離散行動空間における Soft Actor-Critic の評価

合田 拓矢[1,a)]　金子 知適[1,b)]

概要：Soft Actor-Critic (SAC) [1], [2] は連続行動空間を対象とする State of the Art の強化学習手法であり，そのサンプル効率の高さと学習の頑健さから広く用いられている．本研究では，離散行動空間を対象とする場合，SAC は Soft Q-Learning (SoftQ) [3] と等価な手法となることを示し，実験を通して SoftQ を用いる方が SAC の目的関数をナイーブに離散行動空間に適用するよりもサンプル効率が高くなることを示す．また，行動時の方策が学習にどのような影響を与えるかについてや，ソフト行動価値関数を用いることによる特性を実験的に検証する．

キーワード：強化学習，最大エントロピー強化学習，Soft Actor-Critic，Soft Q-Learning

Evaluation of Soft Actor-Critic in Discrete Action Space

Takuya Aida[1,a)]　Tomoyuki Kaneko[1,b)]

Abstract: Soft Actor-Critic (SAC) [1], [2] is the state-of-the-art reinforcement learning method for continuous action domains and it is widely used because of its high sample-efficiency and robustness. In this research, we show that SAC is equivalent to Soft Q-Learning (SoftQ) [3] in discrete action space and SoftQ performs better than the naive version of SAC for discrete action space in terms of sample-efficiency. Moreover, we evaluate the effect of the choice of behavior policies and the characteristics of using soft action value functions through experiments.

Keywords: Reinforcement Learning, Maximum Entropy Reinforcement Learning, Soft Actor-Critic, Soft Q-Learning

1. はじめに

強化学習は，エージェントが試行錯誤を通して報酬を最大化する振る舞いを学習する枠組みである．近年，強化学習の研究は大きな進展を遂げており，ゲーム分野においても，アーケードゲーム [4] や囲碁 [5] で大きな成功を収めている．

Soft Actor-Critic (SAC) [1], [2] はロボット制御など連続行動空間における State of the Art の強化学習手法であり，そのサンプル効率の高さと学習の頑健さから広く用いられている．SAC を提案した文献 [1], [2] では連続行動空間を持つ環境に焦点を当てて手法を提案し，評価実験を行っている．そのため文献 [1], [2] に記述されている SAC を離散行動空間に対して直接適用することはできない．文献 [6] では離散行動空間を前提とした定式化を行い，離散行動空間を対象とした SAC (DiscSAC) を提案している．

本研究では，行動空間が離散の場合，SAC は Soft Q-Learning (SoftQ) [3] と等価な手法であることを示し，実用上も DiscSAC のように方策を別のニューラルネットワークに分離せず，ソフト行動価値関数を用いて方策を構成するほうがサンプル効率がよいことを実験的に示す．さらに，実用上はサンプル収集時に ϵ-greedy 方策を用いるとさらにサンプル効率がよいことを実験的に示す．また，ソフト行動価値関数を用いる場合と通常の行動価値関数を用いる場合を比較し，その特性を示す．

2. 関連研究

2.1 強化学習

本研究では，既存研究に倣って強化学習の環境としてマルコフ決定過程 (MDP) を考え 4 つ組 $(\mathcal{S}, \mathcal{A}, p, r)$ で表現する．ただし，$\mathcal{S}$ は状態空間，$\mathcal{A}$ は行動空間をそれぞれ表し，

[1] 東京大学大学院情報学環・学際情報学府
[a)] takuyaaida@g.ecc.u-tokyo.ac.jp
[b)] kaneko@acm.org

$p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ は状態遷移確率関数，$r : \mathcal{S} \times \mathcal{A} \to R$ は報酬関数を表す．また，状態 $s \in \mathcal{S}$ におけるエージェントが出力する行動の確率分布 $\pi(\cdot|s)$ を方策と呼ぶ．エージェントが方策 π に従って行動する時，時刻 t に受け取る報酬を r_t とすると，強化学習の目的はエピソードの期待累積報酬 $E_\pi \left[\sum_{t=0}^{T} r_t\right]$ を最大化する方策 π を獲得することである．強化学習に関する詳細は [7] を参照されたい．

2.2 最大エントロピー強化学習

最大エントロピー強化学習は，報酬に方策のエントロピー項を加えた収益を最大化する強化学習の枠組みである．最大エントロピー強化学習では，定数 α で重み付けしたエントロピー項 $\mathcal{H}$ を報酬に加えた，以下の目的関数を最大化する方策 π を求める．

$$J(\pi) = \sum_{t=0}^{T} E_\pi \left[r_t + \alpha \mathcal{H}(\pi)\right] \tag{1}$$

この目的関数は，直感的にはできる限り行動の多様性を保ちながら，かつ，得られる報酬を最大化すると解釈でき，エントロピー項が正則化の役割を果たしている．

最大エントロピー強化学習における行動価値関数をソフト行動価値関数と呼び，以下の式で定義される．

$$Q^\pi(s_t, a_t) = r_t + E_\pi \left[\sum_{i=t+1}^{T} \gamma^{i-t} \left(r_i + \alpha \mathcal{H}(\pi(\cdot|s_i))\right)\right] \tag{2}$$

ただし，$\gamma \in [0, 1)$ は割引率である．

最大エントロピー強化学習において，最適方策 π^* は以下の形で与えられることが知られている． [3]

$$\pi^*(a_t|s_t) \propto \exp\left(Q^*(s_t, a_t)/\alpha\right) \tag{3}$$

ただし，$Q^*(s_t, a_t)$ は最適ソフト行動価値関数である．

2.3 Soft Q-Learning

Soft Q-Learning (SoftQ) [3] は最大エントロピー強化学習の枠組みに基づき，ソフト行動価値関数を更新することにより式 (1) を最大化する方策を求める手法である．Soft Q-Learning では以下の更新式に従いソフト行動価値関数の推定値を更新する．

$$Q(s_t, a_t) \leftarrow r_t + \gamma E_{s_{t+1} \sim p}\left[V(S_{t+1})\right] \tag{4}$$

ただし，ソフト状態価値関数 $V(s_t)$ は以下の式で与えられる．

$$V(s_t) = \alpha \log \sum_{a \in \mathcal{A}} \exp\left(Q(s_t, a)/\alpha\right) da \tag{5}$$

ここで，$\alpha \to 0$ のとき，右辺は $\max_a Q(s_t, a)$ となり，通常の強化学習における行動価値関数の更新式に一致する．

文献 [8] に Arcade Learning Environment [9] の一部の環境を対象として，Soft Q-Learning と Deep Q-Network (DQN) [4] の比較実験がなされており，ソフト行動価値関数を用いることでやや性能が向上することが示されている．

2.4 SAC: Soft Actor-Critic

SAC [1], [2] は最大エントロピー強化学習の枠組みに基づく off-policy の model-free 強化学習手法である．off-policy で学習ができることからサンプル効率が model-free の強化学習手法の中では優れており，また，学習率等のハイパーパラメータの変動に対して頑健に動作することが知られている．

パラメータ θ, ϕ を用いて行動価値関数を $Q_\theta(s_t, a_t)$，方策を $\pi_\phi(a_t|s_t)$ と表すとき，SAC は以下の目的関数を最小化することでパラメータの学習を行う．

$$J_Q(\theta) = E_{(s_t, a_t) \sim \mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(s_t, a_t) - \hat{Q}(s_t, a_t)\right)^2\right] \tag{6}$$

$$J_\pi(\phi) = E_{s_t \sim \mathcal{D}, a_t \sim \pi_\phi}\left[\alpha \log \pi_\phi(a_t|s_t) - Q_\theta(s_t, a_t)\right] \tag{7}$$

ただし

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma E_{s_{t+1} \sim p}\left[V(s_{t+1})\right], \tag{8}$$

$$V(s_t) = E_{a_t \sim \pi}\left[Q_\theta(s_t, a_t) - \alpha \log \pi_\phi(a_t|s_t)\right], \tag{9}$$

であり，$\mathcal{D}$ は過去のエピソードを記録した replay memory である．

式 (8) における期待値計算は環境のモデル $p(s_{t+1}|s_t, a_t)$ を必要とするため，実際には replay memory からサンプリングされた状態遷移 (s_t, a_t, s_{t+1}) を用いて近似する．また，式 (7) と式 (9) では方策 π に関する期待値を計算する必要があるが，行動空間が連続の場合は正確な値を求めることは難しいため，方策 π のサンプルを用いて近似する．特に，式 (7) に関しては reparameterization trick [10] と呼ばれる方法を用いて勾配を切らずに目的関数を計算する．

2.5 DiscSAC: 離散行動空間における SAC

SAC [1], [2] は連続行動空間を念頭に提案された手法であるが，その目的関数は離散行動空間に対しても簡単に拡張可能であり，実際に文献 [6] では，SAC を離散行動空間に対応させるための変更点が述べられている．

離散行動空間の場合，方策 $a \in \mathcal{A}$(ただし，$\mathcal{A}$ は取りうる行動の集合) に対応する確率を出力する．そのため，式 (7) や式 (9) において，方策 π に関する期待値を以下のように定義通り計算できる．

$$J_\pi(\phi) = E_{s_t \sim \mathcal{D}}\left[\sum_{a \in \mathcal{A}} \pi(a|s_t)\left\{\alpha \log \pi_\phi(a|s_t) - Q_\theta(s_t, a)\right\}\right] \tag{10}$$

$$V(s_t) = \sum_{a \in \mathcal{A}} \pi_\phi(a|s_t)\left\{Q_\theta(s_t, a) - \alpha \log \pi_\phi(a|s_t)\right\} \tag{11}$$

3. DiscSAC と Soft Q-Learning の関係

前の節で述べた通り，最大エントロピー強化学習における最適方策は $\pi^* \propto \exp(Q^*(s_t, a_t)/\alpha)$ の形で与えられる．また，離散行動空間における方策は，各行動に対応する確率を計算可能であり，方策 π は

$$\pi(a_t|s_t) = \frac{\exp(Q^*(s_t, a_t)/\alpha)}{\sum_{a\in\mathcal{A}} \exp(Q^*(s_t, a_t)/\alpha)} \tag{12}$$

と表せる．

ここで，SAC の方策の目的関数は以下のように表すこともできる．

$$J_\pi(\phi) = E_{s_t\sim\mathcal{D}}\left[D_{KL}\left[\pi_\phi(\cdot|s_t)\middle\|\frac{\exp(Q_\theta(s_t,\cdot)/\alpha)}{Z(s_t)}\right]\right] \tag{13}$$

ただし，D_{KL} は KL ダイバージェンスであり，$Z(s_t)$ は分配関数である．実際，式 (13) の両辺に α をかけて方策の学習に無関係な分配関数の項を無視すれば式 (7) と一致する．

離散行動空間の場合，式 (13) において方策のパラメータを更新する上でターゲットとすべき方策は，式 (12) と表すことができる．ここで，式 (13) は π_ϕ を式 (12) の右辺とすれば 0 になり，このとき最小である．したがって，離散行動空間においてはソフト行動価値関数を用いて方策を直接表現することが可能であるため，別のニューラルネットワークとして近似する必要はない．

また，ソフト状態価値関数について，式 (9) より

$$V(s_t) = E_{a_t\sim\pi}\left[Q(s_t, a_t) - \alpha\log\frac{\exp(Q(s_t, a_t)/\alpha)}{\sum_{a'\in\mathcal{A}}\exp(Q(s_t, a')/\alpha)}\right] \tag{14}$$

$$= \alpha\log\sum_{a\in\mathcal{A}}\exp(Q(s_t, a)/\alpha) \tag{15}$$

となり，式 (5) と一致する．

4. 実験

4.1 実験環境

本研究では実験環境として OpenAI gym [11] に含まれる LunarLander-v2 という環境を用いる．LunarLander-v2 は宇宙船を制御して月面に着陸させるゲームであり，エージェントが取れる行動は { 左，右，上，何もしない } の 4 種類である．状態は位置，速度，回転角と角速度，左右の足が地面と接しているかどうかの情報を含む 8 次元の連続なベクトルで表現されている．左右や上に行動するたびに小さな負の報酬を，無事着陸すると大きな正の報酬がもらえ，エピソードの累積報酬が 200 以上でクリアとなる．

4.2 DiscSAC vs SoftQ

DiscSAC と SoftQ は理論的には等価な手法であるが，方策を別のニューラルネットワークで表現するという実装上の違いがある．そのため，その差が性能に影響することも十分に考えられる．また，その他にも実用上考慮すべき実装の選択肢について比較するために，以下の 3 つのバリエーションを用意して実験を行った．

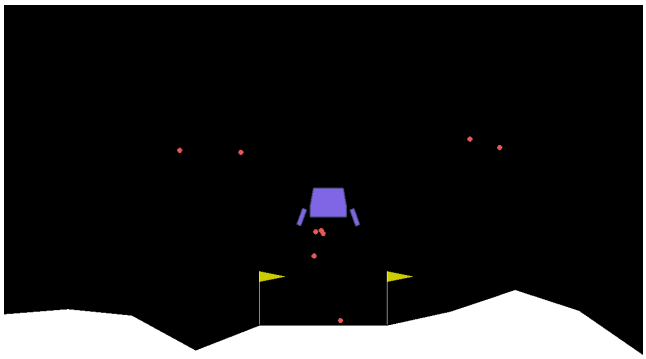

図 1 LunarLander-v2 環境
Fig. 1 LunarLander-v2 environment

- DiscSAC
- SoftQ
- サンプル収集時に ϵ-greedy 方策を用いる SoftQ

これら 3 つの手法は，ソフト行動価値関数の学習に関しては全く同一である．一方で，サンプル収集時における行動の選び方は異なり，DiscSAC は分離した方策ネットワークが出力するカテゴリカル分布から，SoftQ はソフト行動価値関数を用いて算出したカテゴリカル分布からそれぞれサンプリングする．また，サンプル収集時に ϵ-greedy 方策を用いる SoftQ では，確率 ϵ でランダムな行動を選択し，確率 $1-\epsilon$ で最もソフト行動価値関数が高い行動を選択する．なお，ϵ の値に関しては $\epsilon = 1$ から始めて，線形に $\epsilon = 0.1$ まで減衰させている．この実験における，SoftQ の実装の詳細は A.1 を参照されたい．

図 2 に異なるランダムシードを用いて 5 回学習を行った結果を示す．実線は各ランダムシードに対して，一定のタイムステップ間隔で 5 回ずつ，計 25 回の評価用エピソードを実行した際の平均を，影をつけた部分はその平均±標準偏差の範囲を表している．ϵ-greedy 方策を用いる SoftQ が最もサンプル効率がよい一方で，DiscSAC は最もサンプル効率が悪く，理論上も実用上も行動空間が離散の場合は方策を別ネットワークに分離して学習する必要はないといえる．

4.3 行動時の α のスケジューリング

前項で示した通り，SoftQ を用いる際には，データ収集時に $\exp(Q(s, a)/\alpha)$ に比例する確率的な方策を用いずに，ϵ-greedy 方策を用いるとよりサンプル効率が高くなる．そこで，サンプル収集時の α の値をスケジューリングすることで，ϵ-greedy 方策に振る舞いを近づければ，同等のサン

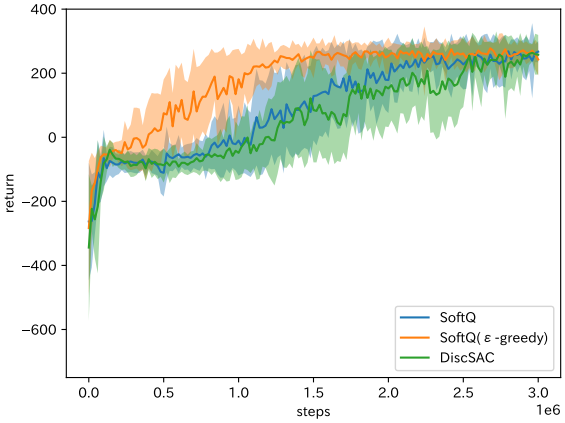

図 2 LunarLander-v2 における DiscSAC および SoftQ の学習結果

Fig. 2 Learning curves of DiscSAC and SoftQ in LunarLander-v2 environment

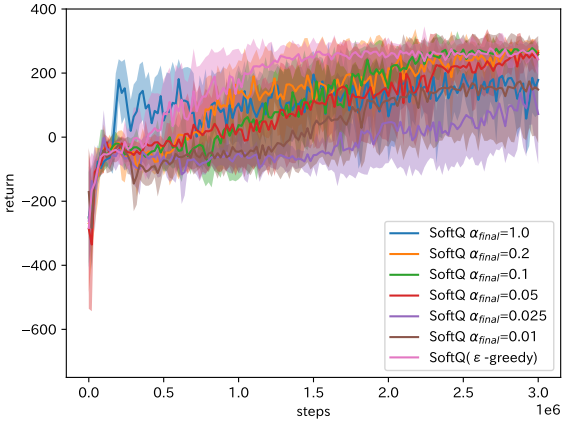

図 3 サンプル収集時に用いる α をスケジューリングした場合の学習結果

Fig. 3 Learning curves of various final values of α for behavior policies

プル効率を達成できるか検証した．

図 3 に $\alpha = 10$ から始めて様々な値に α を線形に減衰させたときの学習結果を，**図 4** に各エピソードにおいて greedy に行動した割合を示す．**図 3** において，最も速くクリアの水準に到達しているのは，サンプル収集時に ϵ-greedy 方策を用いる SoftQ である．一方，**図 4** において ϵ-greedy 方策を用いた SoftQ と比較的近い挙動を示している $\alpha_{final} = 0.05$ や $\alpha_{final} = 0.025$ は，**図 3** においては ϵ-greedy 方策を用いるバージョンと比較して性能が下がっている．学習初期はソフト行動価値関数が不正確なため，その出力から分布が定まる探索方法では影響を受けやすい一方で，ϵ-greedy 方策を用いる場合はその影響が小さくて済むと考えられる．

4.4 状態や行動へのノイズに対する頑健さ

ソフト行動価値関数を用いることで，通常の行動価値関数を用いる場合と比較して，状態や行動のノイズに対してどの程度ロバストに振る舞えるかを検証するために，通常の LunarLander-v2 環境に以下の修正を加えた環境を用意し，DQN との比較実験を行った．

- 状態ベクトルのうち，位置・速度・回転角と角速度に対応する成分について，各成分ごとに $\pm 100s\%$ の範囲で一様分布からサンプリングしたノイズを加える環境 (StateNoise 環境)
- エージェントが取った行動を確率 p でランダムな行動に置き換える環境 (ActionDisturb 環境)

SoftQ と DQN のそれぞれについて，通常の LunarLander-v2 環境で学習させた後，評価時のエピソードの累積報酬の平均が最も良いパラメータを用いて StateNoise 環境と ActionDisturb 環境における zero-shot のパフォーマンスを評価した．この実験における SoftQ の実装は，DQN の実装と条件を揃えるため，どちらも行動時の方策として ϵ-greedy 方策を用いる．また，ターゲットネットワークの更新は，DQN の実装に合わせて，一定タイムステップ間隔ごとにコピーすることとした．それ以外の実装の詳細については，他の実験と同様，A.1 に基づいて行った．

図 5 に SoftQ と DQN を通常環境で学習させた結果を示す．SoftQ と DQN の結果を比較すると，DQN のほうが速くクリア水準に到達していることがわかる．SoftQ と DQN の実装やハイパーパラメータは行動価値関数の更新を除いては共通であり，サンプル収集時にはともに ϵ-greedy 方策を用いていることから，ソフト行動価値関数の学習は通常の行動価値関数と比較すると学習に必要なステップ数が大きいといえる．

次に，学習済みの SoftQ と DQN のパラメータを用いて，StateNoise 環境と ActionDisturb 環境における zero-shot でのパフォーマンスを評価した．ここで，評価対象とする

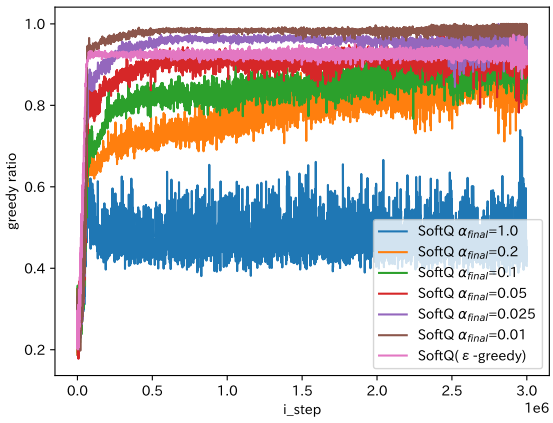

図 4 行動時に用いる α をスケジューリングした場合における，エピソード内で greedy に行動した割合

Fig. 4 ratios of greedy actions in episodes for various final values of α for behavior policies

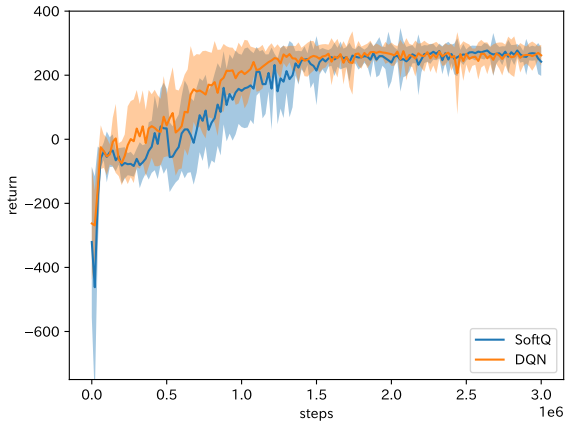

図 5 通常の LunarLander-v2 環境における SoftQ と DQN の学習結果

Fig. 5 Learning curves of SoftQ and DQN in LunarLander-v2

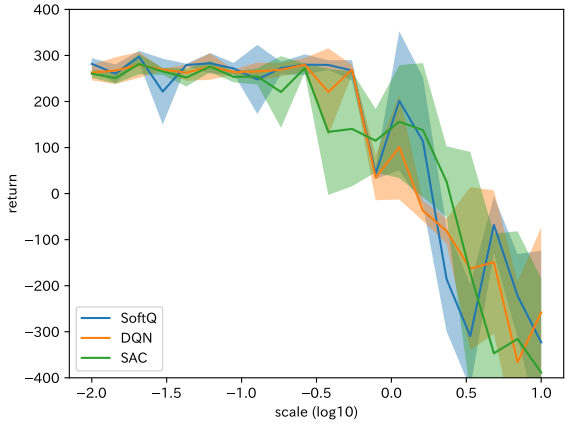

図 6 StateNoise 環境において s を変化させたときの zero-shot での SoftQ と DQN の評価

Fig. 6 Zero-shot evaluation of SoftQ and DQN in StateNoise settings

パラメータは，**図 5** において最もエピソードの累積報酬の平均が高いものを用いた．StateNoise 環境における結果を**図 6** に，ActionDisturb 環境における結果を**図 7** に示す．p や s を大きくすると zero-shot でのパフォーマンスはどちらも同じように下がっていき，SoftQ と DQN で大きな差は見られなかった．

StateNoise 環境については，行動空間が離散か連続かによらず同じ基準で評価することが可能である．そこで，LunarLander-v2 環境の行動空間を連続にした環境 (LunarLanderContinuous-v2) を用いて SAC を学習させ，同様に zero-shot での評価を合わせて行ったが，SoftQ や DQN と同様の性能を示した．SoftQ や SAC は方策のエントロピー項を含めた目的関数を用いて学習するが，状態や行動へのノイズに対しては DQN 以上の頑健さは確認できなかった．

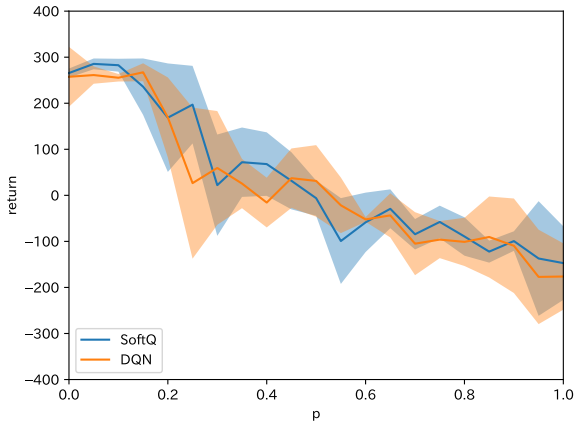

図 7 ActionDisturb 環境において p を変化させたときの zero-shot での SoftQ と DQN の評価

Fig. 7 Zero-shot evaluation of SoftQ and DQN in ActionDisturb settings

5. 結論

離散行動空間における SAC の目的関数は SoftQ と等価である．実装上は行動時の方策を独立に学習するかどうかで違いがあるが，LunarLander の実験においては方策を別のネットワークに分離しない SoftQ の方がサンプル効率が高かった．また，実用上はサンプル収集時に定義通りの方策を用いずに，ϵ-greedy 方策を用いるほうがさらにサンプル効率が高い．最大エントロピー強化学習の枠組みに基づく SoftQ は多少の環境の変化に頑健な方策を学習できると期待される．そこで，環境の状態や行動にノイズが加えられたときに zero-shot でどの程度ロバストに動作するかを DQN と比較したが，本研究における実験の範囲では大きな違いは見られなかった．

今後の課題として，Atari やそれ以外のより複雑な環境において SoftQ と DQN がどの程度性能差を示すかを検証することが挙げられる．特に，環境やタスクに応じて手法を使い分けるという観点から，どのような特性を持つ場合にソフト行動価値関数を用いるべきかを明らかにすることは興味深いテーマである．

参考文献

[1] Haarnoja, T., Zhou, A., Abbeel, P. and Levine, S.: Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, *Proceedings of the 35th International Conference on Machine Learning*, pp. 1861–1870 (2018).

[2] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P. and Levine, S.: Soft Actor-Critic Algorithms and Applications, *CoRR*, Vol. abs/1812.05905 (online), available from ⟨http://arxiv.org/abs/1812.05905⟩ (2018).

[3] Haarnoja, T., Tang, H., Abbeel, P. and Levine, S.:

Reinforcement Learning with Deep Energy-Based Policies, Proceedings of Machine Learning Research, Vol. 70, International Convention Centre, Sydney, Australia, PMLR, pp. 1352–1361 (2017).

[4] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G. et al.: Human-level control through deep reinforcement learning, *nature*, Vol. 518, No. 7540, pp. 529–533 (2015).

[5] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. et al.: Mastering the game of Go with deep neural networks and tree search, *nature*, Vol. 529, No. 7587, pp. 484–489 (2016).

[6] Christodoulou, P.: Soft Actor-Critic for Discrete Action Settings, *CoRR*, Vol. abs/1910.07207 (online), available from ⟨http://arxiv.org/abs/1910.07207⟩ (2019).

[7] Sutton, R. S. and Barto, A. G.: *Reinforcement learning: An introduction*, MIT press (2018).

[8] Schulman, J., Abbeel, P. and Chen, X.: Equivalence Between Policy Gradients and Soft Q-Learning, *CoRR*, Vol. abs/1704.06440 (online), available from ⟨http://arxiv.org/abs/1704.06440⟩ (2017).

[9] Bellemare, M. G., Naddaf, Y., Veness, J. and Bowling, M.: The arcade learning environment: An evaluation platform for general agents, *Journal of Artificial Intelligence Research*, Vol. 47, pp. 253–279 (2013).

[10] Kingma, D. P. and Welling, M.: Auto-Encoding Variational Bayes, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings* (Bengio, Y. and LeCun, Y., eds.), (online), available from ⟨http://arxiv.org/abs/1312.6114⟩ (2014).

[11] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. and Zaremba, W.: OpenAI Gym, *CoRR*, Vol. abs/1606.01540 (online), available from ⟨http://arxiv.org/abs/1606.01540⟩ (2016).

[12] Kingma, D. P. and Ba, J.: Adam: A Method for Stochastic Optimization, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Bengio, Y. and LeCun, Y., eds.), (online), available from ⟨http://arxiv.org/abs/1412.6980⟩ (2015).

付　　録

A.1 実装の詳細

本研究で行った実験の，実装の詳細について説明する．学習にあたりエージェントは合計 300 万タイムステップの行動を行い，16 タイムステップに 1 度パラメータの更新を行った．また，20000 ステップに一度，5 回の評価用エピソードを実行し，各エピソードの累積報酬を記録した．ソフト行動価値関数 $Q_\theta(s,a)$ のターゲットの更新には，以下の更新式のように指数移動平均を用いた．

$$\theta^{\text{target}} \leftarrow \tau\theta + (1-\tau)\theta^{\text{target}} \tag{A.1}$$

ハイパーパラメータは**表 A·1** に示す通りである．

表 A·1　ハイパーパラメータのリスト
Table A·1　List of hyperparameters

ハイパーパラメータ	値
最適化手法	Adam [12]
エントロピー項の係数 (α)	0.2
学習率	3×10^{-4}
割引率 (γ)	0.99
ソフト行動価値関数のターゲット更新の係数 (τ)	0.005
ソフト行動価値関数の隠れ層の数	2
ソフト行動価値関数の隠れ層のユニット数	128