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１報）

廣川　勝久

Simple Comparison of Reinforcement Learnings (Ⅰ)

HIROKAWA Katsuhisa

現在，強化学習には様々な手法が提案され，高い学習能力が示されている。本報告では，それらの基本となる学習手法である Q 学習，DQN および離散方策勾配法について比較し，評価を行った結果について示す。特に，強化学習を始めるに当たり，ポイントとなる点について述べる。

キーワード：Q 学習，DQN, 方策勾配法，モンテカルロ法，機械学習，ニューラルネットワーク，人工知能，python

1. 緒　　言

DeepMind テクノロジー社の人工知能である AlphaGo[1] がプロの囲碁棋士を破ったことにより，強化学習が注目され現在の人工知能研究は第3期を迎えることとなった。

Prologやエキスパートシステムに代表される初期の人工知能は，ルールに基づき判断を行う。この人工知能はルールをハンドコーディングしたものであり，利用される環境のルール全てを把握し，使用環境に合わせたプログラムを必要とした。第２期は階層型ニューラルネットワークによる深層学習である。パーセプトンから始まったニューラルネットワークの研究は，現在の畳み込みニューラルネットワークの原型であるネオコグニトロン [2] に進化していく。この時期のニューラルネットワークでは，使用環境で収集された多くの既知データを使い学習が行われる。これには，ニューラルネットワークの学習パラメータを設定するのみで，環境ごとに処理方法などのプログラムを必要としない利点がある。一方で，学習には使用環境内の全種類のデータを必要とし，学習に利用されない種類のデータが入力された場合，動作が保証されない問題があり汎化能力の実装についても研究が盛んに行われた。[3]

強化学習では，学習前に何らかのデータを用意する必要は無く，環境内を探索しながら学習データを獲得する。探索と学習を繰り返すことで，最も良い行動または最も報酬が得られる行動に収斂していく。強化学習では，この環境内の探索機能により，今までの人工知能では不可能とされた人間を超えた最良の答えや行動を見出すことが可能となった。

強化学習については，現在非常に多くの手法が提案されている [4]。本稿では，価値または方策に基づき行動する強化学習について基本的なモデルを比較，評価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**に示すように学習を行うエージェントと，学習のための探索を行う実行環境から構成される。実行環境sに置かれたエージェントが何かの行動aを起こしたとき，実行環境から報酬rが与えられ，と同時に環境は状態s'に変化すると仮定する。強化学習では，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。強化学習は，行動価値または方策に基づき行動する２種類に大別され，それぞれにテーブルまたは深層学習によるものに分類される。

2.2 行動価値に基づく Q 学習

テーブルを使った強化学習の中で行動価値に基づく学習を Q 学習と呼ぶ [5]。今時刻 t，状態s_tにおいて行動a_tを起こした時の行動価値を$Q(s_t, a_t)$とする。Q 学習における行動価値の更新は次式で行われる。

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha\left(r + \gamma\, maxQ(s_{t+1}, a_{t+1})\right) \qquad (1)$$

ここで，α は学習係数，γ は割引率を示し将来得られる価値を加算している。1 項の$Q(s_t, a_t)$は前回の行動価値を

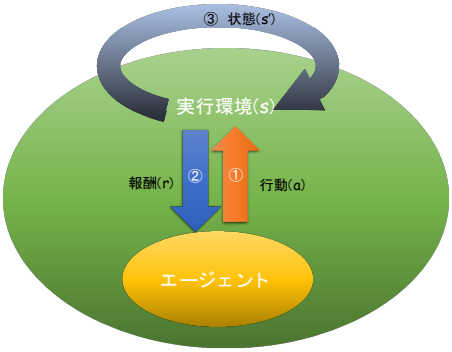

図１　強化学習の環境概念図

2022.7.20　加工技術研究部　受理

表す。$maxQ(s_{t+1},a_{t+1})$は状態s_{t+1}における前回の最大の行動価値を示す。Q 学習では，状態s, と行動aを離散化したテーブルを使い学習が行われ，探索行動毎に行動価値 Q(s, a)の更新が行われる。

2.3 DQN

Q 学習を深層学習により実現したものが，DQN（Deep Q Network）である。Q 学習と異なり状態s_tは連続値の取り扱いが可能であり，テーブルを用いる必要は無い。DQN では（2）式に示すように，状態の入力ベクトル$\mathbb{s}(t)$に対して，行動価値ベクトル$\mathbb{Q}(t)$の出力を得るためのテンソル$\mathbb{W}$を深層学習により求める。

$$\mathbb{Q}(t) = \mathbb{W} \otimes \mathbb{s}(t) \tag{2}$$

一方（1）式は次のように，変形できる。

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left(r + \gamma\, maxQ(s_{t+1},a_{t+1}) - Q(s_t,a_t)\right) \tag{3}$$

これは，行動価値$Q(s_t,a_t)$を$\alpha(\,)$により最大化する学習と見なされる。括弧内は DQN から出力$Q(s_t,a_t)$に対する$r+\gamma maxQ(s_{t+1},a_{t+1})$を教師データとして取り扱う。

図 2 には，DQN におけるデータの流れを示した。DQN では，行動価値$\mathbb{Q}(t)$がベクトルとして出力されるため，行動aは行動価値$\mathbb{Q}(t)$の内，最大値を持つものから選択する。また，学習は選ばれた行動の行動価値のみが，教師データとして学習される。行動aが実行されると，環境から，報酬 r と環境変化$\mathbb{s}(t+1)$が返されるが，DQN では Q 学習とは異なり，環境$\mathbb{s}(t+1)$に対する過去の行動価値$\mathbb{Q}(t+1)$を持たないため，再度$\mathbb{s}(t+1)$をニューラルネットワークに入力し$\mathbb{Q}(t+1)$を得る。以下の式は，行動価値$Q_2(t)$が最大値の場合を示す。その場合，教師ベクトル$\mathbb{T}(t)$は 2 行目のみが学習値となる。

$$\mathbb{Q}(t) = \begin{pmatrix} Q_1(t) \\ Q_2(t) \\ \vdots \\ Q_n(t) \end{pmatrix},\quad a \leftarrow argmax \begin{pmatrix} Q_1(t) \\ \boldsymbol{Q_2(t)} \\ \vdots \\ Q_n(t) \end{pmatrix}$$

$$\mathbb{T}(t) = \begin{pmatrix} Q_1(t) \\ r + \gamma\, max\mathbb{Q}(t+1) \\ \vdots \\ Q_n(t) \end{pmatrix} \tag{4}$$

次式の関数$\mathbb{L}(t)$を

$$\mathbb{L}(t) = |\mathbb{T}(t) - \mathbb{Q}(t)| \tag{5}$$

DQN の損失関数とし，誤差逆伝搬によりネットワークの重みを更新する。

2.4 離散方策勾配法

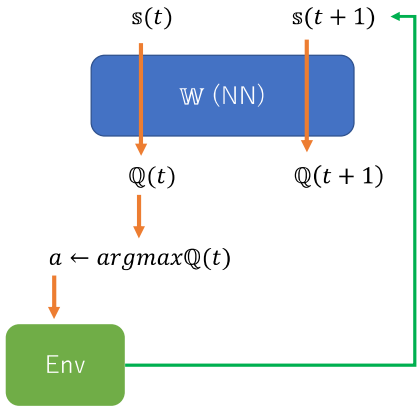

図 2 DQN のデータフロー

2.4.1 方策勾配法

Q 学習や DQN では，将来得られる報酬も含んだ行動価値を基準に行動を決定した。方策勾配法では，状態sにおける，方策（行動）確率に基づき行動を決定する。状態sのとき行動aを選択する方策関数$\pi(a|s)$と行動価値関数$Q(s,a)$から状態価値関数$V(s)$はベルマン方程式により，θ を確率変数として，次式で与えられる。

$$V(s,\theta) = \sum_a \pi(a|s,\theta)\, Q(s,a) \tag{6}$$

方策勾配法では，状態価値関数が最大となる方策確率を環境探索により学習する。方策勾配法による最大化を行う $J(\,\theta)$を目的関数とし，

$$J(\theta) \propto V(s,\theta) \tag{7}$$

方策確率のパラメータ θ の更新（学習）を次式で表す。

$$\theta \leftarrow \theta + \alpha \nabla J(\theta) \tag{8}$$

ここで，目的関数の勾配は

$$\nabla J(\theta) = E\left[Q(s,a)\frac{\partial log\left(\pi(a|s,\theta)\right)}{\partial \theta}\right] \tag{9}$$

により計算できる。E[]は期待値を表す。

2.4.2 モンテカルロ離散方策勾配法

方策勾配法は，Q 学習と同様にテーブルを使った学習が可能である。Q 学習では，行動を決定するテーブルには行動価値の値が割り付けられていたが，方策勾配法では，方策確率パラメータ θ が割り付けられる。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。

$$\pi(a_i|s_t,\theta_i) = \frac{\exp(\theta_i)}{\sum_{k=1}^{n}\exp(\theta_k)} \tag{10}$$

（10）式を(9)に代入し，整理すると

$$\nabla J(\theta_i) = \frac{1}{T}\left(N_i - N_t \pi(a_i|s_t,\theta_i)\right) G(s_t)$$
$$\because Q(s_t,a_i) \approx G(s_t) \qquad (11)$$

と確率変数の具体的な勾配計算式が得られる[6]。ここでN_tは状態s_tとなった回数を表し，N_iはその状態からで行動a_iを選んだ回数を表す。T は試行ごとのステップ数である。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。

$$G(s_t) = r + \gamma G(s_{t+1}) \qquad (12)$$

3. 各種強化学習法の比較

3.1 学習環境

各種手法を比較するための環境には，OpenAI Gym の倒立振子'CartPole-v1'を用いた。この環境は非常にシンプルであり，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）が与えられ，2 つの行動（右に台車を押す，左に台車を押す）を選択する。この環境では 1 回の試行で振子の立った状態が 500 ステップを達成すれば成功となり，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見なされ失敗となる。今回は 20 回連続して成功した場合を学習が安定状態と見なした。

3.2 Q 学習

Q 学習では，**図 3** に示すような行動価値$Q(s_t,a_t)$のテーブルを内部に持つエージェントがテーブル値を参照し，行動を決定する。そのため，連続値である状態s_tを一定間隔で分割した離散値を用いる。**図 4** は振子の角度を 6 分割した例を示す。同様に，台車の位置，台車の速度を 6 分割した場合，**図 3** のようにテーブル数は 2592 個となる。Q 学習では，探索行動を行いながら，2592 個のそれぞれのテーブルを（1）式に従って徐々に更新し最適値を求める。状態s_tの分割数の少しの増加であってもテーブル数は大幅に増加するため，学習速度は大幅に低下する。**図 5** は，状態s_tの分割数を 6 分割，8 分割，10 分割した場合，学習が安定するまでの試行回数のグラフを示す。この学習では，学習係数 α を 0.2 とし，割引率 γ を 0.9 とした。また報酬は振子が立っていれば r=1，倒れた時は r=−200 とした。6 分割の場合，学習は 500 回～3000 回程度で収束するが，8 分割 1500 回～4000 回，さらに 10 分割では，15000 回程度を必要とし，分割数が増加するにつれ，試行回数は飛躍的に伸びる傾向がある。以下に実験で実装した python による学習部分のソースコードを示す。

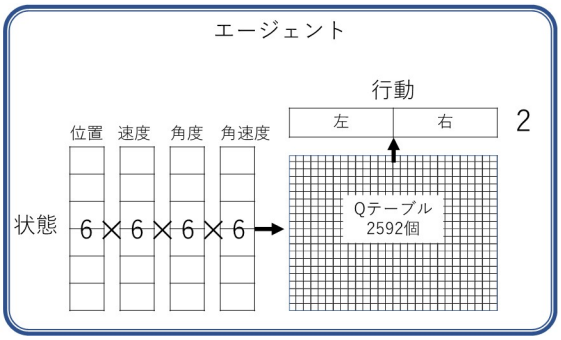

図３　Q学習のテーブル

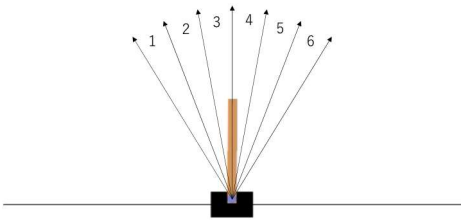

図４　振子の角度を６分割

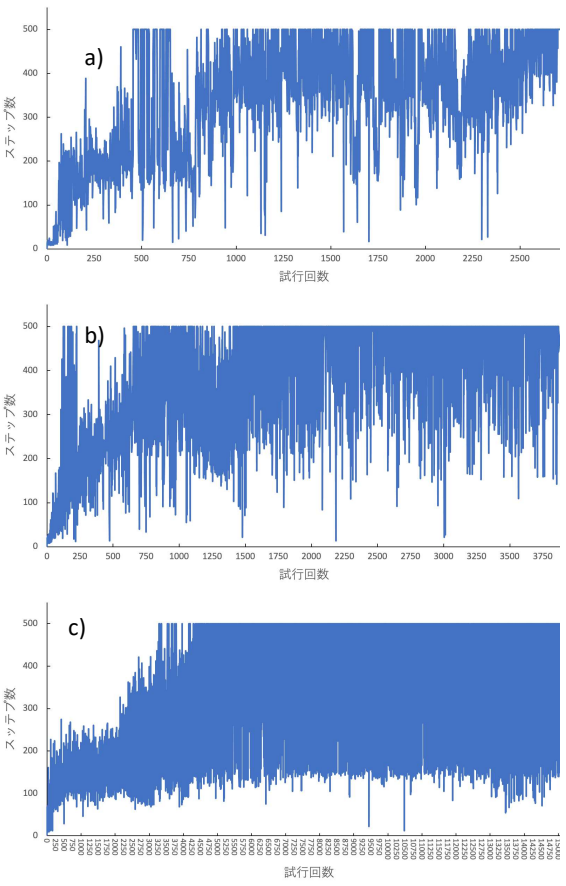

図５　分割数に対する学習速度
(a) 6 分割，(b) 8 分割，(c) 10 分割

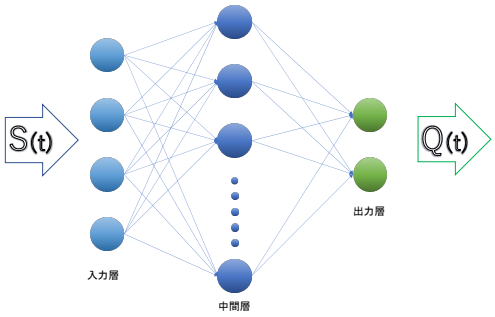

図 6　3 層の DQN

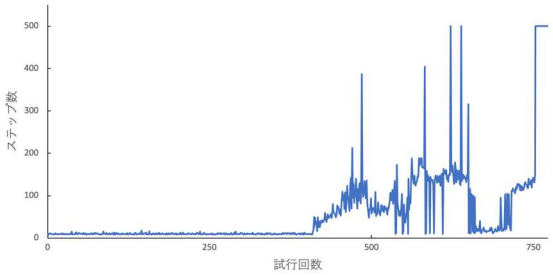

図 7　DQN の学習速度
（中間層 128node）

```
next_state = digitize_state(observation)
next_action = np.argmax(q_table[next_state])

if done:
    q_table[state, action] = (1 - alpha) * q_table[state, action] + alpha * reward

else:
    q_table[state, action] = (1 - alpha) * q_table[state, action] + alpha * (
        reward + gamma * q_table[next_state, next_action])
```

ステップ毎に，Q テーブルの学習を行いながら台車の行動を決定する。試行終了時のみ，次の Q テーブルを参照せず，報酬のみでの更新を行う。また，学習が局所解に収束することを防ぐため 10%はランダムに行動を選択するよう ε -Greedy 法を用いた。

3.2 ＤＱＮ

DQN の実験には**図 6** に示す中間層 1 層からなる全結合 3層ニューラルネットワークを用いた。エージェントは入力層の各ノードに，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）を正規化し入力する。また，出力層の 2 つのノードは左右の行動価値が出力され，エージェントは出力値の大きい方を次の行動として選択し，実行環境に送る。使用した学習部分のソースコードを次に示す。

```
    next_qvalues = model(next_state)
    next_action = next_qvalues.array.argmax()
    if done:
        delta = reward
    else:
        delta = reward + gamma * next_qvalues.array[:,next_action]
    targets = copy.deepcopy(qvalues)
    targets.array[:,action] = delta
```

学習はステップ毎に行われ，ニューラルネットワークの重みが更新される。この実験では，割引率を γ=0.99 とした。また報酬は振子が立った状態を r=1，倒れた時は r=−20 とした。この学習でも局所解に収束することを防ぐため 10%はランダムな行動を選ぶ ε -Greedy 法を用いた。**図 7** は中間層のノード数が 128 個の場合の学習速度を示している。Q 学習と比較し，学習の立ち上がりは遅いものの，安定的に徐々に学習が進む特徴がある。また，入力値が連続値であるのも関わらず，学習速度は Q 学習より速い。また，中間層のノードに対する学習速度は 32 ノードの場合は安定するまで，2000～4000 回の試行を必要としたがノード数の増加に伴い，64 ノードでは 700 回～1800 回，128 ノードでは 600 回～1000 回，256 ノードでは 300 回～1000 回とノード数の増加にも関わらず早期に学習が収束する結果となった。

3.3 モンテカルロ離散方策勾配法

方策勾配法は，Q 学習と全く同じテーブルを使った学習が可能である。参照するテーブルの値は Q 学習がそれぞれの行動に対する価値であったのに対して，方策勾配法では，各行動に対する選択確率のパラメータとなる。Q 学習との比較のための，方策勾配法の実験においても 4 つの環境変数の 6 分割と左右の 2 行動からなるテーブルとした。エージェントは，状態変数に対応するテーブル値 θ から(10)式により左右の行動確率割合を決定し，その割合に応じて行動を決める。具体的なソースコードを次に示す。

```
    tmp=np.array([p_table[state,0],p_table[state,1]])
    tmp=np.exp(tmp)
    tmp/=np.sum(tmp)
    next_action = np.random.choice(tmp.size,p=tmp)
```

学習は，ステップごとでは無く 1 回の試行が終了した後，使用したテーブルの更新を行う。この時，総報酬を(12)式により新しい行動による報酬から順次古いものへと加算を行う。割引総報酬の計算プログラムは下記のとおりである。

```
    r=0.0
    for l in reversed(range(steps)):
        r = buffa[l][2] + gamma*r
        g_reward[buffa[l][0],buffa[l][1]]=r
```

本実験では，学習係数を α=0.2 割引率を γ=0.99 とした。また報酬は振子が立った状態を r=1，成功した時を r=100，失敗し倒れた時は r=−80 とした。また，パラメータ θ の更新は，計算された割引報酬和$G(s_t)$を使い以下のプログラムにより行った。

```
total=np.sum(action_counter)
for l in range (states):
    n=sum(action_counter[l])
    for i in range(actions):
        if action_counter[l,i]:
            p_table[l,i] += alpha * (action_counter[l,i] - p_table[l,2+i] * n) \
                * (g_reward[l,i]-base_line) / total
```

学習の安定化のため，プログラムにはベースライン法も実装している。

図 8 に方策勾配法による学習速度を示す。Q 学習とほぼ同程度の学習速度が得られた。学習に必要な試行回数は，1500 回程度を必要とする。但し，学習が進まない場

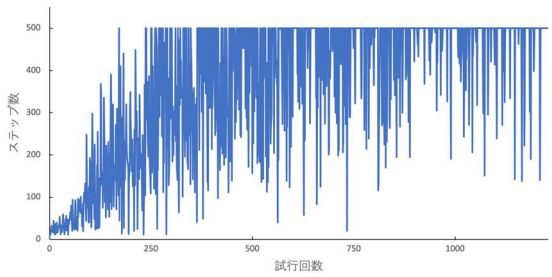

図 8　方策勾配法の学習速度

合もあり，他の強化学習がほぼ一定の学習回数により学習が安定するのに対して，方策勾配法は学習の安定性が非常に悪いと考えられる。またパラメータの選択範囲が非常に狭く，少しの報酬の変更でも，全く学習が進まない状態に陥る。学習が進むパラメータを設定した場合では，700 回程度で学習が安定する場合と，学習が進まない両極端の状態（2 万回以上）となった。このことから，方策勾配法は局所解に陥りやすい傾向があると推察される。

4. 結　　言

Q 学習，DQN, モンテカルロ離散方策勾配法の 3 種類の基本的な強化学習について比較した。Q 学習はテーブルによる学習であるため，学習後テーブルの値を確認し，学習状態を把握できる利点がある。しかし，環境変数やその分割数が多くなるにつれ，飛躍的に学習時間を必要とするため，規模の大きな強化学習には不向きである。DQN では，環境変数を連続値として入力可能であり，学習速度も速い特長がある。但し，ニューラルネットワークの規模などは，入力パラメータ数，出力パラメータ数，学習データ数などに合わせた調整が必要であり，未学習の環境データに対する動作を確実とするためには，過学習とならない汎化能力の高い状態で学習を停止する必要がある。方策勾配法は他の 2 手法と比較し，学習が非常に不安定でありテーブルを用いる手法に利点は無いと考えられる。しかし，今後モータなどの連続値制御が必要な応用については，深層学習による方策勾配法のみが連続値の行動に対応できる利点があり，連続値制御には方策勾配法を用いるほか無い。

文　　献

1) 大槻 知史 他: 最強囲碁 AI アルファ碁解体新書深層学習 モンテカルロ木探索 強化学習から見たその仕組み，翔泳社（2017）.
2) 福島 邦彦：神経回路と情報処理, 朝倉書店(1989).
3) K Hirokawa, K Itoh, Y Ichioka: Invariant pattern recognition by neural networks combined with optical wavelet preprocessor. *Opt. Rev.* **7(4)**, 284-293 (2000).
4) 牧野 貴樹 他：これからの強化学習，森北出版（2016）.
5) 伊藤 真：「強化学習」を学びたい人が最初に読む本，日経 BP マーケティング（2021）.
6) 曽我部東馬：強化学習アルゴリズム入門「平均」からはじめる基礎と応用強化学習アルゴリズム，オーム社（2019）.