環境モデルの誤差による影響を抑える強化学習手法

中田 惇貴[1,a)]　鶴岡 慶雅[2]

概要：近年, 機械学習の手法の 1 つである強化学習は, ゲームやロボット制御などのタスクにおいて高い性能を示している. その一方で, 良い方策を学習するためには, 環境との多くの相互作用を必要とする. そのため, 現実のロボット制御など環境との相互作用を十分に行うことが難しく, 大量のデータを集めることができない環境ではより少ないデータからより良い方策を学習することが求められる. 本研究においてはサンプル効率の良いモデルベース強化学習に着目し, その中でも課題となる環境モデルの誤差の影響を抑えつつ方策の学習を行うことを目的とする. 本稿では, 環境モデルとしてマルチステップ予測モデルを採用し, モデルを用いて方策を学習する際のステップ数を学習が進むにつれて, 徐々に伸ばすことを提案する. 実験においては, CartPole-v0 という環境において, 提案手法により環境モデルを効果的に用いることができるかについて検証を行った. その結果として, 性能の向上が見られることを確認した.

A Reinforcement Learning Method to Reduce the Effects of Environmental Model Errors

Atsuki Nakata[1,a)]　Yoshimasa Tsuruoka[2]

Abstract: Recently, a machine learning paradigm called reinforcement learning has shown high performance in tasks such as games and robot control. On the other hand, learning a good policy requires a lot of interactions with the environment. Therefore, in some environments where it is difficult to fully interact with the environment, or where it is not possible to collect large amounts of data, such as real-life robot control, it is necessary to learn better policy from less data. This study proposes a method for sample efficient model-based reinforcement learning, which aims to learn polices while minimizing the effects of environmental model errors. In this paper, we adopt a multi-step prediction model as an environmental model and propose to gradually increase the number of steps in learning policy using the model as the learning progresses. In our experiments, we verify whether the proposed method can be used effectively in CartPole-v0. As a result, it was confirmed that the performance was improved.

1. はじめに

強化学習は実際の環境に対して行動を繰り返し行うことにより得られた情報から, どのような行動を選択すべきかを改善する手法である. 強化学習を適応する対象としては, Atari2600 といったゲームや, 囲碁, ロボットの制御のようなものが挙げられる. 性能に関しては, 例えば Atari2600 においては近年, 57 個すべてのゲームにおいて人間を上回るようなスコアを達成するような方策を学習することができることが主張されている [1].

このように, 環境に対する試行錯誤を十分な回数行うことができれば人間を上回るような方策を学習できる一方で, ロボット制御など, 実際の環境で多くの動作を行うことが難しい環境においては, より少ない試行回数からより良い方策を学習できるようにサンプル効率を改善する必要がある. サンプル効率の良い強化学習手法として知られているモデルベース強化学習はこの課題を解決するための方法の 1 つである.

[1] 東京大学工学部電子情報工学科
Department of Information and Communication Engineering, The University of Tokyo
[2] 東京大学大学院情報理工学系研究科電子情報学専攻
Department of Information and Communication Engineering, Graduate School of Information Science and Technology, The University of Tokyo
[a)] nakata@logos.t.u-tokyo.ac.jp

しかしモデルベース強化学習では, 環境モデルの不正確さの影響により, 最終的な性能がモデルフリーの手法と比較して劣ってしまうという課題がある [2]. この課題に対して, モデルの誤差による影響を抑えつつ方策の学習を行う手法が研究されている.

先行研究 [3] では, 環境モデルとしてはマルチステップ予測モデルを採用することで, 状態の予測の誤差が拡大してしまうことを抑える手法が提案された. しかし先行研究においては, モデルを用いて方策を学習する際, 常に固定長のステップ数を用いることにより学習を行っていた. 一般に, モデルの精度は学習が進むにつれて向上するものであるが, 先行研究においては, この精度を考慮せずにモデルを利用していた. したがって, モデルを有効的に活用する方法については改善の余地がある.

本研究では, 環境モデルとしてマルチステップ予測モデルを採用し, モデルの精度を考慮した上で方策の学習に用いることにより, モデルが効果的に活用できるかについて, 性能を評価することを目的とする.

2. 関連研究

2.1 強化学習

強化学習とは環境に対する行動を繰り返し行うことによって得られた結果から, 与えられた状態に対してより優れた行動を選択する方法を学習する手法である. この枠組みでは環境とエージェントという 2 つの対象を考える. 環境は, 行動, 状態, 行動に応じた状態の変化, ある状態における行動に対して与えられる報酬といった要素を備えている. また, エージェントはどのような行動をとるかという方策に従い, 環境に対して実際に行動を行う主体である. 環境によって状態が与えられ, その状態に対してエージェントが行動を選択, 実行することで, 環境の状態が変化し, 即時報酬を得る. 強化学習における目的は, 方策に従って行動することにより環境から得られる期待収益を最大化するものである.

他によく知られている機械学習手法として教師あり学習と教師なし学習がある. 教師あり学習においてはデータセットと正解のラベルが与えられ, 未知のデータに対して, 正解のラベルを予測するように学習するものである. また, 教師なし学習ではデータセットが与えられ, それらのデータをいくつかのグループに分類し, 未知のデータがどのグループに属するものであるかを予測するように学習する. それに対して, 強化学習はデータセットそのものは与えられず, 環境との試行錯誤により自らデータを集め, そのデータにより学習する点において両者と異なる.

2.1.1 マルコフ決定過程

また, 強化学習における枠組みとしてよくマルコフ決定過程 (Markov Decision Process, MDP) が用いられる. これは, ある状態は 1 つ前の状態と行動によってのみ決定されるというものである. 強化学習における MDP は, 状態集合である $\mathcal{S}$, 行動の集合である $\mathcal{A}$, ある状態 $s \in \mathcal{S}$, 行動 $a \in \mathcal{A}$ に対して, 次にどのような状態 $s' \in \mathcal{S}$ に遷移するのかという遷移関数 $T(s'|s,a)$, 同じく行動, 状態に対して, どのような報酬が与えられるのかという報酬関数 $R(s,a)$ によって定められる. このように表現された MDP におけるエージェントと環境の関係は図 1 に示される.

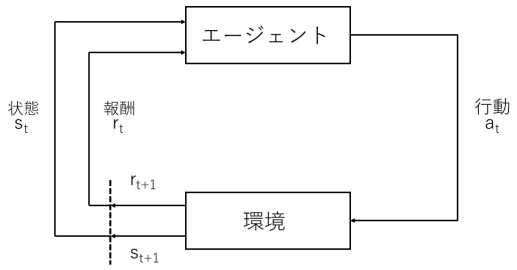

図 1 マルコフ決定過程 (MDP) におけるエージェントと環境の関係

2.1.2 強化学習における表記

強化学習では, 環境から与えられた状態をもとに, 環境に対してエージェントが何かしらの行動を実行し, その結果エージェントは環境から報酬と次の状態が与えられる. これらの一連の流れを 1 ステップとし, 環境の状態が終端状態になるか, 一定のステップ数が経過するまで繰り返す. このように環境の開始から終了までをまとめてエピソードと言う. また, あるエピソードの中で, 時刻 t における状態を s_t, 行動を a_t, 報酬を r_t とそれぞれ表記する. ある状態 s が与えられたとき, エージェントがどのような行動 a を選択するかという方策を $\pi(a \mid s)$ と表す.

エピソードが時刻 T で終了する場合, ある時刻 t における報酬の総和は, 将来得られる報酬は不確かな値となることを考慮し, その分の値を割り引いて計算する. そのための係数を割引率と言い, 0 以上 1 以下である γ を用いて報酬の総和は以下のように定義される.

$$\begin{aligned} G_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^{T-t-1} r_T \\ &= \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k \end{aligned}$$

このように定義した G_t を割引報酬和と言う. 強化学習ではこの G_t を最大化することを目的とする.

状態 s からある方策 π に基づいて行動することによって得られる状態価値 $V_\pi(s)$ が以下のように定義される.

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi \left[G_t | s_t = s \right] \\ &= \mathbb{E}_\pi \left[\sum_{k=t+1}^{T} \gamma^{k-t-1} r_k \middle| s_t = s \right] \end{aligned}$$

同様に, 状態 s において行動 a を選択した後, ある方策 π に基づいて行動することによって得られる状態行動価値 $Q_\pi(s,a)$ は以下のように定義される.

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}_\pi\left[G_t | s_t = s, a_t = a\right] \\ &= \mathbb{E}_\pi\left[\sum_{k=t+1}^{T} \gamma^{k-t-1} r_k \middle| s_t = s, a_t = a\right] \end{aligned}$$

2.2 Q 学習 (Q-learning)

強化学習のアルゴリズムの 1 つとして Q 学習 [4] がある. Q 学習では環境に対する試行錯誤により状態行動価値 $Q(s,a)$ を学習するアルゴリズムである. Q 値は, ある時刻 t における状態 s_t, 行動 a_t, 行動によって得られた報酬 r_{t+1}, 次の状態 s_{t+1} を用いて以下の式によって更新される.

$$\begin{aligned} Q(s_t,a_t) \leftarrow\ & Q(s_t,a_t) \\ & +\alpha(r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)) \end{aligned} \tag{1}$$

上記の式において α は学習率と呼ばれるパラメータである. 更新式は, 既存の状態行動価値の見込み値である $Q(s,a)$ が, 実際の行動から得られた報酬に基づいた期待収益 $r_{t+1} + \gamma \max_a Q(s_{t+1},a)$ により近づくように学習されていくということを示している.

Q 学習における行動選択の手法の 1 つとしては, 時刻 t において, 最も状態行動価値が高い行動を選択するという方策が考えられる. つまり

$$a_t = \arg\max_a Q(s_t,a)$$

として行動を選択する. このように行動を選択する手法を greedy 法と呼ぶ. しかし, このように行動を選択してしまうと, 初期のランダムに決まった Q 値が最も高い行動のみが選択されてしまうため, 状態を十分に探索することができない. これを解決するための方策の 1 つとして, ϵ-greedy 法がある. これは, ある ϵ をパラメータとして設定し, ϵ の確率で行動をランダムに選択, $1-\epsilon$ の確率で状態行動価値が最も高い行動を選択するという方策である. つまり, 以下のように行動を選択する.

$$a_t = \begin{cases} \arg\max_a Q(s_t,a) & (\text{with probability } 1-\epsilon) \\ \text{a random action} & (\text{with probability } \epsilon) \end{cases}$$

Q 学習における課題として, 状態空間や行動空間の次元数が大きくなると, それぞれの Q 値を表すことが難しくなるということが挙げられる. 近年, 強化学習において Atari2600 のようなゲームを対象とするタスクなど, 入力が画像となるような環境での実験も行われており, そういった環境においてはこのような Q 値の学習が難しい.

2.3 Deep Q-Network (DQN)

Q 学習で問題であった次元の問題を解決するために, Minh により Q テーブルをニューラルネットワークで近似する Deep Q-Network (DQN) [5] が提案された. DQN においては, Q 関数は状態を入力とし, 出力は取りうる行動数と同じ次元のベクトルを出力するニューラルネットワークを用いる. 出力層におけるそれぞれの値はある行動を選択した場合の Q 値を表している. また, DQN ではニューラルネットを用いていることを明示的にするため, ニューラルネットのパラメータを θ として, Q 値を $Q(s,a;\theta)$ と表す.

Q 学習では (1) により Q 値の更新を行うが, これは現在の Q 値を, 実際の行動によって得られた報酬に基づく期待収益に近づくように更新するものであった. つまり, $Q(s_t,a_t) \approx r_{t+1} + \gamma \max_a Q(s_{t+1},a)$ となるように Q 値を更新した. DQN においては損失関数 $L(\theta)$ を以下のように設定し, 損失関数が最小となるようにニューラルネットの学習を行う.

$$L(\theta) = \mathbb{E}\left[\left(r_{t+1} + \gamma \max_a Q(s_{t+1},a;\theta) - Q(s_t,a_t;\theta)\right)^2\right] \tag{2}$$

実際に, DQN を用いて方策を学習させる際には, 学習の安定化のためにいくつかの工夫を行う.

1 つ目の工夫として, Experience Replay と呼ばれる手法を用いる. 従来の Q 学習においては, 1 ステップごとに得られた状態, 行動, 報酬をもとに Q 値の更新を行った. しかし, これでは時間的に相関が高い内容をニューラルネットワークが学習するため, 学習が安定しづらいという欠点がある. これを改善するために, 各ステップごとの結果をメモリに保存しておき, ニューラルネットを学習させる際に過去の経験からランダムに内容を取り出して学習に用いる, Experience Replay の考え方が取り入れられた.

2 つ目の工夫として, Fixed Target Q-Network と呼ばれる手法がある. これは (2) 式においてニューラルネットを更新する際, 遷移先の価値を求める際にも同じニューラルネットを用いているが, この場合, 学習のたびにパラメータが変化してしまうため学習が安定しないという欠点がある. これに対して, 遷移先の価値を求めるために, 一定期間パラメータを固定したネットワークを用いる手法が用いられる. 一般に遷移先の価値を求めるネットワークは, メインで用いるネットワークの少し前の時間のパラメータを用いる.

今回行った予備実験においては, エージェントの方策を学習させるために, DQN を用いた学習を行った.

2.4 モデルベース強化学習

強化学習は環境との試行錯誤により期待される収益を最大化するような方策を学習する手法であるが, その一方で

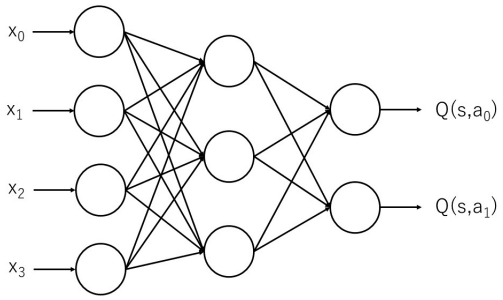

図 2 状態空間が 4 次元, 行動空間が 2 次元である環境における, DQN [5] による Q 関数の表現

大量のデータを集めるために, 環境に対する試行の回数が非常に多くなってしまうという課題がある. 強化学習は実環境に対する行動により得られた情報から, 明示的に環境のモデルを作成し, そのモデルを用いて学習を行うか否かにより大きく 2 つの手法に分類される. 前者をモデルベース強化学習, 後者をモデルフリー強化学習と呼ぶ. モデルベース強化学習は環境のモデルを用いることにより, 実環境におけるサンプル効率を改善する手法である.

環境のモデルを作成するためには, 実際の環境と同じ要素を備える必要がある. 具体的には, 環境のモデルとして, 状態, 行動から次の状態を出力する遷移関数 $T(s,a)$, 同じく状態, 行動を入力として報酬を出力する報酬関数 $R(s,a)$, 状態が終端状態に達したか否かを判定する終了モデル $d(s,a)$ の 3 つのモデルを学習する必要がある.

モデルを利用した手法の 1 つである Dyna [6] では, 環境のモデルを実環境のシミュレータとして用いることで, 実環境を利用した学習に加え, 作成した環境モデル内においても同様に方策を学習する手法が提案された. Dyna では実環境と作成したモデルを併用してエージェントの学習に用いることにより, 実環境のみから学習した場合と比較して, より少ないサンプル数で多くの学習を行うことができるため, サンプル効率が良いという利点がある.

2.5 環境モデルの信頼性

モデルベース強化学習はサンプル効率が良いという利点がある一方で, モデルが不正確である場合, そのモデルを利用することで, 不正確なモデルにおいて期待収益を最大化するように学習する. そのため, モデルフリーの手法と比較して最終的な性能が劣ってしまうことが知られている.

特にモデルベース強化学習で課題となるものとして, 環境モデルの誤差が積み重なってしまうという問題がある. これはモデルを実際の環境のように用いる際, 遷移関数モデルに基づいて次の状態を出力するが, 出力された状態はさらに次の状態を出力するための入力として用いられる. したがって, 遷移関数モデルの出力に誤差がある場合, その誤差を含んだ出力が次の入力として用いられるため, 誤差が拡大してしまうという問題である.

誤差の拡大の影響を抑えつつ, 学習を行う先行研究はいくつか存在する. その 1 つに, Kaiser らにより提案された Simulated Policy Learning (SimPLe) [7] という手法がある. SimPLe は Atari2600 の環境を対象とするモデルベース強化学習手法である. この手法においては, モデルを用いたロールアウトを行う際, モデルの誤差があまり拡大しないように, 本来のエピソードのステップ数よりも少ないステップ数において学習を行った. しかし, そのままでは環境の情報を十分に得ることができないため, 各エピソードの開始状態は実際の環境から得られたデータからをランダムに取り出すことで対応した. これにより SimPLe は他のモデルフリーの手法と比較して高いサンプル効率が得られることを示した.

ロールアウトの長さと環境内を十分に学習できるかはトレードオフの関係にあり, 適切な長さを設定することで, モデルから得られる情報を方策の学習に最大限利用することができる. これを考慮し, Janner らはロールアウトの長さを, モデルの学習が進むにつれて線形に増加させる手法である model-based policy optimization (MBPO) [8] を提案した.

また, モデル内で得られた情報を, 状態価値関数の改善に用いる手法である Stochastic Ensemble Value Expansion (STEVE) [9] において, Buckman らはモデルを複数用いるアンサンブル手法により, 各状態ごとの不確かさを推定し, その情報をもとに状態価値関数を推定する際に用いるロールアウトの長さを動的に調整した.

モデルの誤差が拡大してしまう影響を抑える別の試みとして, Asadi らはマルチステップ予測モデルを用いる手法を提案した [3]. この手法では, H ステップ先の状態の予測のために, 各 $h \in \{1, 2...H\}$ に対して h ステップ先の状態を予測する, 計 H 個のモデルをそれぞれ用いる. この手法においては, ひとつ前のモデルの出力を次のモデルの入力として用いないため, 誤差が拡大するのを抑えることが可能となる. これにより, 1 ステップ先の予測を H 回用いた場合と比較して高いサンプル効率が得られることを示した. マルチステップ予測モデルを用いた状態の予測と 1 ステップモデルを用いた状態の予測は図 3 に示されている.

3. 提案手法

本研究では, モデルベース強化学習においてモデルの誤差の影響を抑えつつ, より効率的な学習手法を提案する. 具体的には, まずマルチステップ予測モデルを用いることにより, モデルの誤差そのものを抑える. その後モデルの活用法として, MBPO [8] において提案されたように, 学習が進むにつれて, より先のステップまでの予測を学習に用い

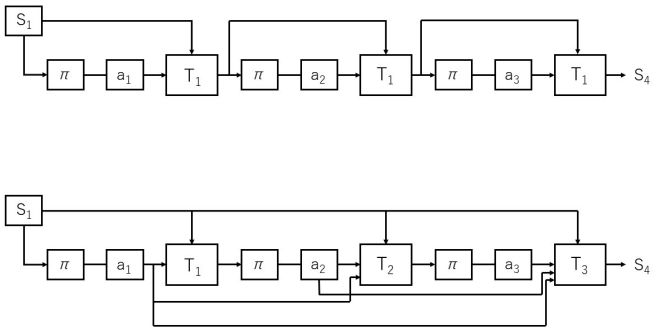

図 3 1 ステップ予測モデルによる状態の予測 (上) とマルチステップモデル [3] による状態の予測 (下)

アルゴリズム 1 環境モデルを用いた学習

Input model dataset D_{model}, and environment dataset D_{env}
Input a policy $\pi(a|s)$, a Q-function Q(s, a), a reward model $f_r(s,a)$, and a multi-step transition model $f_t^h(s,a)\ \forall h \in [1,n]$
set step size k
for M model rollouts **do**
  Sample s_0 uniformaly from D_{env}
  for $i = 0 : k-1$ **do**
    $a_i \sim \pi(\cdot|s_i)$
    $s_{i+1} \leftarrow f_t^{i+1}(s_0, a_0, ..., a_i)$
    $r_i \leftarrow f_r(s_i, a_i)$
    $D_{\text{model}} \leftarrow D_{\text{model}} \cup (s_i, a_i, r_i, s_{i+1})$
    update Q-function using D_{model}

ることを提案する.

マルチステップ予測モデルを提案した先行研究においては, 常に固定長のステップ数の予測データを用いることにより方策の学習を行っていたため, ステップ数を可変とした用いた際の性能についてはまだ検証がなされていない.

MBPO においては 1 ステップ予測モデルの誤差が拡大する悪影響を抑えるために, この学習手法を用いたが, マルチステップモデルの各予測器はより先の状態を予測するためには, 入力するデータの次元が増えたり, 学習するデータ自体が少なくなったりするため, 学習の初期では誤差が大きいことが予想される. したがって, この誤差の影響を抑えるために MBPO と同様の手法を用いることで, 効率の良い学習が行われることを期待する.

4. 実験

4.1 実験の概要

実験として, 先行研究 [3] において用いられたマルチステップ予測モデルを用いた DQN の学習を行った. ここで用いた DQN の Q 関数は 3 層のニューラルネットで実装を行った. また, マルチステップ予測モデルの有効性を確認するために, 1 ステップ予測モデルを用いて学習を行った場合に得られた結果との比較を行った. またモデルの用いる方法としては, Dyna [6] のように, 実環境で得られたデータによりモデルを学習し, さらにそのモデルを実環境のシミュレータとして使用することで得られたデータを用いて, 方策の学習を行った.

モデルの学習, モデルを用いた方策の学習は実環境で 1 エピソードが終わるたびに行った. モデルを用いた学習の際, 開始地点となる状態は実環境で集めたデータからランダムに取り出した. なお, モデル内で学習する総ステップ数は, どのステップサイズを選択した場合であっても一定となるようにした.

また, モデルを用いて方策の学習を行う際, ステップ数を動的に変化させる方法として, MBPO と同様に, ステップ数を線形に増加させる方法を採用した. これは a 番目のエピソード数から b 番目のエピソード数にかけてステップ数を x から y に増加させるとすると, ある e 番目のエピソードにおけるステップ数 $S(e)$ は以下で表されるものとなる.

$$S(e) = \min\left(\max\left(\left\lfloor x + \frac{e-a}{b-a}\cdot(y-x)\right\rfloor, x\right), y\right) \quad (3)$$

なお, 式 (3) において $\lfloor x \rfloor$ はガウス記号を表しており, x を超えない最大の整数を表すものとする.

4.2 実験環境

実験における環境として, OpenAI Gym のタスクの 1 つである CartPole-v0 を用いた. CartPole-v0 ではカートの位置, カートの速度, 棒の角度, 棒の角速度といった 4 つの状態が与えられ, 行動としてカートを右に押すか左に押すかの 2 つのうち, どちらかを選択する. エピソードの長さは 200 ステップであり, 各ステップごとに棒が一定以上傾くか, カートが中心から離れすぎると終了と判定し, 終了しなければ報酬+1 を得る. つまり, 棒が倒れないように台車の制御をうまく行うことがこの実験におけるタスクである. CartPole-v0 での学習の様子は図 4 に示されている.

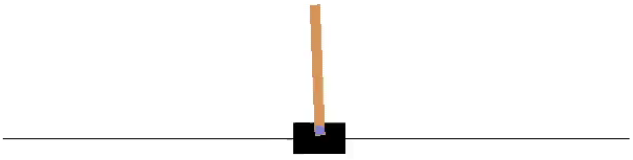

図 4 CartPole-v0 における学習の様子

4.3 環境モデル

実験において, 環境のモデルとして以下の 3 つのものを作成した.

- 遷移モデル: 時刻 t における状態 s_t と, h 個の連続する行動系列 $a_t, a_{t+1}, ..., a_{t+h-1}$ を入力として, h ステッ

プ先の状態 s_{t+h} を出力するモデル

- 報酬モデル: 時刻 t における状態 s_t と, 行動 a_t を入力とし, 報酬を出力するモデル
- 終了モデル: 時刻 t における状態 s_t と, 行動 a_t を入力とし, 状態が終端状態に達したか否かを出力するモデル

遷移モデルは計 H 個実装し, それぞれ複数ステップ先の状態を出力するために用いた. また, 比較として 1 ステップ予測モデルを用いる場合は, 遷移モデルとして時刻 t における状態 s_t と, 行動 a_t を入力とし, 次の状態 s_{t+1} を出力するモデルを 1 つのみ実装した. これらのモデルはすべてニューラルネットワークを用いて実装し, 実環境から得られたデータをもとに, 教師あり学習を行うことによりモデルを学習した. それぞれのモデルについて遷移モデルを f_t, 報酬モデルを f_r, 終了モデルを f_d と表す. また, ニューラルネットワークの最適化手法としては Adam [10] を採用した.

遷移モデル, 報酬モデルは 2 層のニューラルネットで実装し, 終了モデルは 3 層のニューラルネットで実装した. 各モデルの学習は損失関数をそれぞれ L_t, L_r, L_d として以下のように設定し, これらを最小化するように学習を行った.

なお, 遷移モデルは H 個存在し, h ステップ先を予測するモデルを明示的にするため, f_t^h と表し, 損失関数を L_t^h と書く.

$$L_t^h = (s_{t+h} - f_t^h(s_t, a_t, a_{t+1}, ..., a_{t+h-1}))^2$$
$$L_r = (r_t - f_r(s_t, a_t))^2$$
$$L_d = \text{cross\_entropy}(f_d(s_t, a_t), d_t)$$

ここで $\text{cross\_entropy}(y, t)$ は, モデルによって出力された予測 y, 正解のラベル t に対する交差エントロピー誤差を表すものとする.

4.4 結果

マルチステップ予測モデルを用いた CartPole-v0 において, モデルを用いた学習においてステップ数を学習を通して固定長として用いたものと, 線形に増加させたものとを比較した. 学習させるエピソード数は 200 とした.

また, 提案手法において増加させるステップ数としては, 式 (3) において, $a : 20$, $b : 170$ としてそれぞれ設定した. また x, y に関しては $x : 1$, $y : 5$ と設定したものと, $x : 1$, $y : 10$ と設定したものとの 2 つをそれぞれ設定した. 各パラメータにおける結果はそれぞれ図 5, 6 に示されている.

比較として, ステップ数を固定長としてマルチステップ予測モデルを用いたものと 1 ステップ予測モデルを複数回用いたものとの 2 つを各パラメータについて設定して実験を行った.

また, 実験におけるランダムさによる影響を減らすため, 各パラメータにおける実験はそれぞれ 15 回ずつ行い, 図中の実線を中央値, 色付き領域を標準偏差として表した.

図 5 を見ると, 報酬の推移として, まず 1 ステップ予測モデルを用いた場合とマルチステップ予測モデルを用いた場合ではマルチステップ予測モデルを用いた方がより高い報酬を得ることができていることが分かる. またマルチステップ予測モデルを用いた場合においても, ステップ数を固定長で用いる場合よりも線形に増加させた方が性能が良いということが分かる.

次に図 6 を見ると, 図 5 の場合と同様に, マルチステップ予測モデルを用いた際の性能が良いことが分かる. また, ステップ数を固定長としたものと可変長としたものとの比較では, 性能の差異はほぼ同じか, やや可変長にしたものの方が良いということが分かる.

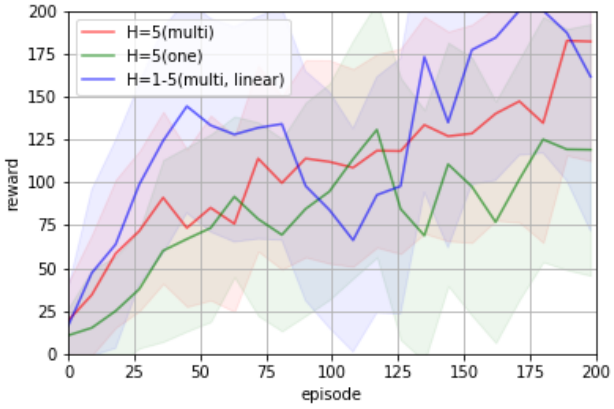

図 5 H=5 としたときの報酬の推移

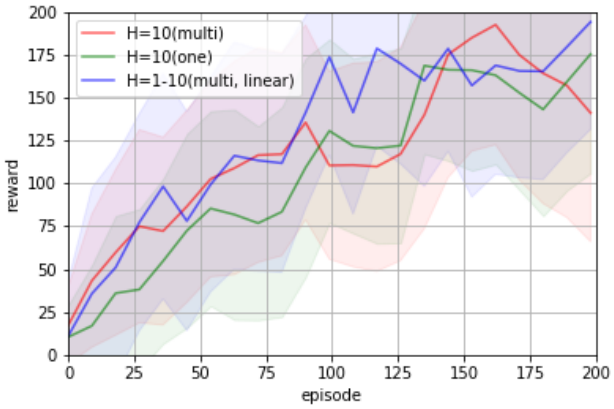

図 6 H=10 としたときの報酬の推移

4.5 考察

2 つのパラメータにおける実験から, 提案手法において, マルチステップ予測モデルを有効的に活用できることが分かった. しかし, 図 6 の場合には固定長のものを用いた場

合と比較してあまり差がなかった.

これはモデルを用いる際のステップ数を手動で設定したことによる影響が大きいと考えられる. 学習が進むにつれ, モデルの精度は向上する一方で, 開始地点となる状態は実環境から得られたデータを採用するため, 学習後半においても, あまり学習できていないような状態が入力として用いられることがあると考えられる. 逆に, 学習初期において精度よく学習できている状態に対しても, ステップ数が少ないことにより, 環境モデルを十分に活用できていないということも考えられる.

ステップ数をエピソード数に対して線形に増加させることではこういった問題に対処することができず, 今回の実験において性能にバラツキが生じたと考えられる.

5. 終わりに

本稿では, モデルベース強化学習において問題となっている環境モデルの誤差を抑える手法の 1 つである, マルチステップ予測モデルを効果的に活用する方法について検証を行った.

今後の課題としては, まず異なる環境においても実験を行うことであると考える. 今回実験を行うために用いた CartPole-v0 という環境は単純なタスクであるため, より状態空間, 行動空間の次元が大きい環境においても実験を行いたいと考えている.

また, 今回はモデルを用いた学習において, ステップ数の増加はエピソード数に対して線形に増加させる方法を採用した. しかしこれはどのような遷移で増加させるかを手動で決定する必要があり, 未知の環境を対象とする際などに応用できない. したがって, モデルの不確かさを用いたステップ数の動的調整を行いたいと考えている.

1 ステップ予測モデルの場合だと, 環境モデルを複数学習させ, 各モデルの予測の分散といった情報などを不確かさの予測に用いることが多い. 一方, マルチステップ予測モデルを用いる場合では, そもそもモデルを複数学習させているため, さらに各ステップ数における予測モデルを複数学習させることは計算コストの問題がある. したがって, ある H ステップ先の予測に対し, h ステップ先の予測と $H-h$ ステップ先の予測を組み合わせて用いることによって得られた複数の予測をもとに, 不確かさを推定することを考えている. その後, 得られた情報をもとに, より効果的なモデルの活用法を検証することを今後の課題としたい.

参考文献

[1] Adrià Puigdomènech Badia, B. Piot, Steven Kapturowski, P. Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. *ArXiv*, abs/2003.13350, 2020.

[2] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *CoRR*, abs/1805.12114, 2018.

[3] Kavosh Asadi, Dipendra Misra, Seungchan Kim, and Michael L. Littman. Combating the compounding-error problem with a multi-step model. *CoRR*, abs/1905.13320, 2019.

[4] C. J. C. H. WATKINS. Learning from delayed rewards. *PhD thesis, Cambridge University, Cambridge, England*, 1989.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[6] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990.

[7] Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2020.

[8] Michael Janner, Justin Fu, Marvin Zhang, and S. Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.

[9] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, pages 8224–8234, 2018.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.