目 次

第6章　アルファ碁からアルファ碁ゼロへ

2017年10月19日。ついにアルファ碁の全貌が明らかになりました。新しいネイチャー論文「Mastering the Game of Go without Human Knowledge (人間の知識なしに囲碁をマスターする)」が発表されたのです。新しい論文では、これまで秘密のベールに包まれていた強化学習手法を中心に据え、ゼロから囲碁AIを学習するための、洗練された手法について説明されています。論文では、「たった3日で」「ゼロから」「1台のマシンでも動く」最強囲碁AIができたことをアピールしています。アルファ碁ゼロの技術は、これまで説明してきたディープラーニング、探索、強化学習の組み合わせであり、決して難しいものではありません。従来版のアルファ碁からアルファ碁ゼロに至る技術を、本章では解説していきます。[1]

rev.1 発行 (2017/11/26)

[1]本稿は大槻知史著「アルファ碁解体新書」の第6章として追加することを想定して書いたものですが、なるべく本章だけでも理解できるように記述したつもりです。

6.1 はじめに

ここまで説明してきたアルファ碁 (以下、従来版アルファ碁と呼びます) が、「時間のかかる処理であるディープラーニングをいかに囲碁の探索にうまく活用するか」という観点で設計されていたのに対し、アルファ碁ゼロは、「いかにして人間の知識なしに囲碁 AI を作るか」「いかにしてゲーム固有の情報を使わないか」を主題に、設計・開発されたようです。

図 6.1 に示すように従来版アルファ碁に対するアルファ碁ゼロの改良ポイントは主に 3 点です。

第 1 にアルファ碁ゼロは、デュアルネットワークと呼ばれる、**従来版アルファ碁のポリシーネットワークとバリューネットワーク (MEMO 参照)** とを統合したディープラーニングのモデルを用いています。

第 2 にアルファ碁ゼロのモンテカルロ木探索の手法は、デュアルネットワークの勝率予測の性能が大幅にアップしたため、従来版アルファ碁では行っていたプレイアウトの処理が不要になり、単純明快な分かり易い処理となっています。

第 3 にアルファ碁ゼロの頭脳にあたるデュアルネットワークのパラメータは、強化学習により獲得されます。このネットワークを、モンテカルロ木探索に組み込むことで、最強囲碁 AI「アルファ碁ゼロ」が完成します。

本章では、まず 6.2 節でアルファ碁ゼロのネットワーク構造を、次に 6.3 節でアルファ碁ゼロのモンテカルロ木探索手法を説明します。その後 6.4 節で、ネットワークパラメータを強化学習により獲得する手法について説明します。

MEMO: 従来版アルファ碁のポリシーネットワークとバリューネットワーク

従来版アルファ碁のポリシーネットワークは次の一手予測をおこなうディープラーニングのモデルです。出力は、19 × 19 の各位置に対し、その手が一番良い手となる確率を出力します。

一方、従来版アルファ碁のバリューワークは、今の手番 (黒または白) からみた勝率を予測するディープラーニングのモデルです。出力は、-1.0 以上 1.0 の数値で、1.0 のとき勝率 100%, -1.0 のとき勝率 0%であることを表します。

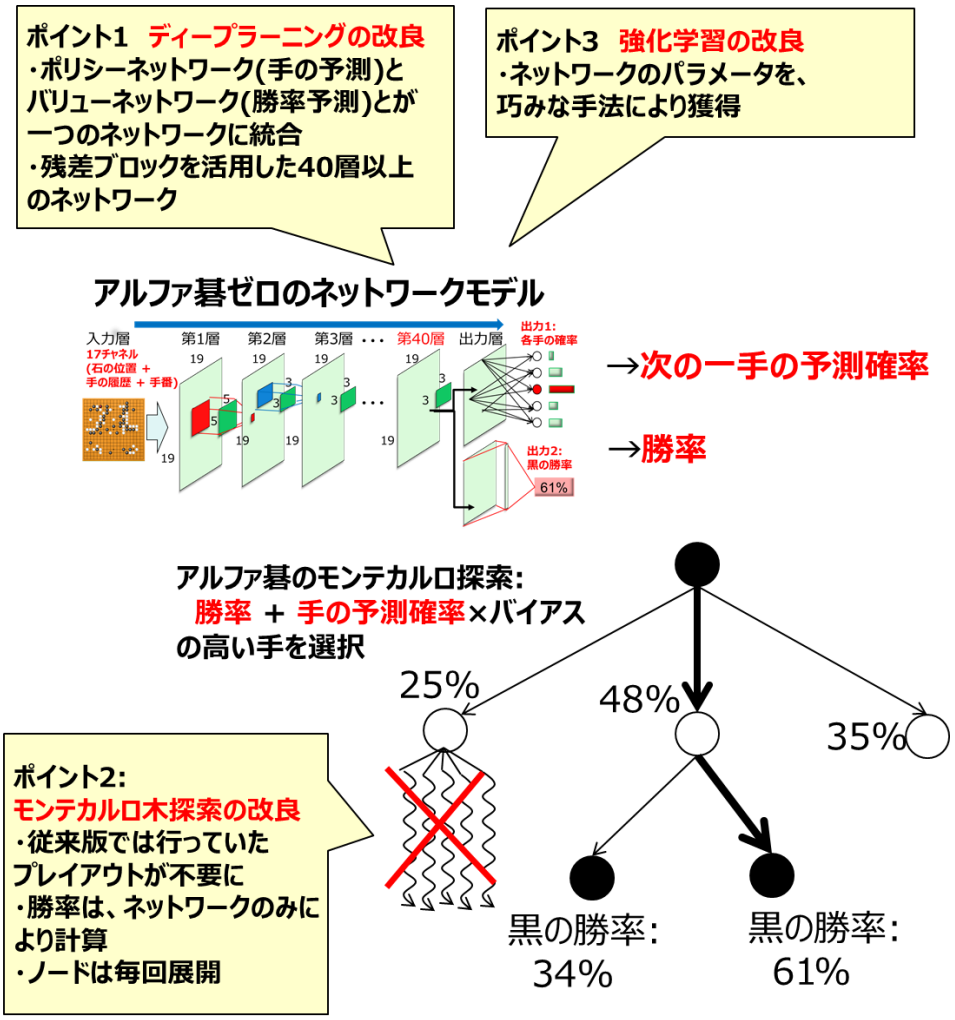


図 6.1: 従来版アルファ碁に対するアルファ碁ゼロの 3 つの改良点

6.2　アルファ碁ゼロにおけるディープラーニング

アルファ碁ゼロでは、従来版アルファ碁における次の一手予測をつかさどるポリシーネットワークと、勝率予測をつかさどるバリューネットワークとを統合したディープラーニングモデルが用いられます。**2017 年度版の新しいネイチャー論文** (MEMO 参照) では、このネットワークを dual と言及しているため、本稿では**デュアルネットワーク** と呼ぶことにします。

本節では、アルファ碁ゼロのデュアルネットワークと、従来版アルファ碁のポリシーネットワーク・バリューネットワークとの構造の違いを中心に説明します。

MEMO: 2016 年度版と 2017 年度版のネイチャー論文について

グーグル・ディープマインド社のメンバーは、2017 年 10 月に、アルファ碁ゼロの仕組みを解説した下記の新しい論文をネイチャー誌に発表しました。本稿は、基本的にこの 2017 年論文の解説となります。この論文のことは、以下**ネイチャー 2017 論文**と呼びます。

『Mastering the Game of Go without Human Knowledge (人間の知識なしに囲碁を究める)』, (David Silver, et al., Nature, 2017)

一方、従来版アルファ碁については、2016 年 1 月にネイチャー誌に掲載された論文に記載されています。この論文のことは、以下**ネイチャー 2016 論文** と呼びます。

『Mastering the game of go with deep neural networks and tree search (深層ニューラルネットワークと木探索により囲碁を究める)』(David Silver, et al., Nature, 2016)

6.2.1　デュアルネットワークの構造

図 6.2 に示すように、デュアルネットワークの入力層から出力層までの構造は、下記のようになっています。

- 入力層: 17 チャネル
- 第 1 層: 3 × 3 サイズ 256 種類のフィルタを持つ畳み込み層と、バッチ正規化、ReLU 活性化関数
- 第 2 層〜第 39 層: 19 個の残差ブロック[2]。なお各残差ブロックは、3 × 3 サイズ 256 種類のフィルタを持つ畳みこみ層、バッチ正規化、ReLU 活性化関数各 2 個からなります。
- 各手の予測確率を計算する**次の一手予測部**と、勝率を予測する**勝率予測部**に分岐
- 次の一手予測部の構造:

[2]ネイチャー 2017 論文では、この 19 個の残差ブロックを中心に解説がされていますが、最強のバージョンは 39 個の残差ブロックからなるそうです。

– 次の一手予測部第 1 層: 1 × 1 サイズの 2 種類のフィルタを持つ畳み込み層と、バッチ正規化、ReLU 活性化関数
– 次の一手予測部第 2 層: 362 ノードに出力する全結合層とソフトマックス関数
– 次の一手予測部の出力: 362 ノード (361 個の盤面上の位置と、パスのいずれかを出力する確率に対応)

- 勝利予測部の構造:
  – 勝率予測部第 1 層: 1 × 1 サイズ 1 種類のフィルタを持つ畳み込み層と、バッチ正規化、ReLU 活性化関数
  – 勝率予測部第 2 層: 256 ノードに出力する全結合層と ReLU 活性化関数
  – 勝率予測部第 3 層: 1 ノードに出力する全結合層と tanh 活性化関数
  – 勝率予測部の出力: 1 ノード (-1.0 以上 1.0 以下の値。+1.0 が黒勝ち、-1.0 が白勝ちに対応)

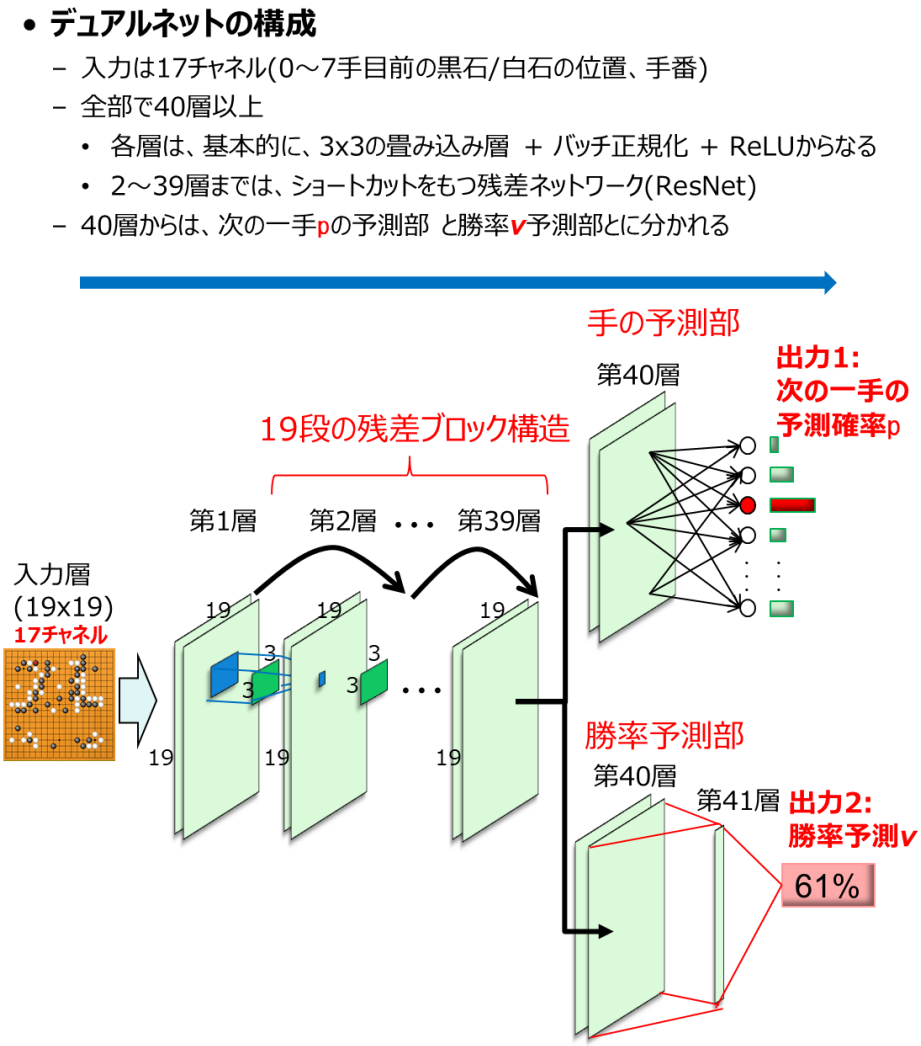


図 6.2: デュアルネットワークの構造

デュアルネットワークの入力に関して

以下、デュアルネットワークの特長的な部分にフォーカスして説明していきます。

最初にデュアルネットワークの入力に関して説明します。従来版アルファ碁では 48 チャネルあった入力情報 (図 6.3(a)) が、デュアルネットワークでは石の位置、履歴、手番の計 17 チャネル (図 6.3(b)) となっています。従来版アルファ碁で入力特徴として使われた、空点の位置、連に関する呼吸点の数の情報、シチョウに関する情報などは、使わなくなりました。これらの情報がある方が、学習は進みやすいと考えられますが、アルファ碁ゼロの「ゲーム固有の情報をなるべく使わない」設計ポリシーに従ったのでしょう。

一方、履歴の入力方法は、従来版アルファ碁 (図 6.3(a)) では、n(=1〜7) 手前に打たれた交点そのものだけを入力していましたが、アルファ碁ゼロのデュアルネットワーク (図 6.3(b)) では、n 手前の黒、白それぞれの石の位置情報を全て入力しています。これらの情報により、直前に重要となった、連に関する情報などを補う効果があるのかもしれません。

(a)従来版アルファ碁で使われた入力48チャネル

入力チャネルの種類	チャネル数
黒石の位置	1
白石の位置	1
空白の位置	1
k手前に打たれた位置(k=1〜8)	8
石がある場合の当該連の呼吸点の数(k=1〜8)	8
そこに打った後石を取れるか(取る数:k=1〜8)	8
そこに打った後、当該連を取られる場合に、何個石を取られるか? (石の数;k=1〜8)	8
その石に打った後の、当該連の呼吸点の数(呼吸点の数:k=1〜8)	8
そこに打った後、隣接する相手の連をシチョウで取れるか?	1
そこに打たれた後、隣接する味方の連をシチョウで取られるか?	1
合法手か?	1
すべて1で埋める	1
すべて0で埋める	1
合計	48

(b)アルファ碁ゼロのデュアルネットワークで使われた入力17チャネル

入力チャネルの種類	チャネル数
黒石の位置	1
白石の位置	1
k手前の黒石の位置(k=1〜7)	7
k手前の白石の位置(k=1〜7)	7
手番(黒番なら全て1, 白番なら全て0)	1
合計	17

図 6.3: (a) 従来版アルファ碁で使われた入力 48 チャネル と (b) アルファ碁ゼロのデュアルネットワークの入力 17 チャネルの内訳

次の一手予測部と勝率予測部とを統合した構造

次に、デュアルネットワークの特長である「次の一手予測部」と「勝率予測部」とを統合した構造について述べます。複数のタスクでモデルの一部を共有し同時に学習する手法のことを、一般に**マルチタスク学習 (MEMO 参照)** と呼びます。

ネイチャー 2017 論文によると、統合した結果として、次の一手予測器としての性能は、従来のポリシーネット単独の場合の方が高いようです。ただし、モンテカルロ木探索に組み込んだ場合には、統合したデュアルネットワークの方が性能が高まるようです。試行錯誤の結果として、総合力としては、デュアルネットワークの方が優れているとの結論に至ったのかもしれません。

次節で説明するモンテカルロ木探索では、デュアルネットワークの次の一手予測部が出力する手の予測確率 p により探索深さが制御され、勝率予測部が出力する勝率予測 v が局面評価関数となります。ゲーム木の探索において重要な、「探索深さ」「局面評価関数」という 2 要素がここでも登場します。アルファ碁ゼロでは、この重要な 2 要素が、統合された一つのネットワークモデルにより計算されるわけです。

MEMO: マルチタスク学習

複数のタスクの学習に対し、モデルの一部を共有し、同時に学習する手法のことをマルチタスク学習と呼びます。モデルを共有することで、複数のタスクに共通する要因を認識しやすくなり、予測精度の向上が期待できます。デュアルネットワークの場合も、次の一手予測と勝率予測のモデルを一部共有することで、これらの認識に必要となる共通の構造の学習が進みやすくなると考えられます。

残差ネットワーク

最後に残差ネットワーク (ResNet) について少し詳しく説明します。

残差ネットワーク (MEMO 参照) は、図 6.4(a) に示した残差ブロックと呼ばれる構造を多段に重ねたものです。デュアルネットワークではこの残差ブロックを 19 段重ねたものを用いています (図 6.4(b))。

デュアルネットワークの残差ブロックは、この場合 3 × 3 サイズ 256 種類のフィルタを持つ畳み込み層 (3x3 Conv 256) と、バッチ正規化 (Bn)、ReLU 活性化関数 (ReLU) を 2 回繰り返したものがメイン経路になっていますが、特徴的なのは、このメイン経路を通らないショートカット[3]を持つことです。このショートカットの存在により、実は深いネットワークであっても、浅いネットワークの機能を包含することが知られています。浅いネットワークのみで学習できるようなデータの場合には、浅い段までのパラメータのみが重点的に学習され、残りはショートカット経路を通り、実質的に意味をもたないよというようなイメージです。

MEMO: 残差ネットワークに関する文献

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep residual learning for image recognition. Computer Vision and Pattern Recognition (CVPR), 2016.

[3]厳密にいうと図 6.4(a) のように、あるショートカット終了から次のショートカット開始までの間に、ReLU をはさんでいます。なぜこのようにしているのかは筆者にはよくわかりません。

デュアルネットワークの規模と計算量

従来版アルファ碁のポリシーネットワークの場合と同様に、デュアルネットワークの畳み込み層の足し算回数とパラメータの個数を計算してみましょう。簡単のため、ここではデュアルネットワークを 3x3 サイズの畳み込み計算 39 層であるとみなして計算します。この仮定の下では、

- 畳み込みの足し算回数: $19 \times 19 \times 3 \times 3 \times 256 \times 256 \times$ (層の数:39) = 約 83 億回
- フィルタ重みパラメータの個数: $3 \times 3 \times 256 \times 256 \times$ (層の数:39) = 約 2500 万個

となります。足し算回数、パラメータ個数共に、従来版アルファ碁で使われたフィルタ 192 枚の SL ポリシーネットの約 5 倍となっています。

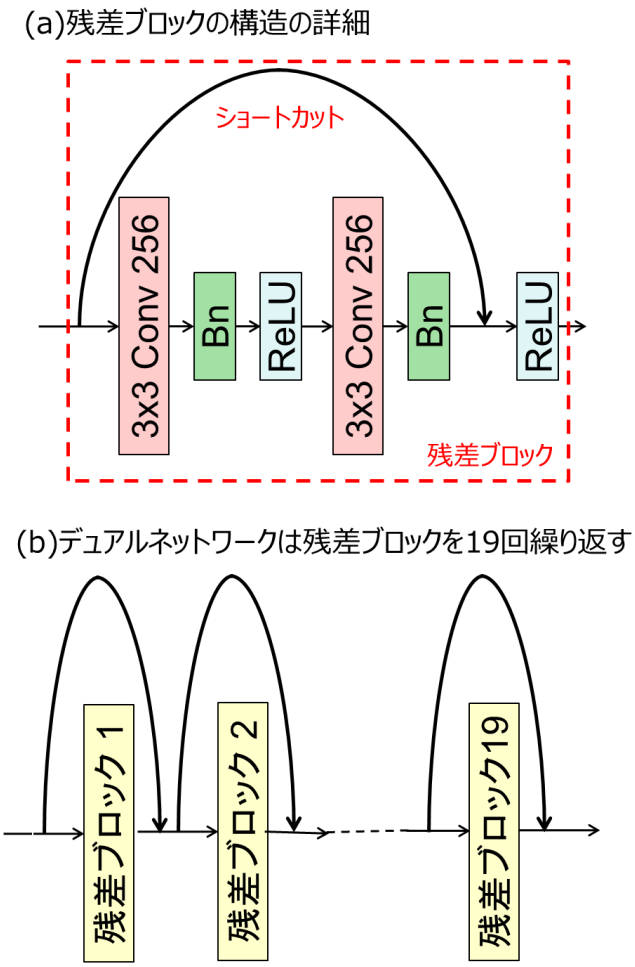


図 6.4: デュアルネットワークの残差ネットワーク部分の詳細。

6.2.2 デュアルネットワークの学習

ここでは、入力局面 s^k と正解データ π^k, z^k の組からなる M 個の学習データ $\{(s^k, \pi^k, z^k)\}_{k=1}^{M}$ が得られたとして、デュアルネットワークの学習手法を説明しましょう。

なお学習手法の詳細に興味のない読者は、この節は読まなくてもかまいません。

ここで正解データの一つ π は、局面 s において、各手 a が打たれる確率を、A 次元ベクトルで表したものです。つまり正解手を a^* とする場合は、a^* 番目の成分だけ 1 とし残りを 0 とした A 次元ベクトル $\pi = \{\pi_a\}_{a=1}^{A}$ で表します。ここで A は手の候補の総数を表します。具体的には、全ての交点の合計 361 に、何もしないパスを加えた 362 となります。

次にもう一つの正解データ z は、最終的な勝ち負け (黒勝ちならば $+1$, 白勝ちならば -1) を表します。

また重みパラメータを θ とするデュアルネットワーク $f_\theta(s)$ は局面 s を入力に取り、手 a が出力である確率 $p(s,a)$ と黒の勝率 $v(s)$ とを出力します。したがって出力の組 (p,v) と正解データの組 (π,z) との誤差を最小化することが目的となります。このための損失関数 L を、アルファ碁ゼロでは下記の式で定義しています。

$$(p,v) = f_\theta(s) \tag{6.1}$$

$$L_\theta = \sum_{k=1}^{M}\Big\{(z^k - v^k)^2 - \sum_{a=1}^{A}\pi_a^k \log p_a^k\Big\} + c\sum_i \theta_i^2 \tag{6.2}$$

このうち $(z^k - v^k)^2$ の部分は、z と v の間の二乗誤差関数であり、これは従来版アルファ碁のバリューネットの損失関数と同じです。次に $\sum_{a=1}^{A}\pi_a^k \log p_a^k$ の部分は、$\pi = \{\pi_a\}_{a=1}^{A}$ と $p = \{p_a\}_{a=1}^{A}$ の間の交差エントロピー損失関数を表し、これは従来版アルファ碁のポリシーネットワークの損失関数と同じです。最後に $\sum_i \theta_i^2$ は、パラメータ θ が過学習することを防ぐための正則化項であり、ニューラルネットの場合は過重減衰 (weight decay) とも呼ばれます。正則化は、過学習を防ぎ、データに合った単純なモデルを作るための手法です。c は、この正則化項と損失関数のバランスを決定するハイパーパラメータであり、ネイチャー 2017 論文では、$c = 10^{-4}$ に設定されています。この損失関数の定義はオーソドックスなものです。

この L_θ を θ で微分した $\frac{\partial L_\theta}{\partial \theta}$ を用いて、パラメータ更新式を簡単に書くと

$$\theta \rightarrow \theta - \alpha \cdot \Delta\theta \tag{6.3}$$

$$\Delta\theta = \frac{\partial L_\theta}{\partial \theta} \tag{6.4}$$

となります。α はこれまで同様学習率です。実際には α をパラメータごとにうまく調整する、モメンタム SGD という手法が使われています。

デュアルネットワークの学習の効果

従来版アルファ碁同様に、3000 万局面を使って教師あり学習した結果、強いプレイヤの手との一致率は 60.4% となることが示されており、最高 57.0% であった従来版アルファ碁の一致率よりも高い値となっています。残差ブロックを導入し、かつネットワークの深さを深くした効果と言えるでしょう。実際には、強化学習を使って、デュアルネットワークのパラメータを学習をしていくわけですが、その手法については 6.4 節で述べることにします。

デュアルネットワークの学習部を書く

図 6.5 に、従来版アルファ碁のポリシーネットワークの場合と同様に、デュアルネットワークを Chainer で記述した場合のネットワーク定義部の例を示します。細かい説明は省略しますが、Chainer

を使えば、これまで説明した比較的複雑な構造を、比較的簡潔に記述できるということだけ指摘しておきます。

(a)各層のサイズと形状の定義部

```
def __init__(self, train=True):
    super(CNN, self).__init__(
        conv0 = L.Convolution2D(17, 256, 3, pad=1),
        conv1 = L.Convolution2D(256, 256, 3, pad=1),
        conv2 = L.Convolution2D(256, 256, 3, pad=1),
         …
        conv38 = L.Convolution2D(256, 256, 3, pad=1),

        bn0=L.BatchNormalization(256),
         …
        bn38=L.BatchNormalization(256),

        conv_p1 = L.Convolution2D(256, 2, 1),
        bn_p1   = L.BatchNormalization(2),
        fc_p2   = L.Linear(19*19*2, 19*19)
,
        conv_v1 = L.Convolution2D(256, 1, 1),
        bn_v1   = L.BatchNormalization(1),
        fc_v2   = L.Linear(19*19, 256),
        fc_v3   = L.Linear(256, 1),)
```

(b)ネットワークの接続の定義部

```
def __call__(self, x):
    h0 = F.relu(self.bn0(self.conv0(x)))
    h1 = F.relu(self.bn1(self.conv1(h0)))
    h2 = F.relu(self.bn2(self.conv2(h1)) + h0)
    h3 = F.relu(self.bn3(self.conv3(h2)))
    h4 = F.relu(self.bn4(self.conv4(h3)) + h2)
         …
    h37 = F.relu(self.bn37(self.conv37(h36)))
    h38 = F.relu(self.bn38(self.conv38(h37)) + h36)

    #policy outpt
    h_p1  = F.relu(self.bn_p1(self.conv_p1(h38)))
    out_p = self.fc_p2(h_p1)

    #value output
    h_v1  = F.relu(self.bn_v1(self.conv_v1(h38)))
    h_v2  = F.relu(self.fc_v2(h_v1))
    out_v = F.tanh(self.fc_v3(h_v2))
    return out_p , out_v
```

残差ブロック

図 6.5: デュアルネットワークを Chainer で記述した場合のネットワーク定義部。

6.2.3 アルファ碁ゼロのディープラーニングのまとめ

本節で説明した通り、ディープラーニングは、人間の直観を代替する機能を持つと言えるでしょう。技術的には、機械学習では従来重要とされた特徴設計のプロセスが不要になったという点が大き

いです。囲碁の場合、局面評価関数作成のための特徴量設計がとても難しかったのですが、アルファ碁ゼロは、ついに石の配置と履歴の情報だけから、局面評価する方法を生み出しました。人間の大局観に相当する、囲碁の局面評価関数が作れないという課題は見事に解決されたのです。

6.3　アルファ碁ゼロにおけるモンテカルロ木探索

ここでは精度の高いデュアルネットワークが得られたとして、このデュアルネットワークを活用するモンテカルロ木探索 (MCTS) について述べます。ここでは、アルファ碁ゼロの MCTS と**従来型のモンテカルロ木探索手法** (MEMO 参照) の違いを中心に説明します。

MEMO: 従来型のモンテカルロ木探索手法

MCTS は、ランダムシミュレーションを繰り返し、最終的に、ルート局面で最もシミュレーション回数の大きい手を選択する手法でした。この手法は、多腕バンディット問題で使われる UCB アルゴリズムをゲーム木探索に拡張した手法であり、各シミュレーションでは、そのノードの手番から見た、楽観的な勝率予測値 (勝率 + バイアス) の大きい手を選択 (Selection) して木を降りていきます。そして、リーフノード (末端局面) に至った後は、プレイアウトを実行して、そのノードの勝率を評価 (Evaluation) し、結果を更新 (Backup) しながら再びルートノードまで昇っていきます。さらにシミュレーション回数が一定値を超えたノードは子ノードを展開 (Expansion) します。

この選択、展開、評価、更新からなるシミュレーション処理を繰り返すことで、結果として、より重要な手を深く展開することができることが MCTS の特長でした。

これに対し、従来版アルファ碁では、選択処理において、勝率の評価にバリューネットワークとプレイアウトの勝率評価の重み和を、バイアスの評価にポリシーネットワークの「次の一手予測確率」を用いているという特長がありました。

6.3.1　アルファ碁ゼロのモンテカルロ木探索の概要

アルファ碁ゼロの MCTS と、従来版アルファ碁の MCTS の最大の違いは、プレイアウトと呼ばれる「乱数を元に終局まで手を進め勝ち負けを得るプロセス」がないことです。アルファ碁ゼロでは、従来の複数回プレイアウトの代わりに、1 回だけデュアルネットワークを計算し、勝率を予測しています。デュアルネットワークによる勝率予測の精度が上がったため、プレイアウトが不要になったということだと思います。結果として、シミュレーション 1 回で勝率が得られるので、シミュレーション 1 回ごとに新しいノードを 1 個展開できるようになりました。

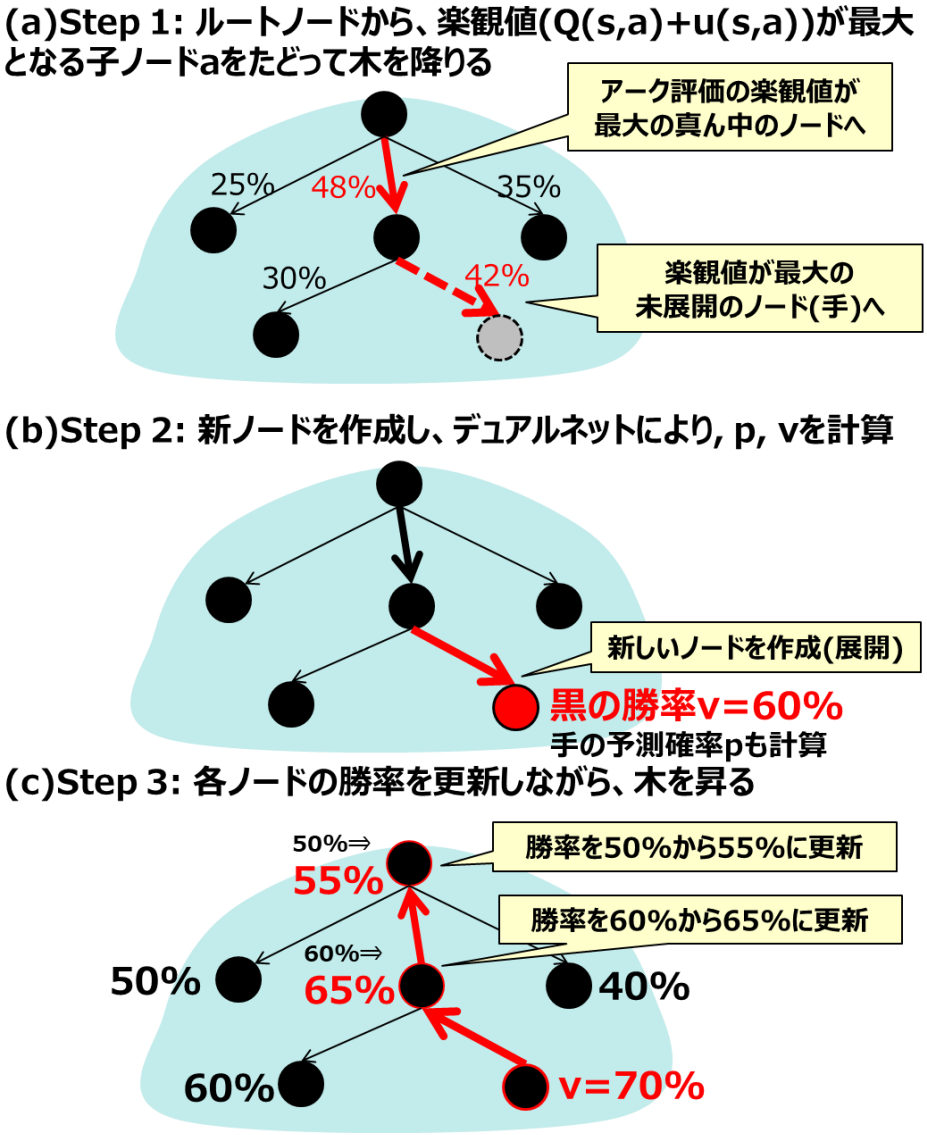


図 6.6: アルファ碁ゼロのモンテカルロ木探索。(a) ルートノードから、アーク評価の楽観値 $(Q(s,a)+u(s,a))$ が最大となる手 a をたどって木を降りる。(b) 新しいノードを作成し、 デュアルネットワークにより p, v を計算。(c) 各ノードの勝率を更新しながら木を昇る。

6.3.2　モンテカルロ木探索のフローチャート

次にモンテカルロ木探索の処理の詳細を見てみましょう。

アルファ碁ゼロのモンテカルロ木探索では、図 6.7 のフローチャートに示すような Step 1〜3 からなるシミュレーション処理を繰り返します。

Step 1(選択処理)

まずStep 1では、局面sにおいて、下記で計算されるアーク評価値[4]$Q(s,a)+u(s,a)$が最大となる手aをたどって木を降りていきます。

$$Q(s,a) = \frac{W(s,a)}{N(s,a)} \tag{6.5}$$

$$u(s,a) = c_{puct}\cdot p(s,a)\cdot\frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)} \tag{6.6}$$

ここで$Q(s,a)$は勝率です。$u(s,a)$は、デュアルネットワークが出力する手aの予測確率$p(s,a)$と、バイアス$\frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}$の積となっています。バイアスは、$a$以下のシミュレーション回数が少ない内は大きな値となり、回数が多くなると小さくなってきます。つまり信頼区間の大きさを表しているとみなせます。したがって$u(s,a)$全体では、予測確率$p(s,a)$が大きい、もしくはa以下のシミュレーション回数が小さい内は大きな値となります。最後に、c_{puct}は、勝率$Q(s,a)$と$u(s,a)$のバランスを決めるパラメータを表します。以上の計算の内容は、勝率$Q(s,a)$が、デュアルネットワークの結果のみで評価されていることを除けば、従来版アルファ碁と全く同じです。

Step 2(展開・局面評価処理)

Step 2では、子ノードs'を展開し、デュアルネットワークf_θにより, $p(s',a)$, $v(s')$を計算します。従来版アルファ碁ではノード展開の頻度は、$n(=40)$シミュレーションに1回でしたが、アルファ碁ゼロでは、毎回ノードが作られます。

Step 3(更新処理)

Step 3では、ルートノードまでの途中の全てのノードsにおいて、勝率の合計$W(s,a)$とシミュレーション回数の合計$N(s,a)$を更新しながら、木を昇ります。

$$N(s,a) = N(s,a)+1 \tag{6.7}$$

$$W(s,a) = W(s,a)+v(s) \tag{6.8}$$

アルファ碁ゼロでは、勝率予測が1種類なため、更新処理もシンプルになっています。

この処理により、結果として$N(s,a), W(s,a)$には、それぞれ、局面sの手a以下のシミュレーション回数と、勝率の合計値が格納されることを確認ください。

結果、$W(s,a)/N(s,a)$により、当該局面sの手a以下に展開された全てのシミュレーションの勝率の平均が計算されます。MCTSの、重要な変化が重点的に展開される性質と合わせると、シミュレーションを重ねるほと、$W(s,a)/N(s,a)$による勝率の精度が高まっていくと考えられます[5]。

[4]評価という用語は紛らわしいため、選択処理による各手の評価は「アーク評価」とよぶことにします。一方、局面をする場合は、「局面評価」と呼びます。

[5]アルファ碁ゼロのMCTSでは、勝負がつくまでプレイアウトしているわけではなく、単にデュアルネットワークの勝率予測で近似しているので、従来型のMCTSのように、理論的に最善手に収束するといった性質はありません。

Step 4(最終的な手の選択処理)

アルファ碁ゼロでは、上記 Step 1〜3 のシミュレーション処理を N 回 (たとえば 1600 回) 繰り返した後に、ルート局面において、最もシミュレーション回数が多かった手を採用します (Step 4)。この点も従来版アルファ碁と同じです。

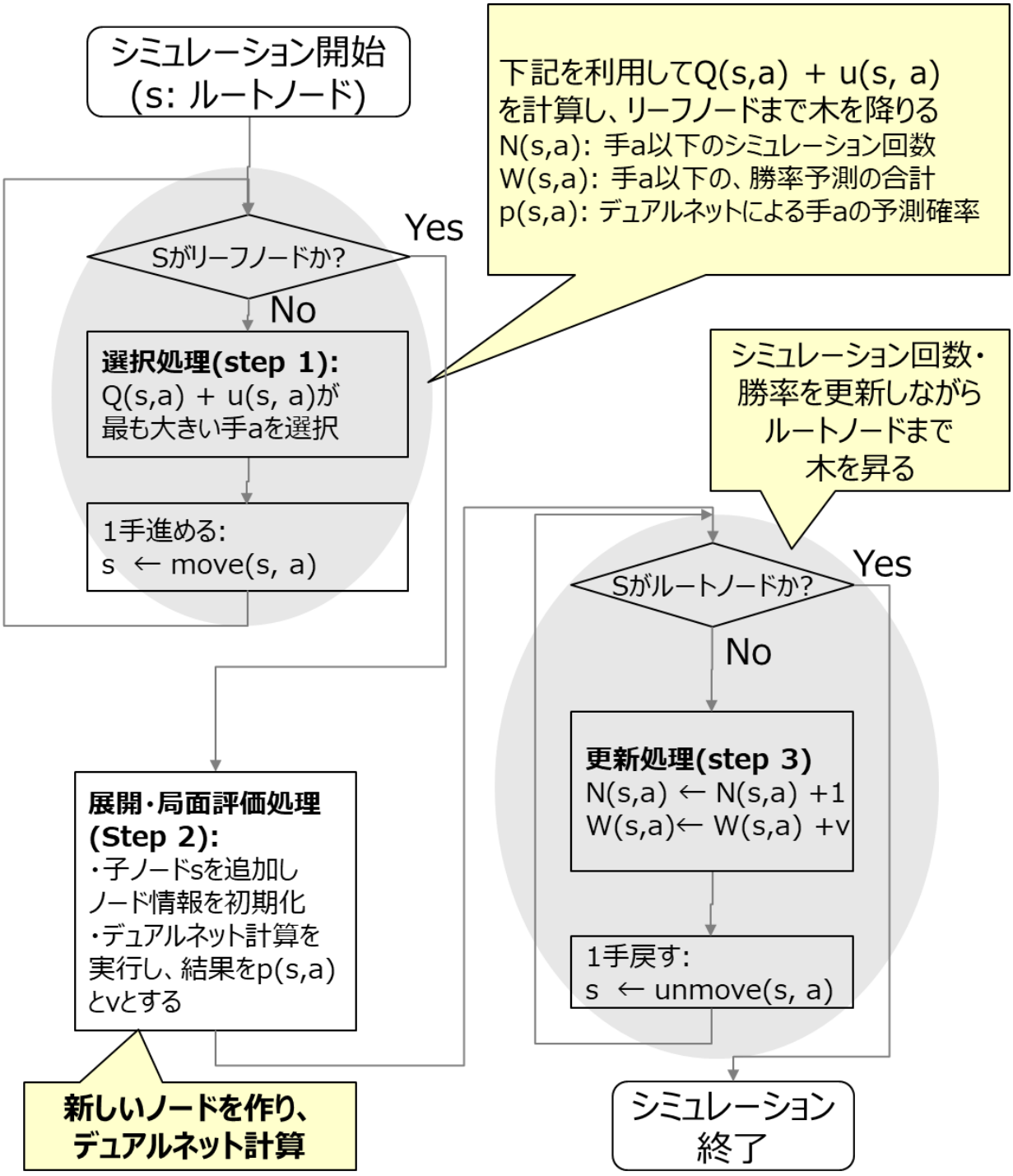


図 6.7: アルファ碁ゼロのモンテカルロ木探索のシミュレーション 1 回の処理に関するフローチャート。

6.3.3　アルファ碁ゼロのモンテカルロ木探索のまとめ

アルファ碁ゼロのモンテカルロ木探索では、デュアルネットワークの勝率予測の精度が向上したことにより、プレイアウトが不要になったことが最大のポイントです。また、このプレイアウトをなくしたことで、1 秒間に生成できるノード数が増えたことも大きなメリットになっています。アルファ碁ゼロでは、TPU という強力なハードウェア使用の効果もあり、4000 ノード/秒程度のノード生成が可能となります。結果として、かなり深く読めそうなことが分かります。

この探索手法は、もはや MCTS というよりも、従来型の将棋などの探索手法に近づいてきたイメージです。実際、アーク評価の楽観値が最も良い順に 1 個ずつノードを追加し、追加したノードを局面評価するということを繰り返しているだけなので、一種の最良優先探索とみなすことができます[6]。

つまり、アルファ碁ゼロの MCTS による先読み方法は、圧倒的な物量で広く深く探索していくというよりも、有望な枝のみを深く読んでいくような手法となっています。人間の思考方法に近づいてきたと言えるかもしれません。

[6] ちょっとマニアックですが、あるノードの評価値と楽観的な評価値を計算して探索をすすめる B*探索という手法に似ている気がします。

6.4　アルファ碁ゼロにおける強化学習

ここまで精度の高いデュアルネットワークが得られた場合の、アルファ碁ゼロの探索手法について説明してきました。後は、精度の高いデュアルネットワークをどのように得るかという問題が残ります。

図 6.8 に示すように、アルファ碁ゼロの強化学習は、最初はランダムプレイから始まります。ここから**セルフプレイ (自己対戦)**(MEMO 参照) を繰り返し、かつセルフプレイの結果を元に、デュアルネットワークのパラメータ θ を逐次更新していくことで、だんだん強くしていく方針です。この強化学習を実現するには、セルフプレイにおいて、より勝ち易い手を選びやすくなるようなパラメータ更新手法を生み出す必要があります。

MEMO: セルフプレイ (自己対戦)

セルフプレイとは、ここでは黒番の囲碁 AI と白番の囲碁 AI の自己対戦のことで、1 手目から勝敗が決まるまで、1 手ずつそれぞれの AI が交互に打つことを言います。本章では、セルフプレイの目的は二つあります。

1 つ目の目的は、別の AI を用いた自己対戦を繰り返すことで、どちらの AI が強いかを決定することです。これは対戦数を増やしていって、勝敗をカウントするだけで評価することができます。

セルフプレイの 2 つ目の目的は、強化学習のための自己評価の手法としての役割です。ある AI を用いた場合の勝敗を得ることで、この AI をより勝ち易くするためのパラメータ更新の方向を得ることが目的となります。たとえば、勝った方の手をより打ちやすくするように、負けた方の手をより打ちにくくするように、強化学習することが考えられます。アルファ碁ゼロでは、後述するように、もう少し複雑な強化学習の枠組みが使われています。

ではどのような強化学習を使えばよいでしょうか? 最もシンプルには、従来版アルファ碁において、SL ポリシーネットワークから、RL ポリシーネットワークを作ったように、セルフプレイの勝敗を元に**方策勾配法** (MEMO 参照) によりパラメータを行うということが考えられます。ただ、さすがのグーグルの計算機リソースをもってしても、この直球勝負の方針では難しかったのかもしれません。

MEMO: 方策勾配法

方策勾配法は、強化学習における、方策ベースの学習手法の一つです。Q 学習のように行動価値関数を使うのではなく、方策そのものを確率的な関数 (方策関数) として表現し、行動を確率的に決定します。この方策関数を更新することにより学習を進めます。従来版アルファ碁のポリシーネットの強化学習に使われたのは、この方策勾配法でした。方策勾配法については、3.4 節で説明しています。

実は、アルファ碁ゼロでは、これまであまり見たことのない、巧みな強化学習手法が使われています。本節では順を追って解説していきます。

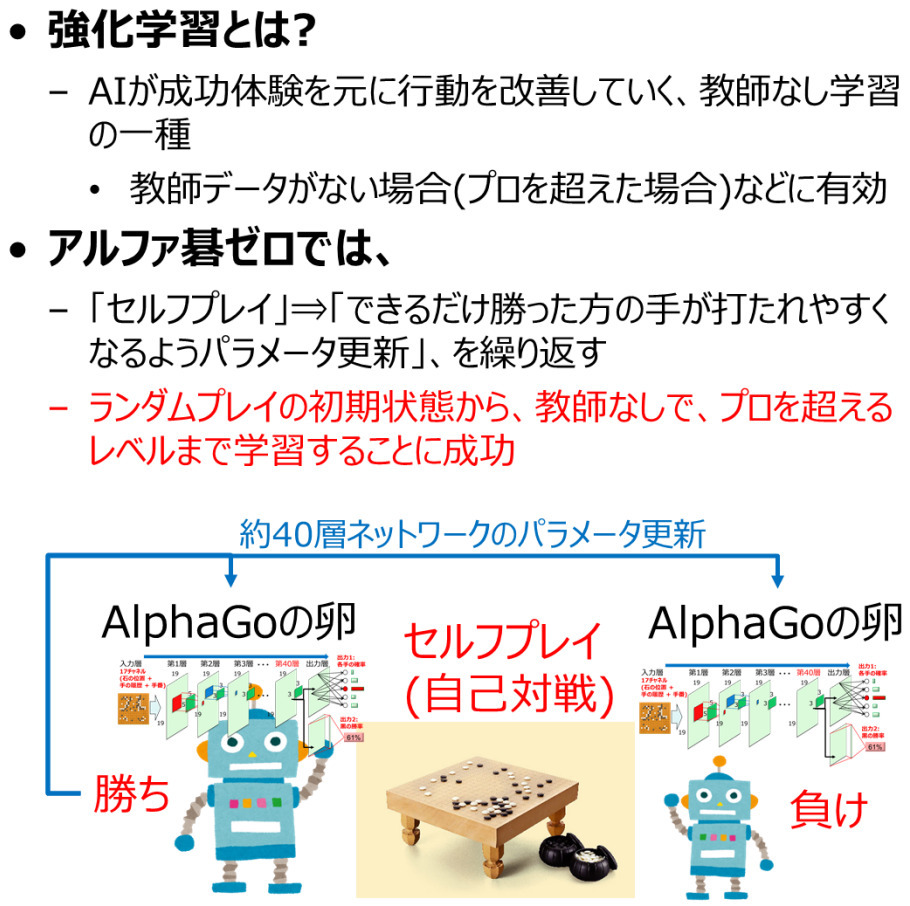


図 6.8: アルファ碁ゼロの強化学習手法。

6.4.1 アルファ碁ゼロの強化学習手法

図 6.9 にアルファ碁ゼロの強化学習のフローチャートを示します。アルファ碁ゼロの強化学習は、セルフプレイ部, 学習部, 新パラメータ評価部の 3 つの部分からなっています。

最初にセルフプレイ部では、現在一番強いパラメータ (暫定最強パラメータ)θ^* によるデュアルネットワーク f_{θ^*} を利用したセルフプレイが行われます。次に、学習部では、過去のセルフプレイの情報 (具体的には z と π) を用いてパラメータを更新し、新しいパラメータ θ' を得ます。最後に、新パラメータ評価部では、暫定最強なデュアルネットワーク f_{θ^*} と、今回得られた新パラメータ θ' による $f_{\theta'}$ による対戦を実施し、$f_{\theta'}$ 側が十分勝ち越した場合には、θ^* を新パラメータ θ' に置き換える処理を行います。

暫定最強パラメータ θ^* の変化に注目してみてみると、Step 1 でランダムに初期化された θ^* は、Step 3 のパラメータ更新部で更新され、更新結果による強さが有意だと認められた場合は新しい θ^* として採用されます (Step 4)。強化学習によりアルファ碁が強くなっていく現象は、この θ^* がどんどん強力なパラメータへと更新されることで実現されます。

なお余談ですが、パラメータ θ^* の更新のプロセスは、人手で開発する場合とよく似ています。人手で開発するときは、新たなアイデアに基づきコードを書き換えたうえで、暫定最強版と新アイデア

実装版による対戦を実施し、新アイデア側が十分勝ち越した場合は、暫定最強パラメータを置き換えます[7]。アルファ碁ゼロの強化学習の枠組みは、コードの書き換え処理をデュアルネットワークのパラメータ更新に落とし込むことで、人間が様々なアイデアを試行錯誤するプロセスを自動化したと見ることもできます。

ここで図 6.9 では、分かり易さのためシーケンシャルに記述していますが、実際には、セルフプレイ部と、パラメータ更新部・新パラメータ評価部とは、非同期並列に実行されており、パラメータ更新部・新パラメータ評価部は、セルフプレイ部が 2.5 万回のセルフプレイを終わるのを待つのではなく、逐次、学習・評価を行っています。

[7]さらに余談ですが、開発が煮詰まってくると、新アイデアのほとんどは、十分勝ち越すことができなくなります。たまに勝ち越した時の喜びは何事にも代えられません。

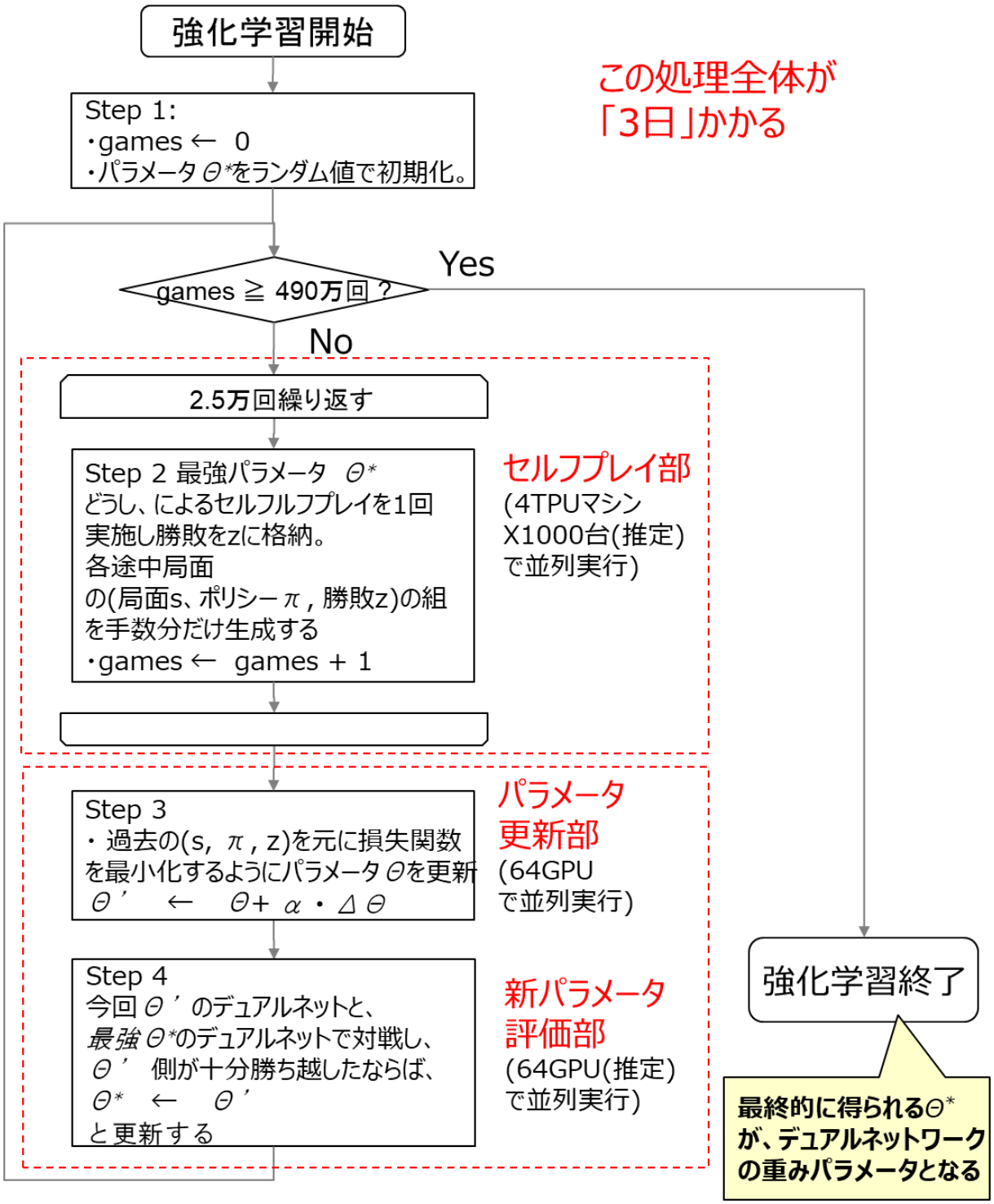


図 6.9: アルファ碁ゼロの強化学習のフローチャート。

強化学習におけるセルフプレイ部の処理

セルフプレイ部の目的は、デュアルネットワーク f_θ^* を利用したモンテカルロ木探索に基づき、高品質かつ多様な棋譜を、できるだけ大量に生成することです。

セルフプレイは、実際の探索時と同じ処理であることが望ましいので、基本的には、6.3 節で述べたのとほぼ同じやり方で探索します。この場合、1 手あたり 0.4 秒かけるとすると、1600 回のシミュレーションが可能となるそうです。

なお通常 MCTS では、最終的な手の選択 (図 6.7 の Step 4) は、最もシミュレーション回数の大きい子ノードの手を選ぶのですが、セルフプレイでは、棋譜の多様性が重要なため、初手〜30 手目までに限り、シミュレーション回数に比例した確率で子ノードの手を選択しています。また 30 手目までに限らず、最終的な手の選択では、乱数を用いて、シミュレーション回数が最大のノード以外も選択されるような工夫も施しています[8]。

一方、セルフプレイを短い手数で終わらせるための工夫として、勝率が 5%以下になったら投了とみなすという判断が入っています[9]。

以上の結果から、1 手進めるごとに、各ルート局面 s に対する、各手 a がシミュレーションされた回数 $N(s,a)$ と最終的な勝敗 z を保存しておきます。最終的にどの手を生成したかだけではなく、ルートノードで各手をシミュレーションした回数を保存し、学習に使うというところに、これまでにない発想があります。

強化学習におけるパラメータ更新部の処理

パラメータ更新部の目的は、セルフプレイの結果に基づき、現状のデュアルネットワークのパラメータを、より優れたパラメータに更新することです。

直前の 50 万回のセルフプレイの中から、ランダムに $A = 2048$ 個の学習データを取り出します。

6.2 節で見たように、デュアルネットワークのパラメータ学習には、次にどの手を選ぶかの正解データ π と勝率の正解データ z とが必要でした。ここで注目すべきは、正解データ π として、6.2 節で述べたように正解手 1 個だけを 1 とするのではなく、複数の候補を上げ、それらに合計が 100%になるような、確率値を付けたものを利用する方針を採っていることです。

学習の結果欲しいのは、MCTS の手を展開する際の事前確率 p ですので、どの手を選択するかの 0-1 を正解とするのではなく、各手の確率の分布を正解としておいた方がよい、という発想です。正解として複数の候補手の分布 π を使えば、単純にどの手を選択したかの 0-1 を使うよりも情報量が増えるので、学習が進みやすくなるのだと思います。

この各手の確率 π_a の作り方としては、MCTS の結果を最大限活用し、シミュレーション回数の情報 $N(s,a)$ を利用して、

$$\pi_a = \frac{N(s,a)^{1/\tau}}{\sum_b N(s,a)^{1/\tau}} \tag{6.9}$$

のように計算しています。このためにセルフプレイ部では $N(s,a)$ を保存しておいたのです。

ここで τ は**温度パラメータ**と呼ばれる、手のバラつきを制御するパラメータになっています。$\tau = 0$ の極限を考えると $N(s,a)$ が最大となった手 a を 100%選ぶことになり、これは MCTS の考え方と同じです。一方 $\tau = 1$ とすると、$N(s,a)$ の大きさに比例して、手 a を選ぶことになります。アルファ碁ゼロでは、実は、この τ を適宜調整しながらシミュレーションを実行し、得られる棋譜の多様性を調整してます。

[8]詳細は省略しますが、多項分布の共役事前分布であるディリクレ分布を利用しています

[9]単純には、デュアルネットワークの勝率予測値を使えば良いように思えますが、より慎重な判断をしています。

強化学習における新パラメータ評価部の処理

新パラメータ評価部の目的は、セルフプレイおよびパラメータ更新処理で得られた新しいパラメータ θ' が、元のパラメータ θ^* よりも優れているかを判定することです。パラメータ更新処理を 1,000 回実行するごとに、以下の処理を行います。

具体的には、今回得られたデュアルネットワーク $f_{\theta'}$ と、従来最強のデュアルネットワーク f_{θ^*} との間で 400 回自己対戦 (セルフプレイ) させます。セルフプレイ部の場合と同様、1 手進めるために 1600 回のシミレーションを実行しています。

結果、$f_{\theta'}$ 側が f_{θ^*} に対し、220 勝以上勝った場合は、$\theta^* \rightarrow \theta'$ と暫定最強パラメータを更新します[10]。

6.4.2 強化学習の計算時間

次にアルファ碁ゼロの強化学習の計算時間について詳しく見ていきます。

前節で説明したように、アルファ碁ゼロの強化学習は、セルフプレイ部、パラメータ更新部、新パラメータ評価部、の 3 つから構成されますが、強化学習に要する時間のほとんどは、セルフブレイ部に費やされます。ネイチャー 2017 論文によるとセルフプレイにおいて 1 手あたり 1600 シミュレーションを実行し、その計算時間は 0.4 秒程度だそうです。

この場合、490 万局のセルフプレイには、1 局のセルフプレイが 150 手で終了すると仮定すると

$$\begin{aligned} 0.4(\text{秒/手}) \times 150(\text{手/局}) \times 490\,\text{万 (局)} &\sim 2.9\,\text{億秒} \\ &\sim 3400\,\text{日} \\ &\sim 9.3\,\text{年} \end{aligned} \tag{6.10}$$

と 10 年近くかかることになります。これが 3 日で終わったということは、1000 台近い並列化を行ったことが推定されます。

一方、1600 シミュレーションが 0.4 秒で終わるというスピードも、ネイチャー 2016 論文の水準に比較し圧倒的です。1 回のシミュレーションの中で一番重い処理は、デュアルネットワークを計算する部分で、シミュレーション 1 回あたり 1 回実行されます。6.2 節でデュアルネットワークの計算量が、従来版アルファ碁のポリシーネットワークと比べると約 5 倍の計算量 (積和計算の量) となることを説明しました。また、従来版アルファ碁においてポリシーネットワークの前向き計算が 5 ミリ秒程度かかっていたことを利用しましょう。その場合、1600 回の計算には、

$$5.0(\text{ミリ秒/シミュレーション}) \times 5 \times 1600(\text{シミュレーション}) = 40\,\text{秒} \tag{6.11}$$

かかることになります。

この計算をアルファ碁ゼロでは 0.4 秒と、100 倍以上も速く計算できるというのはどういうことでしょうか?

これは想像に過ぎませんが、グーグルクラウドで使用可能な TPU を 4 個搭載したマシンを使っただろうというのが、私の仮説です。TPU は従来の GPU に比べて最大 30 倍程度高速に処理できると言われています。30 倍を 4 個使うと、上記の差である 100 倍とほぼ一致することになります。

[10] これは、(大雑把な言い方をすると) 約 98%の確率で強くなったということに相当します。

つまり、まとめるとアルファ碁ゼロの強化学習では、「TPU を 4 個搭載したマシン」を「1000 台程度並列」することで、計算が 3 日で終わったのではないかということです。

ちなみに、この計算を CPU が 1 個しかないマシンで行うとどうなるでしょうか? 従来型の GPU は、CPU の 20 倍高速でした。TPU は、従来型の GPU の 30 倍程度高速で、これが 4 個搭載されたマシンを 1000 台使ったということです。全て掛け合わせると、

$$\begin{aligned} 3(\text{日}) \times 20(\text{倍}) \times 30(\text{倍}) \times 4(\text{個}) \times 1000(\text{台}) &= 720\text{ 万日} \\ &\sim 1.97\text{ 万年} \end{aligned} \tag{6.12}$$

となり、つまり 2 万年近く要するということになります。グーグルが 3 日でできたと言っている計算量が、実はとてつもなく膨大な計算量であったということが、この概算からも分かっていただけるかと思います。

6.4.3 アルファ碁ゼロの強化学習は何をやっているのか?

ではどのような仕組みで、デュアルネットワーク f_θ のパラメータ θ は、強いパラメータになっていくのでしょうか? 実は、局面評価関数の質を上げれば探索の質が上がる、深さ制御の質を上げれば探索の質が上がる、という**ゲーム木探索の性質 (MEMO 参照)** をうまく利用しています。ここでは、そのカラクリを、勝率予測部と次の一手予測部とに分けて考えてみましょう。

MEMO: ゲーム木探索の性質

ゲーム木探索で重要なことは、端的に言えば、次の 2 点に集約されます。

- いかにして重要な手を深く読むか
- リーフノード (末端局面) をいかに正確に評価するか

前者に対しては、重要な変化を深く読むための手法が、後者に対しては、勝率を正確に予測できる「評価関数」が重要となります。ゲーム木探索においては、より深い探索が重要であると同時に、評価関数の精度も同じように重要です。

たとえば両極端を考えてみましょう。まず完璧な評価関数があれば、深い読みは必要ありません。この場合、1 手先まで読んで勝率が 100%となる手を選べば良いのです。

一方、評価関数がまったくあてにならない乱数だとすると、探索結果は乱数だけに左右され、どんなに深く探索しても、まったく価値のない結果が得られるだけです。

まず、分かり易い勝率予測部のパラメータから考えます。図 6.10(a) に示すように、まずパラメータ更新部における局面 s の処理に着目すると、デュアルネットワーク f_θ の勝率予測 v を、ルートノードの勝率 z に近づけるように新たなパラメータ θ' を学習します (図 6.10(a-1))。すると、次のセルフプレイのシミュレーションでは、新しいデュアルネットワーク $f_{\theta'}$ により、リーフノード s の勝率予測 v を計算します。したがって、この v の精度は、前回のシミュレーションよりも高まっていることが期待できます (図 6.10(a-2))。一方、最終的に計算されるルートノードの勝率 z_{new} は、リーフノードの勝率の平均値として決定されますので、z_{new} の精度も前回より高まると考えられます (図 6.10(a-3))。すると次のパラメータ更新では、精度の高まった z_{new} に合わせるように更新することとなります。こ

のようにパラメータ更新とセルフプレイのプロセスの繰り返しには、正のフィードバック構造があり、θ の精度はどんどん高まっていくことが期待できます。

次に、「次の一手予測部」のパラメータに関してですが、これも同様の正のフィードバック構造があることを見ていきましょう。図 6.10(b) に示すように、まずパラメータ更新部におけるある局面 s の処理に着目すると、デュアルネットワーク f_θ の次の一手予測確率 p を、ルートノードの $N(s,a)$ の分布 π に近づけるように新たなパラメータ θ' を学習します (図 6.10(b-1))。すると、次のセルフプレイのシミュレーションの選択処理では、$p = f_{\theta'}$ に基づくバイアスを用いるために、より重要な展開をより深く読むことが期待できます (図 6.10(b-2))。一方、最終的に計算されるルートノードの $N(s,a)$ の分布 π_{new} は、より重要な展開を深く読んだ結果として得られるため、π_{new} の精度よりも高まると考えられます (図 6.10(b-3))。このように「次の一手予測」の観点からも、正のフィードバック構造があり、θ の精度はどんどん高まっていくことが期待できます。

つまりこれらは、局面評価関数の質を上げれば探索の質が上がる、深さ制御の質を上げれば探索の質が上がる、という探索の基本原理をうまく利用しているわけです。特に、深い探索の結果得られる評価値を再び局面評価関数の学習対象とすると、見かけの探索深さがどんどん深くなっていきそうな気がします。

本当に、このような正のフィードバックは生じるのでしょうか? 実際のところ、以上の議論は仮説にすぎず、試してみないことには、何とも言えません。場合によると、勝率予測パラメータの更新が誤った方向に進んでしまい、勝率の低い局面を高いと誤認識してしまう可能性があります。また次の一手予測パラメータも、重要でない局面を深く探索するようになってしまう可能性もあります。アルファ碁ゼロの開発者達も、様々な仮説・アイデアを試す中で、今回の強化学習のフレームワークに到達したのだと思います。

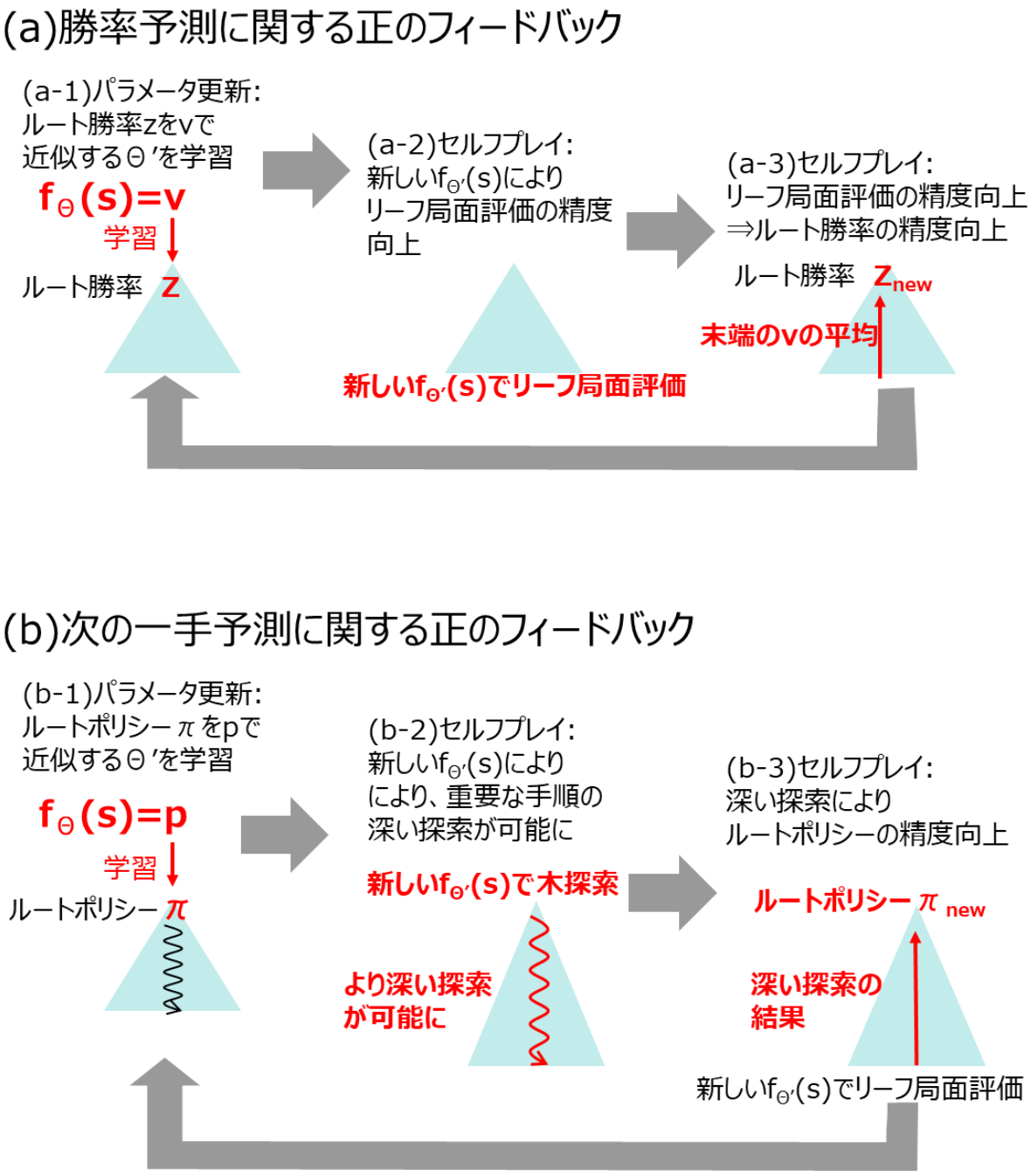


図 6.10: アルファ碁ゼロの強化学習における二つの正のフィードバック構造

6.4.4 強化学習の効果

アルファ碁の強化学習の効果については、ネイチャー 2017 論文に挙げられていた図 6.11 を見るのが良いでしょう。ランダムプレイの初期状態は、**イロ・レーティング** (MEMO 参照) でいうと −3500 点程度に相当するようです。ここから強化学習を進め、開始から 24 時間後には、3000 点程度のプロレベルに到達しています。さらに開始から 36 時間後には、イ・セドル九段と対戦した時点のアルファ

碁 (後述の AlphaGo Lee) を超え、開始から 72 時間後には人類最強レベルである 4500 点程度に達したとのことです。

MEMO: イロ・レーティング

複数のプレイヤの勝敗を元に、各プレイヤの実力を点数付けして表す手法です。チェスの世界では標準的に使われており、100 点差の場合に強いほうが 64%勝つ ようなモデルとなっています。

チェスでは 1200〜1400 点が初級者、1400〜1800 点が中級者、1800〜2000 点が上級者の目安となっています。2017 年 5 月現在、世界一のプレイヤとされるマグナス・カールセンのレーティングは 2800〜2900 点程度であり日本一のチェスプレイヤである羽生善治のチェスのレーティングは 2400 点程度です。羽生善治は、将棋では言わずと知れたトップ棋士の 1 人ですが、実は、チェスでも日本一のプレイヤです。

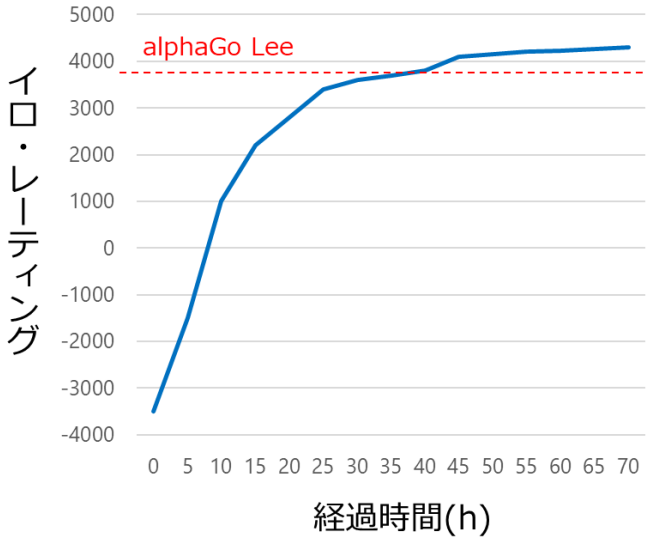


図 6.11: アルファ碁ゼロの強化学習の結果。ネイチャー 2017 論文の図を一部改変。

6.4.5 アルファ碁ゼロの強化学習のまとめ

ゲーム AI に強化学習を適用するメリットとして、教師あり学習の場合必須となる訓練データが必要ないということが挙げられます。そのため、そもそもプレイデータを得られにくいゲームや、既に強くなり教師データが作りにくくなった AI に対しては有効な技術となります。一方、強化学習を適用する別のメリットとして、何の知識も持たない状態から、人並みの知識や、これまで知られていなかった新たな知見を獲得できることが挙げられます。

以前から囲碁や将棋においては、従来版よりも少し強い AI を作るための強化学習手法や、何の知識もない状態から少し強くなる (たとえば将棋の駒価値を学習する) ような、強化学習手法は知られていました。これに対し、アルファ碁ゼロは、知識を全く持たない状態から、プロ棋士をはるかに超えるレベルまで「一気に」強くするという、極めて強力な強化学習フレームワークを作りあげました。また強化学習の中で得られた新たな知見の例として、ネイチャー 2017 論文は、これまで知られ

ていなかった新たな定石をアルファ碁ゼロが生み出したことについても触れています。アルファ碁ゼロは、「経験に学ぶ」強化学習技術の集大成と言えるでしょう。

最近は、将棋 AI でも強化学習がうまくいっているそうです。セルフプレイの勝敗に合わせるように局面評価関数を学習させる手法などが知られており、駒価値だけ与えた状態を起点として最強レベルの将棋 AI を作ったという報告 (MEMO 参照) もあります。またアルファ碁で今回使われたような手法を使うと、新しいゲームに対する攻略法を自ら生み出す AI を作ることができるようになるかもしれません。現時点では膨大な計算量を要していますが、計算量の問題が解決されれば、ゲーム開発の実装工数の削減や、実問題の解決など応用の裾野が広がっていくかもしれません。

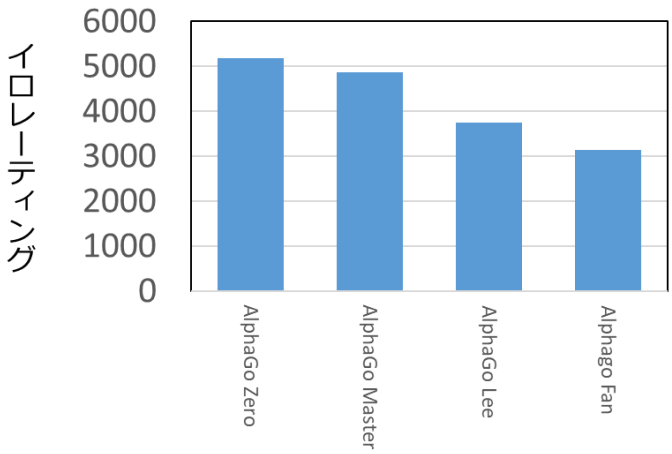
MEMO: 将棋の強化学習に関するウェブページ
「人間の棋譜を用いずに評価関数の学習に成功」,
Webpage: http://yaneuraou.yaneu.com/2017/06/12/ (Last access: 2017/11/4)

6.5　アルファ碁ゼロの強さ

最後にアルファ碁ゼロの強さの変遷についてネイチャー 2017 論文を元に触れてみましょう。ネイチャー 2017 論文では、アルファ碁の進化の過程を以下の 4 段階に分け、これら 4 つを、1 手 4 秒以内のルールで、連続対戦させることで、強さの指標であるイロ・レーティングを算出しています。

- 従来版アルファ碁 (AlphaGoFan)
- イ・セドル戦版 (AlphaGoLee)
- マスター (AlphaGoMaster)
- ゼロ (AlphaGoZero)[11]

(a)アルファ碁の4バージョンのイロ・レーティング



(b)アルファ碁の4バージョンの詳細

	対戦成績	使われている技術	対戦に用いたHWスペック
AlphaGo Fan (従来版アルファ碁)	2015年10月に Fan Hui 二段に勝利	ネイチャー2016論文(本書5章までで説明したもの)	176GPU, 48 TPU
AlphaGo Lee	2016年3月に イ・セドル九段に勝利	AlphaGo Fanと下記の点で異なる ・ポリシーネットではなくアルファ碁の自己対戦により強化学習 ・14層より深いネットワークを利用	176GPU, 48 TPU
AlphaGo Master	2017年1月に 囲碁対戦サイトで 60-0と人間に圧勝	AlphaGo Zeroと下記の点で異なる ・MCTSでは、AlphaGo Fan同様の特徴量を用いたプレイアウトを利用	4 TPU
AlphaGo Zero (アルファ碁ゼロ)	AlphaGo Master に 89-11	ネイチャー2017論文(ただし残差ブロック39段のネットワークを利用)	4 TPU

図 6.12: アルファ碁の (a) 強さの変遷 (ネイチャー 2017 論文より引用) と (b) 各バージョンの概要

[11] ここでは、6.2 節で説明した 19 段ではなく、39 段の残差ブロックをもつデュアルネットワークが使われています。つまり全体で 80 層程度のネットワークになります。

図 6.12(a) に最終的に得られたイロレーティングと、4 バージョンの詳細と対戦時のハードウェア構成を、図 6.12(b) に対戦の結果得られたイロ・レーティングを示しています。

AlphaGoFan, AlphaGoLee については、当時の強さを再現するため、当時用いられたハードウェア構成をそのまま使っています。一方、AlphaGoMaster, AlphaGoZero については、4TPU マシン 1 台で動かしています。結果、AlphaGoZero のレーティングは、5185 点に達しています。これは人類最強プレイヤのレーティングをかなり高めに見積もって 4000 点であったとしても、1000 回に 1 回程度しか勝てない水準です。

6.6 まとめ

本稿では、アルファ碁ゼロで使われている、ディープラーニング、探索、強化学習の各技術について説明しました。これまでの説明により、いずれの手法も、これまでの囲碁 AI に関する研究や、従来版アルファ碁の技術や知見を利用したものとなっており、いきなりゼロから作られたものではないことが分かっていただけたかと思います。

最後に、アルファ碁の謳い文句となっている「たった 3 日で」「ゼロから」「1 台のマシンでも動く」の各ポイントについて、私の見解を述べることで本章の筆を置くことにします。

アルファ碁ゼロの強化学習はたった 3 日で終わったのか?

この 3 日間という数字は、私の仮説では、4TPU 搭載マシン 1000 台程度の環境を使った結果です。CPU を 1 個だけ搭載した普通の PC を使った場合、手元の計算では 20,000 年かかる計算量です。グーグルの計算リソースには脱帽するしかないですが、3 日というのは、簡単に計算できるという意味ではないことは強調しておきたいと思います。

アルファ碁ゼロは人の知識を使わずにゼロから学習したのか?

最終的に作られた強化学習フレームワークは、確かに、ほとんど囲碁の知識なしに動いています。ただしフレームワークの詳細を見ていくと、デュアルネットワークの構造の与え方、新しい MCTS の考え方、強化学習における訓練データ π, v の与え方などは、これまでの知見を元に巧みに設計されたものです。また論文に現れていない部分では、今回の手法に至るまでの様々な試行錯誤やパラメータチューニングの職人芸が含まれていると思われます。決して、アルファ碁ゼロが自らの力のみで強くなったわけではないという点は強調しておきます。

アルファ碁ゼロはマシン 1 台でも強いのか?

ここでのマシン 1 台は、4TPU 搭載マシンを指しています。これは CPU2400 個もしくは、GPU120 個程度の並列に相当する膨大な計算量になります。決して、我々が簡単に入手できるようなマシン上でもアルファ碁ゼロの強さを再現できるわけではないことを強調しておきます。

以上のように、多少宣伝先行な部分があるものの、ゲーム最大の難関の一つである「囲碁」が、強化学習により解決したということは事実です。ゲーム AI は、コンピュータハードウェアと、機械学習・強化学習技術の発展にともなって進歩してきましたが、その集大成であるアルファ碁の登場により、ゲームは、AI の最初のモチーフとしての役割を終えたのかもしれません。