令和 3 年度　卒業研究

カードゲームでプレイヤーの腕前に合わせて手加減してくれる AI の研究

-プレイヤーの腕前を判定する AI プログラムの開発-

Research on AI that can adjust to the player's skill level in card games

--- Development of an AI program to judge the skill of player’s ---

函館工業高等専門学校

生産システム工学科　情報コース

25 番　平井　聖哉

指導教員　東海林　智也

目次

第１章　序論

第１節　英文アブストラクト

In recent years, the number of players of competitive card games has been decreasing. I believe that the cause of this problem is the lack of familiar opponents. Therefore, I started this research because I thought that by having artificial intelligence play the role of the opponent, and by having virtual opponents play against each other, people can learn the fun of competitive card games. As the first step of this research, I developed a program to judge the skill of the player.

Key words: AI, Tensorflow, Keras

第 2 節　研究背景

近年、競技用カードゲームのプレイヤー人口が減少傾向にある。この問題には様々な理由が考えられる。その中でも対戦相手の有無という問題に焦点を当てた。

競技用カードゲームはそもそも対戦相手がいて初めて成立するルールである以上、友人がいないなどの理由で、対戦相手となる人間がいないとゲーム自体を遊ぶことができない。これでは楽しさを理解することができない。Zoom などのアプリとカメラを用いた対戦方法もあるが、Web カメラを買う、パソコンを用意するなどといった環境構築に手間を要してしまう。

また既存のプレイヤーも減少しており、同等レベルの対戦相手が身近にいないなど初心者と同様の問題が発生している。

この問題はディープラーニングを用いて、対戦相手用の人工知能(以下、AI とする)を開発することで解決できる可能性がある。

第 3 節　研究目的

この対戦相手用 AI は初心者と既存のプレイヤーのどちらも使用できるものである必要がある。つまり対戦相手の腕前を判定し、その腕前によってプレイの仕方を変えなければならない。

よって、相手の腕前を予測し必要に応じてある程度手加減をする対戦相手用 AI を構築する必要がある。

そこで当研究では、その最初の段階としてプレイヤーの腕前を判定するアプリケーションを AI を用いて開発する。

取り上げるカードゲームについては、誰でもルールがわかりやすく楽しめ、簡単で認知度も高いという理由からトランプのポーカーを使用する。

第 4 節　開発環境

開発環境は以下の通りである。

使用言語　　：Python3.7

ライブラリ　　：Tensorflow [1]
　　Keras [2]

開発環境　　：Jupyter Notebook6.4.6

本研究ではウェブ上で無料公開無料公開されている AI ライブラリである「Tensorflow[1]」とその Tensorflow 上で動作するライブラリである「Keras[2]」を使用した。

Tensolflow は機械学習のライブラリであり、ニューラルネットワークの構築とその訓練ができることが特徴として挙げられる。

Keras はニューラルネットワークライブラリの一つであり、他のライブラリよりもニューラルネットワークを簡易なコードで容易に構築できることが特徴である。

第 2 章　アプリケーションの開発

第 1 節　概要

本研究では、Tensorflow と Keras を用いて、手札と捨て札を入力すると腕前を判定するアプリケーションを開発する。具体的には、ランダムに配られた手札とそこからプレイヤーが選んだ捨て札を未知データとして入力し、腕前を判定させる。

第 2 節　アプリケーション詳細

本研究で開発する腕前判定アプリケーションは「プレイ用のポーカープログラム」と「腕前判定用の AI プログラム」の 2 つのプログラムから構成されている。

まず、「プレイ用のポーカープログラム」を作成した[3]。このプログラムは、カード群と山札を生成し、山札の中からランダムに 5 つのカードを取り出して手札としてプレイヤーに提示する。次にプレイヤーは手札の中から捨てる札を選択する。その後、交換前の手札と捨て札を数値化し、交換後の手札の手役の判定を行って点数を計算する。このプログラムを実行すると図 1 のようになる。

次に、「腕前判定用の AI プログラム」を作成した。このプログラムはニューラルネットワークへの入力データとして数値化された交換前の手札と捨て札のデータ、ラベルとして腕前の強弱を使って学習し、学習後に未知データを手札と捨て札を入力して腕前の判別を行うものとなっている。

なお入力データとラベルの数値化アルゴリズムについては次節で説明する。

```
Poker Game start
player's hands：[♦-6, ♠-Q, ♥-J, ♥-10, ♥-3]
カードを変えたいならyを押してEnter,変えたくないなら何もせずにEnter
0
♦-6：y
1
♠-Q：y
2
♥-J：
3
♥-10：
4
♥-3：i
[306, 112, 211, 210, 203, 1, 1, 0, 0, 0]
player's hands：[♠-3, ♦-4, ♥-J, ♥-10, ♥-3]
{'is_flash': False, 'is_straight': False, 'same_number_count': 0, 'same_number': 0, 'match_pair_count': 1}
True
Qでゲーム終了、それ以外でゲームスタート：Q
```

図 1　ポーカープログラム実行例

第 3 節　数値化のアルゴリズム

交換前手札の数値化は次の通りで行った。まずマークに数値(スペードが 100、ハートが 200、ダイヤが 300、クローバーが 400)を割り当て、マークの数値にカードの数字を加算した値を札の数値とした。捨て札の数値は、捨てたら 1、捨てなかったら 0 とした。

またラベルとして使用する腕前の強弱は、交換後の手札が交換前の手札と比較したとき、いかに役を揃えやすい捨て方をしているか及び点数が高くなる可能性が高いか低いかを経験則で判定し、強ければ 1、弱ければ 0 の one - bot ベクトル形式とした。

第 4 節　MLP(多層パーセプトロン)の仕様

腕前判定用 AI プログラムの AI として、3 層全結合型フィードフォワードニューラルネットワーク[4]を使用した。入力層が交換前の手札 5 枚と捨て札 5 枚を使用しているため計 10 個、中間層には双曲線活性化関数を利用したものを 100 個、出力層には softmax 活性化関数を利用したものを 2 個(強い・弱い)とした。

第 3 章　実験

第 1 節　学習データセットによる学習結果

学習データセット(図 2)を 14 個、学習率を 0.001、エポック数を 500 回、バッチサイズを 2 で学習させた様子を図 3 に示す。結果として正解率が 50%から 100%に変化した。

手札					捨て札					腕前	
										強い	弱い
101	110	111	112	113	0	0	0	0	0	1	0
202	203	204	205	408	0	0	0	0	1	1	0
109	207	307	312	407	1	0	0	1	0	1	0
102	203	213	302	303	0	0	1	0	0	1	0
202	304	305	308	405	1	1	0	1	0	1	0
108	406	407	408	410	0	1	0	1	1	0	1
106	303	304	306	406	1	0	0	0	1	0	1
204	206	303	304	306	1	1	0	0	0	0	1
109	113	209	311	412	0	0	1	0	0	0	1
110	206	303	302	313	1	1	1	1	1	0	1
111	205	308	406	407	1	0	0	0	0	1	0
104	112	403	405	206	0	1	0	0	0	1	0
102	111	407	303	212	1	1	1	1	1	0	1
401	402	202	208	213	1	0	0	1	1	0	1

図 2　学習データセット

```
学習前
学習データセット : 損失=1.78, 正解率=0.50
学習データの予測値
[[ 0.967  0.033]
 [ 0.929  0.071]
 [ 0.929  0.071]
 [ 0.964  0.036]
 [ 0.971  0.029]
 [ 0.949  0.051]
 [ 0.972  0.028]
 [ 0.987  0.013]
 [ 0.972  0.028]
 [ 0.944  0.056]
 [ 0.959  0.041]
 [ 0.947  0.053]
 [ 0.948  0.052]
 [ 0.987  0.013]]


学習後
学習データセット : 損失=0.02, 正解率=1.00
学習データの予測値
[[ 0.984  0.016]
 [ 0.999  0.001]
 [ 0.988  0.012]
 [ 0.987  0.013]
 [ 0.973  0.027]
 [ 0.002  0.998]
 [ 0.019  0.981]
 [ 0.024  0.976]
 [ 0.021  0.979]
 [ 0.032  0.968]
 [ 0.957  0.043]
 [ 0.994  0.006]
 [ 0.006  0.994]
 [ 0.006  0.994]]
```

図3　学習結果

第 2 節　テストデータを用いた実験

20 個のテストデータセットを腕前判定プログラムに入力し、強いか弱いかを判別させる。得られた結果が想定する腕前と同等になれば腕前判別プログラムは正しいデータを算出したと言える。

結果を図 4 に示す。実験結果から腕前の判定には成功していると言える。しかし、強いとも弱いともどちらともとれない判定が甘いデータがいくつか算出された。これは腕前判定のラベルを強いと弱いの 2 つしか設定していなかったためだと推測できる。

手札					捨札					強さ		AI	
										強い	弱い	強い	弱い
411	304	308	403	204	1	0	1	1	0	0	1	0.344	0.656
112	309	109	302	212	1	0	0	1	1	1	0	0.688	0.312
404	313	203	304	401	1	0	0	1	0	1	0	0.846	0.154
103	106	113	302	309	1	1	0	1	1	1	0	0.446	0.554
106	405	401	307	210	0	0	1	0	1	0	1	0.000	1.000
402	106	109	407	313	1	1	1	1	1	0	1	0.040	0.960
202	402	401	111	112	0	0	1	1	1	0	1	0.206	0.794
403	109	304	308	312	1	1	0	0	0	0	1	0.970	0.030
303	112	304	403	213	0	1	1	0	1	1	0	0.939	0.061
307	404	308	411	402	1	0	1	0	0	1	0	0.953	0.047
307	113	308	403	402	1	0	1	1	1	1	0	0.988	0.012
101	411	203	103	313	0	0	1	1	0	0	1	0.012	0.988
304	103	402	410	211	0	0	0	1	1	1	0	0.997	0.003
204	411	406	112	409	1	0	0	1	0	0	1	0.007	0.993
306	106	305	109	101	1	1	0	0	1	0	1	0.115	0.885
112	202	213	403	302	0	1	0	0	1	1	0	0.968	0.032
212	202	108	112	205	1	1	0	0	1	1	0	0.596	0.404
307	212	303	404	405	0	1	0	0	0	0	1	0.320	0.680
311	302	211	306	207	0	0	1	0	1	1	0	0.950	0.050
304	301	111	112	401	1	0	1	1	0	0	1	0.076	0.924
									平均	0.5	0.5	0.5231	0.47695

図 4　実験結果

第 4 章　まとめ

本研究ではプレイヤーのカードゲームの腕前を判定するアプリケーションを開発した。今後の改善点として細かい数値の調整や学習データの増量、ラベルの要素の再設定などが挙げられる。とくに学習データに関しては現在の学習データセットの数が 14 個と少ないため、数を増やすだけでも十分良い結果が得られると考えられる。またラベルに関しても、強い・弱いだけでなく段階を分けたり、大穴狙いや安定志向といった、個々の人間の考え方にも考慮するように設定をするべきだと思われる。また腕前の判定の仕方も、交換後の手札や自分の手札も参照するなど改良の余地がみられる。

今後は本研究を更に進め、最終的に相手の腕前を予測し必要に応じてある程度手加減をする対戦相手用 AI を構築することを目指す。

参考文献

[1] Tensorflow ホームページ：https://www.tensorflow.orgMLP　(最終参考 2022 年 2 月)

[2] Keras ホームページ： https://keras.io/ja/　(最終参考 2022 年 2 月)

[3] Pyhon でドラクエのポーカーを実装してみた：
https://quita.com/p-t-a-p-1/items/784d97de3a6cab49a70c　(最終参考 2022 年 2 月)

[4] フィードフォワードニューラルネットワーク wikipedia ：
http://en.wikipedia.org/wiki/Feedforward_neural_network　(最終参考 2022 年 2 月)