# 麻雀のポリシー関数に適したネットワークモデルの構築と評価

清水大志[1,a)]　田中哲朗[2,b)]

**概要**：入力にゲーム固有の特徴量をほとんど用いずに自己対戦による強化学習のみで，AlphaGo Zero は囲碁のトッププレイヤを大きく超える強さを達成した．この成功を受け，他のゲームにおいてもゲーム固有の特徴量をなるべく入力に使わないニューラルネットを強化学習により学習させて強いプレイヤを作成する試みが行われている．強化学習を用いた自己対戦には大量の計算機を使った実験が必要になるが，本研究ではあるゲームにおいて強化学習をさせる前に，事前にそのゲームの性質を持つ小さいゲームを教師あり学習で学習させて，適したネットワークモデルを求める方法を提案する．小さいゲームに対する教師あり学習は短い時間で終了するため，ハイパーパラメータ自動最適化ツールを用いて様々なネットワークモデルの中から適したモデルを選択することが可能である．本研究では，麻雀のゲームとしての特徴を保持しつつ，理論的な最善手が求められるミニゲームを対象として教師あり学習により，ニューラルネットワークのモデルを評価した．提案したモデルは先行研究のモデルよりも高い正解率が得られた．高評価を得たモデルに対して強化学習を適用したが，得られた正解率は低かった．

# Building and evaluating neural networks for policy functions of mahjong

Taishi Shimizu[1,a)]　Tetsuro Tanaka[2,b)]

***Abstract:*** With the success of AlphaGo Zero, which achieved strength far exceeding the top players of Go by using only the reinforcement learning by self-training using almost no game-specific features for input, many people have been attempting to create strong players by learning a neural network that does not use game-specific features for input as much as possible. Reinforcement learning with self-training requires experiments using a large amount of computation. We propose a method for finding a suitable network model, which is learned by using a small game with the characteristics of the game with supervised learning before using reinforcement learning. Since supervised learning for a small game can be completed in a short time, we can select a suitable model from various network models using hyperparameter automatic optimization tools. In this study, we evaluated the neural network model by supervised learning for mini-games that require the best of the theory while retaining the characteristics of mahjong. We call this variation of games as mini-mahjong. The proposed model achieves higher accuracy than the models previously proposed. This highly evaluated model applied to reinforcement learning, but the accuracy was low.

## 1. はじめに

囲碁プログラム AlphaGo [1] は畳み込みニューラルネットワークとモンテカルロ木探索を組み合わせて作られたプログラムであり 2016 年 3 月にトッププロのイ・セドルとの 5 回戦を戦い 4 勝 1 敗で勝利した．その後継版である AlphaGo Zero[2] は人間の棋譜を用いずに自己対戦による強化学習のみで元の AlphaGo を大きく超える強さを達成したが，それに加えて入力にゲーム固有の特徴量をほとんど用いずに実現したという特徴を持つ．

AlphaGo Zero の成功を受け，他のゲームでもゲーム固

[1] 東京大学大学院総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo
[2] 東京大学情報基盤センター
Information Technology Center, The University of Tokyo
[a)] shimizu-taishi650@g.ecc.u-tokyo.ac.jp
[b)] ktanaka@tanaka.ecc.u-tokyo.ac.jp

© 2019 Information Processing Society of Japan

有の特徴量をなるべく入力に使わないニューラルネットによるポリシー関数や価値関数の学習が試みられている．麻雀に関しては，築地ら [3] や Gao ら [4] による先行研究があるが，どのような入力を与えるのが適切か，またどのようなネットワークモデルが適しているかについては，十分な検討がおこなわれていない．その原因として，先行研究の多くが人間の牌譜を用いた教師あり学習を使っているために，多くのエラー含む人間の牌譜を用いることによる教師データの質の確保が難しい点，学習に必要な局面数を増やすのが難しい点が考えられる．

AlphaGo Zore のような自己対戦による強化学習で，この問題が解決できる可能性はある．しかし，強化学習を用いた自己対戦には大量の計算機を使った実験が必要になる．そのため，あるゲームにおいて強化学習をさせる前に，事前にそのゲームの性質を持つ小さいゲームを教師あり学習で学習させて，適したネットワークモデルを求める方法を提案する．

小さいゲームに対する教師あり学習は短い時間で終了する．そのため optuna[5] などのハイパーパラメータ自動最適化ツールを用いることによって，幅広いネットワークモデルの中から通常は人の手で最適化されるハイパーパラメータを含めて適したモデルを選択することが可能となる．

その例として本研究では，麻雀のゲームとしての特徴を保持しつつ，理論的な最善手が求められるミニゲーム (以下ではミニ麻雀と呼ぶ) を対象として教師あり学習によりニューラルネットワークのモデルを評価した．また，そこで高評価を得たモデルに対しては強化学習を適用した．

以下，第 2 節ではミニ麻雀のルールについて，第 3 節では先行研究について，第 4 節では実際に行った研究のうち教師あり学習について，第 5 節では実際に行った研究のうち強化学習について，第 6 節ではまとめと今後の課題について述べる．

## 2. ミニ麻雀のルール

本研究で用いたミニ麻雀は以下のような点で麻雀を簡略化したゲームである．

- 使用する牌の種類は萬子のみの 9 種類とする．同種の牌の枚数は 4 枚とする．
- 手牌の牌数は 8 枚とし，雀頭 1 つと面子を 2 つ構成することであがりとする．
- 1 人でプレイする．したがって，ツモあがりしか存在しない．
- ポン，チー，カンなどの鳴きは存在しない．
- ツモ牌は手持ち以外の牌から等確率で引く．例えば手牌に一萬が 3 枚, 二萬が 2 枚, 三萬が 0 枚ある時，二萬を引く確率は一萬の倍であり，は更にその倍になる．
- ツモの回数に制限はなく，あがるまでゲームを続ける[*1]．
- プレイの目的は割引された累積報酬和の期待値を最大化することとする．
- 役は導入するが，タンヤオ，一盃口，対々和，チャンタのみである．
- あがり時の役の個数を $v$，基本点を $c$ として，あがった時の得点は $2^v c$ とする (ただしタンヤオとチャンタ，一盃口と対々和のように複合しない役があるため，得点の最大値は $4c$ である)．

割引された累積報酬和は強化学習で用いられるものであるが，このゲームは報酬はあがった時の得点 $r$ のみの episodic task に対応する．スタートから $n$ 手であがった時の累積報酬和 $R$ は割引率を $\gamma$ として，

$$R = \gamma^n r$$

となる．本研究では割引率 $\gamma$ は 0.9 とし，あがり点の基本点は 1 とする ($c = 1$)．

一般に episodic task の時には割引率を 1 に設定することもあるが，高い役であがることと早くあがることという相反する目的を達成することは麻雀の重要な特徴であり，割引率を 1 とした時には前者のみを達成することが目的となるため，割引率は 1 未満に設定している．

一方で，このミニ麻雀は以下のような麻雀の要素を持っていない．

- 他のプレイヤからのロンあがり，他のプレイヤへの振り込みを考慮した打牌選択
- ポン，チー，カンなどの鳴きを考慮した打牌選択
- 他のプレイヤーとの得点差，順位を考慮した手作り

他のプレイヤの打牌や鳴きをどのような入力として表現するか，これらをうまく扱うネットワークモデルはどのようなものかということは非常に興味深い問題であるが，本研究では扱わない．本研究では手牌の入力表現および，手牌から役も考慮して有効な打牌を選択するのに必要なネットワークモデルを検討するという段階にとどまっている．

## 3. 先行研究

水上ら [6] は教師あり学習と強化学習を組み合わせて麻雀プログラムを作成する手法を提案した．この手法は，はじめに人間の牌譜から教師あり学習によりコンピュータプレイヤを構築し，このプレイヤ同士を対局させて生成された牌譜を用いて，手牌から和了の翻数を予測モデルを機械学習により構築し，このモデルの出力と期待最終順位を用いて点数状況を考慮する麻雀プログラムを作成するものである．このプログラムでは入力として水上らが選択した麻雀固有の特徴量も使用している．

築地ら [3] は，麻雀の特徴を自動抽出して捨て牌を自動

[*1] 後半の強化学習の実験では実験の都合上このツモの回数に制限を設けている

© 2019 Information Processing Society of Japan

選択する学習を行うことを目的として，麻雀用の学習用CNN構成を提案した．これは4つのサイズの異なるフィルタを用いた二次元畳み込み層，プーリング層，全結合層からなるネットワークである．二次元畳み込み層のフィルタ枚数は32，プーリング層では最大プーリングを行い，全結合層の深さは1で，活性化関数にはReLUを用いる．全結合層では選定率が0.5であるドロップアウトも行なっている．学習データとして「東風荘」[*2]の牌譜40000局面を用い，テストデータとの一致率が53.93%になった．この研究では面子に関する特徴を画像的に表現できる手牌の入力方法も提案している．この入力は34行5列，要素は0 or 1の二値の行列 $M$ で手牌を表現するもので，種類 $i$ の牌が $j$ 枚あった時，$M_{i,j} = 1$ でそれ以外は全て0とするというデータ構造になる．本研究ではこの手牌の表現方法を「手牌モード1」と名付ける．このモデルの出力は各牌34種に対応した，捨て牌となる確率である．この表現方法をミニ麻雀に適用した時の入力は9行5列になる．手牌11244999の入力の表現を図1に示す．

| | 0枚 | 1枚 | 2枚 | 3枚 | 4枚 |
|---|---|---|---|---|---|
| 1m | [0, | 0, | 1, | 0, | 0] |
| 2m | [0, | 1, | 0, | 0, | 0] |
| 3m | [1, | 0, | 0, | 0, | 0] |
| 4m | [0, | 0, | 1, | 0, | 0] |
| 5m | [1, | 0, | 0, | 0, | 0] |
| 6m | [1, | 0, | 0, | 0, | 0] |
| 7m | [1, | 0, | 0, | 0, | 0] |
| 8m | [1, | 0, | 0, | 0, | 0] |
| 9m | [0, | 0, | 0, | 1, | 0] |

**図 1** 手牌モード1の例

gaoら[4]は深層畳み込みニューラルネットワークを用いて不完全情報ゲームである麻雀の教師あり学習を行った．手牌から捨てる牌を出力するモデルにおいては，「二次元畳み込み層，Batch Normalization，ドロップアウト」というネットワークを三つ重ねて，最後に全結合層を入れるモデルを使用している．二次元畳み込み層において，カーネルのサイズは $5 \times 2$ で，フィルターの枚数は100，ドロップアウトの選定率は0.5，全結合層のユニット数は35で活性化関数にはsoftmax関数を用いている．学習データとして「天鳳」[*3]の牌譜を用いており，accuracyとして68.8%を達成している．またこの研究でもニューラルネットの構造の提案だけでなく，入力データである手牌の表現方法も紹介もしている．この入力は34行4列，要素は0 or 1の二値の行列 $M$ で手牌を表現するもので，種類 $i$ の牌が $k$ 枚あった時，$j < k$ について，$M_{i,j} = 1$ でそれ以外は全て0とするというデータ構造になる．本研究ではこの手牌の表現方法を「手牌モード2」と名付ける．このモデルの出力としては各牌34種に対応した，捨て牌となる確率である．この表現方法をミニ麻雀に適用した時の入力は9行4列になる．手牌11244999の入力の表現を図2に示す．

| | 1枚 | 2枚 | 3枚 | 4枚 |
|---|---|---|---|---|
| 1m | [1, | 1, | 0, | 0] |
| 2m | [1, | 0, | 0, | 0] |
| 3m | [0, | 0, | 0, | 0] |
| 4m | [1, | 1, | 0, | 0] |
| 5m | [0, | 0, | 0, | 0] |
| 6m | [0, | 0, | 0, | 0] |
| 7m | [0, | 0, | 0, | 0] |
| 8m | [0, | 0, | 0, | 0] |
| 9m | [1, | 1, | 1, | 0] |

**図 2** 手牌モード2の例

## 4. 教師あり学習

ミニ麻雀においては，現在の巡目や捨牌の記録は最善手を求めるにあたって不要である．手牌から1枚捨てた時の手牌の組み合わせから，累積報酬和の期待値を求めることができる．上記の条件での7枚の組み合わせは6,030通りあり[*4]，[7]において「Value Iteration」として紹介されているアルゴリズムによりその期待値を求めた．

ここで得られた結果について少し考察する．役がある状態とない状態でもっとも状態価値に差があった手牌は2233344(6677788)である．これは役がある状態ではもっとも価値が高い手牌の一つであり，2から4のどの手牌においても役が二つ確定する[*5]．役がある時とない時で最善手が変わる手牌は11385個のうち6570個ある．その価値の差がもっとも大きい手牌は11222334で，役なしにおける最善手は3で価値が0.768であるが，役ありにおける最

[*2] http://mj.giganet.net
[*3] http://tenhou.net
[*4] 1から9までを9から1までに置き換える対称性を利用して組み合わせ数を減らすことが可能だが，この数字はそれを考慮していない
[*5] 2か4であがるとタンヤオと対々和が，3であがるとタンヤオと一盃口が確定する．

© 2019 Information Processing Society of Japan

善手は 4 で 2.703 となっている．

このミニ麻雀で，最善の捨牌を求めるポリシー関数となるニューラルネットワークについていくつか評価した．麻雀を行うプログラムにはポリシー関数を学習するもの，価値関数を学習するもの，その両方を学習するものなど存在するが，ここでポリシー関数のみを対象としたのは，ポリシー関数の方が理論上の最適プレイヤにどのくらい近づいたかが評価しやすいためである．あがっていない状態の 8 枚の手牌の組み合わせは 10,389 通りあり，牌を捨てた後の累積報酬和の期待値が最大となる手を 1(複数ある時は最大となる種類数で 1 を割った値)，それ以外を 0 とするような教師データを作成して教師あり学習を行う．

ここで，入力手牌の表現方法には複数考えられ，その表現方法によって学習速度に差が出ると考えられる．本研究では以下の 3 種類の表現方法を検討した．

1 つ目は築地らの提案した入力をミニ麻雀に適用したもので，9 行 5 列，要素は 0 or 1 の二値の行列 $M$ で手牌を表現するもので，種類 $i$ の牌が $j$ 枚あった時，$M_{i,j} = 1$ でそれ以外は全て 0 とする．図 1 に，例として手牌が 11244999 であった時の表し方を示した．先ほどにも述べたように，これは「手牌モード 1」と名付ける．

2 つ目は gao らの提案した入力をミニ麻雀に適用したもので，9 行 4 列，要素は 0 or 1 の二値の行列 $M$ で手牌を表現するもので，種類 $i$ の牌が $k$ 枚あった時，$j < k$ について，$M_{i,j} = 1$ でそれ以外は全て 0 とする．図 2 に，例として手牌が 11244999 であった時の表し方を示した．先ほどにも述べたように，これは「手牌モード 2」と名付ける．

3 つ目は 8 行 9 列，要素は 0 or 1 の二値の行列 $M$ で手牌を表現するもので，手牌を種類によってソートした後で，$i$ 枚目の牌の種類が $j$ の時，$M_{i,j} = 1$ でそれ以外は全て 0 とする．図 3 に，例として手牌が 11244999 であった時の表し方を示す．これを「手牌モード 3」と名付ける．

| | 1m | 2m | 3m | | 8m | 9m |
|---|---|---|---|---|---|---|
| 1枚目 | [1, | 0, | 0, | … | 0, | 0] |
| 2枚目 | [1, | 0, | 0, | … | 0, | 0] |
| 3枚目 | [0, | 1, | 0, | … | 0, | 0] |
| 4枚目 | [0, | 0, | 0, | … | 0, | 0] |
| 5枚目 | [0, | 0, | 0, | … | 0, | 0] |
| 6枚目 | [0, | 0, | 0, | … | 0, | 1] |
| 7枚目 | [0, | 0, | 0, | … | 0, | 1] |
| 8枚目 | [0, | 0, | 0, | … | 0, | 1] |

**図 3** 手牌モード 3 の例

この節における実験の設定は以下の通りである．

- 10,389 通りのうち，75 % を訓練データとし，25 % をテストデータとして用いる．
- 訓練データを一回ずつ学習させるのを 1 エポックとし， エポック数は 1000 回とする．
- loss は categorical cross entropy とする．

### 提案するモデルにおける教師あり学習

ポリシー関数に適切なネットワーク構造を見つけるため，以下の 3 つの構造を考える．

(1) 多層 MLP
(2) conv1d + Dense
(3) conv2d + Dense

これらのネットワークは先行研究を元に作ったものである．

### 多層 MLP

「多層 MLP」について，optuna を用いて最適なモデルを求めた．中間層の数は 1 から 3，それぞれのユニットの数は 50 から 1000 の範囲から取るものとし，optuna における trial の回数は 200 回とした．この条件においてもっとも良い値が得られたネットワークの構造は以下である．

- 中間層が 2 層の 4 層 MLP で，はじめにデータを 1 次元配列に直す．
- 中間層 1 層目のユニットの数は 302 である．
- 中間層 2 層目のユニットの数は 125 である．
- 出力層のユニットの数は action の数，すなわち 9 である．
- 出力層の活性化関数は softmax 関数で，それ以外では ReLU を用いる．

### conv1d + Dense

次に「conv1d + Dense」について，optuna を用いて最適なモデルを求めた．1 次元畳み込み層の数は 1 から 4，それぞれのフィルタの数は 8 から 64，全結合層の数は 1 か 2，それぞれのフィルタの数は 50 から 1000 の範囲から取るものとし，optuna における trial の回数は 200 回とした．この条件においてもっとも良い値が得られたネットワークの構造は以下である．

- はじめの 2 層は 1 次元畳み込み層とし，データを 1 次元配列に変更してから全結合層を 1 層入れる．
- 1 次元畳み込み層における kernal のサイズは 3, padding は SAME とし，活性化関数には ReLU を用いる．
- 1 次元畳み込み層におけるフィルタの数は，はじめの層からそれぞれ 56，64 である．
- 全結合層の 1 層目のユニットの数は 711 である．
- 全結合層の出力層のユニットの数は action の数，すなわち 9 である．

© 2019 Information Processing Society of Japan

- 全結合層の出力層の活性化関数は softmax 関数で，それ以外では ReLU を用いる．

**conv2d + Dense**

次に「conv2d + Dense」について，optuna を用いて最適なモデルを求めた．2 次元畳み込み層の数は 1 から 4，それぞれのフィルタの数は 8 から 64，全結合層の数は 1 か 2，それぞれのフィルタの数は 50 から 1000 の範囲から取るものとし，optuna における trial の回数は 200 回とした．この条件においてもっとも良い値が得られたネットワークの構造は以下である．

- はじめの 2 層は 2 次元畳み込み層とし，データを 1 次元配列に変更してから全結合層を 2 層入れる．
- 2 次元畳み込み層における kernel のサイズは 3, padding は SAME，活性化関数には ReLU を用いる．
- 2 次元畳み込み層におけるフィルタの数は，はじめの層からそれぞれ 64，43 である．
- 全結合層の 1 層目のユニットの数は 925 である．
- 全結合層の 2 層目のユニットの数は 492 である．
- 全結合層の出力層のユニットの数は action の数，すなわち 9 である．
- 全結合層の出力層の活性化関数は softmax 関数で，それ以外では ReLU を用いる．

次にベースラインとして，築地らの実験 [3] と Gao らの実験 [4] を参考にしたモデルを作成した．

築地らの論文の実験では，入力として手牌モード 2 を使用した手牌を入れ，出力としては各牌 (論文と同じ 34 種) に対応した捨て牌となる確率を出すモデルを使用した．現在用いている牌は 9 種類だが，論文では 34 種類すべての牌を用いており，原論文と全く同じ形を採用している．なお，論文では kernel のサイズとして $9 \times 5$ よりも大きいものも使用していたが，今回の実験では萬子以外の牌を使用していないことから，二次元畳み込み層における kernel のサイズは $3 \times 5$ と $9 \times 5$ のみを使用している．フィルタも原論文では 4 種類用いているが，3 種類は複数の牌種を使用していることが前提のフィルタであるため，1 種類のフィルタしか用いていない．

Gao らの論文の実験では，入力として手牌モード 3(ただし行数は論文と同じ 34) を使用した手牌を入れ，出力としては各牌 (論文と同じ 34 種) に対応した捨て牌となる確率を出すモデルを使用した．なお，batch normalization を行うテンソルの軸を適切と思われるものに修正 (通常 batch normalization は channel の軸に沿って行う) し，出力の次元も牌の種類数の 34 種に修正している．また原論文においては手牌から捨てる牌を出力するモデルだけでなく，リーチを判定するモデルや鳴きについて判定するモデルも作成しているが，ミニ麻雀においてはどちらも考慮しないためこれらのモデルは作成していない．

はじめに学習が進んでいるかを確認するために，役がない単純な状態 ($r = 1$) で教師あり学習を行い，正解率を調べた．ここでの「正解」とは，出力された確率がもっとも高い牌がその手牌における最善手に含まれていることであり，あがっていない全ての手牌において「正解」の割合を調べたものが「正解率」である．結果を表 1 に示す．表中の x は原論文との入力の形が違うために行なっていない実験である．

**表 1** 各手牌モード，モデルにおける正解率 (教師あり学習，役なし)

| 名前 | 手牌モード 1 | 手牌モード 2 | 手牌モード 3 |
|---|---|---|---|
| 多層 MLP | 0.848 | 0.847 | 0.793 |
| conv1d + Dense | 0.855 | 0.847 | 0.822 |
| conv2d + Dense | 0.846 | 0.853 | 0.828 |
| 築地ら | 0.807 | x | x |
| cao ら | x | 0.828 | x |

手牌モード 1 や手牌モード 2 において，先行研究よりも約 2 から 4 ポイント高い正解率が出た．これは optuna による自動最適化ツールを用いたことにより，先行研究のモデルよりも教師あり学習に適したモデルを用いることができたからであると考えられる．また手牌モード 3 は他の手牌モードに比べて値が悪くなっている．これは手牌モード 1 や 2 の方が，手牌を画像として表現しやすくなっていることが原因として考えられる．

次に役がある状態（$r = 2^v$，$v$ は成立した役の数）で Value Iteration を行い，理論値を出した上でその状態の価値を学習できているかを調べた．正解率の結果を表 2 に示す．

**表 2** 各手牌モード，モデルにおける正解率 (教師あり学習，役あり)

| 名前 | 手牌モード 1 | 手牌モード 2 | 手牌モード 3 |
|---|---|---|---|
| 多層 MLP | 0.854 | 0.888 | 0.881 |
| conv1d + Dense | 0.861 | 0.869 | 0.886 |
| conv2d + Dense | 0.873 | 0.889 | 0.889 |

これも先ほどと同じように手牌モード 1 の正解率が低くなっていることが読み取れる．

## 5. 提案するモデルを用いた強化学習

先ほどと対応するネットワーク構造を使って強化学習により価値関数を学習させた．使用したモジュールは OpenAI の gym[8] と baselines[9] である．

強化学習のアルゴリズムとしては DQN[10] を用いている．詳細な設定は以下である．

- 手牌から 1 枚牌を捨てて，1 枚引くのを 1 ステップとする．
- 切れない牌を切ろうとした時は-1 の報酬とし，その時

© 2019 Information Processing Society of Japan

の手牌は変わらない．これも 1 ステップとする．

- 学習は全部で $3 \times 10^4$ ステップとする[*6]．
- 手牌が配られてからあがるまでを 1 エピソードとする．ただし 100 ステップ経ってもあがれない時はそこでエピソードを終了する．
- 最初の時点であがっていた場合は，報酬を何も与えずに引き直しとする．
- あがったら $2^v c$ の報酬を与える (今回は $c = 1$)．
- 割引率 $\gamma$ は 0.9 を用いる．
- その他のハイパーパラメータは openAI baselines のデフォルト値に従った．

はじめに学習がうまく進んでいるかを確認するために，あがり役を導入せずにあがったら一律で報酬 $r = 1$ を与えるという条件で，Value Iteration で求めた値を正解としてその正解率を求めた．その結果は表 3 に示す．

**表 3** 各手牌モード，モデルにおける正解率 (強化学習，役なし)

| 名前 | 手牌モード 1 | 手牌モード 2 | 手牌モード 3 |
|---|---|---|---|
| 多層 MLP | 0.301 | 0.198 | 0.272 |
| conv1d + Dense | 0.316 | 0.212 | 0.219 |
| conv2d + Dense | 0.345 | 0.265 | 0.201 |

次に役あり（$r = 2^v$，$v$ は成立した役の数）として，残りは同じ条件で正解率を出した．結果は表 4 に示す．

**表 4** 各手牌モード，モデルにおける正解率 (強化学習，役あり)

| 名前 | 手牌モード 1 | 手牌モード 2 | 手牌モード 3 |
|---|---|---|---|
| 多層 MLP | 0.163 | 0.207 | 0.256 |
| conv1d + Dense | 0.229 | 0.198 | 0.316 |
| conv2d + Dense | 0.178 | 0.227 | 0.247 |

どの設定においても 0.2 から 0.3 程度の正解率で，学習があまりうまくいっていないことがわかる．ここで学習がどのようになっているかを調べるため，学習時の 100 エピソードごとの平均の報酬をプロットした．どの手牌モードにおいても同じような状況だったので，手牌モード 3 のときのみを示す．役なし時は図 4 に，役あり時は図 5 に表す．

グラフ上では正の値の報酬を得られていることがわかる．教師あり学習のような高い正解率を得るためには実験のステップ数が足りていないのは明らかであり，今後時間をかけて実験を行う予定である．

## 6. まとめ

本研究では理論的な最善手が求められるミニ麻雀を対象として教師あり学習と強化学習を行った．入力としては 3 つの手牌の表現方法を用い，モデルのネットワークの構成

[*6] 教師あり学習の結果から考えて $10^6 \sim 10^7$ ステップ数ほど必要であるが，時間の都合上このステップ数となっている

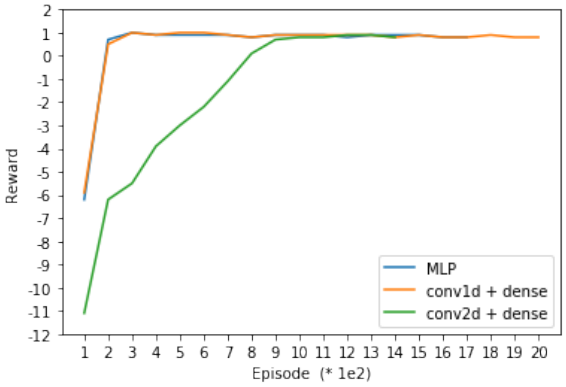


**図 4** 役なし時のエピソードごとの平均獲得報酬

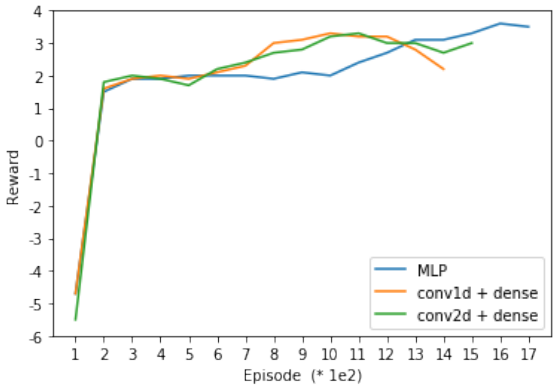


**図 5** 役あり時のエピソードごとの平均獲得報酬

には optuna という自動最適化ツールを用いた．提案したモデルにおいては先行研究よりも 2 から 4 ポイント高い正解率を得た．次に，このモデルを価値関数として強化学習を適用し学習させたが，計算時間の関係もあり，うまく学習できなかった．

今回はミニ麻雀の中でも簡単な 8 枚麻雀のみで実験を行ったが，牌の種類や手牌の枚数を増やしたり，役をさらに増やした時に関しても同様の実験をおこなう予定である．今後は状態数が増える結果として理論的な最善手を求めるのが難しい場合には，探索を用いて近似値を求め，その値を教師データとすることも必要になると考えられる．

なおこの研究で用いたコードは github(`https://github.com/minnsou/gpw2019`) で公開している．

**謝辞** 本研究は JSPS 科研費 18K11600 の助成を受けておこなわれた．

### 参考文献

[1] Silver, David, et al.: Mastering the game of go without human knowledge, Nature, 550, 7676, pp.354 (2017)
[2] Silver, David, et al.: Mastering the game of Go with deep neural networks and tree search, Nature, 529, 7587, pp.484 (2016)
[3] 築地毅, 柴原一友: CNN 麻雀-麻雀向け CNN 構成の有効性, ゲームプログラミングワークショップ 2017 論文集, 2017, pp.163-170 (2017)

© 2019 Information Processing Society of Japan

[4] Gao, Shiqi, et al.: Supervised Learning of Imperfect Information Data in the Game of Mahjong via Deep Convolutional Neural Networks, Information Processing Society of Japan (2018), (2018)
[5] AKIBA, Takuya, et al.: Optuna: A Next-generation Hyperparameter Optimization Framework, In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, 2019, p.2623-2631 (2019)
[6] 水上直紀, 鶴岡慶雅.: 自動対戦棋譜の教師あり学習による翻数予測に基づく麻雀プレイヤ, 情報処理学会論文誌, 2019, pp.1325-1336 (2019)
[7] Sutton, Richard S., Andrew G. Barto.: Reinforcement learning: An introduction, MIT press (2018)
[8] BROCKMAN, Greg, et al.: Openai gym, arXiv preprint arXiv:1606.01540 (2016)
[9] DHARIWAL, Prafulla, et al.: Openai baselines, URL https://github.com/opfenai/baselines (2017)
[10] MNIH, Volodymyr, et al.: Playing atari with deep reinforcement learning, arXiv preprint arXiv:1312.5602 (2013)

© 2019 Information Processing Society of Japan 