広島県立総合技術研究所東部工業技術センター研究報告　No. 35（2022）
技術ノート

強化学習における各種手法の比較（第 2 報）

廣川　勝久

Simple Comparison of Reinforcement Learnings (Ⅱ)

HIROKAWA Katsuhisa

テーブルによる強化学習は，状態の分割数の増加に伴う学習コストの増加が課題であり，加えてテーブルを用いた方策勾配法については，学習の不安定も明らかとなった。本報告では，方策勾配法を，３種類の深層学習により実装し比較した。これらの深層学習を実装する上で，ポイントとなる点について述べる。

キーワード：方策勾配法，Actor-Critic，連続値出力，PyTorch, モンテカルロ法，機械学習，深層学習，python

1. 緒　　言

これまでのコンピュータ制御では，人間がコンピュータにルールや処理方法をプログラミングすることが必要であった。深層学習は（ディープ）ニューラルネットワークにターゲットとするデータを与え自動的にルールや制御方法を生成する技術である。深層学習の技術はここ 10 年の間に著しい進歩を成し遂げた。画像認識の分野では，人間の能力をはるかに超え，すでに実用化が進んでいる。深層学習の技術開発は子供の成長と類似しており，認識の次は行動や制御，その後人間とのコミュニケーションや感情の理解へと技術的な進歩が予想され，それぞれの技術分野ではすでに一定程度の成果が得られている。

強化学習は，認識技術の次に実用化が期待される深層学習分野である。プラントの制御や自動運転，ロボット制御など産業応用が特に期待されている。これまで，強化学習については，数多くの手法が提案されている。以前の報告では，基本的な強化学習の方法として，Q 学習，DQN, テーブルによる方策勾配法について報告を行った[1)]。本稿では，テーブルによる方策勾配法を深層学習に置き換えた方策勾配法の 3 種類の基本的な手法について比較・検討を行ったので報告する。

2. 強化学習の概要

2.1 エージェント環境

強化学習では，**図 1** に示すように頭脳となるエージェントと，実際に行動する実行環境から構成される。強化学習の研究開発では，頭脳となるエージェントの学習プログラムの開発を行う。エージェントは，学習により実行環境に適した最適な行動の選択を可能とする。**図 1** のように実行環境s（台車の位置や棒の角度など）に置かれたエージェントが何かの行動a（台車を押す）を起こしたとき，実行環境から報酬rが与えられ，同時に環境は変化し状態s'（台車の位置や角度が変わる）となる。強化学習では，報酬rから環境内の状態sの時の行動aの価値を最大化するように学習が行われる。

2.2 方策勾配法

方策勾配法では，Q 学習や DQN とは異なり行動価値基準ではなく，状態sにおける方策確率に基づき行動を決定する。状態価値関数を$V(s)$とすると，状態価値関数は方策関数$\pi(a|s)$と行動価値関数$Q(s,a)$から次式で与えられる。

$$V(s,\theta) = \sum_a \pi(a|s,\theta)\, Q(s,a) \tag{1}$$

ここで，状態sのときの行動をaとする。また，θは各方策を選択する確率パラメータを表す。方策勾配法を用いた方策確率の探索による状態価値関数を最大とする目的関数$J(\theta)$とすると，方策確率のパラメータθの更新（学習）は次式で与えられる。

$$J(\theta) \propto V(s,\theta) \tag{2}$$

$$\theta \leftarrow \theta + \alpha \nabla J(\theta) \tag{3}$$

ここで，目的関数の勾配は

$$\nabla J(\theta) = E\left[Q(s,a) \frac{\partial log\left(\pi(a|s,\theta)\right)}{\partial \theta} \right] \tag{4}$$

により計算できる。E[]は期待値を表す。

2.3 深層学習による方策勾配法

深層学習においても，モンテカルロ離散方策勾配法と

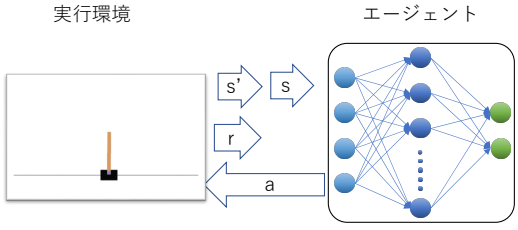

図 1　強化学習の環境とエージェント

2022.8.30　加工技術研究部　受理

同様にパラメータ θ からの方策確率の計算には softmax 関数を用いる。また，行動価値$Q(s_t, a_i)$はモンテカルロ法による割引総報酬$G(s_t)$を用いて近似する。

$$\pi(a_i|s_t, \theta_i) = \frac{\exp(\theta_i)}{\sum_{k=1}^{n} \exp(\theta_k)} \quad (5)$$

$$G(s_t) = r + \gamma G(s_{t+1}) \quad (6)$$

従って，ニューラルネットワークの学習に必要な損失関数は

$$\nabla J(\theta_p) = log\left(\pi(a_p|s_t, \theta_p)\right) G(s_t) \quad (7)$$

で与えられる。ここで，pは選択した方策を示す。

2.4 Actor-Critic 方策勾配法

前節の学習では，行動価値をモンテカルロ法による割引総報酬$G(s_t)$により近似したが，それぞれの状態における価値を加味していない。例えば，倒立振子の場合，棒がほぼ垂直に立っている場合は，倒れる確率低く，状態価値は高いと言える。また，棒が倒れ込んでいる場合は，どのような行動を選択しても，倒れる確率が高く状態価値は低いと言える。Actor-Critic 方策勾配法は Actor と呼ばれる方策を学習するニューラルネットワークと Critic と呼ばれる状態価値を学習する 2 つのネットワークから構成され，状態価値を学習に用いることで，方策勾配法の学習安定性を向上させる方法である。

図 2 に Actor-Critic 方策勾配法のデータ処理の流れを示す。オレンジ色のラインは行動毎に計算を行い，緑色のラインはバッファリング後，一括して処理を行う。まず Actor 側が，状態sの変化に応じて行動aを選択する。行動後報酬rを得てa, s, rの 3 値を記憶する。一定程度値の記憶がなされた後，今度は Critic 側に最後に記憶された状態s_tを入力し，最初の記憶にさかのぼりながら，状態価値$V(s_t)$を求める。同時に割引総報酬$G(s_t)$も計算を行い，Actor 側，Critic 側双方の誤差を求める。

Actor-Critic 方策勾配法では(7)式の割引総報酬$G(s_t)$部分を Critic 側の状態価値$V(s_t)$を用いて，次式の Advantage

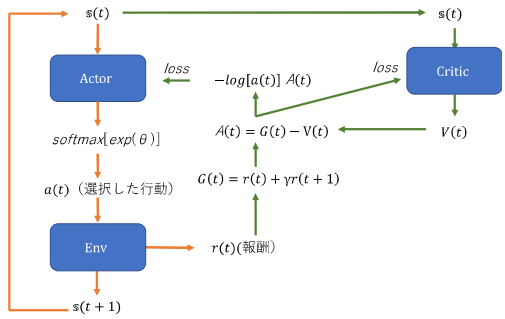

図 2　Actor-Critic の動作

関数に置き換える。

$$A(s_t) = G(s_t) - V(s_t) \quad (8)$$

従って，Actor 側の損失関数は

$$\nabla J(\theta_p) = log\left(\pi(a_p|s_t, \theta_p)\right) A(s_t) \quad (9)$$

となる。また，Critic 側の損失関数は DQN(Deep Q Networks)と同等と考えて良い。

2.5 連続出力型方策勾配法

これまでの方策勾配法の出力は，複数の行動の中から適切な行動の離散値が得られた。しかし，モータのトルクや電流の制御などへの応用を考えた場合，離散値の出力では不都合な場合がある。離散型の方策勾配法では，それぞれの行動が，ニューラルネットワークの出力値の大きさの割合に応じた確率により選択された。連続出力型方策勾配法 [2] では，状態sにおける連続出力値である行動aはガウシアン分布により確率的に選択されると仮定する。この時の行動の値は次式より確率的に決定する。

$$\pi(a_t|s_t) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(a_t-\mu)^2}{2\sigma^2}\right) \quad (10)$$

ここで，μはガウシアン分布の平均値，σは標準偏差を表す。連続出力型の方策勾配法では，この平均値μと標準偏差σの最適値を学習により探索する。(4) 式の方策勾配の損失関数の計算には

$$log(\pi(a_t|s_t)) = -\frac{1}{2} log(2\pi\sigma^2) - \frac{(a_t-\mu)^2}{2\sigma^2} \quad (11)$$

を代入する。

3. 各種強化学習法の比較

3.1 学習環境

深層学習による方策勾配法を実装するためには python と連係するフレームワークの標準的な損失関数は使用することが出来ない。本報告では，(7), (9)，(11) 式で表される独自の損失関数を実装するために，フレームワークとして，PyTorch を用いた。PyTorch 逆伝搬に必要な勾配を自動計算する機能が備わっており，独自の損失関数を定義するのみで，損失に応じた逆伝搬と重みの更新が可能である。

各種手法の比較のため倒立振子の環境として，OpenAI Gym の'CartPole-v1'と ByBullet の'CartPoleBulletEnv'を用いた（**図 3**）[3]。この環境では，4 つ環境変数（台車の位置，台車の速度，ポールの角度，ポールの角速度）

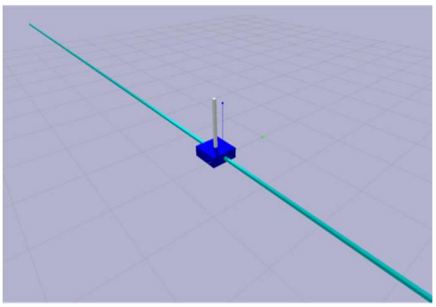

図３　PyBullet の CartPole

が与えられる。OpenAI Gym では，行動は右に台車を押すか左に台車を押すかの離散的な選択を行う。ByBullet では台車へ与えるトルクの範囲を-10～10 の連続値から，台車を左右に押す行動として選択が可能である。実験では 1 回の試行で振子の立った状態が 500 ステップを達成とし，倒立振子の角度が 12 度以上，または台車が 2.4 台以上ずれた場合は倒れた状態とし，2 つの環境の基準を統一した。また，学習が安定状態は 20 回連続の成功と設定した。報酬については，試行後 500 ステップを達成出来ない場合を-20 とし，試行中ステップ毎に立っていれば 1 とした。

3.2 深層学習による方策勾配法

方策勾配法のニューラルネットワーク部の実装には，次のソースコードに示す 3 層構造を用いた。出力ノードから softmax 関数を通過し，確率割合を出力するニューラルネットワークとした。

```
    def forward(self, x):
        h=self.l1(x)
        h=F.relu(h)
        h=self.l2(h)
        y=F.softmax(h,dim=1)
        return y
```

また，ニューラルネットワークから出力割合に応じて方策を確率的に選択するため以下の PyTorch の関数を用いた。

```
probs = model(x)
m = Categorical(probs)
action=m.sample()
observation, reward, done, info = env.step(action.item())
```

これにより，(7)式の log 関数部分が，m.log_prob(action)として計算でき，勾配を使った逆伝搬が可能となる。また，実際の損失関数のソースコードを以下に示す。

```
def forward(seif,log_p_values,trace,steps):
    delta = 0
    for l in range(steps):
        _ , g_reward = trace[l]
        delta -= alpha * log_p_values[l] * g_reward
    delta/=steps
    return delta
```

方策勾配法では，状態価値関数が最大となる方策確率を探索する。即ち(7)式の最大値を深層学習により求めるこ

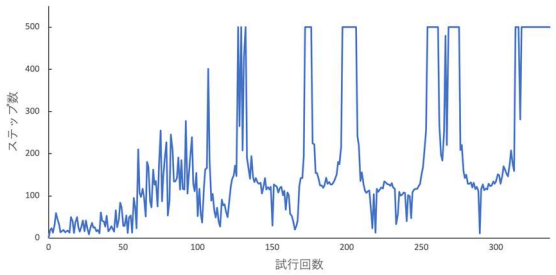

図４　深層学習による方策勾配法の学習速度

とであるが，フレームワークには損失を最小にする最適化関数しか利用出来ないため，ソースコードでは符号を反転することで最大値を求めている。

図４にニューラルネットワーク部の中間層のノード数が 128 個の場合の深層学習の学習グラフを示す。テーブルを用いた方策勾配法が最低でも 700 回の試行回数が必要であったが，深層学習では，ほぼ半分程度の試行回数で学習は完了した。また，テーブルを用いた方策勾配法では，全く学習が行われない場合や非常に長い学習回数を必要とする場合など，学習が不安定であったが，深層学習では，ほぼ同程度の回数で学習が完了し，高い学習安定性が示された。

3.3 Actor-Critic による方策勾配法

Actor-Critic 法では，Actor 部分は前述の方策勾配を学習するニューラルネットワーク部をそのまま用いる。Critic 部分は中間層 1 層の 3 層構造の単純な構造とした。(8)式の Advantage の値を得るためのソースコードを示す。

```
value = model_ctc(states)
q_value = reward + gamma * last_value
advantage=q_value - value.item()
buffa[l][1] = advantage
loss += F.smooth_l1_loss(value, torch.tensor([[q_value]]))
last_value=q_value
```

このプログラムでは，Advantage の値を計算すると同時に Critic 部分の誤差を Huber 損失関数 smooth_l1_loss（）により求めた。また，2 種類のネットワークの更新が必要であるため，以下のように 2 種類のネットワークを続けて更新を行った。

```
model_act.zero_grad()
loss_act.backward(retain_graph=True)
optimizer_act.step()

model_ctc.zero_grad()
loss_critic.backward()
optimizer_ctc.step()
```

図５に中間層のノード数が 128 個の学習結果を示す。ノード数が 128 個の場合，深層学習のネットワークを単独で使用した場合と学習速度に大きな差は無かった。そこで，ノード数の違いによる両者の学習速度について比較を行った。**表１**に方策勾配法のみを実装したニューラルネットワークの学習回数，**表 2** には Actor-Critic 法によ

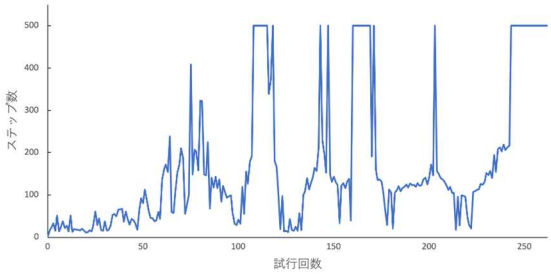

図５　Actor-Critic による方策勾配法の学習速度

る学習回数を示す。各表はノード数対する 3 回の学習回数を示す。**表 1** では，ノード数が 64，128 個では学習速度も速く安定しているが，ノード数が少ない 32 個や多い 256 個では学習速度が低下し，学習も不安定になる。

表 1 ノード数に対する試行回数（方策勾配法）

ノード数			
32	64	128	256
517	264	392	853
513	273	327	163
288	321	336	1076

一方**表 2** に示す Actor-Critic 法では，ノード数の増減に対して学習速度は同程度であり，学習安定性も高いことが明らかとなった。

表 2 ノード数に対する試行回数（Actor-Critic）

ノード数			
32	64	128	256
297	267	262	307
399	274	512	213
426	286	472	212

3.4 連続値出力型方策勾配法

連続値出力型の方策勾配法には，**図 6** に示すような 3 層構造を用いているが，ネットワークからの出力を平均値と標準偏差の対数値を出力する構造とした。ソースコードは以下のとおり。

```
def forward(self, x):
    h = self.l1(x)
    h = torch.tanh(h)
    mean = self.lmean(h)
    log_div = self.ldiv(h)
    div = log_div.exp()
    div = torch.clamp(div, min=0.00001, max=5)
    return mean, div
```

標準偏差が負の値とならないよう，まずニューラルネットワークからの出力を標準偏差の対数とし，指数関数により標準偏差を求めている。また，学習時に標準偏差が極端に大きな値になることを防ぐため，クランプも行っている。

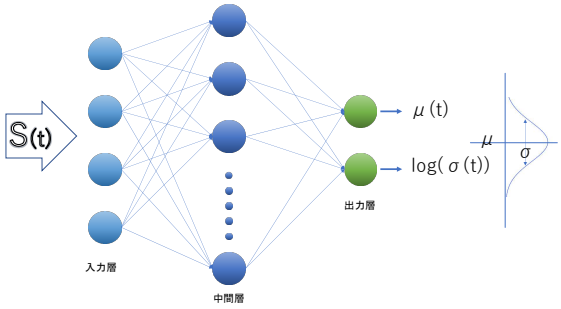

図６　連続値出力型の構造

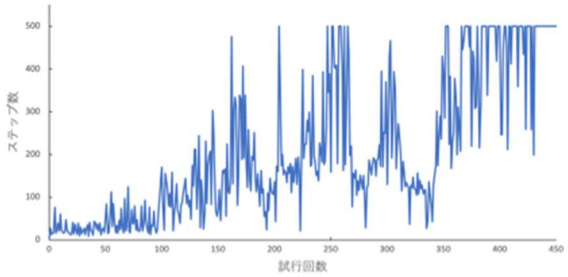

図７　連続値出力型方策勾配法の学習速度

下記のプログラムにより，平均値 μ と標準偏差 σ のガウス分布確率から行動（方策）の値が得られる。CartPole の入力範囲が-10～10 であるため，行動の値を一旦，双曲線正接関数 tanh()により-1～1 の範囲に正規化している。

```
mean, div = model_act(x)
m = Normal(mean, div)
action = m.sample()
torque = torch.tanh(action) * max_torque
```

また（11）式の損失関数の計算は以下のとおり。報酬については，Actor-Critic の Advantage を用いた。

```
mu, sigma = probs[l]
tmp1 = -0.5 * torch.log(2 * torch.pi * sigma**2)
tmp2 = -0.5 * ((action - mu) ** 2) / sigma**2
delta -= (tmp1 + tmp2) * g_reward
```

他の方策勾配法と同様に最大値を求めるため，損失の符号は反転している。**図 7** に連続値出力型の方策勾配法の学習速度を示す。学習は試行回数が 500 回程度で完了するが，学習については，Actor- Critic 法を用いても不安定であり，全く学習が進まない状態に陥ることが度々あった。また，殆どの場合，標準偏差が，最大値に張り付いた状態で学習が完了した。

4. 結　　言

強化学習の方策勾配法について，テーブルを使った手法を 3 種類の深層学習に置き換え実装，比較した。方策勾配法に深層学習を用いることにより，テーブルを使った手法より高い学習速度，学習に安定性が得られた。Actor-Critic 法では方策勾配法を単独のニューラルネットワーク

のみで実装したものと比較して，中間層のノード数の影響が少なく高い学習安定性が示された。また，Actor-Criticを応用した連続値出力型の方策勾配法では，学習は可能であるが，学習が不安定であり，さらに標準偏差が最大値に陥った。連続値出力型の場合，標準偏差に対するなんらかの制約を損失関数に付加することが必要と考えられる。

文　　献

1) 廣川 勝久：広島県立総合技術研究所東部工業技術センター研究報告，強化学習における各種手法の比較（第1報）**35**（2022）.
2) 伊藤多一 他：現場で使える！Python 深層強化学習入門 強化学習と深層学習による探索と制御，翔泳社（2019）.
3) 牧野 浩二，西崎 博光：TensorFlowによる深層強化学習入門 —OpenAI Gym+PyBullet によるシミュレーション—（2021）.