AlphaZero 班　活動報告書

立命館コンピュータクラブ
2020 年度プロジェクト活動

2021 年 2 月 8 日

深田 紘希[*1]阿部 竜也[*2]堀田 隆成[*3]
田邉 雄士[*4]原 佑馬[*5]海原 義樹[*6]
高山 紗世梨[*7]青木 雅典[*8]玄元 奏[*9]

*1　情報理工学部 情報理工学科 実世界情報コース 二回生
*2　情報理工学部 情報理工学科 実世界情報コース 二回生
*3　情報理工学部 情報理工学科 セキュリティ・ネットワークコース 二回生
*4　情報理工学部 情報理工学科 システムアーキテクトコース 二回生
*5　情報理工学部 情報理工学科 システムアーキテクトコース 三回生
*6　情報理工学部 情報理工学科 システムアーキテクトコース 三回生
*7　情報理工学部 情報理工学科 システムアーキテクトコース 三回生
*8　理工学部 電子情報工学科 四回生
*9　情報理工学部 情報理工学科 実世界情報コース 四回生

目次

1　はじめに

文責: 深田 紘希

2015 年に AlphaGo という囲碁 AI が囲碁チャンピオンに勝利したことで世界を驚かせた．さらに 2017 年には自己学習のみで AlphaGo に勝利するプログラムである AlphaGoZero が制作された．同年，AlphaGoZero を汎用化させて他のボードゲームにも応用可能なように改良された AlphaZero が制作されることになった．AlphaZero は様々なゲームに応用することができ，棋譜などの教師データを必要とせず，また簡単ではないが比較的シンプルなコード構成をしているため，基本的な強化学習を勉強した後に入門しやすいと想定しプロジェクトを設立した．

2　活動概要

文責: 深田 紘希

本班はプロジェクトリーダーが事前に指定した参考書 [1] を参考にしながら毎週決められた担当者が担当範囲を解説し，その後班員の間で議論するという，輪講の形式で進めた．コロナ禍の影響で実際に集まることが難しいため，Zoom を用いて活動を進めることとした．その際には動画を撮影し，見返すことができるようにした．

表 1　各週の活動内容

週	活動内容
第一週	顔合わせ，環境構築
第二週	パーセプトロンとニューラルネットワーク
第三週	画像認識
第四週	多腕バンディット問題，方策勾配法
第五週	Sarsa，Q 学習，DQN
第六週	min-max 法，アルファベータ法
第七週	原始モンテカルロ探索，モンテカルロ木探索

3　活動内容

本活動で取り扱った内容について以下に記述する．

3.1　環境構築

文責: 深田 紘希

今回用いた環境は Google Colaboratory である．Google Colaboratory とは Google が提供するサービスのひとつで，オンライン上で Python プログラミングとコードの実行ができる．導入理由としてはローカルで Python の環境構築する必要がないことが挙げられる．また高性能な GPU を無料で使用することができることも大きな利点である．さらに複数の有用なライブラリが既にインストールされていることも円滑な進行に寄与した．

しかし，メモリやストレージ，1 ノートブックのサイズが制限されているほか，何も操作せずに 90 分が経過したりインスタンスが起動してから 12 時間が経過するとリセットされるため，使用する際はこれらに注意する必要がある．

ローカルランタイムの接続をすることで時間制限を無くすことが可能であるが，本プロジェクトではホスト型ランタイムで GoogleDrive から接続するかたちをとった．

3.2　パーセプトロンとニューラルネットワーク

文責: 高山 紗世梨

パーセプトロンはニューロンの一種で，複数の入力に対して一つの信号を出力する．

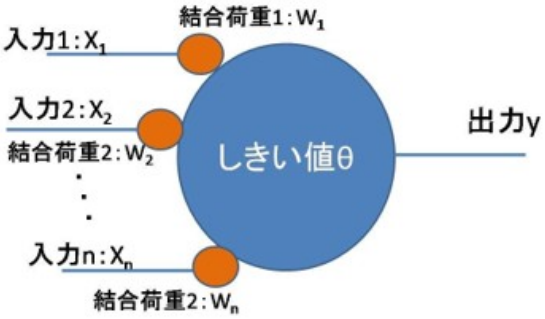

図 1　図式化したパーセプトロン

複数の入力値 x に信号の重要度を表す総合荷重 w を掛ける．総和が閾値を超えるか判断し，その結果によって出力値 y が変わる．ニューロンのみでは複雑な問題に対処することはできない．そのため，ニューロンを並べ層を作り，その層を積み重ねることで，複雑な問題に対応できるニューラルネットワークを作成する．ニューラルネットワークを作成するだけでは正しい結果を得ること

ができない．学習データを入力した時の予測値と実際の答えの差から重みパラメータとバイアスを更新していくことで，適切に分類をすることができるようになる．

3.3 画像認識

文責: 阿部 竜也

画像認識セクションでは，畳み込みニューラルネットワーク [2]（CNN：Convolutional Neural Network）を用いて手書き文字の識別を行った．実装には TensorFlow という深層学習ライブラリを使用し，MNIST データセットを利用する．実装及び検証は TensorFlow 公式チュートリアル [3] に沿って行った．

3.3.1 TensorFlow のインポート

まずは自分の環境に TensorFlow をインストールする．

```
pip install tensorflow
```

その後，Google Colabratory に下記プログラムを追加していく．

```
from tensorflow.keras import datasets, layers, models
```

3.3.2 MNIST データセットのダウンロードと前処理

データセットのダウンロードは，load_data() を呼び出すことで簡単に行える．ただし，このままでは各ピクセル値が 0~255 と学習には不向きであるため，0~1 に正規化する．

```
(train_images, train_labels), (test_images, test_labels) =
      datasets.mnist.load_data()

train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))

# ピクセルの値を 0~1 の間に正規化
train_images, test_images = train_images / 255.0, test_images / 255.0
```

3.3.3 CNN を作成

複雑な畳み込み演算やプーリング処理などは，TensorFlow を使えば簡単に呼び出すことができる．

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28,
    28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.summary()
```

summary() では作成したモデルのアーキテクチャを出力している．下記が今回扱う CNN のアーキテクチャである．

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 64)                36928
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
```

3.3.4 モデルのコンパイルと学習

最後に学習ステップへと移る．今回は最適化関数を Adam，損失関数を交差エントロピーとする．この設定でコンパイルしたモデルを，先ほど用意した学習用の MNIST データセットを用い

て，epoch 数 5 で学習する．

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(train_images, train_labels, epochs=5)
```

事前に用意した検証用データを用いて精度を測定すると，99%以上の精度を達成していることがわかる．

3.4 多腕バンディット問題

文責: 海原 義樹

「多腕バンディット」とは，アームが複数あるスロットマシーンの一種であり，「多腕バンディット問題」は，そのゲームをプレイしたときに得られる報酬を最大化する方法を問う問題である．この問題は，エージェントはアームを選んでそれを引くという一つの行動しかしないためシンプルに強化学習について考えることができる．

この問題は，アームの選択を行う際の方針として情報収集のために行う「探索」と収集した情報を利用して行う「利用」のどちらかを選んで計算する．この二つのバランスとるために，ε -greedy アルゴリズムと UCB1 アルゴリズムの二つが主に知られている．ε -greedy アルゴリズムは一定の確率でランダムに「探索」と「利用」を選択する．UCB1 アルゴリズムは成功率と試行回数を基に組み合わせる．UCB1 アルゴリズムは試行回数が少ないと「探索」を選び，試行回数が増えてくると「利用」に偏るようになっており，

$$\frac{\omega}{n} + (\frac{2log(t)}{2})^{\frac{1}{2}}$$

の式で表す（ここでωはそのアームを選んだ時の成功数，n はそのアームを選んだ回数，t は試行回数である）．これらを用いて，ゲームをシミュレーションし，アームごとの期待値を計算することで多腕バンディット問題を解くことができる．

3.5 方策勾配法

文責: 原 佑馬

方策反復法とは，方策に従って行動し，成功時の行動を重要と考えてその行動を多く取り入れるように方策を更新するような手法であり，方策勾配法はこれを利用したアルゴリズムの 1 つである．

ここで，迷路ゲームを例に上げて考えてみる．迷路ゲームにおいては，ゴールまで辿り着くことを目的，現在の位置が状態，上下左右に移動する事を行動とし，ゴールした行動を重視するようにする．1 エピソードはゴールするまでとし，パラメータの更新は 1 エピソード毎とする．

まず，方策に変換される値パラメータ θ を状態数×行動数の配列で用意し，それを方策に変換する．方策とは，ある状態においてある行動をとる確率を表す．また，パラメータ θ を方策に変換するためにはソフトマックス関数を用いる方法がある．

次に，方策に従って行動をゴールするまで，つまり 1 エピソード分繰り返し，成功した行動を多く取り入れるようにパラメータ θ を更新し，方策の変化量が閾値以下になるまでこれを繰り返す．

以上のようにして，より良い結果が得られる方策を導き出す手法が，方策勾配法である．

3.6 Sarsa，Q 学習

文責: 深田 紘希

方策は方策反復法の他に価値反復法で求めることができる．方策反復法が方策を直接更新したのと異なり，価値反復法では価値を更新する．

価値とは状態と方策を固定したときの条件付きの収益のことである．収益は割引報酬和として計算する．具体的にはステップ数が増えるほど重みを小さくしてそれをすべて足し合わせる．

さらに方策反復法はスタートからゴールまで一通りエージェントに行動させてからパラメータを更新していたが，価値反復法は行動するたびにパラメータを更新するということが特徴である．差分計算をするため，スタートからゴールまでのステップ数が多くても計算がしやすい．

方策を求めるときは方策勾配法という計算式を使ったが，収益を計算するひとつの方法として行動価値関数を用いる．

行動価値関数は Q 関数とも呼ばれ，引数に状態と行動をもつ．報酬はゴールするかしないかで 1 か 0 で与えられる．収益は時間で割り引かれるため，早くゴールしたほうが収益の増加を見込める．

行動価値関数の更新方法には Sarsa と Q 学習の 2 種類が存在する．Sarsa での更新には次ステップの行動を用いるため，ε -greedy のランダム性が存在することになる．対して Q 学習での更新は次ステップの最大価値の行動を使用するためランダム性が存在しない．Q 学習は Sarsa に比べて収束が早いが局所解に陥りやすいという特徴を持っているが，強化学習は膨大な状態数を持つ場合が多く収束の早さを重視した Q 学習が用いられることが多い．また歴史的にも Sarsa が先に考案され，その収束の遅さを改善したのが Q 学習の手法である．

3.7 DQN

文責: 深田 紘希

行動評価関数は引数に状態と行動の表形式で表現できる．しかし行動や状態が多くなると，その積である状態数が多くなりすぎてきちんと学習することが現実的に難しくなる．そこで行動評価関数を表形式からニューラルネットワークで表現しようと考案されたのが，Deep Q-Network（通称DQN）である．

DQN は引数に状態をとり，返り値に各々の行動確率をリストをとる．ニューラルネットワークを用いてある状態である行動をとる確率を推論する．

DQN は学習を進めやすくするために様々な工夫をしている．

まず時系列順での学習はデータ間の時系列の特徴に影響を受けてしまうため，メモリにデータを蓄積しておいてのちにランダム学習を行うように変更されている．また更新を計算するための行動価値関数をもう一つ別に用意している．さらに環境によるスケール変化をできるだけ吸収するために報酬のスケールを-1, 0, 1 に固定するほか，NN の誤差が大きくても安定している誤差関数である Huber 関数を用いている．

OpenAI Gym という強化学習用のツールキットを用いることで DQN を利用した深層強化学習を試すことができる．有名なものとして棒を倒さないようにバランスを取る CartPole の学習がある．

2015 年には DQN を用いてゲームのルールを全く教わらずにゲーム画像の画面のみで学習することに成功した．ブロック崩しに関しては人間を超えるほどの実力まで到達し，有名な技術となった．

3.8 min-max 法

文責: 深田 紘希

次の一手を決めるために数手先までゲーム木を展開して盤面を評価する必要があり，これを探索と呼ぶ．

min-max 法は自分は自分にとって最善手を選び，相手は自分にとって最悪手を選ぶと仮定して，最善手を探す探索アルゴリズムである．自局面のノードはその子ノードの状態評価値の最大値を状態価値にとり，相手局面のノードはその子ノードの状態評価値の最小値を状態価値にとる．するとルートノードの子ノードの中で評価値の高いノードが最善手となるのが min-max 法のアルゴリズムである．

この手法はゲーム木で展開されるすべての局面の評価が分かるので最強の手法であるが，複雑なゲームでは探索に時間がかかりすぎるため計算量を減らす工夫をする必要がある．その工夫については後述する．

3.9 アルファベータ法

文責: 深田 紘希

前述した min-max 法は計算に時間がかかりすぎてしまうため，それを改善した手法としてアルファベータ法がある．ミニマックス法の性質を生かすことで，不必要な探索を避けることができる．全探索するゲーム木の一部の探索を取りやめることからこれを枝刈りと呼び，α カットと β カットが存在する．

α カットとは，先手がわざわざ評価値の小さな手は打たないことを利用して，後手の行動をカットすることである．ゲーム木ではノードの評価がある値以下になると探索を打ち切ることで実現できる．

β カットとは，後手がわざわざ評価値の大きな手を打たないことを利用して，先手の行動をカットすることである．ゲーム木で見るとノードの評価がある値以下になると探索を打ち切ることで実現できる．ゲーム木ではあるノードの評価がある値以上になると探索を打ち切ることで実現できる．

3.10 原始モンテカルロ探索

文責: 田邉 雄士

前項で扱った min-max 法やアルファベータ法は理論上最強のアルゴリズムであるが，将棋やオセロでは計算量が膨大になってしまうため現実的手法ではない．原始モンテカルロ探索とは，上記の計算量の問題を解決すべく，全てを計算するのではなくランダムに 1 つ選択したノードにおいて，状態評価するものである．

3.11 モンテカルロ木探索

文責: 田邉 雄士

原始モンテカルロ探索ではその特性上，9 勝 1 敗の手があった場合相手に 1 敗の手を選ばれ続けると必ず負けてしまう．その問題を解決すべく原始モンテカルロで高い評価を受けた手を選び最善の手を選択する手法がモンテカルロ木探索である．モンテカルロ木探索の操作は以下の 4 つからなる．

ノードの選択

UCB1 アルゴリズムの値が一番大きいノードを，子ノードが無い状態まで選択し続ける操作を行う．

評価

選択したノードからランダムに手を選び, 勝敗をカウントして価値を算出する操作を行う.

展開

選択したノードの評価回数が任意の回数に到達したものを拡張し, 選択対象を増やす操作を行う．

更新

選択したノードを評価した後に, ノードを辿り戻り, 評価回数と評価結果を反映する作業を行う.

4 展望

文責: 深田 紘希

強化学習の基礎部分の勉強に留まったため，今後は AlphaZero 特有の部分を学習することを課題としたい.

この後の勉強方針としてデュアルネットワークの構築がある．これは CPU が CPU と勝ち残り戦をして，強い個体のみを残して学習する手法である．興味深い部分であるため，本報告書の活動内容を進めたのちに取り組むと理解がさらに深まると思われる.

根幹の学習部分を作成すれば，様々な二人零和有限確定完全情報ゲームに応用することができるのが AlphaZero の特徴である．個人で利用可能な計算資源であれば，五目並べやオセロ，どうぶつ将棋ほどの状態数を持つゲームが対象となる．更なる計算資源が存在するのであれば，チェスや将棋といったプロの存在するゲームを学習させるのもよいだろう.

さらには完全情報ゲームにとどまらずテキサス・ホールデムやぷよぷよといった不完全情報ゲームへの応用も研究されており，今後の発展が期待される.

5 運営上の問題点

文責: 深田 紘希

本プロジェクトでの顕著な問題点として，活動時間の不足が挙げられる．特に後期活動に関しては一度もしていないという結果となった．後期に活動しなかった理由を以下に記述することとする.

輪講形式で行うため進行の肝となる参考書には，コード誤りや数式誤りが大量に発見された．公式サイトに正誤表が掲示されているものの，そこで取り上げられていない誤りも十数個発見した．担当者の適切な割り振りや，最終的な議論をまとめるためにもプロジェクトリーダーは輪講範囲を理解しておく必要があったが，参考書にあるコードを生かしつつミスを修正していく作業が間に合わず，それによって当初の計画通りに進めることができなくなった結果，リーダーのモチベーション低下へとつながった.

プロジェクトを立ち上げたときのリーダーの見通しが甘く，さらにリーダー本人の実力が不足していたことが一因である．プロジェクトを立ち上げた時点では参考書の 1/3 ほどが理解できていた状態であり，1 年かけて班員とともに 2/3 の範囲を読み切ることを目標としていた．しかし，実際は後半になるにつれ参考書が誤りにより参考にならなくなる現状となった.

対策として，まず一番に参考書はすべて精査したのちにプロジェクトを立てるのが望ましい．それが叶わない場合は，複数の参考文献を用意することが適切である．

本プロジェクトの進行に類似した企画書が提出された場合，複数の根拠となる書籍や情報源を提示してプロジェクトを進めることができるように助言していくことが，多様なプロジェクト設立に貢献する行動であろうと思う．

6 おわりに

文責: 深田 紘希

AlphaZero で用いられている，強化学習の基礎となる部分を輪講の方式で勉強した．具体的には多腕バンディット問題から方策勾配法，DQN という強化学習の歴史に沿って勉強を進めた．さらに最善の手を選ぶための min-max 法，アルファベータ法，原始モンテカルロ探索，モンテカルロ木探索を学習し，今後ゲーム AI を制作していく上で必須となる知識をこの活動で得ることができた．

このプロジェクトを機に，より強化学習への理解を深めゲーム AI の作成に結びつくことを期待し，以上を報告書とする．

参考文献

[1] AlphaZero 深層学習・強化学習・探索： 人工知能プログラミング実践入門 / 布留川英一著
https://www.borndigital.co.jp/book/14383.html

[2] https://developers.google.com/machine-learning/glossary/?hl=ja#convolutional_neural_network

[3] https://www.tensorflow.org/tutorials/images/cnn?hl=ja