Vol.2009-GI-22 No.5
2009/6/26

勝率に近似させた評価関数の性能について

大崎 泰寛[†] 築地 毅[†] 但馬 康宏[††] 小谷 善行[††]

シミュレーションによって獲得された勝率に基づいて着手を選択する研究が近年盛んに行われている．ところが対局のオンライン時に精度の高い勝率をシミュレーションによって獲得するためには時間がかかるという問題点がある．そこで本論文では機械学習を用いて評価関数を，あらかじめ対局のオフライン時に獲得した勝率に近似する手法を提案した．そしてブロックスデュオを実験環境として，得られた勝率近似関数の学習成果について考察し，オンライン時にモンテカルロシミュレーションから着手を選択するプログラムと対局することで本手法の方が優れた対局結果を示すことを確認した．

Performance of Evaluation Function Approximated Probability of Winning

Yasuhiro OSAKI[†] Tsuyoshi TSUKIJI[†]
Yasuhiro TAJIMA[††] Yoshiyuki KOTANI[††]

In these years, the researches of selecting moves using the winning probability obtained through simulations are treated as hot topics. However, there is a problem that obtaining the accurate winning probability through simulations needs too much time at on-line during games. In this paper, we suggested a method as to evaluation function is approximated previously by the probability of winning at off-line during games. Using BlokusDuo as a testing environment, we evaluate the learning result of approximating winning probability function and confirm that our learned player, in comparison to on-line Monte-Carlo simulation player, has been seen to yield better results of games.

1. はじめに

ゲーム情報学研究の主要なテーマの一つに，評価関数の機械学習が挙げられる．評価関数は与えられたゲーム局面において，様々な特徴とその重みから局面の優劣を数値化するものであり，ゲームプログラムの思考ルーチンが最適な着手を選択する指針となる．一方で局面評価が複雑なゲームでは，評価関数の設計自体が困難であるため，シミュレーションによって得られた勝率に基づいた着手選択が有効な思考ルーチンとなる．しかし精度の高い勝率を得るためには多くのサンプルを必要とするため，シミュレーションに時間がかかるという問題がある．

また近年では，将棋のプロ棋士が指した手を教師として評価関数の精度を向上させる兄弟局面学習がコンピュータ将棋プログラムにおいて高い成果を挙げている [1]．ところが誕生してまだ新しいゲームでは，精通したトッププレイヤが存在しないために前述の学習規則は適用できない．

そこで本論文では，各局面の勝率に注目することで比較的歴史の浅いブロックスデュオにおける評価関数の学習方法について考える．そして，静的な評価値を各非終端局面の勝率へ近似するように学習させた評価関数がどのような振る舞いをするかについて考察する．また実際にサンプルをオンラインで獲得しながら着手するモンテカルロシミュレーションプレイヤ（以下 MC プレイヤ）に対して，本手法で学習させた評価関数は勝ち越すことに成功した．さらに第 2 回コンピュータブロックスデュオ大会で 3 位に入賞したプログラム [2]である Mate_with_pawn_drop Club ver.22 ―僕と私と打ち歩詰めの会―（以下 MWPD）に対して，ほぼ互角といえる棋力を得ることができた．

以降，2 章では関連研究について紹介し，3 章では本研究の関数近似手法を示す．そして 4 章では実験方法ならびに実験結果について示し，最後 5 章ではまとめと今後の課題について述べる．

2. 関連研究

近年コンピュータ囲碁の分野において，モンテカルロシミュレーションに木探索を組み合わせた UCT(Upper Confidence bounds applied to Trees)[3]を適用したゲームプログラムが大きな成功を収めている．さらには Bradley-Terry モデルに基づいてシミュレーションの質を向上させた Crazy Stone[4]や，盤上から得られる局所的な特徴から構成された線形価値関数を強化学習させて利用した MoGo[5]はその中でも特に顕著な成果を収め，囲碁のプロ棋士の棋力に肉薄してきているといえる．

[†] 東京農工大学大学院 工学府 情報工学専攻
Department of Computer and Information Sciences, Graduate School of Technology, Tokyo University of Agriculture and Technology
[††] 東京農工大学大学院 工学府
Department of Computer and Information Sciences, Tokyo University of Agriculture and Technology

そこで，このモンテカルロシミュレーションによって得られた勝率に注目することで，著者らはゲームに適用した学習アルゴリズムである TDMC(λ)[6][7]を提案してきた．これは，ゲームの終端局面で環境から与えられる勝敗を報酬とするのみならず，各非終端局面からシミュレーションによって近似される勝率を代理的な報酬と見立てて，環境から与えられた代理報酬の総和の最大化を目的とした強化学習の一種である．

本論文では最急降下法を用いて，学習局面の勝率とその局面の評価値との平均二乗誤差(Mean Squared Error)が最小となるように調整して，得られた勝率近似関数がどのような振る舞いをするか考察した．また本論文の先行研究には，機械学習によって得られた勝率近似関数を UCB1 アルゴリズムの初期値推定に利用する手法がある[8]．

3. 勝率を近似させるアルゴリズム

本章では，まずブロックスデュオのルールならびにゲームの性質について述べる．続いて学習させる線形評価関数の構造と，入力に対して[0,1]の出力を返すシグモイド関数について述べる．そして学習局面の勝率とその局面の静的評価値との平均二乗誤差が最小となるように，最急降下法を用いて評価関数の重みを調整する方法について説明する．

3.1 ブロックスデュオ

ここでは，本論文において学習対象としたブロックスデュオのルールについて本頁右上の囲みに示す．なおブロックスデュオのルールについては，情報処理学会プログラミングシンポジウムの分科会である GPCC が提案する 2009 年公式ルール☆から一部抜粋したものである．また，同研究会ではコンピュータにプレイさせることを考慮して，図 1 に示したブロックスデュオの盤面に座標(1,1)から座標(E,E)までを定義した．

3.2 評価関数の構造

続いて，評価関数$f(w,x_t)$の構造について述べる．まず，エピソードの終端局面の時刻をTとしたとき，時刻ステップは$t\epsilon[1,\mathrm{T}]$と定義する．そして，時刻tの局面P_tにおける静的な評価値v_tを単純な線形結合$f(w,x_t)=\sum_i w_i\times x_{ti}$としたときに，その局面における重みベクトルは$w=(w_1,w_2,\cdots,w_N)$，特徴ベクトルは$x_t=(x_{t1},x_{t2},\cdots,x_{tN})$とそれぞれ$N$次元ベクトルで表すことができる．

局面P_tの特徴ベクトルx_tについては，有効マス，死角マス，勢力範囲およびスコアの盤上の分布から構成されている．盤上の分割については以下の図 2 にあるとおり，個別に割り振られた 1～105 のインデックス番号にしたがった．このときブロックスデュオの性質から，座標(5,5)と座標(A,A)とを結んだ線に対称となるようにインデックスを設けた．例えば図 2 の太枠で示させた座標(8,2)と，色が塗られた座標(2,8)には同じインデックス番号の”30”が割り振られている．

☆ GPCC2009 年問題，http://hp.vector.co.jp/authors/VA003988/gpcc/gpcc09.htm

- 14×14 マスの正方の盤を用いて遊ぶ 2 人完全情報ゲームである．
- 各プレイヤは 1～5 のポリオミノ（総数 21 種類）を交互に使用する．
- 各プレイヤは初手にそれぞれ指定されたマスを覆うようにピースを置く．
- 先手は座標(5,5)のマスで，後手は座標(A,A)のマスが指定されたマスである．
- 2 手目以降は，自分のピースの角同士が少なくとも一か所以上接し，辺同士は接しないような位置に置かなければならない．
- ピースが置けなくなったプレイヤはパスをし，2 人ともパスの場合ゲームが終了する．
- 盤上に置かれたピースが占めるマスの総数の多いプレイヤが勝ちとなる．

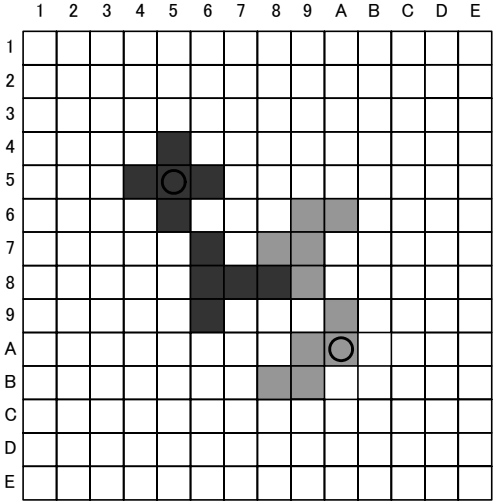

図 1　ブロックスデュオの盤面

続いて，勢力範囲について説明する．ブロックスデュオでは，2 手目以降は自分の置いたピースの角マス（以下有効マス）につながるように自分のピースを置いていくため，有効マスは大変重要な役割を持つ．そこで，有効マスからのマンハッタン距離が近ければ近いほど，自分のピースが占有する確率が高いことになる．ところが，21 種類のピースは様々な形をしているため，有効マスからのマンハッタン距離が等距離であっても，同一の価値を持った勢力範囲であると見なすのは粗いといえる．

	1	2	3	4	5	6	7	8	9	A	B	C	D	E
1	1	2	4	7	11	16	22	29	37	46	56	67	79	92
2		3	5	8	12	17	23	30	38	47	57	68	80	93
3			6	9	13	18	24	31	39	48	58	69	81	94
4				10	14	19	25	32	40	49	59	70	82	95
5					(15)	20	26	33	41	50	60	71	83	96
6						21	27	34	42	51	61	72	84	97
7							28	35	43	52	62	73	85	98
8								36	44	53	63	74	86	99
9									45	54	64	75	87	100
A										(55)	65	76	88	101
B											66	77	89	102
C												78	90	103
D													91	104
E														105

図 2　盤上に割り振られたインデックス

			3				
		5	2	5			
	5	4	1	4	5		
3	2	1	0	1	2	3	
	5	4	1	4	5		
		5	2	5			
			3				

図 3　勢力範囲と有効マスからの距離

そこで本論文では,有効マスからのマンハッタン距離が 3 以内のマスを(図 3 参照), *d=0*～*d=5* まで計 6 種類の勢力範囲と特徴付けることで，評価関数の特徴ベクトルとして用いることにした．ただし勢力範囲に該当するためには，対象のマスにいずれのプレイヤのピースも置かれていないかつ,上下左右に自分のピースが置かれたマス(以下死角マス）でないことが条件となる．さらに有効マスからの最短経路のすべてに邪魔がないことを考慮するために，経路上に相手のピースがないことも勢力範囲の条件として加える．また勢力範囲は各有効マスからそれぞれ計算したとき，複数の勢力範囲の距離を持ったマスが現れた場合は重複を認め,単純に重みを加算することにした．

表 1　評価関数の特徴

特徴	固有特徴の数	共有特徴の数	詳細
d=0	105	1	勢力範囲の距離 0（有効マス）
d=1	105	1	勢力範囲の距離 1
d=2	105	1	勢力範囲の距離 2
d=3	105	1	勢力範囲の距離 3
d=4	105	1	勢力範囲の距離 4
d=5	105	1	勢力範囲の距離 5
死角マス	105	1	手番側が置けないマス
スコア	105	1	手番側が盤上に置いたマス
計	840	8	

ただし評価関数の特徴に関しては表 1 のとおりで，それぞれのインデックス上に存在する特徴を別個としてみなす固有特徴と，いずれのインデックス上に存在しようが同一のものと見なす共有特徴から構成されている．これは学習の際に，学習データに十分登場しないようなスパースな特徴が overfitting してしまうことを回避するための手法として，将棋の評価関数で先行研究されている [9].

3.3　シグモイド関数

局面P_tの勝率は$r_t \in [0,1]$の範囲を表す変数であるため，評価値v_tを入力としたシグモイド関数$\sigma(v_t)$は以下の式で定義される．

$$\sigma(v_t) = \frac{1}{1+\exp(-kv_t)}$$

評価値v_tは$\langle w, x_t \rangle$から求められて理論上値域が$[-\infty, +\infty]$をとるのに対して，シグモイド関数の出力する値は$\sigma(v_t) \in [0,1]$となる．また，ゲイン係数kは以下に示す図 4 のように$v_t = 0$付近における傾斜を制御する．

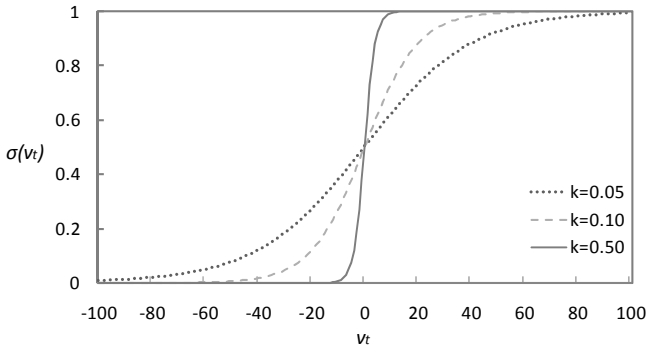

図 4　シグモイド関数

3.4 重みベクトルの更新式

局面P_tにおける勝率r_tと，シグモイド関数で正規化した評価値$V_t = \sigma(f(w, x_t))$との平均二乗誤差を最小にするために，最急降下法を用いて重みベクトルwを更新する．

まず，学習データ内の任意の 1 エピソードにおける目的関数OFを，

$$OF = \sum_{t=1}^{T} [r_t - V_t]^2 \quad (1)$$

とする．目的関数OFは学習データの初手から対局終了を表すT手までに観測された二乗誤差の総和を表している．このように，学習データの平均二乗誤差を最小化する 2 次計画問題に帰着させた．続いて，式(1)で示された目的関数OFを最小化するために重みベクトルwの勾配ベクトル$\nabla_w OF$を求める．

$$\nabla_w OF = \nabla_w \sum_{t=1}^{T} [r_t - V_t]^2$$

$$= -2 \sum_{t=1}^{T} [r_t - V_t] \nabla_w V_t$$

$$= 2k \sum_{t=1}^{T} [r_t - V_t] \sigma(v_t)(1 - \sigma(v_t)) x_t \quad (2)$$

式(2)から求められた勾配ベクトル$\nabla_w OF$を利用することで，重みベクトルwを更新することができる．本論文における学習モデルでは，それぞれの着手ごとに逐次的な更新をせずに，1 エピソードの終端までの差分を求め，終端と同時に一括したバッチ更新を行った．例えば，j番目の要素の重みw_jを 1 エピソードの対局データから更新する場合には，以下の更新式(3)のとおりである．

$$w_j := w_j + \alpha \Delta w_j$$

$$:= w_j + \alpha k \sum_{t=1}^{T} [r_t - V_t] \sigma(v_t)(1 - \sigma(v_t)) x_{tj} \quad (3)$$

重みw_jの更新式(3)は，著者らが提案した TDMC(λ)学習アルゴリズムにおいて，$\lambda = 0, \gamma = 0$としたものと完全に一致する [6)7)]．TDMC(λ)は局面P_tから終端局面Tまでの間に期待される代理報酬の総和を最大化する学習規則であったに対して，本手法は学習局面の即時的な勝率r_tへの近似を目指して学習する．一般的には，学習エージェントが即時報酬(immediate reward)にのみ関心にもった近視眼的な学習をした場合は将来的な報酬が得にくくなるため [10)]，「ゲームに勝つこと」というゲームプログラム本来の目標から遠ざかってしまう．ところが，本手法で即時報酬として扱う勝率には，“ゲームの進行にしたがって終端局面から与えられる勝敗に近づく”という性質がある．そのために，先読みからより深い局面の勝率を推測して着手するプログラムはゲームに勝つという目標から反れていない．またシミュレーション回数が十分であれば，勝率は各局面に対して一意に定まるため，学習の収束は期待できる．

本論文では学習によって調整された勝率近似関数を評価関数としてαβ法による反復深化に組み込むこと，言い換えると勝率で枝刈りをしながら最も勝率の高い局面を選んで着手することでプログラムの振る舞いについて考察した．

4. 実験

本章では，前述の手法を用いてブロックスデュオの評価関数の重みを調整した．そして本手法を評価するために，無作為に生成したテスト局面の勝率と静的評価値との平均誤差を調べた．また，本手法で調整された評価関数から構成されたブロックスデュオのプログラムを TOQN と名付け，MC プレイヤならびに第 2 回コンピュータブロックスデュオ大会で 3 位に入賞した MWPD と対局させた．そして対局結果を考察して，勝率に近似させた評価関数の成果を検証した．

4.1 学習方法

評価関数の特徴については表 1 で示した 848 種類を用いた．ただし初手から 9 手目までを序盤とし，10 手目からゲームの終端までを終盤としてそれぞれ独立した重みベクトルを用いるものとした．ε-greedy 方策に基づき，学習途中の評価関数による自己対局から学習データを収集した．このとき次の着手を確率$\varepsilon\epsilon[0,1]$でランダムに子局面を選択し，$1-\varepsilon$で子局面のうち最も評価値の高いものをグリーディに選択した．ただし，子局面の中に最も評価値の高いものが 2 つ以上ある場合は，その中からランダムで着手を選択した．εが小さいと評価関数の構造に依存した学習データへと偏向してしまうが，一方でεが大きすぎると評価関数の学習成果が以降収集する学習データに反映されなくなってしまう．こうして収集された学習データから，本手法を用いて各パラメータを調整した．重みベクトルwの各要素を1.0×10^{-3}に初期化した後，ゲイン係数を$k = 0.02$，更新式の学習率を$\alpha = 0.5$と固定して学習データ収集と重み更新を 1000 回繰り返して行った．このとき，一棋譜からの更新回数を 3 回と決めた．

4.1.1 学習結果

学習時に$\varepsilon = 1$，つまりランダム局面から重みベクトルwを調節した結果，共有特徴の重みは以下の図 5 から図 8 のとおりになった．ただし図の縦軸は各共有特徴の重みに対して，重みベクトルwの norm の逆数$\|w\|^{-1}$をかけて正規化したものとする．

また，図 5，図 6 では乱数シミュレーションの回数$M = 100$から勝率r_tを得たのに対して，図 7，図 8 では$M = 1000$から勝率r_tを得ることでそれぞれ学習させた．図 5 と図 7 から正規化した共有特徴には大きな違いが見られなかった上，重みの順位は同じであった．また，図 6 と図 8 にも大変よく似た重みづけがされていることがわかった．また序盤，終盤ともに盤上に置いたピースの面積（スコア）が最も高い重みを示した．

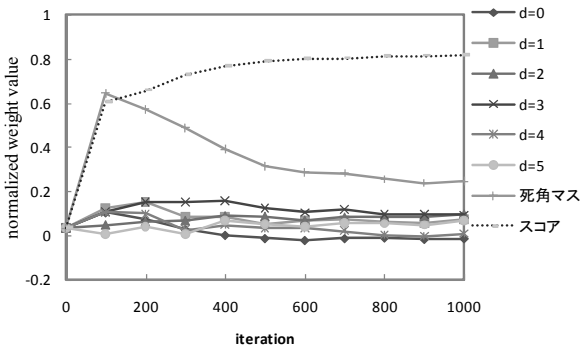

図 5　ゲーム序盤の共有特徴の重み($M = 100$)

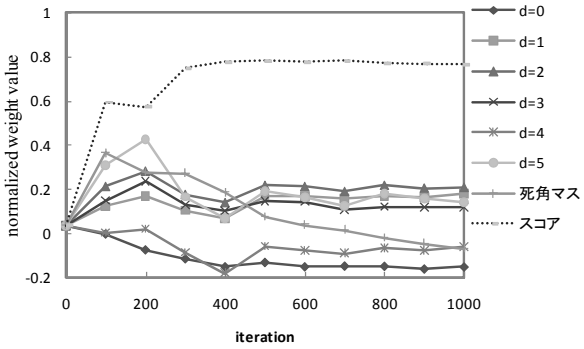

図 6　ゲーム終盤の共有特徴の重み($M = 100$)

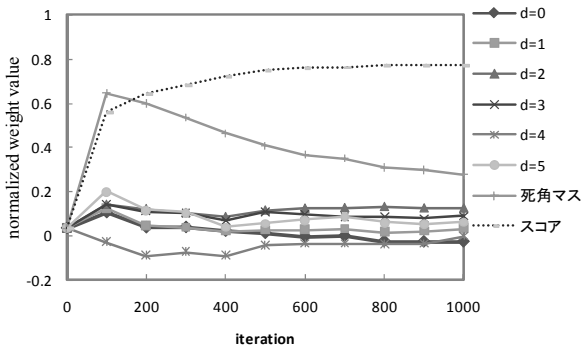

図 7　ゲーム序盤の共有特徴の重み($M = 1000$)

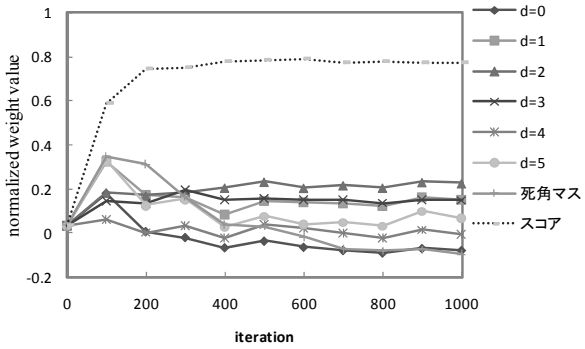

図 8　ゲーム終盤の共有特徴の重み($M = 1000$)

続いて,盤上の各インデックスに対する先手からみたスコアの重みを以下の図 9 図,図 10 に視覚化した．ゲーム序盤（初手から 9 手目まで）において，盤の中央の重みは小さく,それらを囲むように重みが大きくなっていることが図 9 からわかる. 座標(7,4)および座標(4,7)にピースがあるとき，最大値 0.212 をとる．一方，座標(8,7)および座標(7,8)にピースがあるとき，最小値-0.197 をとる.

またゲーム終盤（10 手目から終端まで）もゲーム序盤同様，盤の中央の重みは小さく，外周ほど重みが大きくなっていることがわかる．ここで特徴的なのは，先手が初手で置かなければならない座標(5,5)の周囲を囲むように重みが高いことである．これは座標(1,1)から座標(5,5)辺りを囲むようにピースを置くことで，自分に有利な地を形成することが相手からの侵入を防いで勝率を高くすることを表している．また，座標(8,2)および座標(2,8)にピースがあるとき，最大値 0.392 をとる．一方，座標(8,7)および座標(7,8)にピースがあるとき，最小値-0.462 をとる．重みの詳細については付録の A.1 ならびに A.2 を参照されたい.

4.1.2 学習した評価関数の平均誤差と正解率

学習した評価関数が未知のテスト局面に対してどの程度勝率を予測できるかについて調べた. テスト局面には無作為に生成させた初手から 18 手目までの各 200 局面を利用した. そして学習時のεを$0.01, 0.1, 0.5, 1$の 4 種類として，それぞれの評価関数で予測した結果を以下の表 2，表 3 に示す.

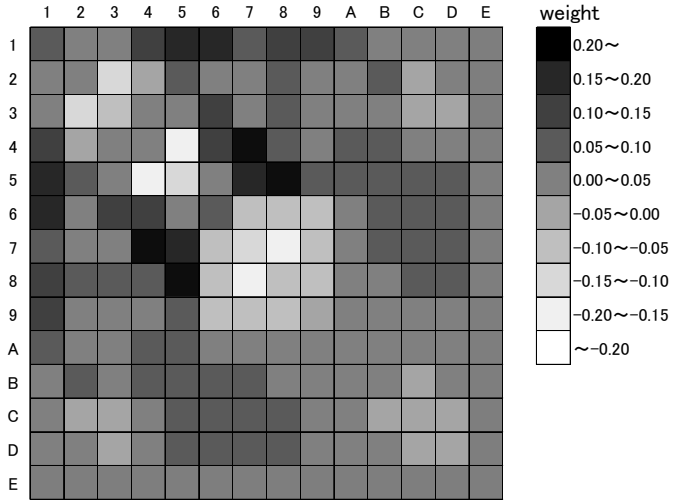

図 9　視覚化したゲーム序盤のスコアの重み($M = 100$)

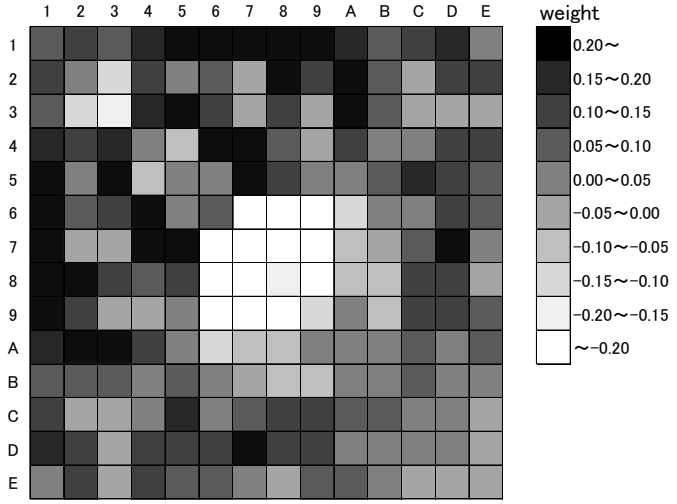

図 10　視覚化したゲーム終盤のスコアの重み($M = 100$)

表 2　平均誤差と正解率($M = 100$)

ε	平均誤差	正解率
0.01	0.1091	73.49%
0.1	0.1009	74.50%
0.5	0.0964	78.00%
1	0.0952	79.65%

表 3　平均誤差と正解率($M = 1000$)

ε	平均誤差	正解率
0.01	0.1029	74.74%
0.1	0.0904	78.31%
0.5	0.0951	76.24%
1	0.0860	81.76%

ただし勝率との誤差を$|r_t - \sigma(v_t)|$として，$M = 100$と$M = 1000$の 2 種類の学習方法で実験した．このとき，各テスト局面に付与される勝率r_tは 1000 回乱数シミュレーションを行ったときの平均値を用いた．また，自分に有利な局面，つまり勝率が 50%を超える局面を正しく評価できているかを調べるために，正解を以下の式(4)に定義した．

$$correct = \begin{cases} 1 & if\ \{(r_t > 0.5) \wedge (\sigma(v_t) > 0.5)\} \vee \{(r_t < 0.5) \wedge (\sigma(v_t) < 0.5)\} \\ 0 & otherwise \end{cases} \quad (4)$$

表 2 および表 3 から，学習時のεが高くなると平均誤差が減少することがわかった．これは，学習時により多岐に渡った局面を体験することで細分化した特徴が学習データに多く登場し，正しく学習できたためであると考えられる．そして，$M = 1000$かつ$\varepsilon = 1$で学習した評価関数は勝率との平均誤差が最小となった．以上の表 2，表 3 から学習時のεが増加するにつれて正解率が増加することがわかった．

しかし，テスト局面と学習局面の用意の仕方が同じであったため，$\varepsilon = 1$で学習した評価関数に有利に働いたとも考えられる．そこで，以下の表 4 に示すとおり平均誤差が最小であった評価関数（表 3 の$M = 1000$かつ$\varepsilon = 1$）の学習成果を他のテスト局面を用いて調べた．テスト局面は，2 手目までは完全にランダムに着手することで十分にばらけさせた後，3 手目以降は 3 種類の評価関数（表 3 の$\varepsilon = 0.01, 0.1, 0.5$）によって生成したものである．

表 4　テスト局面の変化に応じた平均誤差と正解率($M = 1000$)

テスト局面 生成時のε	平均誤差	正解率
0.01	0.0903	81.62%
0.1	0.0878	82.66%
0.5	0.0891	82.47%

表 4 から，無作為に生成させた局面から学習した評価関数は表 3 と同等かそれ以上の正解率を示した．また，表 3 の平均誤差 0.0860 と比べて誤差が大きくなった理由には，評価関数による偏りがテスト局面に生じていたことが原因として考えられる．

表 5　TOQN(思考時間 1[s])と MC プレイヤの対局結果

ε	TOQN wins	draw	MC Player wins
0.01	78	7	115
0.1	100	8	92
0.5	118	11	71
1	135	3	62

表 6　TOQN(思考時間 5[s])と MC プレイヤの対局結果

ε	TOQN wins	draw	MC Player wins
0.01	77	14	109
0.1	122	9	69
0.5	133	10	57
1	141	5	54

表 7　TOQN(思考時間 10[s])と MC プレイヤの対局結果

ε	TOQN wins	draw	MC Player wins
0.01	94	7	99
0.1	139	5	56
0.5	136	11	53
1	134	10	56

4.2 対局方法

続いて，本手法によって学習された評価関数から構成されたブロックスデュオプログラム TOQN と，対局時にサンプルを獲得する MC プレイヤを対局させた．TOQN は学習時にε-greedy 方策に基づいた学習データを収集しているため, $\varepsilon = 0.01, 0.1, 0.5, 1$の 4 種類で比較した．また，勝率$r_t$を得るための乱数シミュレーション回数を$M = 100$として学習させたものを用いた．

4.2.1 MC プレイヤとの対局結果

MC プレイヤは各子局面をルートとして，それぞれ 100 回シミュレーションをした結果，最も勝率の高かったものを着手とした．一方 TOQN はαβ法による反復深化を用いていて，思考時間の終了時に現在得られている最善手を返す．また，MC プレイヤの思考時間は子局面のすべてがシミュレートし終わるまでとして，TOQN の思考時間はそれぞれ 1[s]，5[s]，10[s]とした．対局結果が偏らないようにお互い初手はランダムにピースを置くようにして, 先手後手合わせて 100 回ずつ対局をした結果を表 5, 表 6, 表 7 に示す.

まず表 5 から，学習時のεが高くなるにつれて MC プレイヤに対する勝数が増加していることがわかる．これは，学習時にできるだけ多くの局面を体験したことが，対局結果に反映したといえる．また思考時間を 5[s]として対局させた表 6 からも，εが高くなるにつれて対局成績がよくなることがわかる．ところが表 7 では，$\varepsilon = 0.1$のときに 139 勝 56 敗 5 分と最も対局結果が優れており，MC プレイヤに対する成績が$\varepsilon = 0.1, 0.5, 1$ではほぼ横ばいとなっている．また，TOQN の思考時間が 5[s]と 10[s]の間に大きな勝数の差がないことがわかる.

4.2.2 MWPD との対局結果

続いて，第 2 回コンピュータブロックスデュオ大会で 3 位に入賞した MWPD と対局結果を示す．MWPD の評価関数は，ゲームの序盤と終盤にそれぞれの重みを有し，有効マス，死角マス，勢力範囲などの 424 種類の特徴からなる．また 1 秒間で訪問することができる局面数はおよそ 103,000 局面である. 一方 TOQN の評価関数は, MWPD の倍 848 種類の特徴から構成されているが, 1 秒間で訪問できる局面数はおよそ 77,000 局面と MWPD の約 70 % 程度である．MC プレイヤとの対局同様，初手はお互いランダムにピースを置くようにして，先手後手合わせて 100 回対局をした．また，思考時間はそれぞれ 1[s]として，両プログラムともαβ法による反復深化を用いて思考時間の終了時に現在得られている最善手を返すものとする．学習の教師となる勝率r_tを得るために行った乱数シミュレーション回数をそれぞれ$M = 100$, $M = 1000$として, 表 8, 表 9 に MWPD との対局結果を示す.

表 8　TOQN($M = 100$)と MWPD の対局結果

ε	TOQN wins	draw	MWPD wins
0.01	54	8	139
0.1	76	10	114
0.5	91	9	110
1	94	11	95

表 9　TOQN($M = 1000$)と MWPD の対局結果

ε	TOQN wins	draw	MWPD wins
0.01	59	5	136
0.1	72	12	116
0.5	90	13	97
1	95	6	99

表 8 から，学習時のεが高くなるにつれて MWPD に対する勝数が増加していることがわかる. $\varepsilon = 1$, つまり完全にランダムに学習データを生成して学習した場合, MWPD に対して先手 100 回対局中 61 勝 34 敗 5 分, 後手 100 回対局中 33 勝 61 敗 6 分だった．また表 9 も表 8 同様，学習時のεが高くになるにつれて勝数が増加した．最高勝数は$\varepsilon = 1$のとき，MWPD に対して先手 100 回対局中 64 勝 33 敗 3 分，後手 100 回対局中 31 勝 66 敗 3 分だった．しかし予想と反して，$M = 100$の場合と勝率の精度がより高い$M = 1000$との場合との間に勝数の差があまり生じなかった.

5. おわりに

本論文では各局面の勝率と静的評価値との平均二乗誤差を最小化するように評価関数を調整する手法を提案した．また，本学習モデルは各局面の勝率のみを利用して学習するため，解析が十分させていないゲームにも適用することができる．実験結果から,学習データを無作為に収集した時に初手から 18 手目までの平均二乗誤差を最小にすることができた．さらに，対局時にシミュレーションをして勝率を獲得する MC プレイヤに対して十分勝ち越すことに成功し,前回大会において上位入賞した MPWD に対しても同等の強さを発揮することができた．また対局結果からも，学習データを無作為に収集した方がより強い評価関数を形成することがわかった．一方で，シミュレーション回数を 10 倍に増やして学習の教師となる勝率の精度を向上させても,対局成績に大きな差は見られなかった．

今後の課題として，特徴量の増減と勝率近似の性能の関係について調べる必要がある．さらに，本手法を囲碁や将棋など他のゲームに適用することも課題として挙げられる．またその時，勝率をより近似できるような局面特徴の発見が伴えば，今後のゲーム解析に貢献できると考えられる．

参考文献

1) 保木邦仁,"局面評価の学習を目指した探索結果の最適制御", 第 11 回ゲームプログラミングワークショップ(GPW2006), pp.78−83.
2) 築地毅, 柴原一友, 但馬康宏, 小谷善行,"先読みを教師とした兄弟局面の比較に基づく評価関数の学習", 情報処理学会, ゲーム情報学研究会報告, vol.2009 no.27 2009-GI-21, pp.71-78.
3) Levente Kocsis, Csaba Szepesvári,"Bandit Based Monte-Carlo Planning", Lecture Notes in Computer Science, vol.4212, pp.282-293, 2006.
4) Rémi Coulom,"Computing Elo Ratings of Move Patterns in the Game of Go", Proceedings of the 9th International Conference on Computer Games, pp.113-124, Apr. 2007.
5) Sylvain Gelly, David Silver,"Combining Online and Offline Knowledge in UCT", Proceedings of the 24th International Conference on Machine Learning, pp.273–280, Jun. 2007.
6) 大崎泰寛, 柴原一友, 但馬康宏, 小谷善行,"TDMC(λ)に基づく評価関数の調整", 第 13 回ゲームプログラミングワークショップ(GPW2008), pp.73−79.
7) Yasuhiro Osaki, Kazutomo Shibahara, Yasuhiro Tajima, Yoshiyuki Kotani, "An Othello Evaluation Function Based on Temporal Difference Learning using Probability of Winning", IEEE Symposium on Computational Intelligence and Games, pp.205–211, Dec. 2008.
8) 但馬康宏，小谷善行,"モンテカルロ法における勝率近似関数の組み込み方法", 第 13 回ゲームプログラミングワークショップ(GPW2008), pp.100−103.
9) 築地毅, 柴原一友, 但馬康宏, 小谷善行,"重ね合わせによるデータ構造を用いた評価関数の学習", 第 13 回ゲームプログラミングワークショップ(GPW2008), pp.144−151.
10) Richard Sutton, Andrew Barto,"Introduction to Reinforcement Learning", MIT Press, 1998.

付録

A.1 ゲーム序盤のスコアの重み($M = 100$)

Row＼Col	1	2	3	4	5	6	7	8	9	A	B	C	D	E
1	0.050	0.067	0.041	0.108	0.167	0.166	0.061	0.139	0.117	0.081	0.033	0.021	0.034	0.010
2	0.067	0.005	−0.196	−0.035	0.055	0.043	0.018	0.081	0.040	0.039	0.059	−0.004	0.021	0.016
3	0.041	−0.196	−0.070	0.023	0.035	0.101	0.015	0.054	0.041	0.048	0.014	−0.019	−0.003	0.011
4	0.108	−0.035	0.023	0.037	−0.159	0.115	0.212	0.083	0.026	0.050	0.053	0.039	0.025	0.014
5	0.167	0.055	0.035	−0.159	−0.107	0.008	0.178	0.207	0.091	0.062	0.085	0.056	0.054	0.018
6	0.166	0.043	0.101	0.115	0.008	0.069	−0.072	−0.086	−0.063	0.013	0.057	0.057	0.054	0.027
7	0.061	0.018	0.015	0.212	0.178	−0.072	−0.112	−0.197	−0.074	0.014	0.065	0.069	0.092	0.038
8	0.139	0.081	0.054	0.083	0.207	−0.086	−0.197	−0.094	−0.095	0.012	0.044	0.081	0.071	0.043
9	0.117	0.040	0.041	0.026	0.091	−0.063	−0.074	−0.095	−0.007	0.019	0.030	0.044	0.037	0.026
A	0.081	0.039	0.048	0.050	0.062	0.013	0.014	0.012	0.019	0.001	0.018	0.003	0.006	0.011
B	0.033	0.059	0.014	0.053	0.085	0.057	0.065	0.044	0.030	0.018	0.001	−0.001	0.003	0.003
C	0.021	−0.004	−0.019	0.039	0.056	0.057	0.069	0.081	0.044	0.003	−0.001	−0.003	−0.001	0.001
D	0.034	0.021	−0.003	0.025	0.054	0.054	0.092	0.071	0.037	0.006	0.003	−0.001	−0.002	0.001
E	0.010	0.016	0.011	0.014	0.018	0.027	0.038	0.043	0.026	0.011	0.003	0.001	0.001	0.001

A.2 ゲーム終盤のスコアの重み($M = 100$)

Row＼Col	1	2	3	4	5	6	7	8	9	A	B	C	D	E
1	0.050	0.107	0.091	0.197	0.224	0.312	0.203	0.365	0.279	0.194	0.100	0.117	0.160	0.043
2	0.107	0.035	−0.134	0.120	0.028	0.086	−0.020	0.392	0.111	0.233	0.084	−0.046	0.139	0.106
3	0.091	−0.134	−0.187	0.193	0.293	0.112	−0.037	0.137	−0.015	0.205	0.058	−0.033	−0.035	−0.031
4	0.197	0.120	0.193	0.044	−0.071	0.206	0.285	0.077	−0.008	0.020	0.040	0.043	0.115	0.148
5	0.224	0.028	0.293	−0.071	0.001	0.029	0.205	0.114	0.017	0.010	0.082	0.160	0.118	0.067
6	0.312	0.086	0.112	0.206	0.029	0.062	−0.360	−0.355	−0.221	−0.113	0.043	0.043	0.137	0.058
7	0.203	−0.020	−0.037	0.285	0.205	−0.360	−0.210	−0.462	−0.259	−0.073	−0.022	0.051	0.208	0.049
8	0.365	0.392	0.137	0.077	0.114	−0.355	−0.462	−0.193	−0.351	−0.089	−0.083	0.108	0.117	−0.006
9	0.279	0.111	−0.015	−0.008	0.017	−0.221	−0.259	−0.351	−0.102	0.022	−0.062	0.110	0.111	0.057
A	0.194	0.233	0.205	0.020	0.010	−0.113	−0.073	−0.089	0.022	0.001	0.036	0.074	0.004	0.058
B	0.100	0.084	0.058	0.040	0.082	0.043	−0.022	−0.083	−0.062	0.036	0.021	0.050	0.021	0.012
C	0.117	−0.046	−0.033	0.043	0.160	0.043	0.051	0.108	0.110	0.074	0.050	0.011	0.007	−0.012
D	0.160	0.139	−0.035	0.115	0.118	0.137	0.208	0.117	0.111	0.004	0.021	0.007	0.001	−0.002
E	0.043	0.106	−0.031	0.148	0.067	0.058	0.049	−0.006	0.057	0.058	0.012	−0.012	−0.002	−0.002