制御システム論分野研究会資料

2020 年 11 月 9 日

M2 竹内 維吹

進捗報告 11.9

1 はじめに

近年, 機械学習分野で注目を集めているトピックとして強化学習というものがある. これは, ロボットなどが環境とのインタラクションを通じて, 最適な制御方策を見つけ出していくというものである. その中でも, 強化学習にニューラルネットワークを組み込んだ深層強化学習は, その適応能力の高さから多くの研究がなされている. しかしながら, 深層強化学習にはハイパーパラメータへの強い依存性など, 依然として課題点が多くまだまだ発展途上にある.

本稿では, 対象となる問題を最適セルフトリガー制御問題というものに絞り, 強化学習の課題点に対して有効な特性の調査に向けた準備を行う. まず, 深層強化学習の基礎についての確認を行った後, 最適セルフトリガー制御問題を定式化し, その強化学習に特有の課題点を抽出する. 最後に, その課題点の解決に向けた今後の展望についても簡単に述べる.

2 方策勾配を用いた強化学習

2.1 強化学習の基礎知識

マルコフ決定過程 M を $M = \{S, A, T, d_0, r, \gamma\}$ として与える。ここで S, A はそれぞれ状態, 行動集合, $T(s^{'}|s, a)$ は状態遷移確率を表す. また $d_0, r(s, a), \gamma \in [0, 1]$ はそれぞれ初期状態分布, 報酬, 割引率を示す.

さて, 強化学習の目的は

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, J(\pi) \tag{1}$$

を求めることである. ここで,

$$V^\pi(s) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|_{a_t=\pi(s_t)}, s_0 = s \tag{2}$$

$$J(\pi) = \mathbb{E}_{s_0 \sim d_0}[V^\pi(s_0)] \tag{3}$$

であり, $J(\pi), V^\pi(s)$ をそれぞれ評価関数, (状態) 価値関数とよぶ.

強化学習を解析するツールとして有用な関数として Q 関数がある.

$$\begin{aligned} Q^\pi(s, a) &= r(s, a) + \gamma \sum_{t=1}^{\infty} \gamma^t r(s_t, a_t)|_{a_t=\pi(s_t)} \\ &= r(s, a) + \gamma V^\pi(s') \end{aligned} \tag{4}$$

式 (4) より, Q 関数は開始時刻において自由に行動 a を選択して, 次ステップから方策 π に従った時の価値を表す. したがって, Q 関数は行動価値関数という別名がある.

2.2 方策反復法

式 (1) を達成するためのアルゴリズムとして方策反復法というものがある. これは以下の 2 つのステップを繰り返すというものである.

1. 方策評価: 行動価値関数 $Q^\pi(s,a)$ を求め (または, 近似す) る.
2. 方策改善: 求めた $Q^\pi(s,a)$ に従って, $\pi(s) = \underset{a}{\operatorname{argmax}} Q^\pi(s,a)$ と方策を更新する.

以上の 2 ステップを繰り返すことで最適方策 π^* が得られることが知られている.(方策改善定理)

2.3 状態空間, 行動空間の特性に合わせたアルゴリズム

状態空間も行動空間も離散値をとる場合, $Q^\pi(s,a)$ をテーブルに保存しておくことで, 前節で登場した $\pi(s) = \underset{a}{\operatorname{argmax}} Q^\pi(s,a)$ を容易に求めることができる.

では, 状態空間が連続値の場合はどうか. 状態 s が無限種類の値をとることになるため, テーブルに保存することができない. そこで Minh ら [**?**] は, $Q^\pi(s,a)$ をニューラルネットワークを用いてパラメトライズして近似するアプローチをとった. ここでも, 行動空間は離散であるため $\underset{a}{\operatorname{argmax}} Q^\pi(s,a)$ を求めることは可能である.

最後に, 状態空間も行動空間も連続値である場合は $\underset{a}{\operatorname{argmax}} Q^\pi(s,a)$ を求めるのに膨大なコストがかかるという問題点がある. したがって, これまでは方策 π は Q 関数によって定めていたが, 両空間が連続値の場合にはこのアプローチはとれない. そこで, 方策関数も独立に π_θ のようにパラメトライズした関数として用意し, パラメータ θ を勾配法などで更新する手法が取られることが多い.

2.4 方策勾配による方策関数のパラメータの更新

Silver ら [**?**] は, 方策が $\pi(s)$ のように決定論的に定めるものとして与えた場合に, 評価関数 $J(\pi_\theta)$ に対する勾配を計算する手法を発見した. この勾配は決定論的方策勾配 (DPG:Deterministic Policy Gradient) とよばれ, 以下のように計算することができる.

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi}[\nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s)}] \tag{5}$$

ただし,

$$\rho^\pi(s) = \int_S \sum_{t=0}^{\infty} \gamma^t d_0(s_0) \Pr(s_0 \to s, t, \pi) \mathrm{d}s_0 \tag{6}$$

を割引分布という.

この方策勾配を深層強化学習の枠組みに取り入れたアルゴリズムが DDPG(Deep DPG)[**?**] である. これは Actor-Critic 構造を採用しており, Q^{π_θ} を近似する critic ネットワーク $Q(s,a|\omega)$ と, 方策 π を表現する actor ネットワーク $\pi(s|\theta) = \pi_\theta$ をそれぞれ学習する手法である. 以下に, この近似勾配を用いた actor と critic の更新アルゴリズムを記す.

DDPG はミニバッチ学習を用いる. 過去の経験データ (s_t, a_t, r_t, s_{t+1}) を保存しておき, その中から N 個のデータを取り出し (ミニバッチ, E と記載する), そのデータ集合に対する最適化を行う. まず, critic の更新から記す. critic の目的は Q^π を近似することである. Q 関数は式 (4) のように分解することが可能で有るため, $Q(s,a|\omega)$ もこれを満たすように更新すればよい. そのために TD(Temporal Difference) 誤差

$$\mathrm{TD} = Q(s,a|\omega) - \{r(s,a) + \gamma Q(s,\pi(s)|\omega)\} \tag{7}$$

が最小となる方向に ω を更新する. 全ての (s,a) についてこれを一度に行うことはできないので, 作成したミニバッチ E に対する平均二乗誤差

$$Loss = \frac{1}{N} \sum_{s\in E} \mathrm{TD}^2 \tag{8}$$

を Loss 関数として, この Loss 関数を減らすようにアルゴリズムを働かせる.

さて, 上記の critic の更新方法は教師あり学習そのものである. 従ってミニバッチに含まれるデータは i.i.d. であることが要求される. もしミニバッチ E がエージェントが経験した直近の N ステップのデータを用いると, これらは独立ではなくなってしまう. 従って, エージェントは環境とのインタラクションによって得られた経験データを experience replay に保存しておき, そこから無作為に N このデータを選びとる手法でデータの分散を上げている.

次に actor の更新を記す. actor は方策関数 $\pi(s)$ を表現するものであり, パラメータの更新には方策勾配を用いる. ただし, DDPG では式 (5) のように正しい Q 関数を用いることができないので,

$$\mathbb{E}_{s\sim\rho^\pi}[\nabla_\theta \pi_\theta(s) \nabla_a Q(s,a|\omega)|_{a=\pi_\theta(s)}] \simeq \nabla_\theta J(\pi_\theta) \tag{9}$$

のように critic ネットワークを用いて近似した方策勾配を用いる. さらに, 期待値に関して,

$$\mathbb{E}_{s\sim\rho^\pi}[\nabla_\theta \pi_\theta(s) \nabla_a Q(s,a|\omega)|_{a=\pi_\theta(s)}] \simeq \frac{1}{N}\sum_{s\in E}[\nabla_\theta \pi_\theta(s) \nabla_a Q(s,a|\omega)|_{a=\pi_\theta(s)}] \tag{10}$$

という近似も行っている. 従って, この近似勾配 $g = \frac{1}{N}\sum_{s\in E}[\nabla_\theta \pi_\theta(s) \nabla_a Q(s,a|\omega)|_{a=\pi_\theta(s)}]$ は critic の近似精度とミニバッチの分布によって大きく性能を落としてしまう可能性があり, 大きな問題点であると言える.

3 最適セルフトリガー制御問題に対する強化学習

3.1 セルフトリガー制御

図 1 のような制御系を考える.

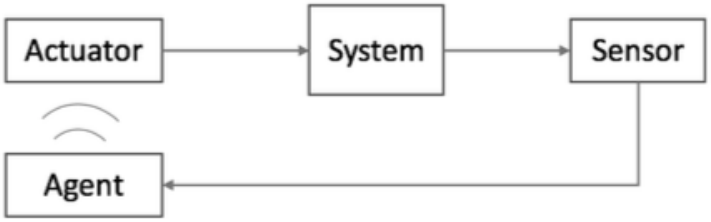

図 1 制御系

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).

3.2 最適セルフトリガー制御

最適なセルフトリガー制御則 π^* を, 以下のように定義する.

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, J(\pi) \tag{11}$$

$$J(\pi) = \mathbb{E}_{s_0\in d_0}[V^\pi(s_0)] \tag{12}$$

$$V^\pi(s_0) = \sum_{i=0}^{\infty} \gamma^i r_i^\pi \tag{13}$$

$$r_i^\pi = -\int_{T_i}^{T_{i+1}} s(t)^\top Q s(t) \mathrm{d}t + \tau_i a_i^\top R a_i + \lambda \tau_i,\ T_i = \sum_{l=0}^{i} \tau_l \tag{14}$$

ここで, 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 の出力であるとする.

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 (13) 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.

4 数値実験による課題点の抽出

上記の問題は, 一般的な強化学習のフレームワークにそのまま落とし込むことができるため, 数値実験を行うことが可能である. 数値実験の結果を通して, 最適セルフトリガー制御の強化学習にどのような課題点があるのかを考察していく.

実験環境は Open-AI Gym の pendulum で, 初期状態が $\theta \sim N(0, \pi), \dot{\theta} \sim U(-\pi, \pi)$ となるような環境で行った. また replay buffer の分散を上げるため, 10 秒間の制御を 1 エピソードと定義し, エピソードを重ねることで経験データを増やしていく. また, pendulum は以下のようなダイナミクスに従っており入力アフィン系の非線型システムである.

$$\frac{\mathrm{d}}{\mathrm{dt}} \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{\theta} \\ \frac{3g}{2l} \sin\theta + \frac{3}{ml^2} a \end{pmatrix} \tag{15}$$

4.1 初期方策

学習の初期方策として

$$\begin{cases} a(s) = a^*_{0.001}(s) \\ \tau(s) = 0.001 \end{cases} \tag{16}$$

とする方策 π_{init} を用いる. ただし, $a^*_{0.001}(s)$ はシステム (15) を, 離散化幅 $\delta_t = 0.001$ として離散化したシステムを原点付近で安定化する制御則として与える.

4.2 学習の様子からみる勾配法の進捗の確認

式 (14) より, τ を大きくするモチベーションは λ に大きく依存するため, それを様々な値にすることでど学習過程にどのような変化が見られるのかを確認してみた. ここでは, λ を $0.05, 0.5, 5, 50$ に設定し, それぞれ 3 回, 200000 ステップ (インタラクションの回数) の強化学習を行った. また, τ は $0.001 \sim 0.1$ の範囲に制限している.

まずそれぞれの学習タスクの中で, 1 エピソードの間の τ の平均値の変化を見てみる.

ここで注意したいのは, τ によって 200000 回のインタラクションに含まれるエピソード数が異なる点である. 例えば τ の学習が進まずにずっと $\tau = 0.001$ であった場合, 10 秒間の制御の間には 10000 回のインタラクションが行われる. したがって 200000 回のインタラクションの間には 20 エピソードしか含まれない. 逆に, 初期から τ の学習が進み $\tau = 0.1$ となることが増えてくると, エピソード数は増えてくる. 図 2 の x 軸の範囲が異なるのはそのためである.

図 2 をみると, 勾配法の収束は (200000 ステップの間は) 見ることができない. このことから, 近似勾配 $g = \frac{1}{N} \sum_{s \in E} [\nabla_\theta \pi_\theta(s) \nabla_a Q(s, a|\omega)|_{a=\pi_\theta(s)}]$ のノルムの 200000 ステップの間の変化を見てみる.

図 3 をみると, 勾配のノルム $\|g\|$ は収束していない. また, 100000 ステップあたりで突然大きな勾配になっていることがわかる. 以下では, このような問題が生じている原因について検討していく.

4.3 勾配法が収束しない原因の検討

2.4 節で示した DDPG のアルゴリズム自体の問題点以外に, 前節の問題点の原因となる候補を考察する.

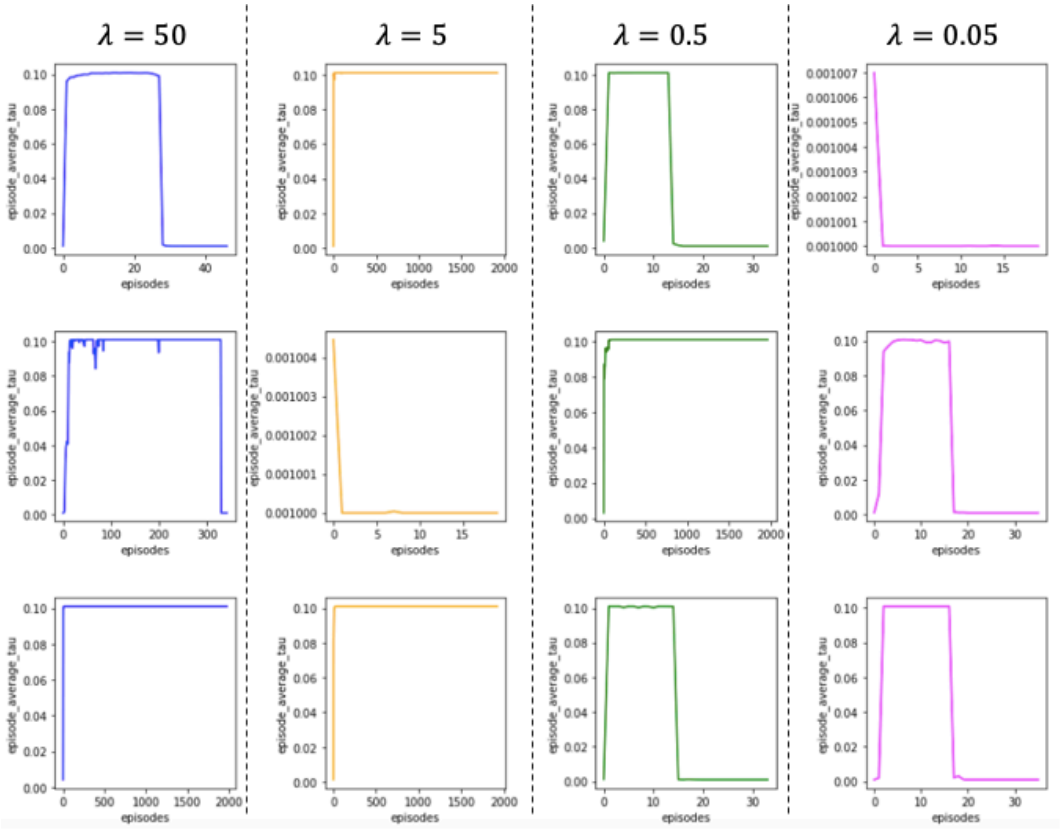

図 2　学習を通しての τ の変化

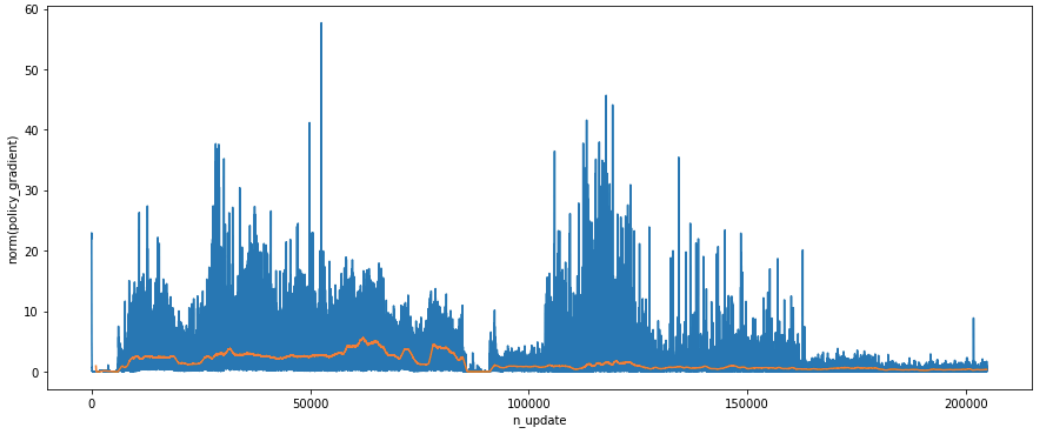

図 3　学習を通しての $\|g\|$ の変化

まず 1 つ目として, Q 関数の近似が正しく行えていないことが考えられる. actor の更新には

$$g = \frac{1}{N}\sum_{s\in E}[\nabla_\theta \pi_\theta(s) \nabla_a Q(s,a|\omega)|_{a=\pi_\theta(s)}]$$

が用いられるため, Q 関数の近似精度は必要不可欠である. しかしながら, DDPG では各ステップにおける方策 π_{θ_k} に対する $Q^{\pi_{\theta_k}}$ の学習が収束する前に次ステップに進む場合があり, この場合は間違った方向に actor の更新がなされることになる.

2 つ目に, 正しい方策勾配 (5) がパラメータ空間上で非常に変化が激しい領域がある可能性があることである. 図 3 をみると, 一度近似勾配 g のノルムが小さくなった後再び大きくなっている. もしノルムが小さくなっているステップ周辺での Q 関数が正しく近似されており, replay buffer の分布が $\rho^{\pi_{\theta_k}}$ に近いのならば, 用いられた勾配は正しい値となる. これが急激に大きくなっているのなら, その場合のステップサイズを小さくして

おく必要がある. また, 方策が大きく変化すると critic のパラメータ ω の再学習も必要となり, 1 つ目の問題点の影響を受けることになる.

5　課題点抽出

前節で行った考察より, まず方策勾配を解析的に計算してパラメータ空間上で急峻な変化をもつ構造かどうかを確認する. もし予想の通りであれば, そのパラメータ付近で意図的にステップサイズを小さくする手法を考案する. 逆に, 急峻となる構造でないのならば, replay buffer のデータの分布に問題があるということになり, システムとのインタラクション方法について考察する必要がある.

方策勾配の解析的な計算についてであるが, 図 3 に見られた問題点は, 最適セルフトリガー制御の強化学習に特有の特徴であったため, $\nabla_{\theta^a} J(\pi_\theta)$, $\nabla_{\theta_\tau} J(\pi_\theta)$ についてそれぞれ計算を行いたいと考えている.

6　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.

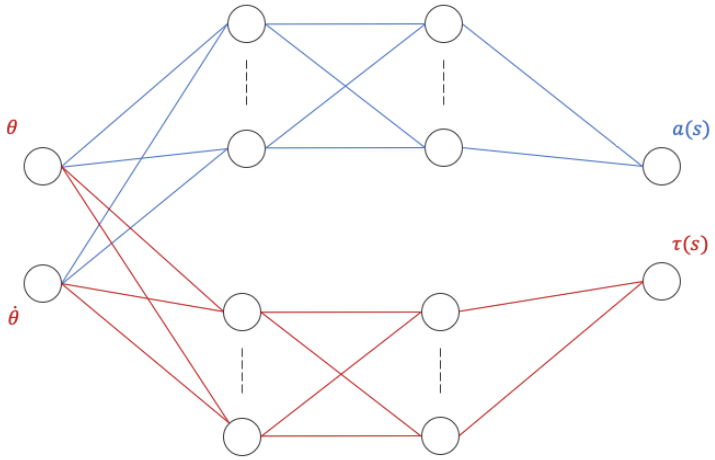

図 4　actor ネットワーク

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.

$$\theta^\mu \leftarrow \theta^\mu - \alpha_a \nabla_{\theta^a} J(\theta^\mu) - \alpha_\tau \nabla_{\theta^\tau} J(\theta^\mu) \tag{17}$$

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

$$\|\nabla_{\theta^a} J(\theta^\mu)\| = 66.70076 \tag{18}$$
$$\|\nabla_{\theta^\tau} J(\theta^\mu)\| = 8.730332e-05 \tag{19}$$

π_2 での値は,

$$\|\nabla_{\theta^a} J(\theta^\mu)\| = 1.9210727 \tag{20}$$
$$\|\nabla_{\theta^\tau} J(\theta^\mu)\| = 4.414095e-05 \tag{21}$$

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.

7　2 つのパラメータに関する方策勾配の考察

前節で行った検証により倒立振子の場合には，actor を表現する 2 つのニューラルネットワークのパラメータに関する方策勾配 $\nabla_{\theta^a} J(\theta^\mu), \nabla_{\theta^\tau} J(\theta^\mu)$ の大きさがかなり異なっていた．このことから, 制御対象に関するどのような前提があれば $\|\nabla_{\theta^a} J(\theta^\mu)\| > \|\nabla_{\theta^\tau} J(\theta^\mu)\|$ となるのかを考えてみる．

評価関数 $J(\theta^\mu) = \mathbb{E}_{s_0 \sim d_0}[V^{\theta^\mu}(s_0)]$ であることから $\nabla_{\theta^\mu} J(\theta^\mu) = \int_S d_0(s) \nabla_{\theta^\mu} V^{\theta^\mu}(s) \mathrm{d}s$ なので, まず $\nabla_{\theta^\mu} V^{\theta^\mu}(s)$ について詳しくみていく．価値関数 $V^{\theta^\mu}(s)$ の (一般的な) 性質から,

$$\nabla_{\theta^\mu} V^{\theta^\mu}(s) = \nabla_{\theta^\mu} [r(s, \mu(s|\theta^\mu)) + \gamma V^{\theta^\mu}(s')] \tag{22}$$

微分の chain rule より,

$$\begin{aligned} \nabla_{\theta^\mu} V^{\theta^\mu}(s) = & \nabla_{\theta^\mu} \mu(s|\theta^\mu) \nabla_u r(s,u)|_{u=\mu(s|\theta^\mu)} \\ & + \gamma \int_S \nabla_{\theta^\mu} \mu(s|\theta^\mu) \nabla_u \mathrm{Pr}(s'|s,u)|_{u=\mu(s|\theta^\mu)} V^{\theta^\mu}(s') \mathrm{d}s' \\ & + \gamma \int_S \mathrm{Pr}(s'|s, \mu(s)) \nabla_{\theta^\mu} V^{\theta^\mu}(s') \mathrm{d}s' \end{aligned} \tag{23}$$

となる (ただし, $u = [a, \tau]$). この式は微分するパラメータを actor 全体のパラメータ θ^μ ではなく, 入力信号 a, 通信間隔 τ のネットワークのパラメータのみに絞った場合にも同じような式が成り立つはずである．つまり,

$$\begin{aligned} \nabla_{\theta^a} V^{\theta^\mu}(s) = & \nabla_{\theta^a} \mu(s|\theta^\mu) \nabla_a r(s,u)|_{u=\mu(s|\theta^\mu)} \\ & + \gamma \int_S \nabla_{\theta^a} \mu(s|\theta^\mu) \nabla_a \mathrm{Pr}(s'|s,u)|_{u=\mu(s|\theta^\mu)} V^{\theta^\mu}(s') \mathrm{d}s' \\ & + \gamma \int_S \mathrm{Pr}(s'|s, \mu(s)) \nabla_{\theta^a} V^{\theta^\mu}(s') \mathrm{d}s' \end{aligned} \tag{24}$$

$$\begin{aligned} \nabla_{\theta^\tau} V^{\theta^\mu}(s) = & \nabla_{\theta^\tau} \mu(s|\theta^\mu) \nabla_\tau r(s,u)|_{u=\mu(s|\theta^\mu)} \\ & + \gamma \int_S \nabla_{\theta^\tau} \mu(s|\theta^\tau) \nabla_\tau \mathrm{Pr}(s'|s,u)|_{u=\mu(s|\theta^\mu)} V^{\theta^\mu}(s') \mathrm{d}s' \\ & + \gamma \int_S \mathrm{Pr}(s'|s, \mu(s)) \nabla_{\theta^\tau} V^{\theta^\mu}(s') \mathrm{d}s' \end{aligned} \tag{25}$$

が成り立つ. 以下 θ_τ, θ_a の両方に成り立つ議論をする場合に, 便宜上 θ を用いることとする.

システム (**??**) は確定系なので,

$$s' = s + \int_0^{\tau(s)} (f(t) + g(t)a(s)) \mathrm{d}t \tag{26}$$

と定まり, 各漸化式の第 3 項目は

$$\int_S \mathrm{Pr}(s'|s, \mu(s)) \nabla_\theta V^{\theta^\mu}(s') \mathrm{d}s' = \nabla_\theta V^{\theta^\mu}(s + \int_0^{\tau(s)} (f(t) + g(t)a(s)) \mathrm{d}t) \tag{27}$$

となる.

またシステムと方策 $\pi(s)$ が確定的であるので, 全インターバルの開始時刻での状態は, 初期状態を引数とする関数で書け, $s^{(n)}(s)$ と表記する．同じく, 漸化式の第 1,2 項目までの和も確定的であり, $k^{(n)}(s)$ とすれば,

$$\nabla_\theta V^{\theta^\mu}(s) = \sum_{n=0}^{\infty} \gamma^i k^{(n)}(s) \tag{28}$$

となるため, $k^{(n)}(s)$ を解析的に表現できれば, 価値関数の勾配を解析的に書けそうである.

7.1　注意

$\nabla_{\theta^\tau} \mu(s|\theta^\mu)$, $\nabla_{\theta^a} \mu(s|\theta^\mu)$ に関しては, ニューラルネットワークモデルの設定にしか依存しないものなので, 今回は特段着目しない.

7.2 各ステップの報酬のパラメータ勾配

初めに, 漸化式 (25) の右辺第 1 項に関して考察する. T を各インターバルの開始時刻だとすると,

$$\begin{aligned}\nabla_\tau r(s,u)|_{u=\mu(s|\theta^\mu)} &= -\nabla_\tau \int_T^{T+\tau} s(t)^\top Q s(t)\mathrm{d}t - \nabla_\tau(\tau a^\top R a) + \nabla_\tau(\lambda\tau) \\ &= -\nabla_\tau \int_0^{\tau} s(t+T)^\top Q s(t+T)\mathrm{d}t - a^\top R a + \lambda \\ &= -s(\tau+T)^\top Q s(\tau+T) - a^\top R a + \lambda \end{aligned} \tag{29}$$

となる. ただし,

$$s(t+T) = s(T) + \int_0^t (f(l+T) + g(l+T)a(T))\mathrm{d}l \tag{30}$$

であることを注意しておく.

次に, 漸化式 (24) の右辺第 1 項に関して考察する.

$$\begin{aligned}\nabla_a r(s,u)|_{u=\mu(s|\theta^\mu)} &= -\nabla_a \int_T^{T+\tau} s(t)^\top Q s(t)\mathrm{d}t - \nabla_a(\tau a^\top R a) + \nabla_a(\lambda\tau) \\ &= -\nabla_a \int_0^{\tau} s(t+T)^\top Q s(t+T)\mathrm{d}t - \tau R a \\ &= -\int_0^{\tau} (\frac{\partial s(t+T)}{\partial a}^\top Q s(t+T) + s(t+T)^\top Q \frac{\partial s(t+T)}{\partial a})\mathrm{d}t - \tau R a \\ &= -2\int_0^{\tau} s(t+T)^\top Q \frac{\partial s(t+T)}{\partial a}\mathrm{d}t - \tau R a \ (\because\ Q : \text{対称}) \end{aligned} \tag{31}$$

式 (30) より,

$$\begin{aligned}\frac{\partial s(t+T)}{\partial a} &= \frac{\partial}{\partial a}\int_0^t (f(l+T) + g(l+T)a)\mathrm{d}l \\ &= \int_0^t g(l+T)\mathrm{d}l \end{aligned} \tag{32}$$

なので,

$$\nabla_a r(s,u)|_{u=\mu(s|\theta^\mu)} = -2\int_0^{\tau} s(t+T)^\top Q \left\{\int_0^t g(l+T)\mathrm{d}l\right\}\mathrm{d}t - \tau R a \tag{33}$$

と表せる. したがって, インターバルの開始状態 $s(T)$ と, $u = \mu(s|\theta^\mu)$ がわかれば各インターバルにおけるステップ報酬の勾配の比較は可能である.

最後に, 各インターバルの開始時刻での状態は

$$s^{(n)}(s) = s^{(0)}(s) + \sum_{i=0}^{n-1}\int_{T_i}^{T_i+\tau(s^{(i)}(s))} (f(T_i+t) + g(T_i+t)a(s^{(i)}(s)))\mathrm{d}t \tag{34}$$

と書けるため, 式 (29), (33) に代入すれば全ステップでの r の勾配は, 関数 V の引数 s の関数として表せる. ただし, $T_i = \sum_{i=0}^{n-1}\tau(s^{(i)}(s))$ である.

8 修論の見通し

方策勾配は

$$\nabla_{\theta^\pi} J(\theta^\pi) = \mathbb{E}_{s\sim\rho^{\pi^{\theta^\pi}}}\left[\nabla_{\theta^\pi}\pi(s|\theta^\pi)\nabla_u Q^{\theta^\pi}(s,u)|_{u=\pi(s|\theta^\pi)}\right] \tag{35}$$

と書けるのであった. 入力アフィン系のセルフトリガー制御に対象を絞ることにより, $Q^\pi(s,a)$ を解析にかけることを活用して, $\nabla_{\theta^\pi} J(\theta^\pi)$ の大きさや形を解析し, 勾配法の設定方法についての理論を構築していきたい.

本研究では方策 π を図 4 のように a, τ をそれぞれ別のモデルに分離していたことから, それぞれのモデルに対する方策勾配は

$$\nabla_{\theta^\tau} J(\theta^\pi) = \mathbb{E}_{s\sim\rho^{\pi^{\theta^\pi}}}\left[\nabla_{\theta^\tau}\tau(s|\theta^\tau)\nabla_\tau Q^{\theta^\pi}(s,a,\tau)|_{a=a(s|\theta^a),\tau=\tau(s|\theta^\tau)}\right] \tag{36}$$

$$\nabla_{\theta^a} J(\theta^\pi) = \mathbb{E}_{s\sim\rho^{\pi^{\theta^\pi}}}\left[\nabla_{\theta^a}a(s|\theta^a)\nabla_a Q^{\theta^\pi}(s,a,\tau)|_{a=a(s|\theta^a),\tau=\tau(s|\theta^\tau)}\right] \tag{37}$$

とそれぞれ書ける. 式 (36), (37) の中で問題の特有性を活かせるのは第 2 項目 (Q 関数の勾配) であるためこれに着目したい. たとえば, $\nabla_\tau Q^{\theta^\pi}(s,a,\tau)|_{a=a(s|\theta^a),\tau=\tau(s|\theta^\tau)}$ や $\nabla_a Q^{\theta^\pi}(s,a,\tau)|_{a=a(s|\theta^a),\tau=\tau(s|\theta^\tau)}$ のある程度の形がわかれば, この問題に適したオプティマイザが何なのかを特徴付けることができるかも知れない. しかしながら,

$$\nabla_\tau Q^{\theta^\pi}(s,a,\tau) = \nabla_\tau[r(s,a,\tau) + \gamma s'(s,a,\tau)\nabla_s V^{\theta^\pi}(s)|_{s=s'(s,a,\tau)}] \tag{38}$$

$$\nabla_a Q^{\theta^\pi}(s,a,\tau) = \nabla_a[r(s,a,\tau) + \gamma s'(s,a,\tau)\nabla_s V^{\theta^\pi}(s)|_{s=s'(s,a,\tau)}] \tag{39}$$

であり, 両方に共通している $\nabla_s V^{\theta^\pi}(s)|_{s=s'(s,a,\tau)}$ についての計算結果が非常に煩雑で, どのような形になるのか評価方法がわかりません.

また, Q 関数が方策依存の関数であるため, 方策近似モデルであるニューラルネットワークに関する仮定をおかないと議論を進めることができない. そのため, 強い仮定をおいてしまうことになるがその方向で議論を進めていこうと思います.

9 イベントトリガーとセルフトリガーの違いについての考察

最適イベントトリガー制御問題に対する DDPG は工夫なく (通信の有無を 2 つの連続値の大小を真偽値とするのが工夫なのかも知れないが) 適用しても最適解を得ることができた. そこで, 似たような問題設定のセルフトリガー制御問題に対しても単純に DDPG を適用することで強化学習を行うことができるのではないかと考え, 実験を行った. その結果 (少なくとも同じステップ数では) うまく学習をすることはできなかった. この「うまくいかない」とは, DDPG アルゴリズムの中でロス関数を収束させ, 方策の更新を収束させることができなかったという意味で使っている. そこでここからは, 何故イベントトリガー制御とセルフトリガー制御でそのような違いが生じたのかを考えていく.

9.1 状況整理

イベントトリガー制御では, 方策 π は制御入力 a とそれをアクチュエータに送信するかを決めるための連続値変数 d_1, d_2 を出力するモデルであるとした (communication $= (d_1 > d_2)$). 一方でセルフトリガー制御の方策関数 π は制御入力 a と次の通信までの時間間隔 τ を出力するモデルであるとした.

DDPG などの深層強化学習では, 方策をニューラルネットワークを用いてパラメータ近似を行う. セルフトリガー制御でもイベントトリガー制御に倣い, 全てのパラメータに対して Adam と呼ばれるオプティマイザを用いて学習してみた. すると, 本節冒頭で述べたような「うまくいかない」という状況におちいったのである. (ちなみに, Adam は更新方向を勾配の移動平均にし, 勾配の移動平均のノルムに反比例 (この時の比例定数が学習率ハイパーパラメータ) するステップサイズにするようなアルゴリズムである.)

9.2 イベントトリガー制御と, セルフトリガー制御に対する強化学習で違いが生じた原因

python で行っている実験では, 近似方策勾配のノルムが 1 以下となるように線形にクリップする工夫を施してから, その補正勾配を Adam に渡して更新則 (更新方向, ステップサイズ) を導出している. したがって, セルフトリガー制御の評価関数に勾配が「非常に」大きくなるようなパラメータ領域が存在しているかどうかは, 今回の実験結果に対して直接悪影響を与えたとは考えられない. それよりも, パラメータの変化に対する割引分布 ρ^π(式 (6)) の変化が, イベントトリガー制御に対して大きくなりすぎているため, critic の学習の loss と方策勾配の近似に悪影響を及ぼしているという可能性がある.

ρ^{π} が急激に変化すると, 当然経験データの分布にも影響が出る.

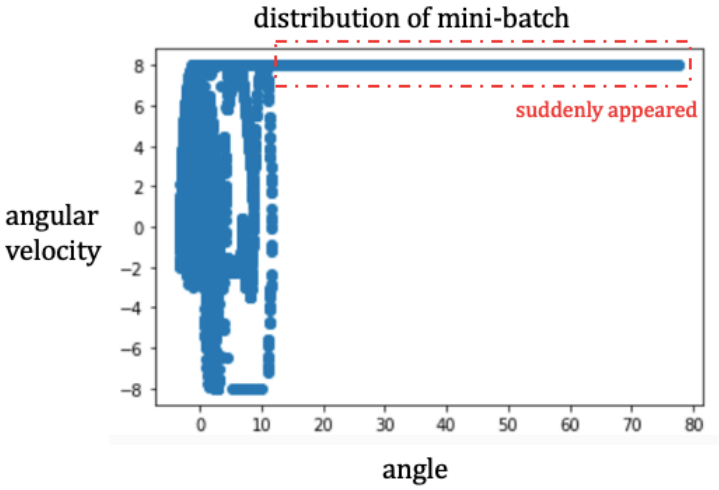

図 5　経験データの分布

図 5 には, 直近の経験データの分布を示した. 赤枠で示された部分 (システムが発散している部分) はその中でも最も直近の経験データであり, それまでの経験データとは明らかに異なる領域であると言える. このように未知のデータが突然入ってくなると, 式 (8) で表された critic の loss は大きくなってしまう. そうなると, 正しい方策勾配は計算できなくなる.

(再掲: 式 (8))

$$Loss = \frac{1}{N}\sum_{s\in E}(Q(s,a|\omega) - \{r(s,a) + \gamma Q(s,\pi(s)|\omega)\})^2$$

さらに, 方策勾配は割引分布に関する期待値をとるが, 式 (10) のように実際の計算には図 5 のデータに対する平均を取ってしまうので, 正しい方策勾配を求めることはできない.(赤枠以外の分布は, 現在の方策に対する割引分布には現れない分布かも知れない)

(再掲: 式 (10))

$$\mathbb{E}_{s\sim\rho^{\pi}}[\nabla_{\theta}\pi_{\theta}(s)\nabla_{a}Q(s,a|\omega)|_{a=\pi_{\theta}(s)}] \simeq \frac{1}{N}\sum_{s\in E}[\nabla_{\theta}\pi_{\theta}(s)\nabla_{a}Q(s,a|\omega)|_{a=\pi_{\theta}(s)}]$$

そこで, セルフトリガー制御では Adam の学習率に関するパラメータをイベントトリガーの時のそれに比べて小さくするべきであると考える.

9.3　今後の研究のすすめ方

前節での検証を踏まえると, actor の学習率を非常に小さくすれば方策の変化も小さくなるため, 長大なステップを重ねれば学習を安定して行えると考えられる. しかしながらこれは自明なことであり, 何らかの工夫を施して高速化を行いたい. もしニューラルネットワークのパラメータのうち, その微少変化に対する割引分布への変化率が小さいものがあれば, そのパラメータに対しては学習率を小さく取っておく必要はない. それを最も議論しやすい形として図 4 のような方策近似モデルを用い, $\nabla_{\theta^a}\rho^{\pi_{\theta^{\pi}}}(s), \nabla_{\theta^{\tau}}\rho^{\pi_{\theta^{\pi}}}(s)$ を解析することで議論を進めていこうと考える.

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.