_i(s), r, s'\}$ の組を保存した replay buffer から mini batch を作成して行った. その時の loss(mini batch の TD 誤差の平均) の値の変化を追った時のプロッ

トを図 3 に示す.

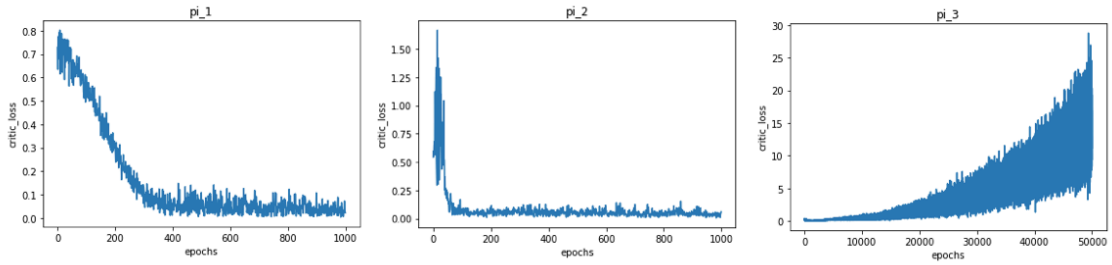

図 3　critic の学習

図 3 より, 方策によって学習の様子が全く異なることが見てとれる. 問題なのは π_3 の学習で, これは loss が指数的に発散している. 学習率は他の 2 つに比べてかなり小さく取っての結果なので, 原因は最適化アルゴリズム以外にあると考える.

6.2　方策勾配の大きさによる, 入力信号 a と通信間隔 τ の学習率の考察

図 4 のように actor ネットワークを, $a(s)$ と $\tau(s)$ を分離したモデルとして用いる.

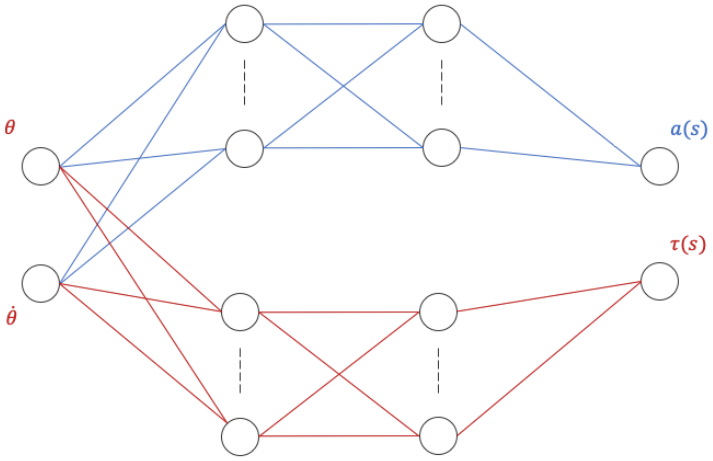

図 4　actor ネットワーク

actor ネットワークの全パラメータを θ^μ, $a(s)$ と $\tau(s)$ を表す部分ネットワークのパラメータをそれぞれ θ^a, θ^τ とする. このとき, 方策勾配 $\nabla_{\theta^\mu} J(\theta^\mu)$ もそれぞれのパラメータが表す部分を分けて, $\nabla_{\theta^a} J(\theta^\mu), \nabla_{\theta^\tau} J(\theta^\mu)$ としたとき, パラメータの更新則は以下のようになる.

$$\theta^\mu \leftarrow \theta^\mu - \alpha_a \nabla_{\theta^a} J(\theta^\mu) - \alpha_\tau \nabla_{\theta^\tau} J(\theta^\mu) \tag{21}$$

研究の 1 つの成果物として, α_a, α_τ の理論的な大きさの設定方法について言及できればと考えている. そのため, 勾配 $\nabla_{\theta^a} J(\theta^\mu), \nabla_{\theta^\tau} J(\theta^\mu)$ の大きさをそれぞれみてみる. (θ^τ, θ^a はそれぞれ同じ長さのパラメータベクトルにしています.)

6.1 節で critic が収束した π_1, π_2 について, その方策勾配 $\nabla_{\theta^a} J(\theta^\mu), \nabla_{\theta^\tau} J(\theta^\mu)$ のノルムを計算すると, π_1 での値は

$$\|\nabla_{\theta^a} J(\theta^\mu)\| = 66.70076 \tag{22}$$
$$\|\nabla_{\theta^\tau} J(\theta^\mu)\| = 8.730332e-05 \tag{23}$$

π_2 での値は,

$$\|\nabla_{\theta^a} J(\theta^\mu)\| = 1.9210727 \tag{24}$$
$$\|\nabla_{\theta^\tau} J(\theta^\mu)\| = 4.414095e-05 \tag{25}$$

となった. このことから, θ^τ の微小変化が J に与える影響が非常に小さいことが見てとれる. 従って, π_1, π_2 は θ^τ に関する local maximam にある可能性が高い.

また, π_3 に関しても調べたいところであるが, 正しい Q 関数が得られていないので計算はできない. おそらく $\|\nabla_{\theta^\tau} J(\theta^\mu)\|$ が 2 つの方策の場合に対してかなり大きいのではないかと考えている. これは $\tau(s)$ がどちらかに張り付くことからの予想である.

以上のことから, 初期方策として π_1, π_2 を選んだ場合, $\tau(s)$ に関しての局所最適解となっている特性と, これらの間にある方策 (例えば π_3) での $\|\nabla_{\theta^\tau} J(\theta^\mu)\|$ が大きいという特性から, 強化学習の途中で $\tau(s)$ が大きく揺れてしまうことが予想される. そのため, 学習が進むにつれて減衰し, 勾配のノルムに反比例するような学習率 α_τ を導入するのはどうかと考える.

参考文献

[1] G. Yang, C. Belta, and R. Tron. “Self-triggered Control for Safety Critical Systems Using Control Barrier Functions.” *In American Control Conference (ACC) Philadelphia, USA*, 2019.