強化学習を用いた評価関数の作成手法の信頼性の分析

嶽 俊太郎[1,a)]　金子 知適[2]

概要：ゲーム AI 分野において、自己対戦により強化学習を行って評価関数を作成する手法は、AlphaGo に代表されるように大きな成功を収めてきた. しかし、強化学習で学習した評価関数は、当然のことながら最適価値関数とは限らず、また最適価値関数からどの程度離れているかもわからない. この研究では、強化学習により学習した評価関数が、最適評価関数と比べてどの程度精度の面で離れているか一定の判断基準を与えることを目的とする. 実験は最適評価関数が解析されているどうぶつしょうぎを用いて行う. 完全解析データにノイズを加えて学習させた評価関数を強化学習による評価関数と見立て、これと最適評価関数との精度を比較をする. 実験から、評価関数のモデルの種類によっては 40%のノイズを加えても精度があまり落ちず、想定していたよりもノイズに対して頑丈であることを示す結果が得られた. また、より高度なモデルの方がノイズの影響を受けやすいことを示唆する結果も得られた. この結果は、より高度で正確な評価関数を作成・学習させるには、学習データの精度もより正確でなければならないということ指し示していると考えることができる.

Analysis of Reliability of Methodology for Building Value Function using Reinforcement Learning

Shuntaro Take[1,a)]　Tomoyuki Kaneko[2]

Abstract: In the field of Game AI, the methodology of making a value function with reinforcement learning (RL) using self-play has been successful including AlphaGo's achievement of defeating human world-top-level Go player. Understandably, however, the value functions learned in the reinforcement learning do not have to be the optimal one, nor is it clear how much they are close to it. The purpose of this study is to give a certain criteria to judge how accurate, compared to the optimal, the value functions learned by RL are. The experiment is held by using Dobutsu shogi (animal chess), whose optimal value function has been completely analyzed and databased. We regard value functions learned with noised data from the complete database as the ones learned by RL. Then we compare their accuracies with the optimal ones. It is found that the functions are more noise-robust than expected, some of which are immune to even 40 percent noise. Also, results imply that the more sophisticated model the more brittle to noise. Accordingly, it may be said that models need more accurate training data as they become more complex and sophisticated.

1. はじめに

チェスや将棋、囲碁に代表される二人零和完全情報ゲームにおいて、強いコンピュータプレイヤを作成する研究は長らくなされてきたが、近年になり、最も難しいとされてきた将棋や囲碁においても、コンピュータの実力が人間のトッププロ棋士を凌駕するようになった. 将棋や囲碁におけるこうした棋力の飛躍的向上は多くの要素技術に支えられているが、そのうちの一つとして、自己対戦や強化学習により評価関数の作成する手法が大きく貢献していると考えられる. 囲碁プログラム AlphaGo はこの技術を他の既存技術とともに組み合わすことで、人間のプロ棋士を凌駕することに成功した [1]. また、Logisthello は棋譜の対局結果を用いて強い評価関数を作ることに成功した [2] [3]. したがってそれらの手法の有効性と限界の検証を試みること

[1] 東京大学教養学部
College of Arts and Sciences、 The University of Tokyo
[2] 東京大学大学院総合文化研究科
Graduate School of Arts and Sciences、 The University of Tokyo
[a)] takesh427@g.ecc.u-tokyo.ac.jp

は、今後の研究開発に大いに貢献する可能性があり、一つの重要な研究課題と言える.

こうした検証に有用であろう手段の一つに、どうぶつしょうぎの完全解析データベースの活用が考えられる. どうぶつしょうぎは、本将棋と比べて盤面を小さくしコマの数も減らして、簡易化したボードゲームである. このゲームは田中 [4] によって完全解析されており、初期局面から到達可能な全ての局面においてその局面からお互いが最善に指した場合の勝敗がわかっている. このデータベースを用いれば、局面と勝敗の関係を正解データとして与えることで、より正確な評価関数の作成ができるはずである.

本研究では、どうぶつしょうぎを題材とし、完全データベースを用いることで正確な評価関数の作成を試みる. そして、完全データベースにノイズを入れて評価関数を学習させることで、それによる精度の悪化具合を調べる. 強化学習を用いた評価関数の作成手法はこのノイズ入りの学習に相当するので、ノイズによりどの程度精度が落ちるを調べることで強化学習による手法の信頼性について検討する.

2. 評価関数の作成における強化学習の利用

評価関数の作成に強化学習を利用している例として、囲碁プログラム AlphaGo を紹介する. AlphaGo において評価関数は、policy networks と value networks と呼ばれる deep convolutional neural networks を用いて、概ね次のように作成されている [1].

(1) policy networks をプロ棋士の棋譜データを教師として学習させて p_σ とする.
(2) p_σ を自己対戦させることで強化学習をして p_ρ とする.
(3) 強化学習をした p_ρ を 最適方策の近似として利用し、状態 s の最適方策に従った場合の状態価値 $v^*(s)$ の近似として方策 p_ρ に従った場合の価値 $v^{p_\rho}(s)$ を求める.

ここで、最適方策の近似として強化学習により学習した方策が用いられていることに注意する. AlphaGo は policy network の方策下での評価関数を学習するが、もちろんこの方策は最適方策とは限らない. また、囲碁の局面の数は膨大であるため、この方策がどの程度最適方策と異なるか推定することも難しい. そこで本研究では、この強化学習で学習した方策と最適方策とがどの程度異なっているかについて以下の手順によって検討する. まず最適方策が既知のどうぶつしょうぎにおいて評価関数を作成する. 次に、最適方策にノイズをかけて評価関数を学習させる. この最適方策にノイズをかけたものは、AlphaGo で用いられる、強化学習による方策を見立てたものである. こうしてできた評価関数の精度を元の最適方策から学習させてできた評価関数の精度と比較することで、強化学習を利用して評価関数を作成する手法の信頼性を分析することができる. ノイズの寡多により精度の変化を調べ、一般に自己対戦で評価関数を作る手法がどの程度最適方策下の評価関数に近しいのかを分析する.

3. どうぶつしょうぎとその完全解析

3.1 どうぶつしょうぎの概要とルール

どうぶつしょうぎは現代における将棋の変種の一つであり、盤面の大きさとコマの種類、及び一部のルール等が本将棋と異なる. 盤面は横 3 マス×縦 4 マスであり本将棋よりも小さい. コマの種類は以下の非成駒 4 種類及び成駒 1 種類がある.

- らいおん：本将棋の玉に相当し、周囲 8 マスの中から一つ選んで移動できる.
- きりん：前後左右の 4 マスから一つ選んで移動できる.
- ぞう：周囲の斜め 4 マスから一つ選んで移動できる.
- ひよこ：本将棋の歩に相当し、前に 1 マスのみ移動できる. 相手陣（最上段または最下段）に進むとにわとりに成ることができる.
- にわとり 本将棋のと金に相当し、ひよこから成ることでのみ登場できる. 前後左右に加えて右前と左前の 6 マスから一つ選んで移動できる.

初期局面は、下の表のような配置である.

表 1 どうぶつしょうぎの初期配置

き（後）	ら（後）	ぞ（後）
	ひ（後）	
	ひ（先）	
ぞ（先）	ら（先）	き（先）

終局条件は以下の 3 つである.

- キャッチ：本将棋における詰みに相当（厳密にはどうぶつしょうぎでは詰みの形で終わらずライオンを明示的に「とる」）. 相手のらいおんをキャッチした方の勝利.
- トライ：本将棋には対応するものがない条件で、自分のらいおんを相手陣（最上または最下の一段）に移動させ、かつ次の相手の手番でキャッチされない場合、勝利となる.
- 引き分け：本将棋の千日手に相当するもので、同じ局面が 3 回目に現れた場合引き分けとなる.

3.2 完全解析

どうぶつしょうぎは完全解析されており、初期局面から先手後手両者が最善手を指し続けると 78 手で後手必勝であることが知られている [4]. また、初期局面から到達可能な全ての局面において、その局面から両者が最善手を指し続けて場合の勝敗と終局までの手数がデータベース化され

ており、到達可能な全ての局面についてその勝敗を調べることができる. 本研究ではこのデータベースを用いてある局面の勝敗を求める.

4. 本研究の目標

本研究での目標は以下の二つである.

- 完全解析データベースを用いてどうぶつしょうぎの評価関数を作成する.
- データベースの勝敗にノイズを加えて評価関数を学習させる. ノイズ入りの勝敗を強化学習で用いる自己対戦での勝敗と見立てて、強化学習による評価関数の作成手法の信頼性について検討する.

後者が主眼となるが、副次的に作成される評価関数について、より精度の高いものの作成も目指す. また、その過程で知見の得られるどうぶつしょうぎの性質（駒の価値や駒組みの価値など）についても議論する.

5. 手法

5.1 評価関数のモデルと学習方法

上記の目標を達成するために、まず評価関数のモデルを立てる. ここで、評価関数とは、ある局面 s の勝敗 z の期待値である価値 v を見積もる状態価値関数である.

まず、局面 s からその局面を表す特徴ベクトル $\boldsymbol{x} = (x_1, x_2, ..., x_m)$ を取り出す. 次に s の価値 v を見積もる評価関数のモデルを $model$ とすると、

$$v = model(\boldsymbol{x})$$

となる. ここではモデルとしてロジスティック回帰モデルを採用することにする. すなわち、ロジスティックシグモイド関数 $\varsigma(t) = \frac{1}{1+exp(-t)}$、重みベクトル $\boldsymbol{W} = (w_1, w_2, ..., w_m)$ 及びバイアス項 b を用いて、

$$v = \varsigma(\boldsymbol{W}\boldsymbol{x} + b) = \varsigma(w_1x_1 + w_2x_2 + ... + w_mx_m + b)$$

と表す.

どうぶつしょうぎにおいては、初期局面から到達できる全ての局面から両プレイヤが最適方策に従った場合の勝敗がわかっている. そこでその局面におけるこの真の勝敗と特徴ベクトルのペアをモデルに与えて教師あり学習させることにより、パラメタ $\boldsymbol{W}$ および b をフィッティングする. 適当に学習された $\boldsymbol{W}$,b を用いれば、上式に特徴ベクトル $\boldsymbol{x}$ を与えることにより s の価値 v が見積もられる. ロジスティック回帰であるので価値 v は先手の勝つ確率を表す. この値が 0.5 より大きければ先手の勝率が高く、反対に小さければ後手の勝率が高いということを表す.

5.2 特徴ベクトルと評価関数の種類

今回の実験では、特徴ベクトル $\boldsymbol{x}$ として、先手の持つ各駒の枚数、先手から見た 1 駒の絶対位置、2 駒の絶対位置関係を用いる.

駒の枚数を表す変数は、各駒（ひよこ、ぞう、きりん、にわとり）の種類 4 つで、取りうる値は、(先手が持つ枚数) - (後手の持つ枚数) の [-2, 2] の整数である.

先手から見た 1 駒の絶対位置は、ひよこ、ぞう、きりんの 3 つの駒については盤上の 12 マスと持ち駒の 13 種類、にわとりについては板状の 12 マスの 13 種類、ライオンについては相手陣にあるときはトライとなり終局してしまうのでそれを除いた 9 種類で、合計 60 個の変数である. 各変数はその位置にその駒があれば 1、それ以外では 0 の値をとる. ただし、後手の駒については、盤を 180 度回転させて重なる位置の変数に-1 を加える. したがって、回転対称の位置に先手後手同一の駒がある場合は 0 になる. 各変数の取りうる値は {-1, 0, 1} である.

2 駒の絶対位置関係も、同様に 180 度回転して同じになるものは同一の変数に符号を変えて格納する. 取りうる値は {-1, 0, 1} で、変数の数は 3330 個である.

なお、本実験においては、左右の対称性を入れないことにする. すなわち A 列のマスと C 列のマスは異なる位置として考える.

これらの特徴ベクトルのどれを変数に持つかで、3 つの評価関数を作成する.

(1) 駒割り：上記の変数のうち各駒の枚数のみを特徴ベクトルとして与える. 学習された $\boldsymbol{W}$ は各駒の価値、すなわち駒割りとなる.

(2) P：上記の変数のうち各駒の枚数及び 1 駒の絶対位置を用いる. 1 駒の絶対位置の変数に対応する $\boldsymbol{W}$ の成分は、その駒がそのマスにいる場合の価値となる. 以後これを単に位置価値と呼ぶことがある.

(3) PP：各駒の枚数、1 駒の絶対位置、2 駒の絶対位置関係の全てを用いる. 2 駒の絶対位置関係の変数に対応する $\boldsymbol{W}$ の成分は、その 2 駒の位置関係の価値を表す.

注意すべきことは、P と PP にも駒割りを表す変数が、PP にも位置価値を表す変数が含まれていることである. 以後、単に、駒割り、P、PP というときには、特に断りのない限りこれら 3 つの評価関数を表すことにする.

6. 実験設定

6.1 局面のデータ

実験には、訓練及びテストデータそれぞれに、ランダムにサンプリングされた各約 10 万局面を用いる. 局面は、後述するランダムプレイヤによるランダムプレイを 10 万対局行い、一対局から一局面ずつを取り出す.

ランダムプレイヤは、初期局面から対局を開始し、以下の二つの条件を満たした上でランダムに合法手を選ぶ.

- 直後に敗北する手は選ばない（ライオンが自分から相手駒の効きに行くことはない）.
- キャッチできるときは必ずする.

上記のランダムプレイヤでは千日手にはまってしまい対局が終わらないことがある．そのため、800 手以上対局が続いたものはそこで対局を打ち切り、その対局からは局面をサンプリングしないことにした．800 手以上続く対局は全対局の 1%程度であるため、これらのデータを使わないことによる影響は限定的であると考えられる．

対局から局面を選ぶ方法は次の通り．ある対局における終局までの手数を n 手とする．0 から n-1 までの整数の中からランダムに整数 k を選ぶ．その対局を k 手まで行なった時の局面をサンプルする．初期局面はサンプルするが結局時の局面はサンプルしない．先手番、後手番両方の局面をサンプルする．

上記の通りランダムに局面をサンプルしたところ、正例（先手の勝ち局面）と負例（後手の勝ち局面）の数では負例の方が多かった．これからどうぶつしょうぎにおいては後手有利な局面が多いことが示唆されるが、本実験では正例負例の数の違いによる影響をなくすため、正例の数に揃えて負例は一部使わずに学習を行なった．

下表に、訓練及びテストデータの局面数を正例負例別に示す．

表 2　各データ数

	打切	正例	負例	使用負例	使用局面
訓練	1281	46926	51793	46926	93852
テスト	1286	46638	52076	46638	93276

また、使用した局面の初期局面からの手数のヒストグラムを図 1 に示す．

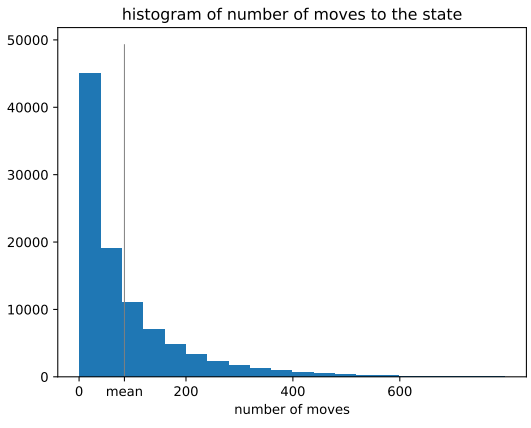

図 1　局面の初期局面からの手数のヒストグラム

6.2　モデルの実装と学習に関する設定

評価関数のモデルは chainer により実装した．上記で述べたモデルに、上述の特徴ベクトル $\boldsymbol{x}$ と完全データベースを引いた勝敗 y のペアを訓練例として与える．先手が勝ちの局面ならば $y = 1$、負けならば $y = 0$ とする．損失関数には最小二乗誤差を用いた．学習器は SGD とし、学習率は 0.1、ミニバッチサイズは 100、エポック数は 10 万回とした．

6.3　評価関数の精度の測定方法

学習した評価関数の精度を測るのにはいくつかの方法が考えられる．最も簡単であるのは、今回データの勝率を 0.5 との大小で勝敗の分類をしていると見ることができるので、正しい勝敗に分類されたものの一致率で表すことである．局面が勝勢か敗勢かの判断の正確さを測っているということができる．以後この勝敗の一致率を accuracy と呼ぶことにする．

また、0.5 との大小で分類すれば 2 値分類の問題であるので、cross entropy を計算できる．accuracy だけではわからないことも cross entropy からわかる可能性があるので、クロスエントロピーも調べる．

このほかの測定方法として、最善手との一致率を測る方法や、対戦結果の勝率で強さを判断する方法も考えられるが、今回の実験では、実装上の簡便さから、 accuracy と cross entropy で以って精度を測ることにした．

6.4　ノイズ入りの学習

本研究においてノイズ p のノイズ入りの学習とは、教師データの y の値を、確率 p で反転させて与えることとする．例えば、 ノイズ 5%の学習では、全訓練データを n 個とすると、n*0.05 個の データ を無作為に抽出し、それらのデータの y が 1 であれば $y = 0$ として、0 であれば $y = 1$ として与える．特に記載のない限り、ノイズ入りの学習においてもテストデータにはノイズを加えていない．

7.　結果

7.1　学習した評価関数の精度

各評価関数の学習結果を図 2 から図 4 に示す．loss は 訓練データのミニバッチにおける最小二乗誤差、accuracy はテストデータについて各評価関数の予測（予測値が 0.5 以上なら先手の勝ち、それ以下なら先手の負けと予測）と完全データベースによる実際の勝敗の一致率、cross entropy はテストデータにおける予測値と実際の勝敗の交差エントロピーである．凡例における all は全テストデータ、win only は正例（先手の勝ち）の平均、lose only は負例（先手の負け）の平均の値である．

学習後の各評価関数の accuracy と cross entropy の最終結果 をまとめて図 **??**に示す．図 2 から図 5 からわかる通り、駒割りのみ、P、PP と特徴量の数が増えるに連れて、accuracy は上がり、cross entropy は下がっている．このことから、駒割りのみよりも P が、P よりも PP が評価関数としての盤面評価の精度が高いといえ、より強い評価関数であること表している．

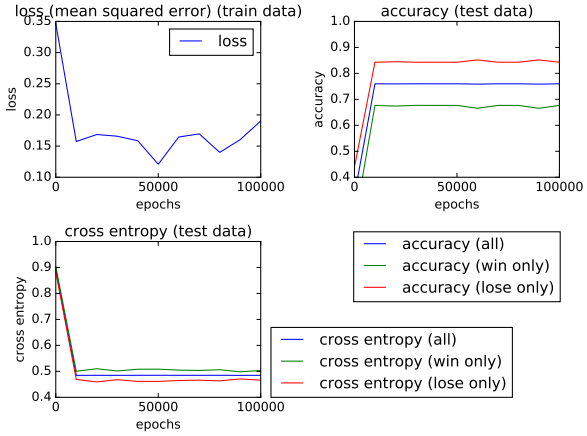

図 2 駒割りのみの評価関数における loss, accuracy 及び cross entropy

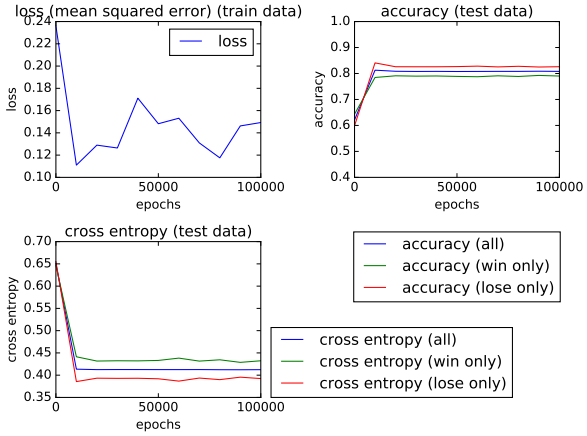

図 3 駒割りと位置価値による評価関数 (P) における loss, accuracy 及び cross entropy

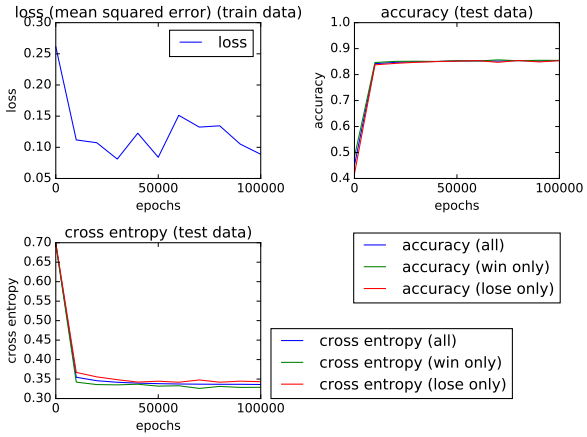

図 4 駒割り、位置価値、2 駒関係による評価関数 (PP) における loss, accuracy 及び cross entropy

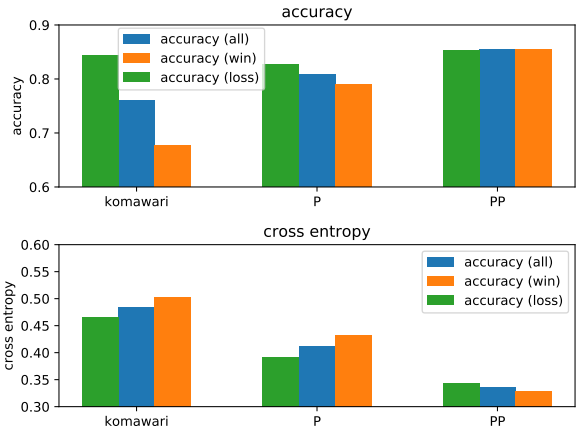

図 5 各評価関数の accuracy 及び cross entropy

特筆すべきは、駒割りは負例に対しての精度が高く、P ではその傾向が多少減少し、PP では正例負例ほぼ等しい精度で判別していることである（PP では正例の精度が少し高い）. これは後で述べる bias 項の正負と関係している. 学習時に正例負例の数を揃えて実験を行なっているため、単純に負例の数が多いためにマイナス方向（サンプルを先手の負けと判断する方向）に偏ってしまっているとは考えられない. 仮説として、負例の方が明白な負けの局面が多く、正例は曖昧な局面が多いということが理由の一つに考えられる. この仮説を確かめるため、また正例負例が正しく分類できていることを確認するために、図 6 から図 8 に、評価関数によるその局面の勝率の予測値（state value）のヒストグラムを示す.

各図において左上のヒストグラムは全データの予測価値のヒストグラム、右上のものはその正例負例による内訳である. 右下のヒストグラムでは見やすさのために右上のヒストグラムでの正例負例を重ねないで表示させた（右側上下の図は同じヒストグラムを表している）. 駒割りでは 0.5 より大きいところに正例が、小さいところに負例が分けられてはいる傾向にあるが、0.5 付近に大きな山がある. P と PP は綺麗に分けられており、PP の方がより極端に分けられている. これから分類能力が駒割り、P、PP の順に大きくなることがわかる. P では 0 付近が 1 付近よりも若干高く、PP では 1 付近の方が若干高い. これは先ほどの、P では負例の方が明白な負け局面が多く PP では正例の方が明白な勝ち局面が多いということを支持する.

7.2 評価関数のパラメタからわかるどうぶつしょうぎの性質

次に、それぞれの評価関数の学習したパラメタの値を調べ、駒の価値や位置の価値を見てみる. ただし、以下で示す値は特に断りのない限りきりんの価値を 1000 とした時の相対値である.

表は各評価関数における駒の価値とバイアス項の値を示

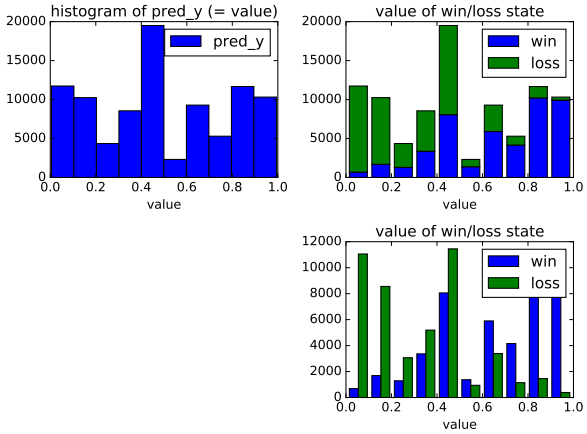

図 6　駒割りによる勝敗の予測値のヒストグラム

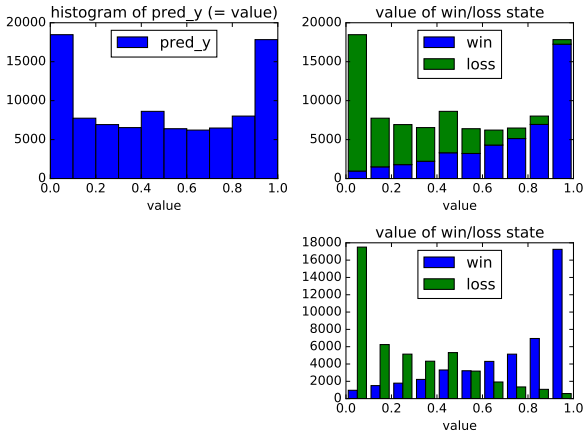

図 7　P による勝敗の予測値のヒストグラム

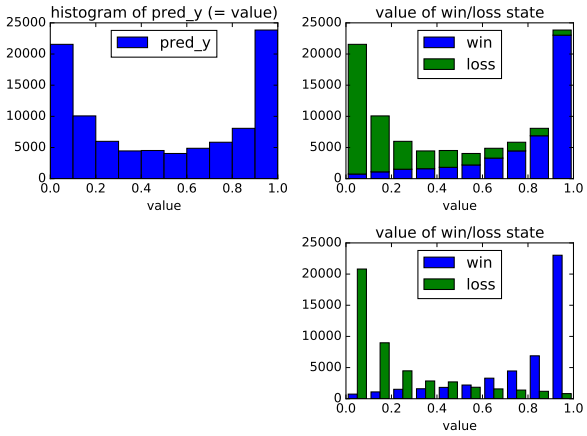

図 8　PP による勝敗の予測値のヒストグラム

表 3　各評価関数における駒の価値

評価関数	ひよこ	ぞう	きりん	にわとり	bias
駒割り	284	826	1000	799	-106
P	123	770	1000	1030	-141
PP	148	919	1000	1111	167

している．全てにおいて、ひよこ＜ぞう＜きりんとなっている．にわとりだけが駒割りではぞうより低く見積もられているが、他の二つではきりんより大きくなっている．効きのマスの数を考えると、周囲 6 箇所に動けるにわとりの方が 4 箇所に動けるぞうやきりんよりも価値が高い方が直感に一致する．

また、バイアス項は先手の有利さを表し、駒割り、P では負であり、PP では正である．この事実は、先ほどのヒストグラムでの山の高さの考察、及び accuracy の正例負例による違いと矛盾しない．

次に、評価関数 P 及び PP について、各駒の絶対位置の価値を図 9 と図 10 に示す．ただしライオンは相手陣にいる場合にはトライとなるので一段目を変数として用意していない．

ぞう、きりんについては、B3 の位置価値が最も高く、駒の効きが最大限に活かせる場所で大きくなっていると考えられる．ライオンは A2、 C2 の価値が高く、これはトライをすることができる位置だからだと考えられる．同じ 2 段目でも B2 は低い．B2 に相手の駒の効きがないことはほぼあり得ないためかもしれない（その場合すでに相手のライオンがトライしている）．また A2、 C2 よりも B2 の方が価値が低いのは、他の駒についても概ね言える．

また、図に表示していない持ち駒の位置価値については、以下のようになった．

表 4　P 及び PP における持ち駒の位置価値

評価関数	ひよこ	ぞう	きりん
P	794 （最大）	554 （B3: 1002）	648 （B3: 717）
PP	1054 （最大）	666 （B3: 808）	1042 （最大）

括弧内には、その駒の位置価値の中で持ち駒の価値が一番大きい場合には「最大」と、それ以外の場合には一番大きな位置価値とその位置を参考値として示した．表や位置価値の図からわかる通り、持ち駒の価値は総じて高くなっている．特にひよこときりんについては持ち駒が最大であるものもある．

全体としては、左右の対称性を考慮せずに学習させたにも関わらず、特に PP について、位置価値の左右対称性を学習している．にわとりについてはあまり対称性が見られないが、そもそもにわとりの出現する局面は他の駒よりも小さいので、学習不足が原因かもしれない．

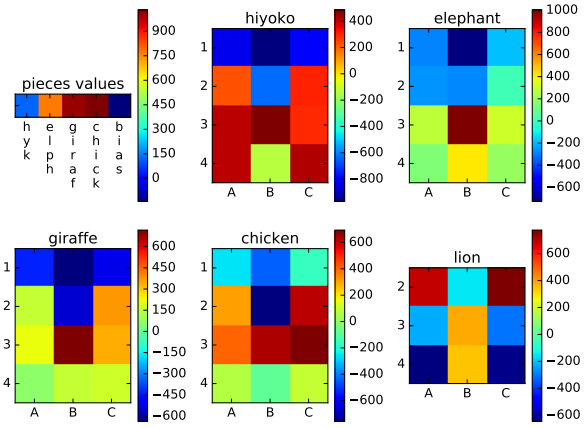

図 9 評価関数 P による各駒の位置価値

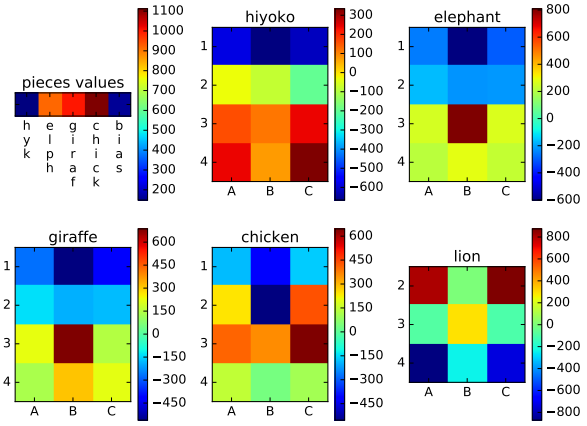

図 10 評価関数 PP による各駒の位置価値

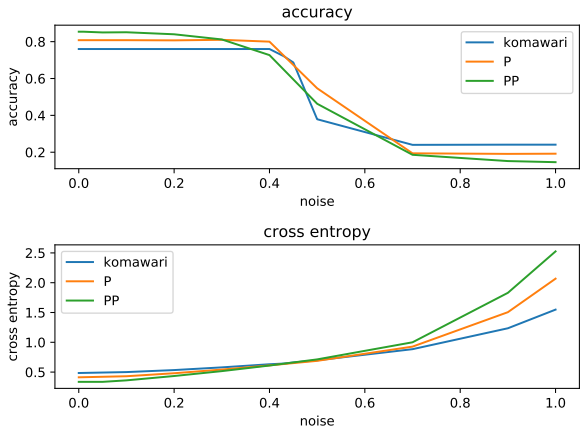

図 11 ノイズ入り学習の accuracy 及び cross entropy ($0 \leq noise \leq 1$)

7.3 ノイズ入りの学習

ノイズを入れた学習後の accuracy 及び cross entropy を図 11 に示す. 図より、ノイズが小さいときは PP、P、駒割りの順で accuracy が高く、ノイズ 0.5 で accuracy も 0.5 程度になり、ノイズがそれ以上になると、accuracy の大小が三者間でが逆転することがわかる. また、ノイズが 1 近くになると、accuracy は ノイズ 0 における accuracy を 1 から引いた値に近くなる.

三者ともに、ノイズが 0.4 程度までは精度がそれほど落ちないことがわかる. 4 割の教師データが反転しているのにもかかわらず精度が落ちずに学習できるということはそれほど自明ではない. 実験前はノイズが 5% や 1% でも精度が落ちる可能性もあり得ると想定していたため、それと比較するとかなりノイズに強いと言える.

ノイズを 0.5 入れると訓練データの半数が反転して与えられることになるため、評価関数による予測もランダムに近くなるので accuracy が 0.5 になることは妥当である. したがって評価関数のノイズに対する耐性を見るには、ノイズが 0.5 以下の時に精度がどう変わるかを見れば良い. そこで図 11 のノイズが 0.1 から 0.4 における部分を図 12 により詳細に示した.

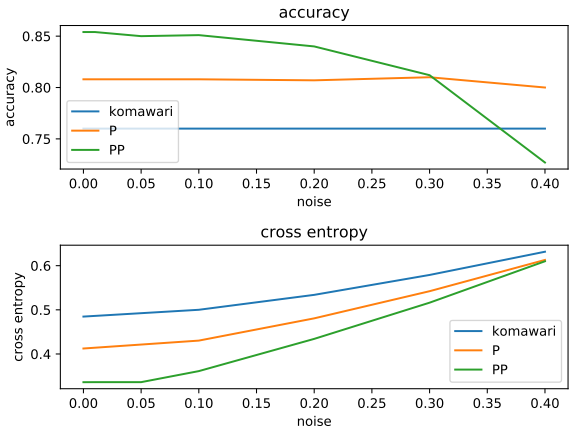

図 12 ノイズ入り学習の accuracy 及び cross entropy ($0 \leq noise \leq 0.4$)

図 12 によると、駒割り及び P、特に前者については、ノイズが増加しても accuracy は常にノイズなしの状態とそれほど変わらないことがわかる. 一方で PP は、ノイズ 0 でこそ最も accuracy が高いが、ノイズが増えるにつれて次第に減少し、ノイズ 0.4 では他の二つよりも低くなっている. このことから、PP の方がノイズに対する耐性が低いといえそうである. PP の cross entropy がノイズ増加に伴い P や駒割りと同程度まで上がってしまうことも PP のノイズ耐性の弱さを支持する.

さて、accuracy だけ見ればノイズの影響は限定的に見えるが、cross entropy は上昇しているのでより詳細に見る

必要がある．10%のノイズを加えた駒割りと PP の state value のヒストグラムを図 13 と図 14 に示す．

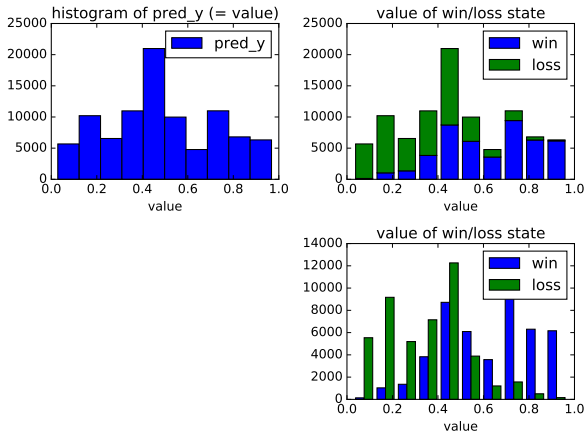

図 13 ノイズ 10%の駒割りによる勝敗の予測値のヒストグラム

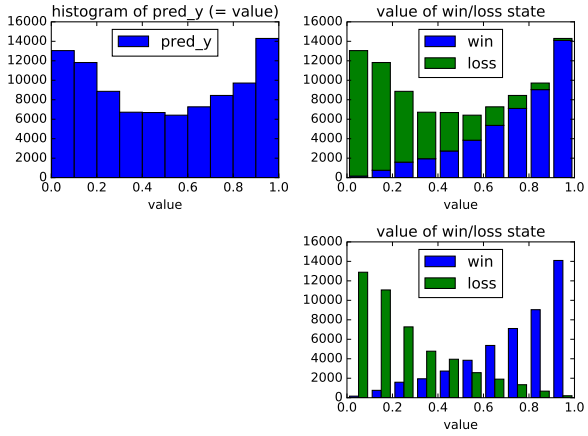

図 14 ノイズ 10%の PP による勝敗の予測値のヒストグラム

図 6 、図 8 と図 13、図 14 を比べると、komawari, PP ともに、10%のノイズでも（accuracy はほとんど変わっていなかったが)、明白だと判断できる局面（value 0,1 付近）が減り、曖昧な局面が増えていることがわかる．

8. 考察

PP の方がノイズ耐性が弱いのは、高度な評価関数の方がノイズに敏感で繊細であるということを示唆する（ただしもちろん、PPP などさらに高度なものを実際に調べてみる必要がある）．この仮説が正しいとすれば、より高度な評価関数を学習させるには、強化学習におけるサンプルデータ（自己対戦の結果）も、より正確なものでなければならないということになる．

なぜ高度な評価関数ほどノイズに弱いのだろうか．そこには議論の余地があるが、一つの理由として、変数（特徴量）の数が増えるとそれだけ訓練データにフィットすることができるため、ノイズのかかった誤ったデータにもフィットしてしまった結果だと考えることができる．特徴量が増えると過学習を起こしやすいことに似ている．

また今回の実験によって、当初期待していたよりもノイズ耐性がある（すなわち頑丈な）ことがわかった．通常の強化学習ではノイズ入りの勝敗で学習させることになるが、ある程度ノイズがあったとしてもノイズの影響をそれほど受けずにある精度の評価関数を作ることはできそうだ．

9. 今後の展望

今回、ノイズ入りの教師データを自己対戦による強化学習で得られるデータと見立てて実験を行なったが、今後は実際に自己対戦をさせて評価関数を作ることにより、実際の強化学習におけるノイズがどの程度のものであるかを測定することを試みる．自己対戦の結果の勝敗と、その対局中の局面でデータベースを引いた真の勝敗との一致率（を 1 から引いた値）が、そのまま今回の実験におけるノイズに相当する．

またすでに述べた通り、今回の結論を検証するためにはより高度な評価関数で実験を行う必要があるため、KPP や KKP などの PPP や、value network といった手法で評価関数を作成することも試み、それらがノイズによってどのように精度が変わるか検証する．

正確さの評価の指標としては、最善手との一致率でも評価することができそうである．どうぶつしょうぎでは各局面から最善方策に従った場合の終局までの手がわかっている．このことを使えば、ある局面での最善手を求めることができる．すなわち、勝ち局面では、次局面が勝ちになる手のうち最短で終局するものを、負け局面では次局面が負けになる手のうち終局まで最長手数かかるものを最善手として採用すればよい．また、勝ち局面から負け局面に転落しないような一致率を考えることもできる（この場合は手数による違いは考慮せずに同列に扱う）．

また学習した評価関数に従うプレイヤ同士の対戦結果の勝率で精度を測ることも考えられる．いわゆる「強さ」を直接的に測る手法である．

参考文献

[1] Mastering the game of Go with deep neural networks and tree search, Silver David, Huang Aja, Maddison Chris J., Guez Arthur, Sifre Laurent, van den Driessche George, Schrittwieser Julian, Antonoglou Ioannis, Panneershelvam Veda, Lanctot Marc, Dieleman Sander, Grewe Dominik, Nham John, Kalchbrenner Nal, Sutskever Ilya, Lillicrap Timothy, Leach Madeleine, Kavukcuoglu Koray, Graepel Thore, and Hassabis Demis, (2016).

[2] From Simple Features to Sophisticated Evaluation Functions, Michael Buro, Computers and Games, (1998).

[3] Improving heuristic mini-max search by supervised learning, Michael Buro, (2002).

[4] どうぶつしょうぎの完全解析, 田中 哲朗, (2009).