修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院　情報理工学研究科　情報・ネットワーク工学専攻　博士前期課程		
氏　　名	HUANG ZIHANG	学籍番号	1931166
論 文 題 目	深層学習を用いたシャンチー上級者の着手及び勝敗予測		

要　旨

ボードゲームの人工知能の意思決定を理解することは、人間にとってはしばしば困難である。初級者が訓練するときには、人工知能の着手ではなく上級者の着手を学んだ方が良い。

本研究で題材としたシャンチーは、チェス系の二人ゼロ和完全情報ゲームである。二人のプレイヤは、一方が赤駒、他方が黒駒を持つ。赤方が先手で、初期配置から、交互に二人のプレイヤが駒を動かし、パスはできない。駒の動き方は、駒種によって異なる。駒種は全部で 7 つである。

シャンチーは、中国において人気があり、様々なトーナメントが開催されている。ただし、筆者の知る限り、シャンチー上級者の着手及び勝敗を予測する深層学習の報告はない。

本研究の目的は、チェスに適用された AlphaZero の深層学習の、入力とする特徴やニューラルネットワーク(NN)の構造などをシャンチーに適用して、上級者の着手及び勝敗の予測精度を調べることにある。なお、シャンチーとチェスの予測精度を比較するために、本研究ではチェスの深層学習実験も行う。

本研究ではまず、上級者の棋譜の着手及び勝敗をラベルとした教師あり学習を行うために、シャンチーとチェス上級者の棋譜を収集した。シャンチーでは、訓練データ 10000 棋譜と、テストデータ 500 棋譜を用意した。そして、ルールの左右対称性を利用して、棋譜を二倍に水増しして、実質的な棋譜数は訓練データが 20000、テストデータが 1000 とした。シャンチーとの比較性のために、チェスの棋譜数も同様である。また、シャンチーとチェスの棋譜のいくつかの統計情報を比較し、ゲームの性質を調べた。

次に、収集した訓練データを用いて着手と勝敗を予測する NN を学習した。この NN は残差ブロック 9 個を含み、畳込み層は大きさ 3×3 のフィルタ 32 個からなる。実験の結果、シャンチー上級者の着手予測は、チェスとほぼ同じ精度でなされることが明らかとなった。また、シャンチーでは「砲」、チェスでは「ルーク」及び「クイーン」の移動を予測することが難しいとの知見も得られた。その一方で、シャンチー上級者の勝敗予測はチェスよりも良い精度でなされることも明らかとなった。これは、シャンチーがチェスよりも引分けになりやすく、引分けの予測が上級者の棋譜の結果とよく一致することに起因する。

電気通信大学　情報理工学研究科
情報・ネットワーク工学専攻　修士学位論文

深層学習を用いたシャンチー上級者の着手及び勝敗予測

2022 年 1 月 27 日

情報数理工学プログラム

学籍番号 1931166

HUANG ZIHANG

指導教員 保木邦仁
村松正和

目次

1　はじめに

ボードゲームは、ルールが明確でコンピューターによるシミュレーションが容易であり、意思決定の良し悪しの判断基準 (例えば勝敗) を提供するため、人工知能技術の性能評価に用いられることが多い。近年、ゲームをプレイする人工知能技術の発展が加速している。1997 年にチェスで DeepBlue がランキング 1 位のガルリ・カスパロフに勝利した。2017 年には囲碁で AlphaGo が中国棋士ランキング 1 位の柯潔九段に、将棋で Ponanza が佐藤天彦名人に勝利した[1]。

チェスやシャンチーは二人ゼロ和完全情報ゲームである。このようなゲームでは、「ゲーム木という形でゲームの状態遷移をすべて表現することが可能であり、この木を調べ尽くせば、両者が最善を尽くしたとき、「先手必勝か後手必勝か引分けになるか」が判明する。これはゲームの必勝法を見つけるということであり、必勝法を見つけることをゲームを解くと呼んでいる」[1]。しかし、ゲーム木複雑さがチェスは 10^{123}、シャンチーは 10^{150} と膨大であるため、ゲームを解くのは実際には困難である [2]。そのため、深層学習によりゲーム固有知識を獲得し、強い人工知能を作る研究が近年注目されている。

深層学習は、大規模な多層ニューラルネットワーク (NN) を用いた機械学習の手法である。NN とは、人間の脳の仕組みを模倣したネットワークと呼ばれる数理的なモデルで、事物の判断を行うものである。NN は、1943 年の文献 [3] に原形が見られ、今では深層学習などに使われている。近年、深層学習は、ゲームの着手確率や勝率推定においても注目されている。

これまで、ボードゲームを題材としたゲームプレイヤを強くする深層学習の成功事例がいくつか報告された。また、人間プレイヤの着手を深層学習で予測した研究もいくつか報告されている。しかし、筆者の知る限り、シャンチーにおいては、人間プレイヤの着手を深層学習で予測した事例は報告されていない。

深層学習の適用は着手予測だけではなく、ゲームの勝敗予測にも適用され始めている。しかし、人間プレイヤの勝敗を深層学習で予測した研究はほとんどなく、シャンチーの人間プレイヤの勝敗を深層学習で予測した事例も報告されていない。

ボードゲームの人工知能の意思決定を理解することは、人間にとってはしばしば困難である。初級者が訓練するときには、人工知能の着手ではなく上級者の着手を学んだ方が良い。

そこで本研究では、深層 NN と、シャンチー上級者の棋譜の着手及び勝敗をラベルとした教師あり学習とを組合せた手法を用いて、着手確率と勝率の推定精度を調査する。

本論文の構成は次のようである。2 章では、順伝播型ニューラルネットワークの基礎知識を紹介する。3 章では、ボードゲームの着手確率と勝率の推定に関する先行研究を紹介する。4 章では、シャンチーの用語とルールを説明する。5 章では、本研究の目的について述べる。6 章では、ラベルとした上級者の棋譜の収集と、収集した棋譜の統計情報について述べる。7 章では、本研究における着手及び勝敗の予測推定の手法について説明し、実験で得られた予測精度を比較した結果を述べる。8 章では、本研究を総括し、今後の展望を述べる。

[1] ワークデータ.com, URL:https://work-data.com/1418#_2017_vs_ponanza, 最終アクセス 2022

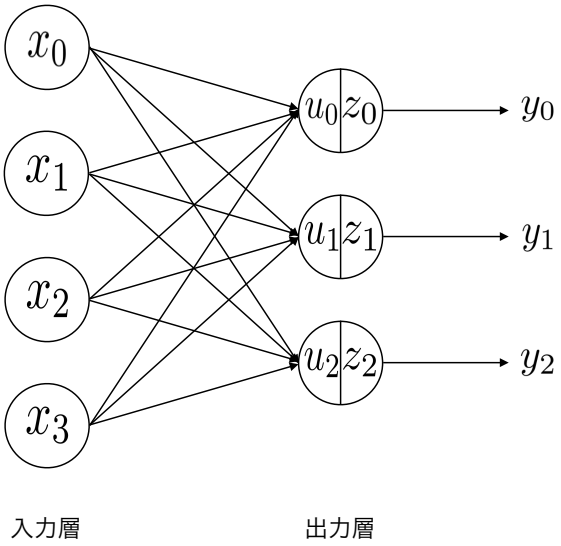


図 1　2 層に並べられたユニットを持つ NN

2　順伝播型ニューラルネットワーク

本章では、書籍 [4] の内容を抜粋して順伝播型 NN を説明する。

順伝播型 NN は、層状に並べたユニットが隣接層間でのみ結合した構造を持ち、情報が入力側から出力側に一方向にのみ伝播する NN である。図 1 に、全結合層 2 層からなる NN の一例を示す。入力層の 4 つのユニットはそれぞれ x_i $(i = 0, 1, 2, 3)$ を出力する。この出力はまた、出力層の入力となる。出力層の 3 つのユニットは z_j $(j = 0, 1, 2)$ を出力する。そして、x_i と z_j は次の条件

$$
\begin{aligned}
u_0 &= w_{0,0}x_0 + w_{0,1}x_1 + w_{0,2}x_2 + w_{0,3}x_3 + b_0 \\
u_1 &= w_{1,0}x_0 + w_{1,1}x_1 + w_{1,2}x_2 + w_{1,3}x_3 + b_1 \\
u_2 &= w_{2,0}x_0 + w_{2,1}x_1 + w_{2,2}x_2 + w_{2,3}x_3 + b_2
\end{aligned}
$$

$$
z_j = f(u_j) \quad (j = 0, 1, 2)
$$

を満たす。ここで、$w_{j,i}$ は重み、b_j はバイアスと呼ばれる NN のパラメータである。また、f は活性化関数と呼ばれる。

2.1 全結合層の順伝播の計算

層数 L からなる順伝播型 NN の計算の手順を説明する。各層を区別するために、NN のパラメータや出力の右肩に層の番号 ($l = 1, 2, \ldots, L$) を付ける。層 $l+1$ の出力 $\mathbf{z}^{(l+1)}$ は、1 つ下の層 l の出力 $\mathbf{z}^{(l)}$ から

$$\mathbf{u}^{(l+1)} = \mathbf{W}^{(l+1)}\mathbf{z}^{(l)} + \mathbf{b}^{(l+1)} \tag{1a}$$

$$\mathbf{z}^{(l+1)} = \mathbf{f}(\mathbf{u}^{(l+1)}) \tag{1b}$$

のように計算される。ただし、各ベクトルと行列は次のように定義する。

$$\mathbf{u} = \begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}, \mathbf{z} = \begin{bmatrix} z_0 \\ \vdots \\ z_{N-1} \end{bmatrix},$$

$$\mathbf{W} = \begin{bmatrix} w_{0,0} & \cdots & w_{0,N^{(l)}-1} \\ \vdots & \ddots & \vdots \\ w_{N^{(l+1)}-1,0} & \cdots & w_{N^{(l+1)}-1,N^{(l)}-1} \end{bmatrix},$$

$$\mathbf{b} = \begin{bmatrix} b_0 \\ \vdots \\ b_{N-1} \end{bmatrix}, \mathbf{f}(\mathbf{u}) = \begin{bmatrix} f(u_0) \\ \vdots \\ f(u_{N-1}) \end{bmatrix}$$

また、$N^{(l)}$ は層 l のユニットの数である。説明の便利ために一部分の層の番号を省略する。

以降では、NN の最終的な出力を $\mathbf{y} = \mathbf{z}^{(L)}$ と表記する。また、NN の最初の入力を $\mathbf{x} = \mathbf{z}^{(1)}$ と表記する。

このような順伝播型 NN では、与えられた最初の入力 $\mathbf{x}$ に対し、入力層側から出力層側へ、式 (1) の計算を繰り返すことで出力を伝播させ、最終的な出力 $\mathbf{y}$ を計算する。この計算は関数 $\mathbf{y} = \mathbf{y}(\mathbf{x})$ として表記できる。この関数を決定するものは、各層間の結合重み $\mathbf{W}^{(l)}$ ($l = 2, \ldots, L$) とバイアス $\mathbf{b}^{(l)}$ ($l = 2, \ldots, L$) である。これらを NN のパラメータと呼ぶ。以降、これらのパラメータ全てを成分に持つベクトル $\mathbf{w}$ を定義し、簡潔に $\mathbf{y}(\mathbf{x}; \mathbf{w})$ と書くことにする。

2.2 活性化関数

ユニットが持つ活性化関数には通常単調増加する非線形関数が用いられる。色々なものがあるが、古くからよく使われているのが、ロジスティック関数

$$f(u) = \frac{1}{1 + e^{-u}}$$

あるいは双曲線正接関数

$$f(u) = \tanh(u)$$

である。いずれの関数も、入力の絶対値が大きな値をとると出力が飽和し一定値となることと、その間の入力値に対して出力値が徐々に滑らかに変化することが特徴であり、一般にシグモイド関数と総称される。近年これらの活性化関数に代わり、正規化線形 (ReL, Rectified Linear) 関数

$$f(u) = \max(u, 0)$$

がよく使われている。これは、$z = u$ の線形関数のうち $u < 0$ の部分を $u = 0$ で置き換えただけの単純な関数である。単純で計算量が小さく、さらに上述の関数よりも学習がより速く進み、最終的にもより良い結果が得られることが多いため、現在よく使われる。この関数を持つユニット (Rectified Linear Unit) のことを ReLU と略記することがある。

2.3 学習の枠組み

上述のように、順伝播型 NN が表現する関数 $\mathbf{y}(\mathbf{x}; \mathbf{w})$ は、パラメータ $\mathbf{w}$ を変えると変化する。よい $\mathbf{w}$ を選ぶことで、この NN が望みの関数を表現するようにすることを考える。1 つの入力 $\mathbf{x}$ に対する望ましい出力 (ラベル) を $\mathbf{d}$ と書き、そのような入出力のペアが複数、

$$(\mathbf{x}_1, \mathbf{d}_1), (\mathbf{x}_2, \mathbf{d}_2), \ldots, (\mathbf{x}_{N^{\text{sample}}}, \mathbf{d}_{N^{\text{sample}}})$$

のように与えられているとする。なお、これらのペア $(\mathbf{x}_n, \mathbf{d}_n)$ 一つ一つを訓練サンプルと呼び、その集合を訓練データと呼ぶことにする。そして、どの $n \in \{1, \ldots, N^{\text{sample}}\}$ に対しても、入力 $\mathbf{x}_n$ を与えた時の NN の出力 $\mathbf{y}(\mathbf{x}_n, \mathbf{w})$ が、なるべく $\mathbf{d}_n$ に近くなるように $\mathbf{w}$ を調整する。そうすることを学習と呼ぶ。

NN が表現する関数と訓練データとの近さ $(\mathbf{y}(\mathbf{x}_n; \mathbf{w}) \approx \mathbf{d}_n)$ の尺度を考える。この尺度のことを、以下では誤差関数と呼ぶことにする。ここでは、回帰問題に適した誤差関数とクラス分類に適した誤差関数を示す。回帰とは、主に出力に連続値をとる関数を対象に、訓練データによく適合するような目標関数を定めることをいう。この場合、NN の出力層に、値域が目標関数のそれと類似する活性化関数を選びたい。例えば、目標関数の値域が $[-1:1]$ の場合、出力層の活性化関数には双曲線正接関数が適する。

このように出力層の活性化関数を選んだ上で、NN の出力 $\mathbf{y}(\mathbf{x}_n)$ が、訓練データの目標出力 $\mathbf{d}_n$ に可能な限り近くするようにすることを考える。それにはまず誤差関数を決める必要があり、二乗誤差

$$\|\mathbf{d}_n - \mathbf{y}(\mathbf{x}_n; \mathbf{w})\|^2$$

を使う場合を考える。二乗誤差を訓練データの全サンプルに渡り平均化したものは

$$E(\mathbf{w}) = \frac{1}{N^{\text{sample}}} \sum_{n=1}^{N^{\text{sample}}} \|\mathbf{y}(\mathbf{x}_n; \mathbf{w}) - \mathbf{d}_n\|^2$$

を考える。

クラス分類とは、入力 $\mathbf{x}$ を内容に応じて有限個のクラスに分類する問題である。このような多クラス分類を対象とする場合、NN の出力層に分類したいクラス数 K と同数のユニットを並べ、この層の $k(k = 0, \ldots, K-1)$ 番目のユニットの値にソフトマックス関数を作用させ、

$$y_k = z_k^{(L)} = \frac{\exp(u_k^{(L)})}{\sum_{j=0}^{K-1} \exp(u_j^{(L)})}$$

を出力層の k 番目の出力とする。こうして決まる出力 $y_0, \ldots, y_{K-1}$ の総和は 1 になる。なお、分類するクラスを $C_0, \ldots, C_{K-1}$ と表すとき、上のように選んだ出力層の k 番目のユニットの出力 y_k は、与えられた入力 $\mathbf{x}$ がクラス C_k に属する確率

$$p(C_k|\mathbf{x}) = y_k$$

を表すものと解釈する。そして、入力 $\mathbf{x}$ をこの確率が最大になるクラスに分類することにする。NN が実現する関数が各クラスの事後確率のモデルであると見なし、そのような確率モデルの下で、訓練データに対する NN のパラメータの尤度を評価し、これを最大化する。

今 n 番目の訓練サンプルとして、入力 $\mathbf{x}_n$ とその正解クラスの組が与えられたとする。このとき NN の目標出力を、2 値の値を K 個並べたベクトル $\mathbf{d}_n = [d_{n,0} \cdots d_{n,K-1}]^\top$ によって表現することにする。$\mathbf{d}_n$ の各成分は、対応するクラスが真のクラスであったときのみ 1 をとり、それ以外は 0 をとるように決める。このように符号化すると、事後確率は

$$p(\mathbf{d}_n|\mathbf{x}_n) = \prod_{k=0}^{K-1} p(C_k|\mathbf{x}_n)^{d_{n,k}}$$

と表せて、訓練データ $\{(\mathbf{x}_n, \mathbf{d}_n)\}(n = 1, \ldots, N)$ に対する $\mathbf{w}$ の尤度を

$$L(\mathbf{w}) = \prod_{n=1}^{N} p(\mathbf{d}_n|\mathbf{x}_n; \mathbf{w}) = \prod_{n=1}^{N} \prod_{k=0}^{K-1} (y_k(\mathbf{x}_n; \mathbf{w}))^{d_{n,k}}$$

のように導出できる。そして、この尤度の対数をとり符号を反転した次を、誤差関数とする。

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \sum_{k=0}^{K-1} d_{n,k} \log y_k(\mathbf{x}_n; \mathbf{w})$$

この関数は、交差エントロピー誤差関数と呼ばれる。

2.4 確率的勾配降下法

順伝播型 NN の学習は、訓練データ

$$D = \{(\mathbf{x}_1, \mathbf{d}_1), \ldots, (\mathbf{x}_{N^{\text{sample}}}, \mathbf{d}_{N^{\text{sample}}})\}$$

を元に計算される誤差関数 $E(\mathbf{w})$ を最小化することにより達成する。しかし、$E(\mathbf{w})$ は一般に凸関数ではなく、大域的な極小値を直接求めるのはほぼ不可能である。また、$E(\mathbf{w})$ の局所的な極小値は一般的に多数存在する。そのような極小値は、何らかの初期値を出発点に $\mathbf{w}$ を繰り返し更新する反復計算によって求める。そうする方法はいくつもあるが、本節では確率的勾配降下法 (SGD) を紹介する。

誤差関数 $E(\mathbf{w})$ は各訓練サンプル 1 個 $(\mathbf{x}, \mathbf{d})$ だけについて計算される誤差 $e(\mathbf{w}, \mathbf{x}, \mathbf{d})$ の和

$$E(\mathbf{w}) = \sum_{(\mathbf{x},\mathbf{d})\in D} e(\mathbf{w}, \mathbf{x}, \mathbf{d})$$

であると考える。SGD は訓練サンプル 1 つを使って $\mathbf{w}$ の更新を一回行う方法である。この方法では、$\mathbf{w}$ の更新は $e(\mathbf{w}, \mathbf{x}, \mathbf{d})$ の勾配

$$\nabla e(\mathbf{w}, \mathbf{x}, \mathbf{d}) = \frac{\partial e}{\partial \mathbf{w}}$$

を計算し、

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \epsilon \nabla e(\mathbf{w}, \mathbf{x}, \mathbf{d})$$

が成り立つように $\mathbf{w}$ を更新する。ここの ϵ は $\mathbf{w}$ の更新の大きさを定める定数で、学習率と呼ばれる。次の $\mathbf{w}^{(t+1)}$ の更新の際は、別の訓練サンプル $(\mathbf{x}', \mathbf{d}')$ を取り出し、これを使って同様に勾配 $\nabla e(\mathbf{w}, \mathbf{x}', \mathbf{d}')$ を評価し、$\mathbf{w}$ を更新する。このように訓練サンプルを毎回取り替え、$\mathbf{w}$ を更新していく。

SGD にはいくつかの長所がある。まず訓練データに冗長性がある場合、計算効率が向上し学習を速く実行できる。さらに反復計算が望まない局所的な極小解にトラップされてしまうリスクを低減できる。

2.5 ミニバッチと L2 正則化

規模が大きい NN の学習は大きな計算コストを要する。そのため、少数のサンプルからなる訓練データを構成し、その単位で $\mathbf{w}$ を更新する。このような訓練データはミニバッチと呼ばれる。

数式では次のように書く。まず t 回目の更新に用いるミニバッチを D_t と書くことにする。そして D_t

が含む全ての訓練サンプルに対する誤差

$$E_t(\mathbf{w}) = \frac{1}{N_t} \sum_{(\mathbf{x},\mathbf{d})\in D_t} e(\mathbf{w}, \mathbf{x}, \mathbf{d})$$

を計算し、$\mathbf{w}$ を更新する。$N_t = |D_t|$ はこのミニバッチが含む訓練サンプルの数である。なお、ミニバッチのサイズを変えたとき、それに合わせて学習率の調整が求められる。$E_t(\mathbf{w})$ を N_t で正規化すると、この煩わしさをある程度避けることができる。

深層学習の結果が訓練データに過適合し、テストデータでの予測精度が落ちる現象を過学習という。このような現象を防ぐために、L2 正則化という手法がある。深層学習は誤差関数を最小化するように重み $\mathbf{w}$ を更新する。一方で、L2 正則化を用いる深層学習は重み $\mathbf{w}$ の各要素の二乗値の和と誤差関数との和を最小化するように重み $\mathbf{w}$ を更新する。重み $\mathbf{w}$ の二乗和の項も誤差関数の一部分と考えると、誤差関数は

$$E_t(\mathbf{w}) = \frac{1}{N_t} \sum_{(\mathbf{x},\mathbf{d})\in D_t} \left\{ e(\mathbf{w}, \mathbf{x}, \mathbf{d}) + c\|\mathbf{w}\|^2 \right\}$$

になる。c は L2 正則化の割合を制御するハイパーパラメータである。

2.6 モメンタム項付き SGD

この節では、SGD の収束性能を向上させるモメンタム項付き SGD を紹介する。これは、重み $\mathbf{w}$ の修正量に、前回の修正量のいくばくかを加算する方法である。ミニバッチ D_{t-1} でなされた $\mathbf{w}$ の修正量を $\Delta\mathbf{w}^{(t)} = \mathbf{w}^{(t)} - \mathbf{w}^{(t-1)}$ と書くと、D_t を使った更新は式

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \epsilon\nabla E_t(\mathbf{w}^{(t)}) + \eta\Delta\mathbf{w}^{(t)} \tag{2}$$

を用いて行われる。$\nabla E_t(\mathbf{w})$ はミニバッチ D_t の誤差の勾配である。η は加算の割合を制御するハイパーパラメータである。

2.7 畳込み層

式 (1) では全結合層で使われる全結合の計算を示した。本節では、畳込み層で使われるストライドが 1 の畳込みの計算を説明する。

畳込み層の入力として多チャネルの画像を考え、画像の大きさが $H \times W$ で、チャネル数が K であるとする。画素をインデックス $(i, j, k)(i = 0, \ldots, H-1, j = 0, \ldots, W-1, k = 0, \ldots, K-1)$ で表すことにする。画素 (i, j, k) の値を $z_{i,j,k}$ と書く。

入力画像に M 種類のフィルタを適用する。フィルタは大きさ $H_\mathrm{F} \times W_\mathrm{F}$ が通常画像よりも小さく、入力画像と同じチャネル数 K を持つ。フィルタの数値一つ一つをインデックス $(p, q, k, m)(p = 0, \ldots, H_\mathrm{F}-1, q = 0, \ldots, W_\mathrm{F}-1, k = 0, \ldots, K-1, m = 0, \ldots, M-1)$ で区別し、(p, q, k, m) の値を $w_{p,q,k,m}$ と書く。

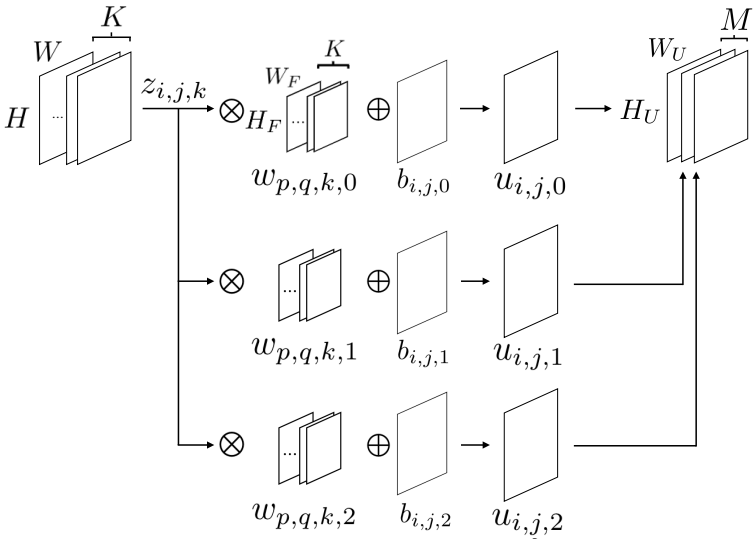


図 2　畳込みの計算。⊗ は畳込み、⊕ は画素ごとの和を表す。

入力画像 $z_{i,j,k}^{(l)}$ とフィルタ $w_{p,q,k,m}^{(l+1)}$ とバイアス $b_m^{(l+1)}$ の畳込みは次の積和計算を行う。

$$\hat{u}_{i,j,m}^{(l+1)} = \sum_{k=0}^{K-1} \sum_{p=0}^{H_F-1} \sum_{q=0}^{W_F-1} z_{i+p-i_0,j+q-j_0,k}^{(l)} w_{p,q,k,m}^{(l+1)} + b_m^{(l+1)} \tag{3}$$

ここで、i_0 と j_0 はパディングと呼ばれる値であり、画像の外の画素の値は 0 とする。パラメータの右肩の (l) は層の番号である。$w_{p,q,k,m}^{(l+1)}$ と $b_m^{(l+1)}$ も $\mathbf{w}$ の一部とみなす。以降、$H_F \times W_F$ はフィルタの大きさ、M はフィルタの数と表記することとする。

式 (3) のように、畳込みは入力画像にフィルタを重ねた時、画像とフィルタの重なり合う画素どうしの積を求めて、フィルタ全体で和を求めて、バイアスを加える計算を行う。したがって、入力画像内にフィルタ全体が収まる範囲内でフィルタを動かすと、$\hat{u}_{i,j,m}^{(l+1)} (i = 0, \ldots, H_\mathrm{U} - 1, j = 0, \ldots, W_\mathrm{U} - 1, m = 0, \ldots, M - 1)$ からなるチャネル数 M を持つ出力画像を得る。

出力画像の大きさは $H_U \times W_U$ で、チャネル数は M である。そして、畳み込み層の出力 $\hat{u}_{i,j,m}^{(l+1)}$ は式 (1) の全結合で使われたようなベクトルに並べ替える必要がある。対応関係は

$$u_{(H_U \times W_U)m + W_U i + j}^{(l)} = \hat{u}_{i,j,m}^{(l)}$$

のようになる。ここで、$u_j^{(l)}$ は並び替えた後のベクトルの j 番目の要素である。

式 (3) の計算で、$M = 3$ の場合を図 2 に示す。$m = 0, 1, 2$ の各フィルタ F_m が入力画像から画素値 $z_{i,j,k}$ を受け取り、畳込みを行った後、1 つのフィルタから 1 チャネルの画像を出力すると、フィルタの数

M と同数のチャネル数を持つ多チャネルの出力画像を得る。

先述の通り、畳込み NN でも順伝播 NN 同様、SGD によるパラメータ最適化を行う。畳込み層の重みはフィルタそのものなので、最適化の対象となるパラメータはフィルタとバイアスということになる。

2.8 バッチノーマライゼーション

深層学習の過程で、各階層の入力データの分布は、下位層のパラメータ $\mathbf{w}$ が更新されることにより変化する。層が深くになるほど分布の変化が拡大する。そのため、学習過程を通して継続的に各階層は新たな分布に適応することとなる。この現象は内部共変量シフトと呼ばれる。これに適応させるため、学習率を下げたり注意深く $\mathbf{w}$ を初期化したりすることにすると、深層学習が遅くなったり、初期値を変えて何度も学習し直さなくてはならなくなったりして、深層学習を行うことはしばしば困難な作業となる。これを回避するために各階層の入力の分布に対し平均と分散を一定に保つバッチノーマライゼーション (BN) という手法が近年注目を集めている [5]。

ミニバッチを使った SGD では、各訓練ステップがミニバッチを単位として実行される。訓練ステップ t 回目の大きさが N_t のミニバッチ

$$D_t = \{(\mathbf{x}_1, \mathbf{d}_1), \ldots, (\mathbf{x}_{N_t}, \mathbf{d}_{N_t})\}$$

でパラメータ w を更新することを考える。層 l の出力 $\mathbf{u}^{(l)}$ を BN で更新する方法は次のようである。

$$\mu_{t,j}^{(l)} = \frac{1}{N_t} \sum_{n=1}^{N_t} u_{n,j}^{(l)} \tag{4a}$$

$$\sigma^{2}{}_{t,j}^{(l)} = \frac{1}{N_t} \sum_{n=1}^{N_t} (u_{n,j}^{(l)} - \mu_{t,j}^{(l)})^2 \tag{4b}$$

$$u_{n,j}^{(l)} \leftarrow \gamma_j^{(l)} \frac{u_{n,j}^{(l)} - \mu_{t,j}^{(l)}}{\sqrt{\sigma^{2}{}_{t,j}^{(l)} + \epsilon}} + \beta_j^{(l)} \tag{4c}$$

ただし、$u_{n,j}^{(l)}$ は $\mathbf{x}_n$ を入力した NN の、活性化関数を作用する前の層 l の出力 $\mathbf{u}^{(l)}$ の j 番目の要素である。$\mu_{t,j}^{(l)}$ は、t 回目のミニバッチ D_t で計算した層 l の出力の、平均 $\boldsymbol{\mu}_t^{(l)}$ の j 番目の要素である。$\sigma^{2}{}_{t,j}^{(l)}$ は、t 回目のミニバッチ D_t で計算した層 l の出力の、分散 $\boldsymbol{\sigma}^{\mathbf{2}}{}_t^{(l)}$ の j 番目の要素である。また、記号 " $\leftarrow$ " は代入、ϵ は分母をゼロにしないよう導入される小さな正の実数である。$\gamma_j^{(l)}$ と $\beta_j^{(l)}$ も学習されるパラメータ $\mathbf{w}$ の一部分となる。

学習した NN を利用して性能をテストするとき、BN の式 (4) の平均と分散は、ミニバッチ D_t のみで計算されるのではなくて、複数のミニバッチに渡って計算される。

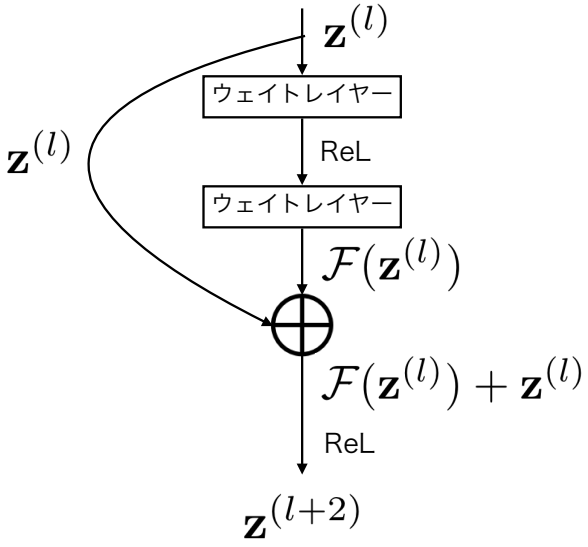


図 3　残差ブロック

2.9　残差 NN

勾配消失・爆発によって、NN は深ければ深いほど、訓練が難くなる。文献 [6] では、He らが深い NN の訓練を簡易にするために、残差学習を提案した。本節では、残差学習の計算法を紹介する。

残差学習は、複数の残差ブロックからなる残差 NN を訓練して達成される。残差ブロック 1 つは図 3 に示されるようなショートカットを持つ NN であり、計算を

$$\mathbf{z}^{(l+2)} = f(\mathbf{W}^{(l+2)} f(\mathbf{W}^{(l+1)}\mathbf{z}^{(l)} + \mathbf{b}^{(l+1)}) + \mathbf{b}^{(l+2)} + \mathbf{z}^{(l)})$$

が満たされるように行う。$\mathbf{z}^{(l)}$ と $\mathbf{z}^{(l+2)}$ は残差ブロックの入力と出力である。$\mathcal{F}(\mathbf{z}^{(l)})$ は $\mathbf{W}^{(l+2)} f(\mathbf{W}^{(l+1)}\mathbf{z}^{(l)} + \mathbf{b}^{(l+1)}) + \mathbf{b}^{(l+2)}$ である。活性化関数 f は ReL である。このようなショートカットを持つ NN は、勾配消失、爆発が起き難いことが知られている。

以上の紹介では、説明を簡易するために、残差ブロック “weight layer"は全結合層としたが、畳込み層としても良い。また、実用上、残差ブロックのウェイトレイヤーの直後にはよく BN が付けられる。

3　先行研究

本研究では、畳込み NN を用いた深層学習を利用して、シャンチーというチャス系ゲームでの上級者の着手及び勝敗を予測する。本章では、他の二人ゼロ和完全情報ゲームにおいて、類似した研究を行った文献を紹介する。

3.1 着手予測の深層学習

ボードゲームで人間プレイヤの着手をラベルとした教師あり学習を行い、着手を予測した研究の予測精度を表 1 にまとめる。ここで、予測精度とは、予測した着手と棋譜に記録された着手との一致率のことである。それぞれの文献で最も高い予測精度を記し、それを与えた NN の構造と題材としたゲームの平均合法手数も併せて示す。文献 [7] は、人間プレイヤではなく、既存人工知能の着手を予測したものであるが、参考のため表 1 に書く。

文献 [8] では、囲碁の着手予測を 20 個の BN 付きの残差ブロックを含む NN で行われた。残差ブロックの畳込み層は大きさ 3×3 のフィルタ 256 個からなる。活性化関数はすべて ReL である。用いられた訓練とテストで使うデータセットは KGS (Kiseido Go Server) のアマチュアの棋譜からなる。データセットの大きさは不明ではあるが、KGS の棋譜は毎年一万局以上のペースで増えていることから、訓練には十万棋譜以上があったことが推察される。

文献 [9] でチェスの着手予測精度 53% 程度を達成した NN は、6 個の Squeeze-and-Excitation (SE) 残差ブロックからなる。SE 残差ブロック [13] は SE ブロックを残差ブロック (図 3) に付けて、チャネル内の依存関係をモデル化して、価値の高いチャネルを強調することで残差ブロックが求めた特徴を調整し性能を向上させたものである。図 4 に SE 残差ブロックの概要を示す。Global pooling では、残差ブロックの結合直前のチャネル K 個それぞれに対し、数値 $H \times W$ 個の平均を取り数値 K 個を得る。これらを全結合層 A に入力し、数値 $\frac{K}{r}$ 個を得る。そしてこれらを全結合層 B に入力し、数値 K 個に復元する。ここで、r (文献 [13] では 16) は、SE ブロックの性能と計算コストのバランスを取るために導入したハイパーパラメータである。全結合層 B の出力は、sigmoid 関数を使って 0 から 1 までの範囲の数値に変換する。そして scale では、変換した数値 K 個をそれぞれに対応する、$\mathbf{z}^{(l+2)}$ のチャネルの数値 $H \times W$ 個にかけることにより特徴を調整する。使ったデータセットは Lichess というデータベースの 2016 年から 2019 年までの人間プレイヤの棋譜からなる。これらの棋譜から、1100 から 1900 までの Elo レイティング 9 段階に分けて分類し、訓練用 9 個、テスト用 9 個のデータセットを用意した。訓練用データセットは 1200 万枚の棋譜を含み、テスト用データセットは 20 万枚の棋譜を含む。

文献 [10] で五目並べの着手予測精度 60% 程度を達成した NN は、5 層と 7 層の BN 付きの畳込み層からなる。そして、畳込み層は大きさ 3×3 のフィルタ 128 個からなる。ただし、第 1 層のみのフィルタの大きさが 5×5 である。活性化関数はすべて ReL である。五目並べの平均合法手数 210 は文献 [2] の平均ゲーム長 30 から見積った。訓練とテストで用いたデータセットは合わせて約 66000 の棋譜からなる。

表 1　囲碁、チェス、五目並べ、オセロ、ヘックスでの深層学習の着手予測精度

ゲーム	合法手数	NN 構造	訓練データ棋譜数 (万)	予測精度 (%)
囲碁 (19×19) [8]	250 [2]	20 個の残差ブロック	–	60
チェス (8×8) [9]	35 [2]	6 個の SE 残差ブロック	1200	53
五目並べ (15×15) [10]	210 [2]	5、7 層の畳込み NN	6.6	42
オセロ (8×8) [11]	10 [2]	8 層の畳込み NN	9	63
ヘックス (13×13) [7]	152 [12]	7〜9 層の畳込み NN	1.35	55

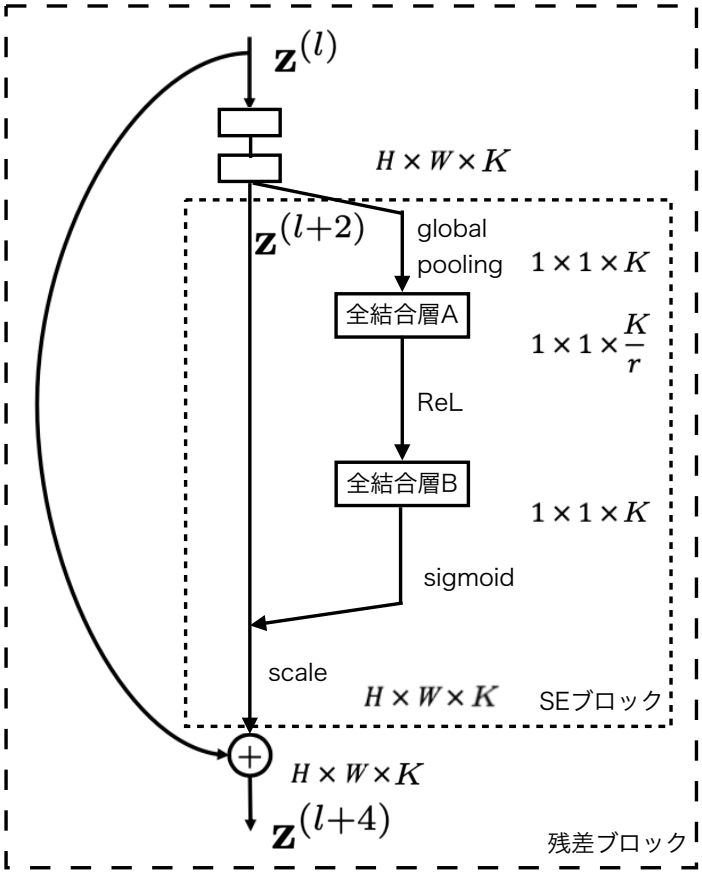


図 4　SE 残差ブロック

文献 [11] では、オセロの着手予測を 8 層の BN 付きの畳込みブロックを含む NN で行われた。全ての畳込み層のフィルタの大きさが 3×3 である。第 1、2 層の畳込み層は 64 個フィルタを持つ。第 3、4 層の畳込み層は 128 個フィルタを持つ。第 5 層から第 8 層までの畳込み層は 256 個フィルタを持つ。活性化関数はすべて ReL である。データセットは 119339 の上級者の棋譜からなる。このデータセットを分割し、75% が訓練で、25% がテストで使われた。

文献 [7] でヘックス (13×13) の着手予測精度 55% 程度を達成した NN は、7 層から 9 層までの畳込み層からなる。そして、畳込み層は大きさ 3×3 のフィルタ 128 個からなる。ただし、第 1 層のみフィルタの大きさが 5×5 である。活性化関数はすべて ReL である。ヘックス (13×13) の平均合法手数 152 は文献 [12] のヘックス (13×13) の平均ゲーム長 35 から見積った。データセットは MoHex 2.0 というヘックスプレイ人工知能の自己対局から生成した 15520 の棋譜からなる。このデータセットを分割し、90% が訓練で、10% がテストで使われた。

Silver らは 2017 年、AlphaGo Zero [8] という大規模な残差 NN を利用した囲碁の強化学習のアルゴリズを提案し、同じ NN の教師あり学習と着手予測の性能を比較した。結果として着手予測では、教師あり学習の方が高性能で、対局実験では、AlphaGo Zero の方が高性能であった。この結果は、プレイヤを強く

することと、人間プレイヤの着手を予測することは必ずしも同じではないということを示唆する。Silver らはまた 2018 年、AlphaZero [14] という強化学習のアルゴリズムを発表した。彼らが報告したチェス、将棋、囲碁での実験結果は、文献 [8] の教師あり深層学習と同程度の規模の深層 NN を利用して、既存の人工知能より強くなるということを示すものである。強化学習により得られた深層 NN は、人間プレイヤの着手と勝敗予測精度もある程度は高いことが伺われるが、これは同文献では示されていない。

文献 [15] は、シャンチーの強化学習のアルゴリズムを発表した。実験では、残差ブロック 7 個からなる NN とモンテカルロ木探索の手法を利用して、自己対局でシャンチーの固有知識を獲得することが示されていた。この文献でも人間プレイヤの着手と勝敗予測の精度は示されていない。

3.2 勝敗予測の深層学習

ボードゲームで人間プレイヤの勝敗を予測する研究がいくつか報告されている。これらの研究は、機械学習やデータマイニングなどの方法 [16] [17] で行われた。一方、深層学習を用いた研究例は少ない。

文献 [18] では、人間プレイヤの棋譜の勝敗を深層学習を用いて予測した。この研究では、棋譜を時系列データとして扱う教師あり学習を行い、チェスの時系列データから NN の入力を与える符号化方法二種 (Bitmap と Algebraic) それぞれ予測精度を報告し、AlphaZero と類似した Bitmap の方が高い一致率を達成することを示した。勝敗の予測は、ゲーム開始から 20、50、80 手目の時点で行った。ゲームプレイの長さが短くて、これらの時点がない棋譜では、棋譜に記録された最後の着手の時点で予測を行った。この研究ではゲームプレイが短い場合、どの着手が棋譜の最後の着手なのかを NN の入力から知ることができる。したがって、予測精度は、着手が最後のものであるという情報を NN に入力しない本研究の方法より高くなると考えられる。

AlphaGo Zero や AlphaZero は、人間プレイヤの勝敗を NN が予測した場合での精度を報告していない。それでも、強化学習で作る人工知能は高性能を示し、既存の人工知能より強くなったことから、教師あり学習の予測精度もある程度は高くなるのではないかと思われる。

3.3 AlphaZero のチェスの実装で用いられた NN の構造

AlphaZero は、囲碁で人間プレイヤの着手の良い一致率を達成した AlphaGo Zero と同等な性能を与え、囲碁以外のボードゲームにも適用された。このことから、AlphaZero の深層学習はボードゲームの着手と勝敗予測を効率的に行うと考えられる。ここでは、チェスで既存人工知能よりも高いパフォーマンスを強化学習により得た AlphaZero の NN の構造を紹介する。

まず、チェスのゲーム状況から AlphaZero の NN の入力 $\mathbf{x}$ を与える符号化方法を紹介する。入力 $\mathbf{x}$ は $H \times W \times [(P+R)T+L]$ の多チャネルの画像であると考える (表 2 参照)。画像の大きさ $H \times W$ は、チェス盤の大きさと対応する 8×8 である。チャネル数は $(P+R)T+L$ である。P はチェスの駒の種類数 12 と[2]、R は盤面の反復回数を表現するチャネル数 2 と、T は履歴の長さ 8 と、L は 7 と対応する[3]。

[2]手番プレイヤと相手プレイヤそれぞれ 6 種

[3]手番プレイヤを表すチャネル数が 1、手数が 1、手番プレイヤのキャスリング 2 種の合法性を表すチャネル数が 2、相手プレイヤのキャスリング 2 種の合法性を表すチャネル数が 2、最後に駒を取ったり、ポーンが動いたりしてから数えた手数を表すチャネ

表 2　AlphaZero の NN に入力するチェスのゲーム状況の特徴

特徴	チャネル数
手番プレイヤの駒の配置	6
相手プレイヤの駒の配置	6
盤面の反復回数	2
手番プレイヤ	1
手数	1
手番プレイヤのキャスリング 2 種の合法性	2
相手プレイヤのキャスリング 2 種の合法性	2
最後に駒を取ったり、ポーンが動いたりしてから数えた手数	1

次に、NN の構造を紹介する (図 5 参照)。AlphaZero が使う NN の構造は文献 [8] での構造とおおよそ同じであり、三つの部分 (“body”、“policy head”、“value head”) からなる。まずは NN の body 部を紹介する。この部分は、BN 付きの畳込み層 1 つと残差ブロック 19 個からなる。ここで、全ての畳込み層は大きさ 3×3 のフィルタ 256 個からなる。活性化関数は ReL である。Policy head 部は、二層の畳込み層からなる。第一層は BN 付きの畳込み層である。活性化関数は ReL である。第二層の畳込み層はフィルタ 73 個からなる。出力に対して、softmax 関数を利用して、ベクトル $\mathbf{P}$ を得る。$\mathbf{P}$ は、入力 $\mathbf{x}$ に対する着手の確率分布の推定である。Value head 部は BN 付きの畳込み層と二層の全結合層からなる。畳込み層は、大きさ 1×1、ストライド 1 のフィルタ 1 個からなる。活性化関数は ReL である。第二層の全結合層は、ユニット 256 個を持つ。活性化関数は ReL である。第三層の全結合層は、ユニット 1 個を持つ。出力に対して、tanh 関数を利用して、-1 から 1 までの v を得る。v は、引分けを 0.5 ポイントとした入力 $\mathbf{x}$ の勝率と対応する。

次に、着手の確率分布 $\mathbf{P}$ の表現法を紹介する。チェスの着手を 2 つの部分から記述する。1 つは、動く駒を記述する部分、もう 1 つは、その駒の移動先を記述する部分である。そして $\mathbf{P}$ は、大きさ 8×8 のチャネル 73 個の画像として表現される。ここで、画像の 8×8 それぞれの位置は、動く駒の位置を表現する。また、56 (8×7 より) チャネルは、クイーンの 8 方向の移動を表現する。8 チャネルは、ナイトの移動を表現する。9 チャネルは、ポーンのアンダープロモーション (三方向のいずれかに移動後、ルーク、ナイト、ビショップに成る) を表現する。これらのようにして $\mathbf{P}$ の各要素が対応する着手が定まり、その値が着手の推定確率を対応する表すこととなる。

次に、AlphaZero の誤差関数を紹介する。ミニバッチを D_t、その大きさを N_t、policy head 部の重み $\mathbf{w}$ と入力 $\mathbf{x}$ に依存する出力を $\mathbf{p}(\mathbf{x};\mathbf{w})$、policy head 部の望ましい出力を $\boldsymbol{\pi}$、value head 部の重み $\mathbf{w}$ と入力 $\mathbf{x}$ に依存する出力を $v(\mathbf{x};\mathbf{w})$、value head 部の望ましい出力を z とする。ミニバッチ D_t が含む訓練サンプル $(\mathbf{x}, z, \boldsymbol{\pi})$ に対する AlphaZero の誤差関数のミニバッチ平均は

$$E_t(\mathbf{w}) = \frac{1}{N_t} \sum_{(\mathbf{x},z,\boldsymbol{\pi})\in D_t} \left\{ (z - v(\mathbf{x};\mathbf{w}))^2 - \boldsymbol{\pi}^\top \log \mathbf{P}(\mathbf{x};\mathbf{w}) + c\|\mathbf{w}\|^2 \right\} \tag{5}$$

である。z は棋譜に記録された勝敗結果 (勝ちが 1、引分けが 0、負けが -1) である。$\boldsymbol{\pi}$ は、入力 $\mathbf{x}$ に対

ル数が 1 で計 7 つ

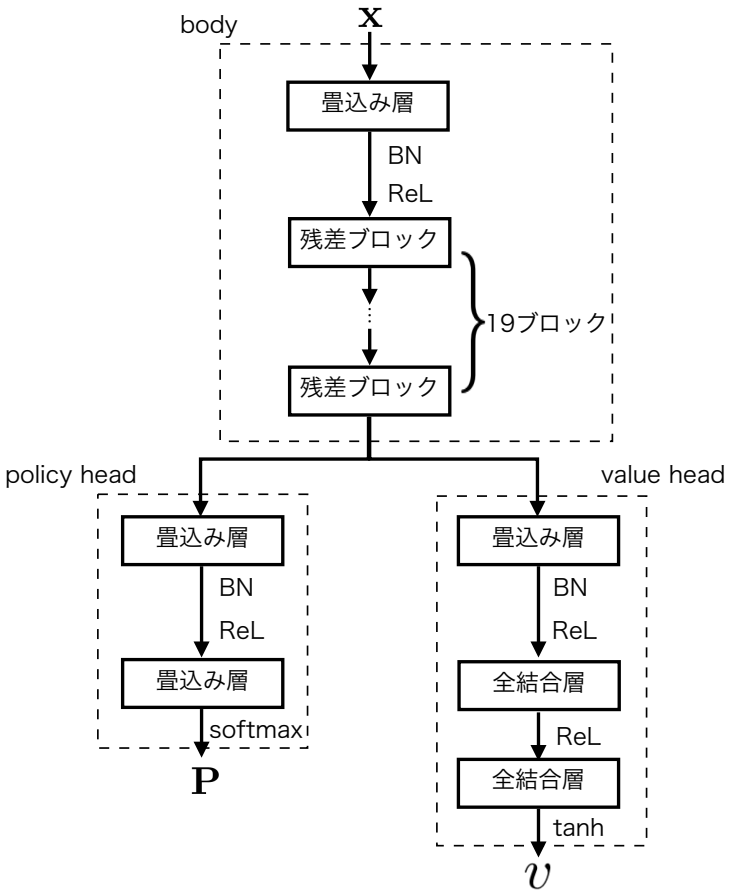


図 5　AlphaZero の NN の構成 (チェスの実験)

するモンテカルロ木探索の着手の確率分布である。AlphaZero では、複数のミニバッチに渡ってこの誤差関数の値を平均的に小さくするように、モメンタム項付き SGD 法を使って重み $\mathbf{w}$ を更新した。

4　シャンチー

本研究の題材としたシャンチーは、中国において人気がある、チェス系の二人ゼロ和完全情報ゲームである。この章では、シャンチーのルールを紹介する。

まずは、シャンチーの駒の配置を行う方法を紹介する。二人のプレイヤは、一方が赤駒、他方が黒駒を持つ。赤駒を持つプレイヤが先手である。駒の動き方は、駒種によって異なる。駒種が全部で 7 つであり、16 枚の駒はいずれかの駒種に属す。駒は図 6 のように、縦 9 本、横 10 本の線が引かれた盤の交点に置く。同じ駒種の駒でも、赤と黒で名前が異なる (以下では黒の駒種の名前を使う)。

初期配置から、赤黒のプレイヤは交互に自分の駒 1 つを動かし (図 7 参照)、パスはできない。自分の駒の移動先に敵の駒があるとき、その駒を取って進むことができる (砲を除く)。取った駒は盤上から除去する。

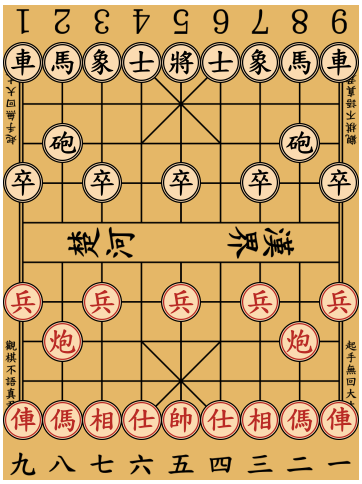


図 6　シャンチーの初期配置[4]

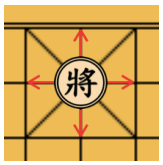
次は、勝敗のルールを紹介する。一方のプレイヤが駒を動かして、そのプレイヤの駒の効きに他方の將がある王手がかかった状況を將 (ジャン) という。將を解消できない状況 (將死、ジャンスー) は負けになる。また、どの駒も動かせない状況 (困斃、クンビー) も負けになる。將を放置する駒移動はルール違反である。赤黒の帥を直接相対させないという王不見王 (ワンブジェンワン) のルールがあり、赤黒の帥が同列で、その間に駒が 1 つもないような状況になる駒移動もルール違反である。連続に將を三回以上することを長將 (チャンジャン) といい、相手の駒 (將以外) を取るために連続で三回以上移動することを長捉 (チャンズオ) という。長將はルール違反である。長捉は一方が手を変えなければ負けになる場合と、双方とも手を変えなければ引分けになる場合がある。長捉は例外の多いルールであり (2011 年試行版のルールブック[5]参照)、本研究では簡単のためこれはルール違反ではないと考える。駒の消耗によって、將死ができなくなった場合、引分けになる。また、相手の致命的なミスがなければ將死ができないと、対局者の合意によって引分けになる場合もある。先後どちらも駒をとらないまま 120 手 (試合により違いがある) 指した場合は、引分けになる。なお、シャンチーでは 1 手は駒を一回動かすことである。

5　目的

本研究の目的は、深層学習を用いてシャンチー上級者の着手及び勝敗を予測し、精度を調べることにある。先行研究より、シャンチーとゲームの性質が類似しているチェスにおいては、深層学習で入力とする特徴や NN の構造に関する知見が得られている。本研究では、チェスの先行研究で用いられた NN をシャンチーに適用して、上級者の着手と勝敗をラベルとした教師あり学習の実験を行う。その上で、得られたシャンチーとチェスの予測精度を比較して、チェスで良いと考えられている方法がシャンチーにおいても適切であるかどうかを調査する。

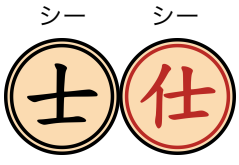
[4]ウィキペディア, シャンチー,URL:https://ja.wikipedia.org/wiki/シャンチー, 最終アクセス 2022
[5]PDF, シャンチーの 2011 年試行版のルールブック,URL:https://www.xqbase.com/protocol/rule2011.pdf, 最後アクセス 2022

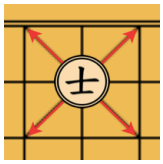
ジャン　シュワイ



九宮(斜線が引かれた交点9個を含む陣地)から出ることができない。前後左右に一路進む。

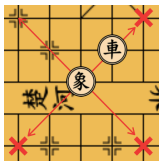
シー　シー





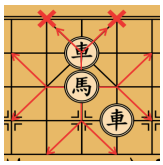
九宮から出ることができない。斜めに一路進む。

シャン　シャン



河を越えることができない。斜めに二路進めるが、駒を飛び越えることができない。

マー　マー



チェスのナイトの動かし方と同じ。しかし、馬に接する上下左右4箇所の駒がある方向に進めない。

チェー　チェー

縦横に何路でも進めるが、駒を飛び越えることができない。

パオ　パオ

縦横に何路でも進めるが、駒を取らずに駒を飛び越えることができない。駒を取るときには、他の駒を1つだけ越えなければならない。

ズウ　ビン

前に一路だけ進める。河を越えると横にも一路進めるようになる。

図 7　シャンチーの駒の動かし方

本研究で行うことは、具体的には以下である。

- シャンチーで教師あり学習を行い着手の予測精度を調べるために、上級者の一万程度の棋譜を収集して、着手をプログラムの処理に適した形式に変換する。また、上級者の棋譜のいくつかの統計情報を調べ、教師あり学習のデータの性質を知る。
- シャンチーとチェスの予測精度を比較するために、チェスの教師あり学習の実験も行う。なお、文献 [9] [18] には、チェスの教師あり学習の着手予測と勝敗予測の報告がある。しかし、これらで用いられた訓練データは大きすぎて、本研究のシャンチーの結果と公平に比較することができない。
- 残差ブロック 9 個からなる NN を用いて上級者の着手及び勝敗の予測精度を調べる。先行研究よりも規模を小さくしたのは、シャンチーの棋譜が少ないためである。

6　訓練データ・テストデータの収集と統計情報

オンライン雑誌「東萍シャンチー」が運営するウェブページ[6]が公開する、シャンチーの主要なトーナメントの棋譜をダウンロードした (図 8 参照)。そして、ダウンロードした棋譜をシャッフルして訓練データに 10000、テストデータに 500 棋譜を用意した。棋譜は中国語で書かれており、人間にとって読みやすい形式[5]で着手が記録されている。例えば、「前砲進二」ならば、2 つの砲が 1 つの列にあって、この列の手番プレイヤから見て遠くにある方の砲を前に 2 マス動かすことを意味する。この形式を、プログラムの処理に適した形式 (駒種、移動元場所、移動先場所) に変換した。そして、シャンチーのルールには左右反転の対称性があり、学習実験ではこの性質を利用して棋譜を二倍に水増しした。したがって、実質的な棋譜数は訓練データが 20000、テストデータが 1000 である。一方、チェスで用いた訓練データの棋譜数は 20000、テストデータの棋譜数は 1000 である。これらは lichess.org[7]からダウンロードしたプレイヤの Elo レイティングが 2400 以上の棋譜であり、棋譜の対局時間タイプは Blitz と Classical である。Blitz では持ち時間は 5 分で、1 手着手するたびに 3 秒追加する。Classical では持ち時間は 30 分で、1 手着手するたびに 20 秒追加する。

表 3 に、データの統計情報を示す。シャンチーの合法手数の平均値はチェスと大体同じであった。また、着手列の長さの平均値も大体同じであった。シャンチーはチェスより引分けになりやすいことも分かった。ここで、チェスでは先手と後手の着手 2 回を合わせて 1 手と数えるのが普通の数え方であるが、本研究ではシャンチーの手数の数え方に合わせるために、先手と後手それぞれ 1 回駒を動かして 2 手と数えた。

図 9 に、棋譜の合法手数ごとの手番の分布を示す。シャンチーでは、手数 44 にピークが見られた。これは、どの棋譜にも必ず 1 つある初期配置によるものである。シャンチーの手数 45 の割合もやや大きくて、これは、後手の初手によるものである。チェスの手数 20 のピークも同じ理由によるものである。こ

[6]東萍シャンチー, URL:http://www.dpxq.com/, 最後アクセス 2022
[7]lichess.org, URL:https://database.lichess.org/, 最後アクセス 2022

こで、チェスでは、先手と後手の初手はどちらも 20 通りである。シャンチーで手数 2 付近にある小さな山は、将 (王手) に起因するものである。シャンチーで王手を回避する合法手数は、通常 1 から 3 までである。チェスの手数 3 付近にある小さな山も同じ理由によるものである。

図 10 に、着手列の長さごとの棋譜の分布を示す。どちらの分布もおおよそ 1 つの山からなり、着手列の長い方向に長く尾を引いている。

表 3　シャンチーとチェスの統計情報。表中の記号 "±" に続く数値は、標準誤差より見積もった 95%の信頼区間を表す

		シャンチー		チェス	
		訓練データ	テストデータ	訓練データ	テストデータ
棋譜数		10000	500	20000	1000
合法手数の平均値		34	35	29	29
着手列の長さの平均値		82	80	85	85
手番の分布 (%) (手番プレイヤの視点)	勝ち	33 ± 0.1	33 ± 0.2	44 ± 0.04	43 ± 0.2
	引分け	34 ± 0.1	35 ± 0.2	13 ± 0.03	15 ± 0.1
	負け	33 ± 0.1	32 ± 0.2	43 ± 0.04	42 ± 0.2
棋譜の分布 (%)	先手勝ち	36 ± 0.5	36 ± 2.2	47 ± 0.4	47 ± 1.6
	引分け	37 ± 0.5	37 ± 2.2	10 ± 0.2	11 ± 1.0
	先手負け	27 ± 0.4	27 ± 2.0	43 ± 0.3	42 ± 1.6

7　着手及び勝敗の深層学習

残差ブロック 9 個からなる NN を用いて、シャンチーとチェス上級者の着手及び勝敗の予測精度を調べて比較する。7.1 節では、深層学習の実験方法を述べる。7.2 節では、実験結果を説明する。

7.1　実験方法

表 4 に、シャンチーのゲーム状況から NN の入力 $\mathbf{x}$ を与える符号化方法を示す。これは、AlphaZero のチェスの符号化方法と大体同じである。入力 $\mathbf{x}$ は、$H \times W \times [(P+R)T+L]$ の多チャネルの画像であると考える。画像の大きさ $H \times W$ は、盤の大きさと対応する 10×9 であり、チャネル数は $(P+R)T+L$ である。P は駒の種類数 14 と[8]、R は盤面の反復回数を表現するチャネル数 3 と、T は履歴の長さ 8 と、L は 3 と対応する[9]。駒種を表すチャネルは、手番プレイヤから見てその駒種がある位置を 1 と、ない位置を 0 とする。盤面の反復回数を表すチャネルは $H \times W$ の値を全て同じにし、チャネル 3 つの値は反復 0 回ならば (0, 0, 0)、1 回ならば (0, 0, 1)、2 回ならば (0, 1, 1)、3 回以上ならば (1, 1, 1) とする。履歴の長さ T は 8 なので、NN には 7 手前までの盤面情報が入力されることとなる。ゲーム開始前の盤面には駒がないとする。手番プレイヤを表すチャネルは $H \times W$ の値を全て同じにし、先手ならば 0、後手ならば 1 とする。手数を表すチャネルも同様で、プレイ開始から数えた着手数を着手列の長さの平均値 82 で

[8]手番プレイヤと相手プレイヤそれぞれ 7 種
[9]手番プレイヤを表すチャネル数が 1、手数が 1、最後に駒を取ってから数えた手数を表すチャネル数が 1 で計 3 つ

棋譜数	トーナメント名	収録年
2011	全国智力运动会象棋比赛 （全国マインドスポーツ大会シャンチートーナメント）	2015-2019
1716	腾讯棋牌“天天象棋”全国象棋甲级联赛 (テンセント「天天シャンチー」全国A級リーグ戦)	2019-2021
546	“威凯杯”全国象棋等级赛 (「ウェカイ杯」全国シャンチー段位戦)	2019
437	“力雅广场杯”全国象棋青年锦标赛 (「力雅広場杯」全国シャンチー青年チャンピオンシップ)	2019
405	“棋王酒业杯”全国象棋个人赛 (「チーワジュエ杯」全国シャンチー個人戦)	2019-2020
304	“万科・拾光杯”全国象棋团队赛 (「ワケ・シグワ杯」全国シャンチー団体戦)	2020
287	“博瑞杯”全国象棋大师公开赛 (「ボルイ杯」全国シャンチーマスターオープンチャンピオンシップ)	2019
257	“宝宝杯”象棋大师公开邀请赛 (「バオバオ杯」シャンチーマスターオープンチャンピオンシップ)	2018-2020
175	“一带一路凌云白毫茶杯”象棋国际公开赛 (「一帯一路凌云白毫茶杯」国際シャンチーオープンチャンピオンシップ)	2019
175	“五龙客家风情园杯”全国象棋团队赛 (「五龍客家風情園杯」全国シャンチー団体戦)	2019

図 8　収集棋譜数トップ 10 のトーナメント

割った値とする。最後に駒を取ってから数えた手数を表すチャネルも同様で、着手数を 120 で割った値とする。ただし、プレイ開始から一度も駒を取っていなければ、プレイ開始から数えた着手数を用いる。

チェスの入力 $\mathbf{x}$ を与える符号化方法も AlphaZero の符号化方法と大体同じである。入力 $\mathbf{x}$ は $H \times W \times [(P+R)T+L]$ の多チャネルの画像であると考える。画像の大きさ $H \times W$ は、盤の大きさと対応する 8×8 であり、チャネル数は $(P+R)T+L$ である。P は駒の種類数 12 と[10]、R は盤面の反復回数を表現するチャネル数 2 と、T は履歴の長さ 8 と、L は 7 と対応する[11]。駒種を表すチャネルは、手番プレイヤから見てその駒種がある位置を 1 と、ない位置を 0 とする。盤面の反復回数を表すチャネルは $H \times W$ の値を全て同じにし、チャネル 2 つの値は反復 0 回ならば (0, 0)、1 回ならば (0, 1)、2 回以上ならば (1, 1) とする。履歴の長さ T は 8 なので、NN には 7 手前までの盤面情報が入力されることとなる。ゲーム開始前の盤面には駒がないとする。手番プレイヤを表すチャネルは $H \times W$ の値を全て同じにし、先手ならば 0、後手ならば 1 とする。手番プレイヤのキャスリング 2 種の合法性を表すチャネルも同様で、キャスリングが合法ならば 1、合法ではないならば 0 とする。相手プレイヤのキャスリング 2 種の合法性を表すチャネルは同じである。手数を表すチャネルも同様で、プレイ開始から数えた着手数を着手列の長さの平均値 85 で割った値とする。最後に駒を取ったり、ポーンを動かしたりしてから数えた手数を表すチャ

[10]手番プレイヤと相手プレイヤそれぞれ 6 種

[11]手番プレイヤを表すチャネル数が 1、手数が 1、手番プレイヤのキャスリング 2 種の合法性を表すチャネルが 2、相手プレイヤのキャスリング 2 種の合法性を表すチャネルが 2、最後に駒を取ったり、ポーンを動かしたりしてから数えた手数を表すチャネル数が 1 計 7 つ

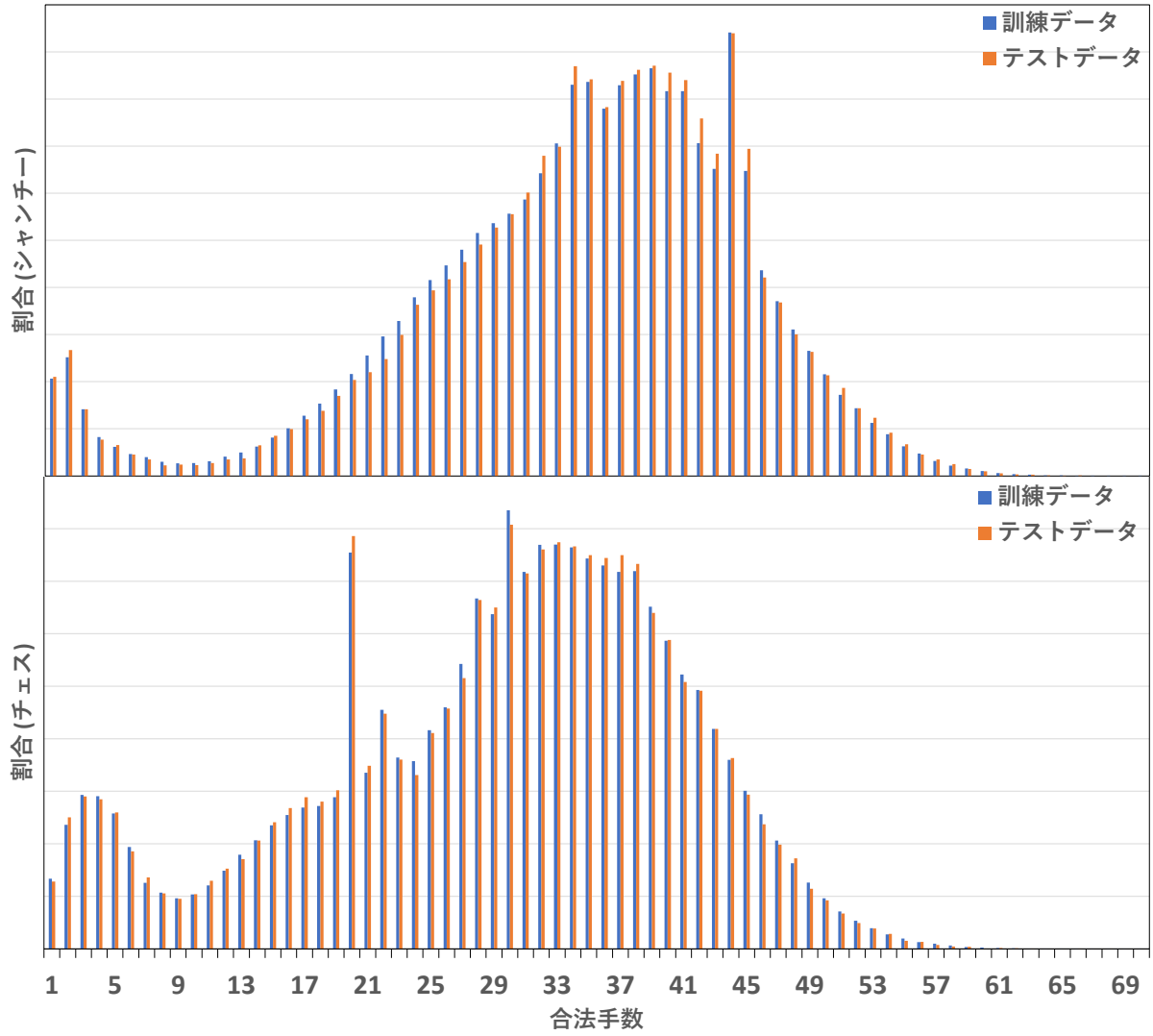


図 9　手番の分布

ネルも同様で、着手数を 100 で割った値とする。ただし、プレイ開始から一度も駒を取っていなければ、プレイ開始から数えた着手数を用いる。

図 11 に、本研究で用いた NN の構造を示す。これは、AlphaZero の NN の構造とおおよそ同じであり、三つの部分 (body、policy head、value head) からなる。NN の全ての畳込みのストライドは 1、パディングは出力の画像の大きさを変えないように設定する。

シャンチープレイヤの着手の確率分布の推定 $\mathbf{P}$ と勝敗の予測 v は、NN の出力と次のように対応する。まず、$\mathbf{P}$ は大きさ 10×9 のチャネル 50 個の画像として表現される。ここで、画像の 10×9 それぞれの画素は、手番プレイヤから見て動く駒の移動元の位置と対応する。画像の 18 (9×2 より) チャネルは、前後の 2 方向の駒の動き方を表現する。16 (8×2 より) チャネルは、左右の 2 方向の駒の動き方を表現する。8 チャネルは、馬 (ナイト) の動き方を表現する。4 チャネルは、象の動き方を表現する。4 チャネル

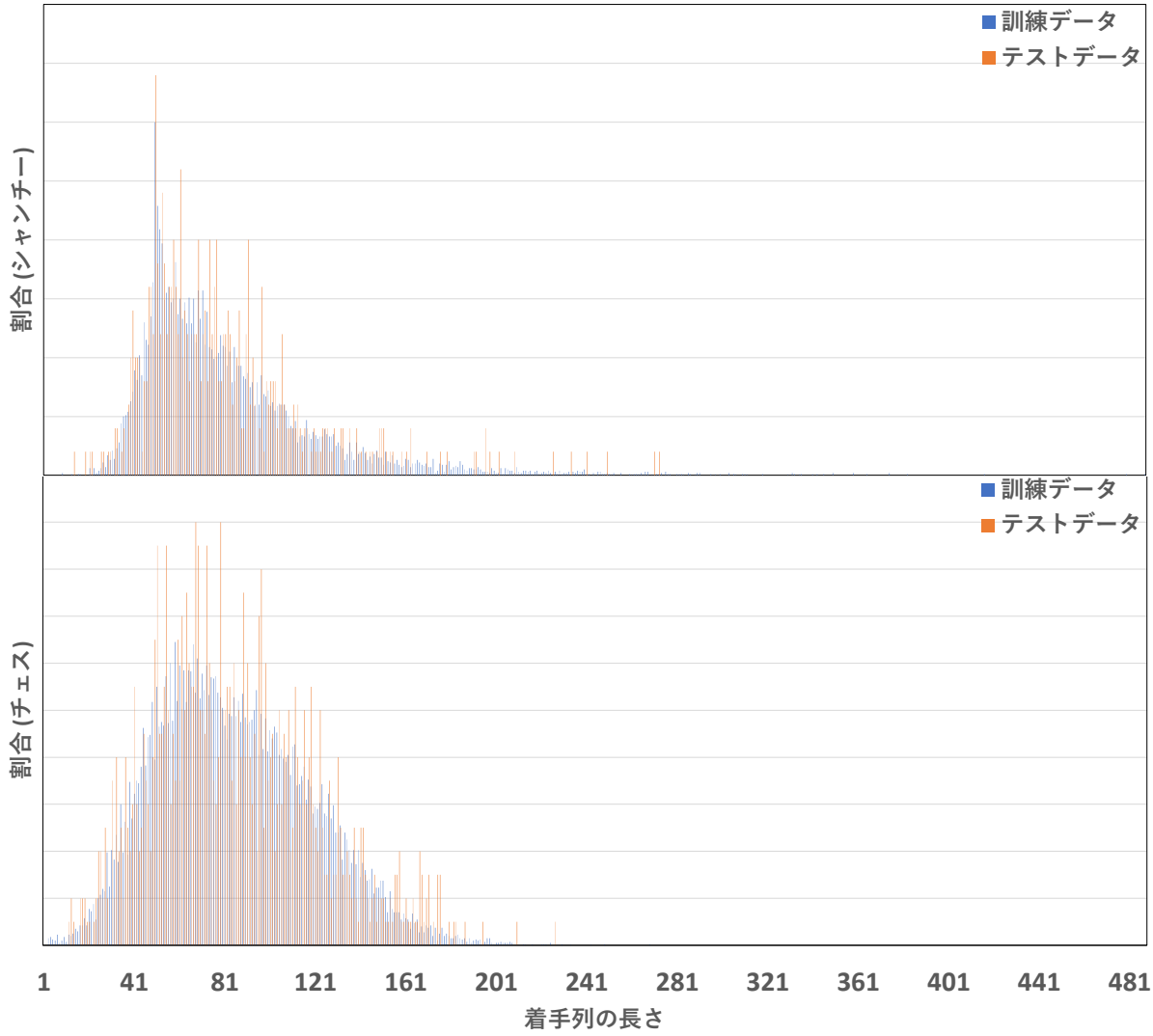


図 10　棋譜の分布

は、士の動き方を表現する。このようにして $\mathbf{P}$ の各要素が対応する着手が定まり、各値がその着手の推定確率を表すこととなる。次に、v は手番プレイヤの勝敗の予測である。$0.5 \leq v \leq 1$ の場合は手番プレイヤの勝ちと、$-0.5 < v < 0.5$ の場合は引分けと、$-1 \leq v \leq -0.5$ の場合は手番プレイヤの負けと対応する。チェスプレイヤの着手の確率分布の推定 $\mathbf{P}$ は AlphaZero のものと同じである。チェスの勝敗の予測 v もシャンチーと同様である。

シャンチーとチェスの深層学習実験は Caffe [19] を利用して行う。Caffe は、NN を利用して、画像認識の深層学習を効率的に行うライブラリである。そして、シャンチーとチェスの実験どちらも、入力 $\mathbf{x}$ とラベルからなるサンプル 128 個でミニバッチ D_t を構成し、モメンタム項付き SGD を行う。各反復回数 t のサンプルは訓練データから重複を許してランダムに抽選される。このミニバッチを使って、重み $\mathbf{w}$ を式 (2) に従い更新する。ここで、学習率 ϵ は、シャンチーとチェスどちらも 0.02 とする。ただし、どちら

表 4　シャンチーの実験で NN に入力するゲーム状況の特徴

特徴	チャネル数
手番プレイヤの駒の配置	7
相手プレイヤの駒の配置	7
盤面の反復回数	3
手番プレイヤ	1
手数	1
最後に駒を取ってから数えた手数	1

も 200 万回の反復ごとに学習率を $\frac{1}{2}$ にする。モメンタム項のハイパーパラメータ η は 0.9 とする。そして、シャンチーとチェスの誤差関数は式 (5) を使う。式 (5) の L2 正則化項は、Caffe の重み減衰 (weight decay) を行う機能で実現する。減衰係数は 0.001 である。重み $\mathbf{w}$ の結合重み $\mathbf{W}$ の部分は MSRA [20] と呼ばれる初期化手法を使い、バイアス $\mathbf{b}$ の部分は 0 で初期化する。

7.2　実験結果

図 12 にシャンチーの深層学習の訓練及びテストの誤差平均値と予測の一致率を示す。図中の訓練の policy 誤差は、式 (5) の policy 部 $\frac{1}{N_t}\sum_{(\mathbf{x},z,\boldsymbol{\pi})\in D_t} -\boldsymbol{\pi}^\top \log \mathbf{P}(\mathbf{x};\mathbf{w})$ の値である。Value 誤差は、式 (5) の value 部 $\frac{1}{N_t}\sum_{(\mathbf{x},z,\boldsymbol{\pi})\in D_t} (z - v(\mathbf{x};\mathbf{w}))^2$ の値である。和誤差は policy 誤差と value 誤差の和である。

図中のテストの誤差平均値も、訓練の誤差平均値と同じように計算し得られた。ただし、標本集合はミニバッチよりも大きくて、テストデータからランダムに重複を許して 12800 個の手番のサンプルを抽選して構成した。Policy 誤差のベースは、入力 $\mathbf{x}$ の合法手全て等確率と考えて得られた $\mathbf{P}(\mathbf{x})$ のテストの誤差平均値である。Value 誤差のベースは、入力 $\mathbf{x}$ によらないと考えて得られた v のテストの誤差平均値である。ここで、v の定数値には、訓練データの value 誤差を最小にする値 0.004 を使った。

着手及び勝敗予測の一致率も、テストの誤差平均値の推定で用いた標本集合で推定する。ここで、着手予測の一致とは、$\mathbf{P}(\mathbf{x};\mathbf{w})$ の推定確率が最も高い着手と棋譜の着手とが一致することを指す。着手予測のベースは、入力 $\mathbf{x}$ の合法手数分の 1 の平均値である。勝敗予測のベースは、常に引分けを予測する時の勝敗予測の一致率である。

反復回数 t が増えるにつれて、訓練とテストの和誤差が小さくなり、着手予測の一致率は 44% に、勝敗予測の一致率は 51% に達した。ここで、反復回数 t が 10 の付近にテストの policy 誤差の点が欠落しているのは、softmax 関数の小さい正の値が数値誤差により 0 になってしまい、誤差関数に log(0) が出現したことによる。

図 13 に、チェスの深層学習の訓練及びテストの誤差と予測の一致率を示す。図中の点の意味は、シャンチーと大体同様である。ここで、value 誤差のベースを与える v の定数値には、訓練データの value 誤差を最小にする値 0.005 を使った。勝敗予測のベースは、常に手番プレイヤの勝ちを予測するときの勝敗予測の一致率である。着手予測の精度はシャンチーと大体同じで、一致率 45% に達した。また、勝敗予測の一致率は 39% 程度で、これはシャンチーより 12 ポイント低い。表 3 の手番の分布を見ると、チェスは勝ちだけではなく負けの割合も大きい。これにより、チェスの勝敗予測は勝つか負けるかほとんど分か

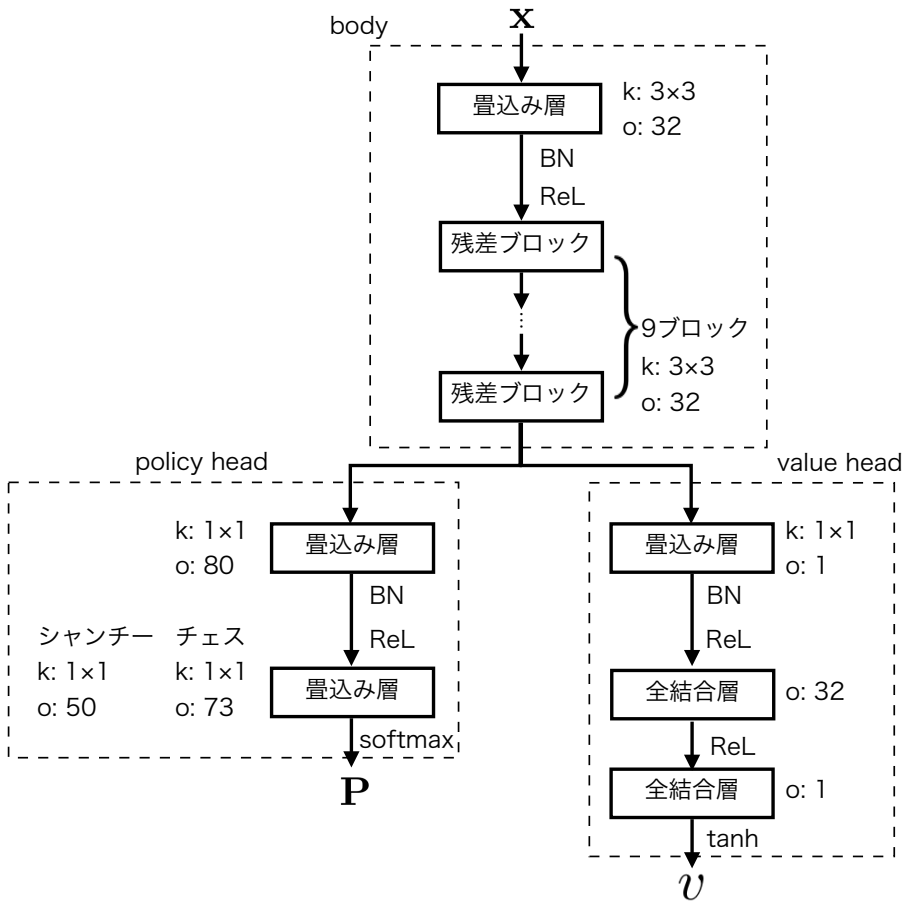


図 11　本研究での NN の構成。図中 “k” はフィルタの大きさ、“o” は畳込みのフィルタや全結合のユニットの数を表す

らないような手番は引分けと予測することとなり、引分けの割合が小さいため、勝敗の予測精度が低くなる傾向が見られた。

図 14 に、棋譜の着手ラベルが $\mathbf{P}(\mathbf{x};\mathbf{w})$ のトップ 16 の着手予測に含まれる確率の平均値を示す。用いた標本集合はテストの誤差平均値の推定で用いたものである。着手予測トップ 16 はおおよそ着手ラベル 1 つを含んでいて、シャンチーとチェスの着手のトップ 1 から 16 までの予測精度は大体同じであった。

図 15 に、予測一致率とプレイ開始から数えた着手数との関係を示す。予測の一致率は、テストデータの全ての棋譜から推定した。初手において、チェスの着手一致率はシャンチーより顕著に良くなる。これは、チェスの初手は約 5 割が「e4」で予測が簡単であることによる。なお、シャンチーで予測した初手は左から数えて 1 番目の駒「砲」を右に 3 マス動かす着手である。これは、訓練・テストデータで最も多く選ばれた初手の 1 つと一致する。他の手数 (131 手目以前) で着手予測の精度はシャンチーとチェスがおおよそ似ている。また、初手の付近において、シャンチーの勝敗一致率はチェスより有意に良くなる。これは、シャンチーの結果は約 4 割が引分けで、引分けの予測が簡単であることによる。終盤に近づくと、勝敗予測の精度がよくなることも見て取れる。

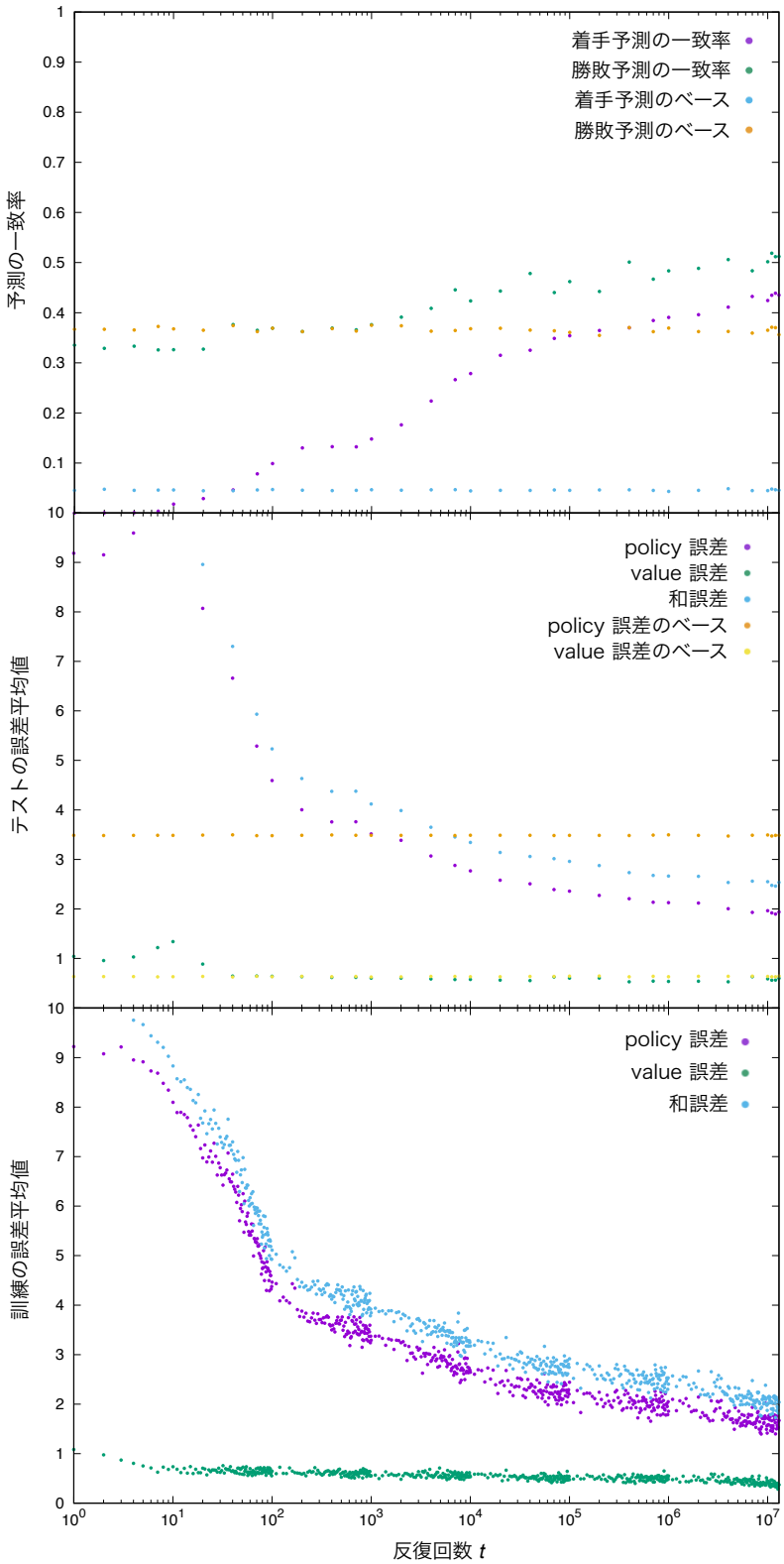


図 12　シャンチーの深層学習の実験

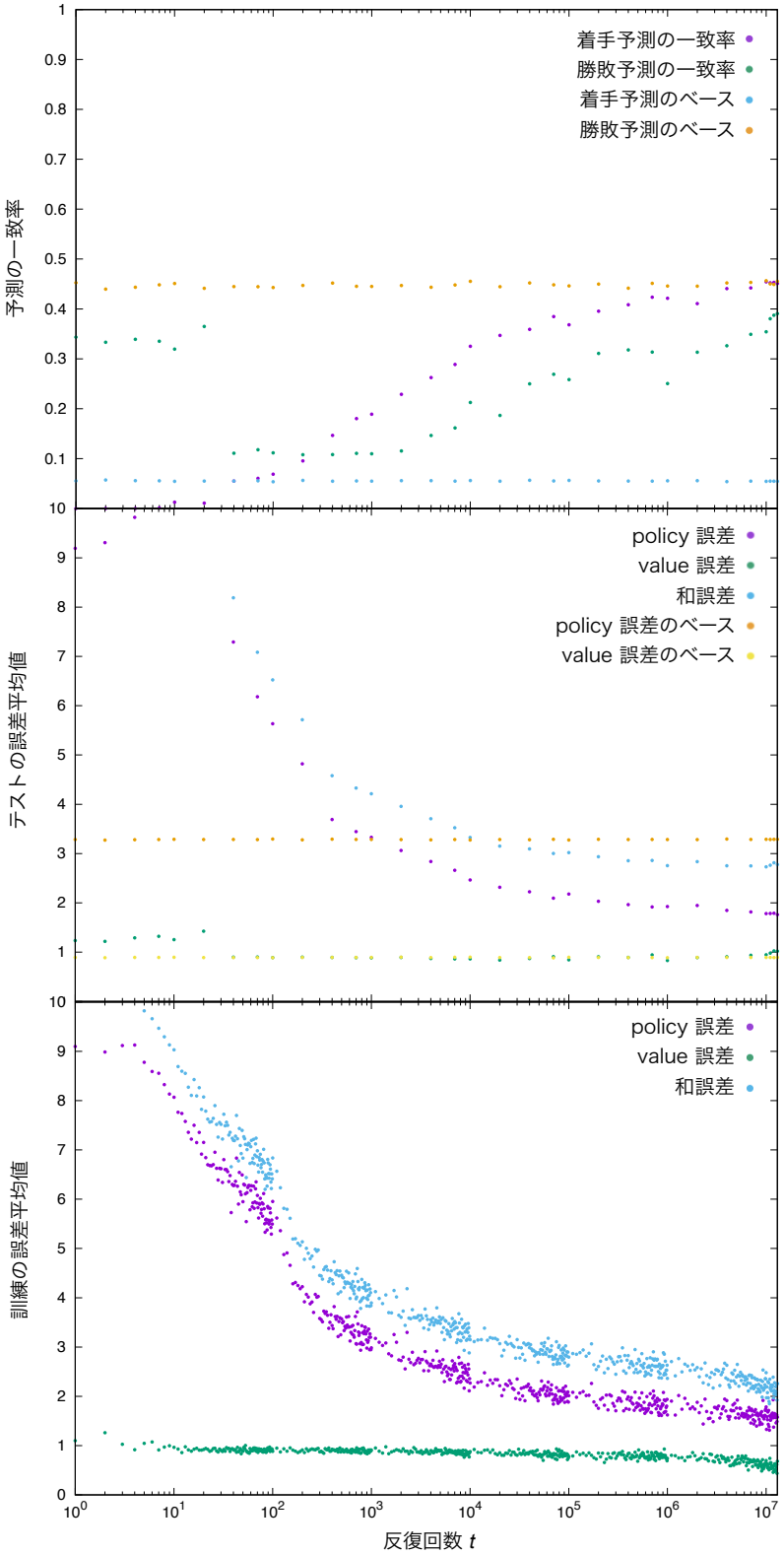


図 13　チェスの深層学習の実験

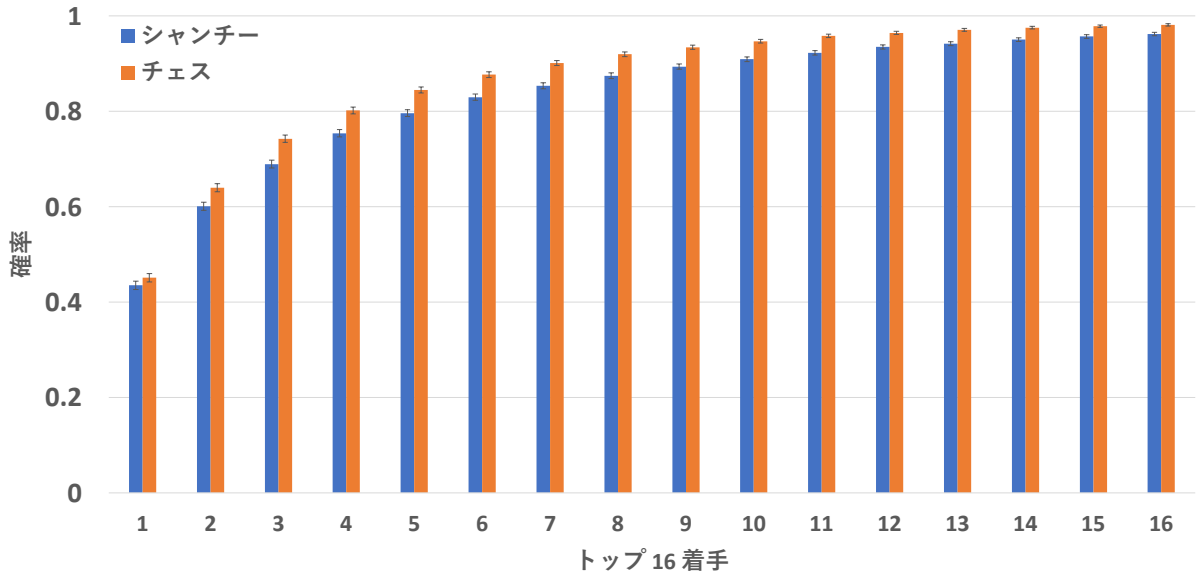


図 14　棋譜の着手がトップ 16 の予測着手に含まれる確率。図中のエラーバーは、標準誤差より見積もった 95%の信頼区間を表す

表 5　駒種ごとの着手の一致率 (%)。駒種は、棋譜で着手されたもの。表中の記号 “±” に続く数値は、標準誤差より見積もった 95%の信頼区間を表す

シャンチー						
卒	車	馬	象	士	砲	將
42 ± 2	46 ± 2	51 ± 2	45 ± 4	41 ± 5	34 ± 2	60 ± 5
チェス						
ポーン	ルーク	ナイト	ビショップ	クイーン	キング	キャスリング
46 ± 2	41 ± 2	45 ± 2	46 ± 2	38 ± 3	50 ± 3	61 ± 5

図 16 に、反復回数ごとに着手予測 $\mathbf{P}(\mathbf{x};\mathbf{w})$ のトップ 1 の予測着手が合法手に含まれる確率を示す。確率は、テストの誤差平均値の推定で用いた標本集合で推定した。反復回数 t が 100 回以降、シャンチーとチェスの予測着手は大体合法手に含まれる。合法手の認識が容易であり、これはチェスの結果と一致する。

表 5 に、駒種ごとの着手の一致率を示す。この一致率は、テストデータから 12800 個の手番のサンプルを重複を許しランダムに抽選して構成した標本集合で推定したものである。この表を見ると、シャンチーでは、駒種「將」と「馬」の着手予測精度が最も良く、「砲」の着手予測精度が悪い。チェスでは、キャスリングの予測精度が最も良く、これ以外は、キング、ナイト、ポーン、ビショップの予測精度が良い。そして、ルークとクイーンの予測精度が悪い。チェスでは、クイーン、ルークような移動距離が遠くなりえる駒の予測が難しくなる。そして、シャンチーでも遠くまで移動可能な駒種「砲」の予測が難しくなった。しかし、同じく遠くまで移動可能な駒種「車」とビショップの予測精度は良かった。この原因は不明である。

図 17 に、着手確率とその推定値との関係を示す。着手確率は、テストの誤差平均値の推定で用いた標本集合で推定した。シャンチーとチェスどちらも着手確率の推定値が大きくなると、着手確率も大きくな

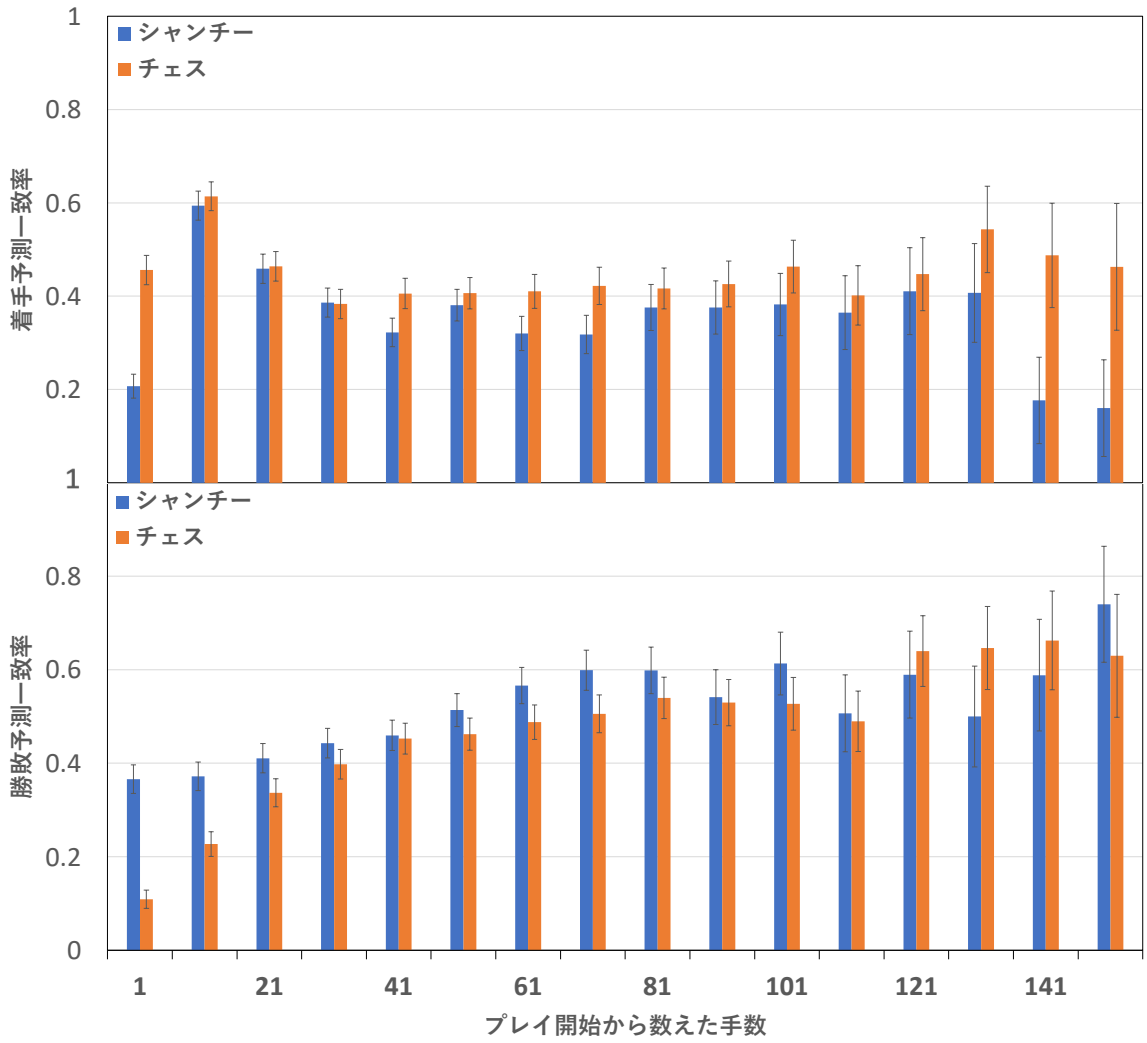


図 15　一致率とプレイ開始から数えた手数との関係。図中のエラーバーは、標準誤差より見積もった 95% の信頼区間を表す

ることが見て取れる。着手確率の推定値が 0.9 以下の着手では、確率がその推定値より低くなっている。

8　終わりに

本研究で、チェスの先行研究で用いられた NN をシャンチーに適用して、シャンチー上級者の着手及び勝敗をラベルとした教師あり学習の実験を行った。そして、得られたシャンチーの予測精度とチェスの予測精度を比較した。また、チェスで良いと考えられた方法がシャンチーにおいても適切であるかどうかを調査した。

シャンチーとチェスの予測精度を比較するために、シャンチーとチェス上級者の棋譜を収集した。収集した棋譜の着手をプログラム処理に適した形式に変換した。変換後の棋譜の性質を知るために、棋譜の

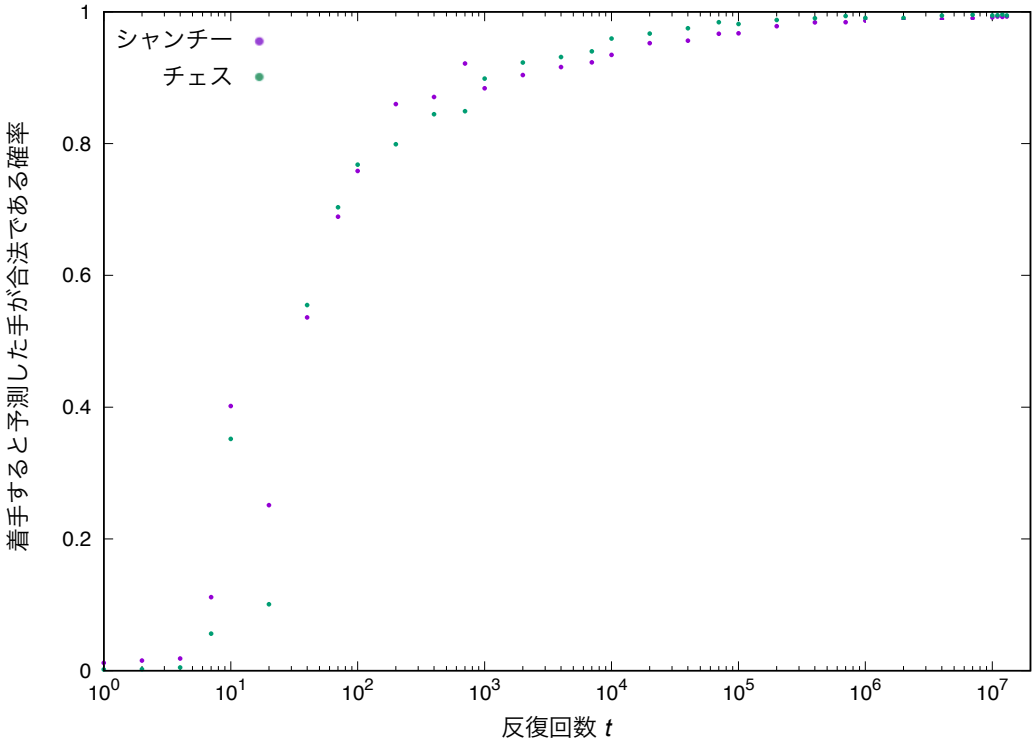


図 16　反復回数ごとの、トップ 1 の予測着手が合法手である確率

いくつかの統計情報を調べた。

実験の NN は残差ブロック 9 個を含み、畳込み層は大きさ 3×3 のフィルタ 32 個からなる。収集した訓練データを用いて着手と勝敗を予測する NN を学習して、予測精度を調べた。実験の結果、シャンチー上級者の着手予測は、チェスとほぼ同じ精度でなされることが分かった。また、シャンチーでは駒種「砲」、チェスでは「ルーク」及び「クイーン」の着手を予測することが難しいとの知見も得られた。合法着手の認識が容易こともが分かった。その一方で、シャンチーの勝敗の予測はチェスより良い精度でなされることが分かった。これは、シャンチーがチェスよりも引分けになりやすく、引分けの予測が上級者の棋譜と良く一致することに起因する。

実験結果により、AlphaZero のチェスの NN の構造と符号化手法をシャンチーに適用することが適切であるとの感触が得られた。

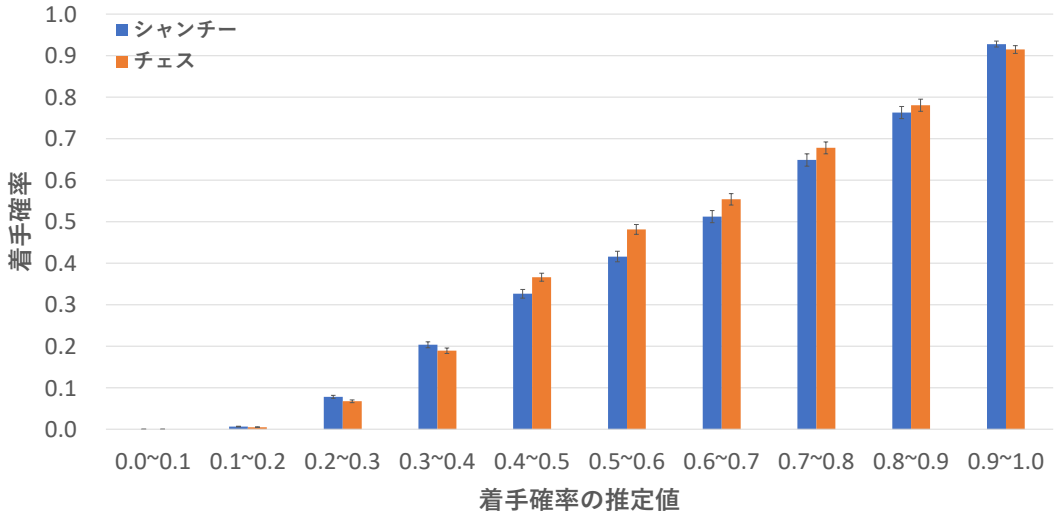


図 17　着手確率とその推定値との関係。図中のエラーバーは、標準誤差より見積もった 95%の信頼区間を表す

参考文献

[1] 伊藤毅志, 保木邦仁, 三宅陽一郎. **ゲーム情報学概論-ゲームを切り拓く人工知能-**. コロナ社, 2018.

[2] Louis Victor Allis. *Searching for solutions in games and artificial intelligence*. PhD thesis, Maastricht University, September 1994.

[3] Warren Sturgis McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[4] 岡谷貴之. **深層学習**. 講談社, 2015.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, volume 37, pages 448–456, 2015.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Chao Gao, Ryan Hayward, and Martin Müller. Move prediction using deep convolutional neural networks in Hex. *IEEE Transactions on Games*, 10(4):336–343, 2017.

[8] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550:354–359, 2017.

[9] Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Aligning superhuman ai with human behavior: Chess as a model system. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1677–1687, 2020.

[10] Kun Shao, Dongbin Zhao, Zhentao Tang, and Yuanheng Zhu. Move prediction in gomoku using deep learning. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation*, pages 292–297, 2016.

[11] Paweł Liskowski, Wojciech Jaśkowski, and Krzysztof Krawiec. Learning to play Othello with deep neural networks. *IEEE Transactions on Games*, 10(4):354–364, 2018.

[12] Yi Yang, Shuigeng Zhou, Jihong Guan, and Chuansheng Shen. The complexities of random-turn Hex, Square and Triangle games. *IEEE Transactions on Games*, Early Access, 2020.

[13] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(8):2011–2023, 2020.

[14] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362:1140–1144, 2018.

[15] Meiyang Li and Wenzhi Huang. Research and implementation of chinese chess game algorithm based on reinforcement learning. In *2020 5th International Conference on Control, Robotics and Cybernetics*, pages 81–86, 2020.

[16] Zheyuan Fan, Yuming Kuang, and Xiaolin Lin. Chess game result prediction system. *CS 229 Machine Learning Project Report*, 2013.

[17] Mohammad M Masud, Ameera Al-Shehhi, Eiman Al-Shamsi, Shamma Al-Hassani, Asmaa Al-Hamoudi, and Latifur Khan. Online prediction of chess match result. In *Advances in Knowledge Discovery and Data Mining*, pages 525–537, 2015.

[18] Rafał Dreżewski and Grzegorz Wątor. Chess as sequential data in a chess match outcome prediction using deep learning with various chessboard representations. *Procedia Computer Science*, 192:1760–1769, 2021.

[19] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 675–678, 2014.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.