# モデルベース強化学習における方策ネットワーク手法の活用

藤田 航輝[1,a)]　鶴岡 慶雅[2]

**概要**：近年モデルベース強化学習がモデルフリー強化学習に対して, 獲得報酬という点で優れた性能を出す手法が多く提案されている. モデルベース強化学習はモデルの学習と方策や状態価値関数の学習を交互に行うことが一般的である. モデルの学習では, その精度を向上させるために, 複数の遷移モデルをニューラルネットワークで近似し, それをアンサンブル学習するという手法が提案されている. また方策の学習では, ある長さの行動列を生成しその行動列自体を最適化するという手法 (モデル予測制御, MPC) が提案されており, 多くのタスクで性能の向上が見られた. 一方で行動次元数の多いタスクでは性能が上がらないという課題がある. 本稿では HalfCheetah という環境で, MPC を用いる POPLIN という既存手法の方策アルゴリズムの一部を変化させることによる性能の変化を検証し, モデルの精度が低い学習前半に元となる手法を上回る報酬を獲得することができた.

# Utilizing Policy Networks in Model-Based Reinforcement Learning

Koki Fujita[1,a)]　Yoshimasa Tsuruoka[2]

***Abstract:*** Recently, many methods in which model-based reinforcement learning performs better than model-free reinforcement learning in terms of acquisition rewards have been proposed. Model-based reinforcement learning generally alternates between model learning and learning of policy or state-value functions. For model learning, in order to improve the accuracy of the model, the method of approximating multiple transition models with neural networks and ensemble learning of them has been proposed. For learning of policy, the method called Model Predictive Control (MPC) that optimizes the action sequence itself by generating action sequences of a certain length has been proposed, and there was an improvement in performance for many tasks. On the other hand, the performance is not improved for tasks with a large number of action dimensions. This paper examines the performance of an existing method, POPLIN, which uses MPC in the HalfCheetah environment, by changing a part of its policy algorithm. We show that the rewards outweighed the original method in the first half of the training, when the model was less accurate.

## 1. はじめに

強化学習の手法は大きく 2 つに分類することができる. モデルフリー強化学習, モデルベース強化学習である. これらは環境を遷移関数としてモデル化するか否かによって分類される.

モデルフリー強化学習は環境から遷移モデルを学習させず, データを収集するたびに環境と相互作用を行う必要がある. モデルフリーな手法としては例えば方策勾配法 [1] が挙げられる. これは収集したデータを用い方策をその勾配方向に変化させることで方策を最適化させていく方法である. その他にも Q 学習 [2] などによる方策の評価と方策の更新を交互に行う Actor-Critic [3] という方法などもあり, 現在非常に多くの手法が提案され, 性能が向上している.

一方モデルベース強化学習では環境から遷移モデルを学習する. このモデルを学習させるためのコストはかかるが, 新たなデータを遷移モデルから取り出すことが可能であるため, 環境と相互作用する回数を少なくできるという意味でサンプル効率がモデルフリーに比べて良いとされている. 理想的には環境と誤差のない関数モデルを作ることができ

[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 Graduate School of Information Science and Technology, The University of Tokyo
[a)] kfujita@logos.t.u-tokyo.ac.jp

れば, 例えばロボットの実験などの環境との相互作用を行わなくて良いことになる. しかし, 学習されたモデルでは環境とのずれが生じてしまう. 1 つは, 環境からデータを集める際に実際に取れるデータにはノイズが乗ってしまい, 理想的なデータを得られない場合があるということ, 2 つめは環境から得られるデータには多くのデータを取れるような状態と, 逆に少ないデータしか取れない状態が存在してしまう. 特にデータ点が少ない部分では学習モデルも不正確になってしまい, そのずれが蓄積することで, 実環境と全く異なる状態に進むことになりうる. このような問題点が存在する. 次にモデルベース強化学習にはモデルを学習した後プランニングを行うことでエージェントが環境に適した行動を行うことができるようにする. これらにはランダムシューティングや交差エントロピー法 [4] などがあり, これらの手法によってある長さの行動列を最適化する. この行動列を構成する 1 つ 1 つの行動はある方策 $\pi$ や, それまでの高い報酬の得られた複数の行動列の各時間ごとの分布から求める. またこのような手法に対し, 近年この方策を求める際にニューラルネットワークを用いる手法が提案されてきている. 本研究ではモデルベース強化学習のプランニング手法に注目している. 本稿で行なったことは以下の通りである. また確認できたこととして, 過去のよい行動列の分布を取り込む比率を変化させることで幅広い探索が可能となることが示唆された.

- 既存手法と提案手法の性能比較
- 方策の学習に必要なデータセットを変化させた時の性能の違いの比較

これらの結果は 6 章で詳しく記載した. また具体的な既存手法のアルゴリズムは以下の 4 章で詳しく記載した.

## 2. 本研究の目的

本研究ではモデルベース強化学習において, その方策を工夫しニューラルネットワークを用いることで, 行動空間やパラメータ空間が広く学習が難しいとされるタスクにおいて, よい性能を上げることができる強化学習のプランニング手法を提案することを目的としている.

## 3. 関連研究

モデルベース強化学習という観点から考えると古くは Dyna [5] と呼ばれる手法がある. これは環境から集めたデータとモデルから得たデータを用いて Q 関数を更新していくという手法の原型に当たる. またゲーム AI の分野ではモンテカルロ木探索 (MCTS) [6] と呼ばれる手法が存在する. これは既知な環境に対し探索を行いより報酬が高くなると考えられる部分や未知の部分を集中的に探索することで探索を効率化し, よりよい方策を求める手法である. これは将棋や囲碁などの環境が既知な状況で使われる手法である. MCTS にニューラルネットワークを用いた AlphaGo [7] や AlphaZero [8] という手法が人間を倒したということで知られている. 近年では環境が未知で行動空間の広い環境に対し, モデルベース強化学習を組み入れる研究がなされている. Nagabandi [9] らは, 遷移モデルの関数をニューラルネットワークで近似し, またプランニングにおいて行動の列をランダムシューティングで生成するという手法を提案し, モデルベース強化学習の性能を向上させた. 加えて Chua ら [10] は, ニューラルネットワークで近似した遷移モデルをアンサンブル化しサンプルに対し過適合しすぎないように工夫し, プランニングでは行動の一連の列に対し交差エントロピー法で行動列をプランニングするという PETS という手法を提案した. また Michael ら [11] は同様にモデルをアンサンブルで学習させ, モデルのロールアウトの長さを変化させ, 方策の学習で Soft Actor-Critic (SAC) [12] を用いる MBPO という手法を用いることで多くのタスクで PETS を上回る性能の向上に成功した.

同時に, Wang ら [13] はプランニング時の行動列を現方策のニューラルネットワークから求める POPLIN という手法を提案し, 多くのタスクにおいて既存のモデルベースな手法を上回る性能を得ることに成功した. また Lai ら [14] は環境から順方向, 逆方向と 2 つのモデルを作ることで, モデルの実環境との複合誤差を小さくし, 方策でも同様に順方向, 逆方向で学習させることでさらに多様なタスクにおいて, 性能を上げることに成功した BMPO という手法がある.

## 4. 背景

本研究において, いくつかの既存手法を用いるため, はじめにこれらの手法のアルゴリズムの概要について説明する.

### 4.1 強化学習の体系

強化学習とは, ゲーム AI やロボットなどのエージェントが環境と相互作用を行い, エージェントが獲得できる累積報酬を最大化させるような方策を求める方法であり, 機械学習の一種とされている. 図 1 で示すように, エージェントは環境内のある状態を取り, そこで取る行動を方策により選択し, 次の状態に進むことが可能となり, またその行動がどれほど最適なものかを示す報酬という値がエージェントに与えられる. 一般に強化学習はマルコフ決定過程 (MDP) と呼ばれる数学的枠組みによって定式化される. 本研究で行われる離散時間 MDP は $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma, \rho)$ によって定義される. $\mathcal{S}$ は状態集合を表し, $\mathcal{A}$ は行動集合を表す. $\mathcal{T}$ はある状態, 行動の組 $(s, a)$ から次の状態 $(s')$ への遷移確率を表す. $r$ はある状態 $s$ において行動 $a$ を選択した時の報酬を表し, $\gamma$ は割引率を, $\rho$ は初期状態分布を表す.

すなわち強化学習では MDP のもとで 以下の式で表される報酬を最大化する方策を求める手法である.

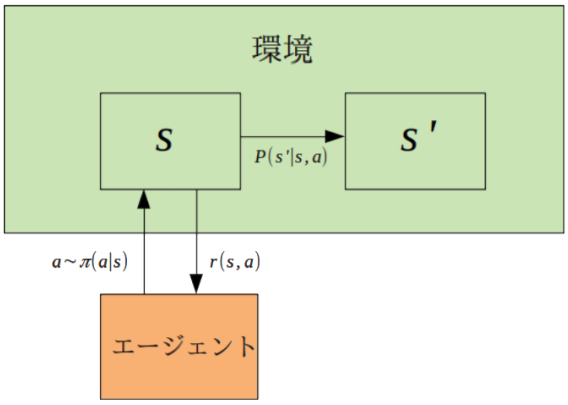


**図 1** 強化学習の模式図

$$J(\pi) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

また状態遷移に関しては, 次の状態は 1 つ前の状態とそこでの行動のみに依って決まり, それ以前の過程は問わないというマルコフ性のもとで遷移する.

### 4.2 Model Predictive Control (MPC)

モデルベース強化学習における, 方策や状態価値関数の推定のために Model Predictive Control (MPC) と呼ばれる手法がある. これはモデルベース強化学習では, 遷移モデルを用いるため, これを利用してある長さの行動の列を行動列単位で最適化していくという手法である. まず, ある分布や現在の方策によって, 行動を生成し, 行動列 $[a_t, a_{t+1}, ..., a_{t+\tau}]$ を作る. この行動列を複数作り, 作成した遷移モデル $f_\theta$ を用いて,

$$s_{t+1} = f_\theta(s_{t+1}|s_t, a_t)$$

上記のように遷移を行う. この遷移のたびに報酬 $r(s_t, a_t)$ を得ることができるため, それを $\tau$ 回行うことで, 行動列に対応する累積報酬を得ることができる. 生成した全行動列のうち最も報酬が得られたものを選び, 最初の行動 $a_t$ を実際の環境で実行し, サンプルデータを得る. これを一定回繰り返し, 得られたデータを基に遷移モデルや方策を学習させる. この一連のプランニングアルゴリズムが MPC と呼ばれる手法である. 本稿で用いる既存手法は全て MPC をモデルとしている.

### 4.3 ランダムシューティングと交差エントロピー法

MPC によって, 行動列を作る際に用いられる手法としてランダムシューティングと交差エントロピー法がある. ランダムシューティングでは行動列のそれぞれの行動をランダムに生成する. これは特定の分布にとらわれずにランダムな行動を選択できるが, 良い行動列を得るために非常に時間を要する. それを解決するための手法として交差エントロピー法がある. これはまず行動列上の各行動をある分布で仮定する. すなわち確率分布を $Pr(\theta)$ として $a_t = Pr(\theta_t)$ のように行動を独立に選び行動列を生成し MPC を実行する. すなわちどのような分布を用いれば高い報酬が得られたかがわかるため, これまでの行動列のうちよい報酬が得られたもの (エリート) の分布を用いて, 新しい分布のパラメータを変化させることができるという手法である. 例えば本稿で取り上げる既存手法ではガウス分布を仮定しているため, それまでの分布のパラメータ, エリートの分布のパラメータをそれぞれ $\mu$, $\Sigma$ $\mu'$, $\Sigma'$ として以下の式のように平均と分散を更新する.

$$\mu = \alpha\mu + (1-\alpha)\mu' \tag{1}$$

$$\Sigma = \alpha\Sigma + (1-\alpha)\Sigma' \tag{2}$$

この方法では分布の更新ごとに報酬が高い行動から抽出された分布が生成されるため, 時間効率は非常に良い.

### 4.4 PETS

PETS という手法はモデルベース強化学習の手法の 1 つであり PE というモデルの構築手法, TS と呼ばれるエージェントの行動の生成手法に分割して考えられる. PE はモデルの遷移関数をニューラルネットワークで近似し, それをアンサンブル化する手法である. ここで遷移関数を

$$f_i(\theta) = Pr(s_{t+1}|s_t, a_t)$$

とし, ある状態とその時の行動を入力とし, 次状態を出力とした遷移関数を複数 (B) 個作成する. これらを基に最終的に選択される状態は

$$f(\theta) = \frac{1}{B}\sum_{i=0}^{B} f_i(\theta)$$

から出力される平均値によって出力される. また本手法では確率分布をガウシアン分布として出力される. そのためニューラルネットワークの損失はパラメータ $\theta$ 平均 $\mu$ 分散 $\Sigma$ を用いて以下の式で表される.

$$loss(\theta) = \sum_{i=1}^{N} [\mu_\theta(s_i, a_i) - s_{i+1}]^{\mathrm{T}} \Sigma_\theta^{-1}(s_i, a_i)[\mu_\theta(s_i, a_i) - s_{i+1}] + \log\det\Sigma_\theta(s_i, a_i)$$

このニューラルネットワークを学習させることでモデルを作成する.

次に行動列のプランニングを行う. このために交差エントロピー法によりある長さの行動の列を作る. 次に行動を開始する初期状態を複数用意する. 作成したモデルに従い各初期状態から状態を遷移させていく. この遷移が TS という手法である. それぞれのデータが累積報酬を獲得することになり, 最終的な報酬値はこれらの報酬の平均とされ

る. 報酬値を基に, 行動列の良さを判断し交差エントロピー法で行動列を求める分布を変更する. この行動列は報酬の高い行動列をエリートとして選択し, 交差エントロピー法により行動列の分布を変更し, 生成する. これを行動列のみを変更し一定ステップ繰り返すことで, 最も報酬が大きい行動列を求めることができる. その最初の行動のみを実際の環境で実行する. これを一定回数繰り返し, ある行動列を実際の環境で実行することになり, そこで得られた状態と行動, 次の状態のタプルをデータセットに保存し, 次のモデル作成に用いる. 以上の方法を繰り返すことでモデルの精度向上とプランニングを行う.

### 4.5 POPLIN

本手法は環境からモデルの作成は PETS と同じ PE というアンサンブル手法を用いる. PETS と異なる点は行動のプランニング手法にニューラルネットワークを組み込むこと, また交差エントロピー法を行動列ではなく空間へのノイズ成分に適用することである. POPLIN には 2 種類あり, POPLIN-A と POPLIN-P である. POPLIN-A では, まず現在の状態での方策によって得られた行動を初期行動列に加える. これをあるステップ数繰り返すことで初期行動列を生成する. 次に, 得られた行動列のそれぞれの行動に対して, ノイズを加え, モデルを用いて状態を遷移させ, 獲得報酬の大きさをもとにノイズの分布を交差エントロピー法を用いて更新していく. 最終的に最も報酬の獲得できる行動列を求め, その最初の行動を実際の環境で実行する. これを繰り返すことで得られる実環境との相互作用のデータを用いて環境モデルと方策のパラメータを更新していく. このときの行動列 $[\hat{a}_t, \hat{a}_{t+1}, ..., \hat{a}_{t+\tau}]$ に対する報酬は $\delta$ をノイズ成分として以下のように定義される.

$$R(s_i, \delta_i) = \mathbb{E}\left[\sum_{t=i}^{i+\tau} r(s_t, a_t = \hat{a}_t + \delta_t)\right]$$

また状態遷移以下のようになる.

$$s_{t+1} = f_\phi(s_{t+1}|s_t, a_t = \hat{a}_t + \delta_t)$$

一方, POPLIN-P では方策のパラメータを更新する流れは同じであるが, POPLIN-A では行動に直接ノイズを加えて行動列を生成したのに対し, パラメータ空間に対してノイズを加えることで行動列を生成するという手法になっている. すなわち $\omega$ をノイズ成分として報酬と状態遷移は以下のように表される.

$$R(s_i, \omega_i) = \mathbb{E}\left[\sum_{t=i}^{i+\tau} r(s_t, a_t = \pi_{\theta+\omega_t}(s_t))\right]$$

$$s_{t+1} = f_\phi(s_{t+1}|s_t, a_t = \pi_{\theta+\omega_t}(s_t))$$

つまり両者の違いは方策のニューラルネットワークを行動空間に適用するかパラメータ空間に適用するかの違いである. また本実験では方策のニューラルネットワークの学習に Behavior cloning (BC) という手法を用いた. BC とは以下の式で表されるように方策を, 実際の環境での行動との差を最小化するようにパラメータを更新するという手法である.

$$\min_\theta \mathbb{E}_{s,a\in D}||\pi_\theta(s) - a||^2$$

### 4.6 既存手法の課題

既存手法ではいくつかのタスクではサンプル効率が高く, それに加えモデルフリー強化学習の手法と同等以上の性能が出ている. 一方 Wang ら [15] はモデルベース強化学習の多くの手法の性能比較を行なっているが, どれもタスク依存であり, 多くのタスクで成功していると考えられる手法が少なく, Humanoid や Walker などの次元数の多い行動を必要とするタスクでは成功しているものが少ない. この理由としては高次元タスクの行動空間の広さに対する探索が不足していることが考えられる.

## 5. 提案手法

本稿で提示した既存手法では MPC の行動列の導出に交差エントロピー法を用い, 分布のパラメータを更新しているが, その際, 取り入れるエリート分布の割合は常に一定である. すなわちイテレーション数に関係なく優秀な分布を同じ比率で取り込む. 一方でモデルの学習は徐々にデータの量が増え, より正確になっていくと考えられる. それゆえ, 学習の前半と後半でエリートと呼ばれる行動列から遷移モデルを介して行われる状態の変化は, 後半になるに連れてより実環境と近い状態になり, その行動列に対する報酬予測の信頼度は高くなると考えられる.

POPLIN という手法が行動空間が広いタスクでは性能が低い理由には, 行動列の探索における多様性が小さいためであるというのが考えられる. これを改善するには, 報酬予測の信頼度が低い学習前半では, ノイズの分布をエリート分布に近づけすぎるのではなく, より多様性のある広い探索を行うことで行動空間の広さに対応できる可能性があると考えられる. これを実現するためには, 分布にエリート分布を取り込む比率を学習前半では, 低く抑えることで, よりランダム性の高い分布でサンプリングを行えるようになり, これが探索の多様性につながると考えられる. またモデルの信頼度が向上する学習後半ではエリート分布を取り込む比率を上げ, 最終的な報酬を向上させることで, 行動空間の広いタスクであっても一定の性能を出すことができると期待される. 具体的には, (1) , (2) 式の $\alpha$ をイテレーション数の変化と共に, 徐々に大きくすることでエリート分布を取り込む比率を動的に変化させ, 学習前半に多様な行動を作り出すことができ, 複雑なタスクであっても, 性能

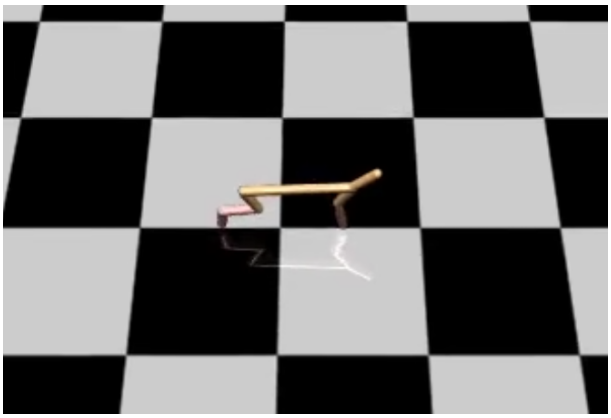

**図 2** HalfCheetah の環境

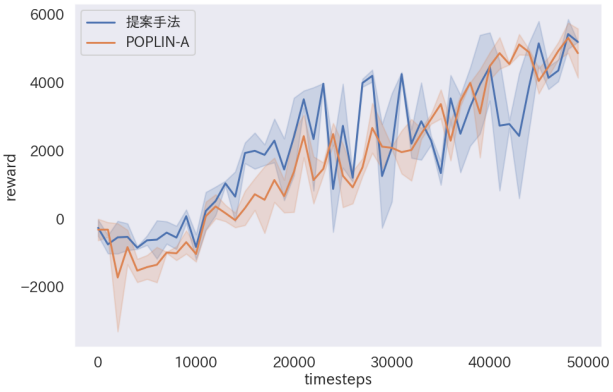


**図 3** HalfCheetah において POPLIN-A と提案手法の性能の比較

が安定することが期待される.

## 6. 実験

### 6.1 実験の概要

実験 1 として, 提案した手法を POPLIN-A の実装を一部変更させ, エリート分布を取り込む割合をイテレーションの回数と共に変化させて, 性能の比較を行なった.

実験 2 では POPLIN-A に対し方策に用いた BC のデータセットを実環境との相互作用のデータのみからなるデータセットと, それに加えモデルから遷移したデータを組み合わせたデータセットを用いた場合で性能にどれほど差が生じるのかの実験を行なった.

実験 2 では提案手法とは異なり, 交差エントロピー法によるエリートの取り組む比率は 0.9 で一定である. また全ての実験結果は 2 回の平均値となる.

本実験で用いたモデルの学習には 3 層の隠れ層を用いた 5 層ニューラルネットワークを用いており, 活性化関数は swish [16] を使用している. またアンサンブルの数は 5 つである.

### 6.2 実験の目的

実験 1 では提案手法と既存手法の差異に注目し, 改善が見込めたか否かを検証する. 実験 2 では方策に必要なデータセットを変化させ, 既存手法がモデルから取り入れたデータを活用できるかどうかを検証することが目的となる.

### 6.3 実験に用いた環境

本実験では全て MuJoCo [17] 上の HalfCheetah の環境を用いた. これは, 前足, 後ろ足の 2 足歩行のシミュレータが走った距離と, 加えたトルクの大きさから報酬を獲得できるタスクである. 図 2 では本実験で用いた HalfCheetah の環境を示す.

### 6.4 実験結果及び考察

実験 1 では (1), (2) 式の $\alpha$ をイテレーション数を t として

$$\alpha = 0.5 - \frac{t}{100}$$

という一次関数的に変化させることで, 学習時の得られる報酬の挙動や, 最終的な獲得報酬が既存手法とどのように異なるのかの実験を行なった. この実験 1 の結果を図 3 で表す. なおイテレーション数は 50 回であるため $\alpha$ は 0.5 から 0 までを 0.01 刻みで変化する. 既存手法では $\alpha$ は 0.1 で固定である.

実験結果から, イテレーション数が 10 から 30 付近において, 既存手法よりも高い報酬が比較的安定して得られるようになった. モデルが不正確な状況である学習前半では性能は不安定で低いことが予想されるが, 本手法のように, ランダム性をやや増やすことで, 探索空間が広くなり, 局所的な解の分布に落ちなかったため, ある一定の性能を出すことができたと考えられる. このことはモデルの学習が難しいタスクにおいて, 良い報酬が得られる可能性があることが示唆される結果となった. 一方で, 最終的な報酬値に大きな変化は見られなかった. これは, 提案手法は既存手法を部分的に変化させただけであり, 手法としての大枠のアルゴリズムは同じである. そのため, 最終的な結果としては大きな変化が見られなかったと考えられる. また実験環境である HalfCheetah は行動次元が少なく, 様々な強化学習の手法で実験の行われている比較的簡単なタスクである. そのため, 手法の差異によらずある程度学習が進めば, 性能が向上するのではないかと考えられる.

次に実験 2 の結果を図 4 で表す. 実験 2 では POPLIN-A という手法において, 方策の学習に BC を用いたが, BC に用いるデータセットを実際の環境からのみ集めたデータを用いた場合 (BC-AR) と, それに加え訓練中のモデルから生成されたデータを加えたを用いた場合 (BC-AI) の性能の違いを比較した. ここで実験 1 では POPLIN-A, 提案手法では BC-AI を用いている. 図 4 から実環境以外からもデータを加えたデータセットのほうがより性能が高くなることが確認された. 特にイテレーション数が 40 を超えた学習後

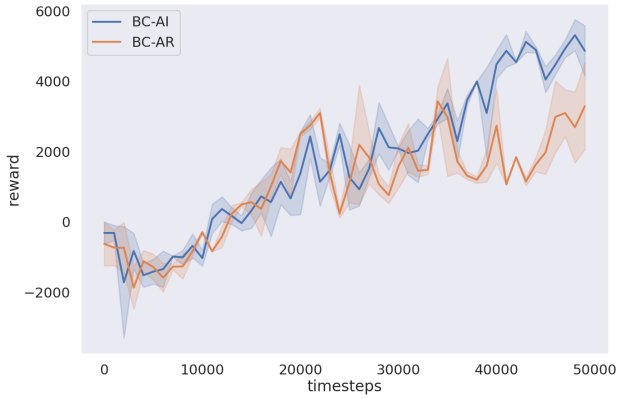


**図 4** BC のデータセットの違いによる性能の比較

半で性能に差が生じていることから, 学習が進むに連れて遷移モデルの性能が向上し, モデルから生成されたデータの信頼度が向上したことが原因だと考えられる. 本実験のタスクである HalfCheetah は行動の次元が少ないため, モデルの学習が進みやすいと考えられる. そのため, モデルの学習に時間を要する高次元タスクでは逆に, モデルからのデータを, この実験と同様に取り入れることが性能の向上につながるとは限らないということが示唆される. すなわちタスクの行動次元数に対応させて, モデルからのデータ取り込みに関して工夫を施す必要があると考えられる.

## 7. 今後の課題

本稿では, プランニングにニューラルネットワークを適切に組み込むことで比較的単純なタスクでは性能が向上することが確認できた. また提案手法の性能を確認するため比較的単純な, 一次関数的にパラメータを変化させることで, 報酬予測の信頼度と共にエリート分布を取り込む比率を変化させ, 実験を行なった. しかし, モデルの精度は必ずしも一次関数的に増加するわけではなく, Chua ら [10] によると HalfCheetah のタスクでは学習前半はモデルの精度が非常に速く向上し, 実環境とのずれが小さくなるが, 学習後半ではその精度は大きくは変化しない. そのため単に一次関数的に変化させるのではなく, モデルの精度を基に比率を変化させる必要がある. モデルの精度を測定するための指標を導入し, それを基に, 報酬予測の信頼度を見積もり, エリート分布を取り込む割合を変化させる必要があると考えられる. また本手法は POPLIN という既存手法を一部, 変化させて実験を行なっているため, 手法としての限界があることも考えられ, 提案手法の改善のみでは, より高次元な行動を要するタスクに対して, 性能が向上できない可能性もある. そのため, モデルベース強化学習, MPC という枠組みの中でより, ベースとしての性能が高い手法を検討する必要があると考えられる.

次に手法のアルゴリズムのみではなくデータセットにも工夫を施すことも考えられる. 行動空間の広いタスクでは, 学習前半ではモデルの性能が低いと考えられるため, モデルから生成されたデータをデータセットに取り込まず, 実環境から得られたデータを使い, 学習後半ではモデルからのデータも部分的に取り込むというような, データセットにも変化を加えることで, より方策の性能が向上するのではないかと考えられるため, この実装も行うことを考える.

本実験では MuJoCo という環境を用いた. これは連続行動であるが, Atari などのゲーム系のタスクの多くは離散行動である. そのため, 本稿で取り上げた手法や提案手法を離散行動を必要とするゲームなどへも適用することも考えられる.

## 参考文献

[1] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, volume 12, pages 1057–1063. MIT Press, 2000.

[2] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. In *Machine Learning*, pages 279–292, 1992.

[3] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In *SIAM Journal on Control and Optimization*, pages 1008–1014. MIT Press, 2000.

[4] Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 1 2005.

[5] Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2:160–163, 1991.

[6] Rémi Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In Paolo Ciancarini and H. Jaap van den Herik, editors, *5th International Conference on Computer and Games*, Turin, Italy, May 2006.

[7] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

[8] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815, 2017.

[9] Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *CoRR*, abs/1708.02596, 2017.

[10] 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.

[11] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *CoRR*, abs/1906.08253, 2019.

[12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.

[13] Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. *CoRR*, abs/1906.08649, 2019.

[14] Hang Lai, Jian Shen, Weinan Zhang, and Yong Yu. Bidirectional model-based policy optimization. *arXiv preprint arXiv:2007.01995*, 2020.

[15] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *CoRR*, abs/1907.02057, 2019.

[16] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017.

[17] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pages 5026–5033. IEEE, 2012.

 