LinUCB のモンテカルロ木探索への応用

万代悠作[1,a)]　金子知適[1]

概要：モンテカルロ木探索は様々な領域で用いられているアルゴリズムであり, そのなかでもゲーム木探索では理論的に裏付けされた UCB という指標を用いる UCT というアルゴリズムが盛んに研究されている. LinUCB は UCB を特徴ベクトル付き multi-armed bandit problem へと拡張したアルゴリズムであり, その手法はゲーム木探索においても有効であると考えられる. 本研究ではモンテカルロ木探索において UCB の代わりに LinUCB を使用した LinUCT やその変種, シミュレーションへの応用を提案し, 基礎的な実験を通してその性能を評価する. 仮想的なゲーム木を用いた性能評価では UCB 以上の性能を発揮することが観察されたが, 実際のゲームとして題材としたオセロにおいてはその特徴ベクトルの構築などにおいて課題が残る.

LinUCB Applied to Monte Carlo Tree Search

Yusaku Mandai[1,a)]　Tomoyuki Kaneko[1]

Abstract: Monte Carlo Tree Search (MCTS) is an algorithm widely used in various domains. In game tree search, UCT (UCB applied to Trees), which employs a value called UCB, has been studied well. LinUCB is a variant of UCB specialized for contextual multi-armed bandit problem, and it should be effective in game tree search. In this study, we propose a brand new MCTS method employing LinUCB instead of UCB, called LinUCT and its variants. Also we applied LinUCB to estimating move probabilities in simulation. In order to show the effectiveness of the presented methods, several experiments were conducted with artificial game trees as well as game of othello. The results showed that our method outperformed UCT in artificial trees, but that, the performance was limited due to the difficulties in the construction of proper feature vectors in practical game.

1. はじめに

近年コンピュータゲームプレイヤは目覚ましい進歩を遂げており, そのなかでもコンピュータ囲碁の棋力はモンテカルロ木探索というモンテカルロシミュレーションと木探索を組み合わせた手法を用いることで飛躍的に向上した. モンテカルロ木探索において, 次にどの局面を探索するかを決定する際に UCB (Upper Confidence Bounds) という指標がよく用いられている. UCB を用いたモンテカルロ木探索は UCT (UCB applied to Tree) と呼ばれ, 現在多くの研究が行われている. UCB は multi-armed bandit problem という問題のアルゴリズムであり [1], UCT ではその局面から得られる期待値の信頼区間の上端である UCB を計算してその局面を評価している.

Multi-armed bandit problem は様々な派生問題が研究されており [2], そのうちの一つが contextual multi-armed bandit problem である. Contextual multi-armed bandit problem とは各腕に特徴ベクトルが割り当てられており, その特徴ベクトルが報酬の期待値に影響する multi-armed bandit problem に一種である. LinUCB (Linear UCB) は contextual multi-armed bandit problem などの環境次第で期待値が変化するような状況に対して UCB1 以上の性能を示したアルゴリズムである [3]. LinUCB は各腕の報酬の期待値をその特徴ベクトルを用いた ridge regression で推定して次の腕を決定している.

本研究ではゲームの局面を特徴ベクトルを持つ腕と考えて LinUCB を用いる手法を提案し, その効果を検証する.

[1] 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)] mandai@graco.c.u-tokyo.ac.jp

また LinUCB が推定する局面の期待値は, モンテカルロ木探索のシミュレーションにおいてより自然な確率で行動選択を行うことが可能であると考えられる. そこで期待値の推定値を用いたシミュレーション中の方策の有用性についても調査する.

2. 関連研究

2.1 モンテカルロ木探索

モンテカルロ木探索は探索木の葉の局面評価をモンテカルロシミュレーションで行う木探索手法である. ゲームにおけるモンテカルロシミュレーションでは, 局面からランダムにゲーム終局までプレイすることで評価を行う. これをプレイアウトと呼び, プレイアウトを複数回行い, その勝率で局面を評価する.

モンテカルロ木探索ではルート局面から始まり, 各候補手に対して Selection Policy に従って葉局面まで選択する. 葉局面から新たな局面を展開し, プレイアウトを行う. プレイアウトで得られた報酬はたどってきた局面全てに伝播される. これを一イテレーションとして, 複数回イテレーションを行う.

2.2 UCT

UCT はモンテカルロ木探索の一つであり, UCB (Upper Confidence Bound) という指標を Selection Policy としたアルゴリズムである. UCB の一つに UCB1 が存在し, t 回目の試行における腕 a の値は以下の式で表される.

$$\mathrm{UCB1}(t,a) = \bar{X}_a + \sqrt{\frac{2\log T_a}{t}} \tag{1}$$

ここで $\bar{X}_a$ はこれまでの試行における腕 a の報酬の平均値, T_a は腕 a を選択した回数である. 第二項は腕 a の信頼度の低さとして解釈でき, 結果として期待値が高いもしくはまだ十分に期待値を推定できていない腕の値が高くなる.

2.3 LinUCB

Contextual multi-armed bandit problem とは各腕と腕を引くユーザに特徴ベクトルが与えられており, その特徴が行動の期待値に影響する問題で, LinUCB は contextual multi-armed bandit problem のアルゴリズムの一つとして提案されたものである.

LinUCB では各腕の報酬の期待値が

$$\boldsymbol{E}[r_{t,a}|\boldsymbol{x}_{t,a}] = \boldsymbol{x}_{t,a}^T\boldsymbol{\theta}_a^* \tag{2}$$

に従うという仮定をしている. ここで $r_{t,a}$ と $\boldsymbol{x}_{t,a}$ は t 回目の試行における腕 a の報酬と特徴ベクトル, $\boldsymbol{\theta}_a^*$ は係数ベクトルである. 文献 [3] では各試行ごとに異なるユーザが腕を選択するという状況をモデル化しているため, 特徴ベクトル $\boldsymbol{x}_{t,a}$ は t にも依存する.

この仮定のもと, t 回目の試行においてこれまでの経験から各腕の $\boldsymbol{\theta}_a^*$ の回帰係数ベクトル $\hat{\boldsymbol{\theta}_a}$ をそれぞれ求め, 期待値の推定値の信頼区間の上端である以下の式を求める:

$$\mathrm{LinUCB}(t,a) = \hat{\boldsymbol{\theta}}_a\boldsymbol{x}_{t,a}^T + \alpha\sqrt{\boldsymbol{x}_{t,a}^T A_a^{-1}\boldsymbol{x}_{t,a}}. \tag{3}$$

LinUCB は各時刻 t でこの評価値が最高の腕を次に引くというアルゴリズムである. ここで α は定数, A_a^{-1} は分散共分散行列の逆行列を表している. LinUCB のアルゴリズム [3] を Algorithm 1 に示す .

Algorithm 1 LinUCB

```
for t = 1, 2, 3, ..., T do
  Observe features of all arms:
  for a in all amrs do
    if a is new then
      A_a ← I_d
      b_a ← 0_d
    end if
    θ̂_a ← A_a^{-1} b_a
    p_{t,a} ← θ̂_a x_{t,a} + α sqrt(x_{t,a}^T A_a x_{t,a})
    Choose arm a_t = arg max_a p_{t,a} with ties broken arbitarily,
    and observe a real-valued payoff r_t
    A_{a_t} ← A_{a_t} + x_{t,a_t}^T x_{t,a_t}
    b_{a_t} ← b_{a_t} + r_t x_{t,a_t}
  end for
end for
```

LinUCB のゲームへの応用については, 一人麻雀へ応用した先行研究が存在する [4]. UCB と同様に第一項が勝率, 第二項がその腕の信頼度を表しているため, UCB1 と本質的に似た値を計算している [3] ため, ゲーム木探索においても同様の運用ができると思われる.

3. 提案手法

3.1 LinUCT: LinUCB の木探索への応用

本研究ではモンテカルロ木探索における Selection Policy に LinUCB を用いる LinUCT やその変種を提案する. LinUCB を木探索に適用する場合, どのような前提を置くかで様々なバリエーションが考えられる.

LinUCT は木探索においても LinUCB 同様に各腕を独立に考え, LinUCB をそのまま用いたものである. LinUCT では, ルート局面から始まりそれぞれのノードの子ノードの LinUCB 値を求め, LinUCB 値が最大のノードを葉ノードへ辿り着くまで選択していく. 葉ノードへ到達したら葉ノードから木を成長させ, プレイアウトを行う. プレイアウトの結果 r_t は辿ってきた各ノードについて LinUCB の計算と同様に $\boldsymbol{A_a} \leftarrow \boldsymbol{A_a} + \boldsymbol{x_a}\boldsymbol{x_a^T}$, $\boldsymbol{b}_a \leftarrow \boldsymbol{b}_a + r_t\boldsymbol{x}_a$ を計算し評価値を更新する. 以上をまとめたものを Algorithm 2 に示す.

前述のとおり LinUCB は UCB と本質的に同様の計算を行っているため, UCT と似た傾向を示すと考えられる.

Algorithm 2 LinUCT

```
for t = 0, 1, ...T do
  i ← 0
  path[i] ← root node
  while path[i] is not leaf nor terminal node do
    for a in all children of path[i] do
      θ̂_a ← A_a^{-1} b_a
      p_{t,a} ← θ̂_a x_a^T + α√(x_a^T A_a^{-1} x_a)
    end for
    path[i+1] ← arg max p_{t,a}
    i ← i + 1
  end while
  if path[i] should be expanded then
    Expand path[i]
  end if
  Playout from path[i] and observe the reward r_t
  for a in path do
    A_a ← A_a + x_a x_a^T
    b_a ← b_a + r_t x_a
  end for
end for
```

3.2 LinUCT$_{\mathrm{Shared}}$ と LinUCT$_{\mathrm{Shared}}$-Rave

先ほどの LinUCT では, 各ノードが独立に回帰を行っているため, 十分な回数を訪問しなければ値が収束しない. しかし評価関数が有効に働くゲームにおいては, 各局面で係数ベクトルである $\hat{\boldsymbol{\theta}}$ は概ね似た値を持つと考えられる. LinUCT$_{\mathrm{Shared}}$ は式 (3) の分散共分散行列や回帰係数を異なる局面で共有させることで上記の性質をモデルに取り入れたものである. これにより各局面が独立に値をもつ場合よりも早く値が収束し, また様々な局面を学習するため過学習を防ぐことができると考えられる.

LinUCT$_{\mathrm{Shared}}$ のアルゴリズムを Algorithm 3 に示す. Algorithm 2 から変更した部分は行頭に * をつけた.

Algorithm 3 LinUCT$_{\mathrm{Shared}}$

```
{ values with prefix _{a,s} mean the shared value }
for t = 0, 1, ..., T do
  i ← 0
  path[i] ← root node
  while path[i] is not leaf nor terminal node do
    for a in all children of path[i] do
      *θ̂_{a,s} ← A_{a,s}^{-1} b_{a,s}
      *p_{t,a} ← θ̂_{a,s} x_a^T + α√(x_a^T A_{a,s}^{-1} x_a)
    end for
    path[i+1] ← arg max p_{t,a}
    i ← i + 1
  end while
  if path[i] should be expanded then
    Expand path[i]
  end if
  Playout from path[i] and observe the reward r_t
  for a in path do
    *A_{a,s} ← A_{a,s} + x_a x_a^T
    *b_{a,s} ← b_{a,s} + r_t x_a
  end for
end for
```

値をどのノード間で共有させるかは様々なやり方が考えられるが, 単純に探索木内のすべての局面で共有させたものを LinUCT$_{\mathrm{RootShared}}$ と呼ぶことにする. すなわち, LinUCT$_{\mathrm{RootShared}}$ では木全体で A, $\boldsymbol{b}$ を共有し, $\hat{\boldsymbol{\theta}}$ を計算する.

一方回帰係数などを異なる局面で共有させてしまうと, 回帰係数と特徴ベクトルから求めた期待値の推定値の誤差が大きくなり, 値の信頼性が低くなると予想される. そこでこの点を解決する手段として LinUCB$_{\mathrm{Shared}}$-Rave を提案する. これは異なる局面で値を共有させた LinUCT$_{\mathrm{Shared}}$ で比較的早く求まる値と, 実際のプレイアウトから求まる UCB 値を Rave [5] のように組み合わせて局面を評価する手法である. LinUCB$_{\mathrm{Shared}}$-Rave ではノードの選択で以下の式

$$\beta \cdot \mathrm{LinUCB}(t,a) \ + \ (1-\beta) \cdot \mathrm{UCB1}(t,a) \tag{4}$$

$$\beta \ = \ \sqrt{\frac{k}{3T_{a,s}+k}} \tag{5}$$

を用いる.

また, LinUCB を計算する際には $d \times d$ 次正方行列の逆行列を計算する. よって逆行列を求めるには $O(d^3)$ の計算量が必要となるため, 特徴空間の増加につれ計算量も増大し, 実行時間の面で不利となる. しかし A の非対角成分の寄与はあまり大きくなく, 対角成分のみを用いたものでも精度にあまり影響しないことが知られている [6] [4].

3.3 LinUCB のシミュレーションへの応用

最後に, 学習した回帰係数 $\hat{\theta}$ による勝率予測をプレイアウト中の着手の精度向上にも活用する LinUCB$_{\mathrm{Shared}}$-Policy を提案する. この手法ではプレイアウト中, 行動 a_i を選択する確率を

$$\pi(a_i) = \frac{\hat{\boldsymbol{\theta}}_{a_i}^T \boldsymbol{x}_{a_i}}{\sum_a \hat{\boldsymbol{\theta}}_a^T \boldsymbol{x}_a} \tag{6}$$

とし, 勝率が高いと予想される着手が高い確率で着手される.

$\hat{\boldsymbol{\theta}}_a$ はプレイアウトを行う際にはすでに計算されているが, 特徴ベクトル $\boldsymbol{x}_a$ はプレイアウト中の行動選択のたびに特徴を抽出しなければならないため, 大きな次元を持つ特徴ベクトルは計算時間とのトレードオフとなる.

4. 実験

4.1 人工木による評価

4.1.1 人工木の概要

LinUCB を用いたモンテカルロ木探索が実際のゲームでどのような振る舞いをするかを検証するために, 仮想的なゲーム木を用いた性能評価を行った. 今回は二人ゲームを模式化した人工木で評価を行った.

このゲーム木では, ゲーム木中の各ノードが二種類の d 次元二値特徴ベクトル $\boldsymbol{x}_m$, $\boldsymbol{x}_o$ を持ち, その局面からの得られる報酬が確率

$$P(r=1) = \boldsymbol{\theta}^* \boldsymbol{x}_m^T - \boldsymbol{\theta}^* \boldsymbol{x}_o^T + 0.5 \quad (7)$$

$$P(r=0) = -\boldsymbol{\theta}^* \boldsymbol{x}_m^T + \boldsymbol{\theta}^* \boldsymbol{x}_o^T + 0.5 \quad (8)$$

に従うと定義する. ここで $\boldsymbol{\theta}^* = (\theta_1^*, \theta_2^*, ..., \theta_d^*)$ は d 次元のベクトルであり, $\theta_i^* \in [-1.0/d, 1.0/d]$ である. $\boldsymbol{x}_m$ は自分にとって有利な特徴, $\boldsymbol{x}_o$ は相手にとって有利な特徴を表現しており, 手番によって勝率が変化し, 親子間は特徴が似ているという状況が生ずる. 各ノードの子ノードの特徴ベクトルは親ノードの特徴ベクトル $\boldsymbol{x}_m$, $\boldsymbol{x}_o$ を確率 p で変化させた $\boldsymbol{x}'_m$, $\boldsymbol{x}'_o$ をそれぞれ新たな $\boldsymbol{x}_o$, $\boldsymbol{x}_m$ として設定する.

図 1 にこの人工木の模式図を示す.

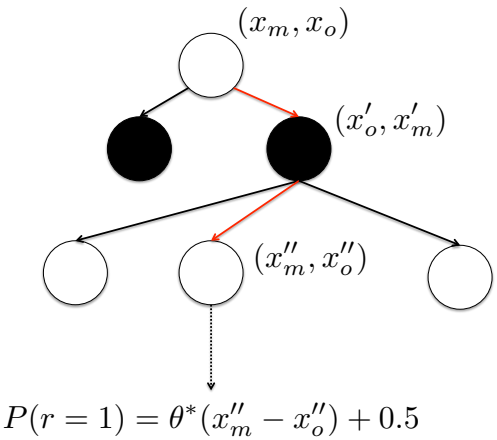

図 1 実験で用いた人工木の模式図

4.1.2 結果

このゲーム木をそれぞれの Selection Policy で探索させ, 振る舞いの傾向を検証した. 実験では $d = 10$, $p = 0.01$ とし, 1000 回のイテレーションを 1 セットとして行った. セット毎に木を作成し, イテレーション毎における到達した葉ノードの真の期待値の累積, $\boldsymbol{\theta}^*$ と推定した $\hat{\boldsymbol{\theta}}$ の差の L^2 ノルム, 実際に推定した期待値と真の期待値との差の絶対値を記録し, 1000 セットの平均を算出した.

また A の対角成分のみを用いるものの性能とすべての成分を用いたものの性能とを比較するために $\mathrm{LinUCT}_{\mathrm{RootShared}}$ の分散共分散行列 A を対角成分のみとした $\mathrm{LinUCT}_{\mathrm{RootShared}}(\mathrm{Diagonal})$ を実験に加えた.

図 2 はイテレーション毎における, 到達した葉ノードの $\theta^*(\boldsymbol{x}_m - \boldsymbol{x}_o)$ の平均値の累積グラフである. $\theta^*(\boldsymbol{x}_m - \boldsymbol{x}_o)$ は負の値になることもあるため, 累積和が減少している Selection Policy もある. 最も性能を発揮したのは $\mathrm{LinUCT}_{\mathrm{Shared}}$-Rave で, 特徴を考慮できていない UCT は当然だが性能が悪い結果となった. LinUCT は前述の通り本質的に UCT と同様の値を計算して選択していくため, 傾向として似たものになった. 一方値を共有する $\mathrm{LinUCT}_{\mathrm{RootShared}}$, $\mathrm{LinUCT}_{\mathrm{Shared}}$-Rave では試行していないノードの期待値も推定しているため, 回数を経るごとに良いノードへ到達できている. また対角成分のみを用いたものも多少精度が悪くなるが傾向としては分散共分散行列の要素全てを用いるものと同様になった.

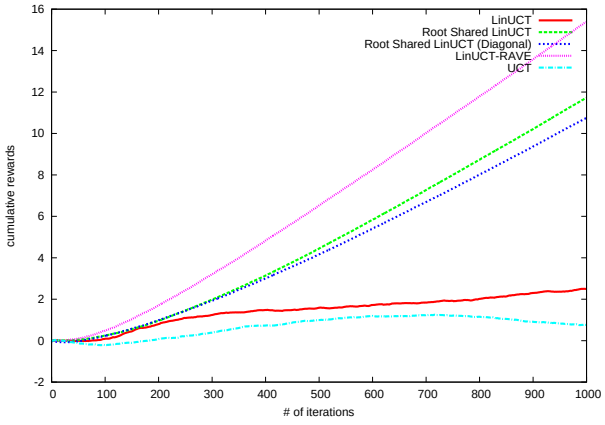

図 2 $\theta^*(\boldsymbol{x}_m - \boldsymbol{x}_o)$ の累積グラフ

図 3 は同じくイテレーション毎の, ルート局面において推定された期待値のゆらぎ $\hat{\boldsymbol{\theta}}(\boldsymbol{x}_m - \boldsymbol{x}_o)$ と真の期待値のゆらぎである $\boldsymbol{\theta}^*(\boldsymbol{x}_m - \boldsymbol{x}_o)$ との差の絶対値 $|(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*)(\boldsymbol{x}_m - \boldsymbol{x}_o)|$ である. また図 4 はイテレーション毎の, ルート局面において回帰を行った $\hat{\boldsymbol{\theta}}$ と真の値である $\boldsymbol{\theta}^*$ の差の L^2 ノルム $\|\boldsymbol{\theta}^* - \hat{\boldsymbol{\theta}}\|_2$ である.

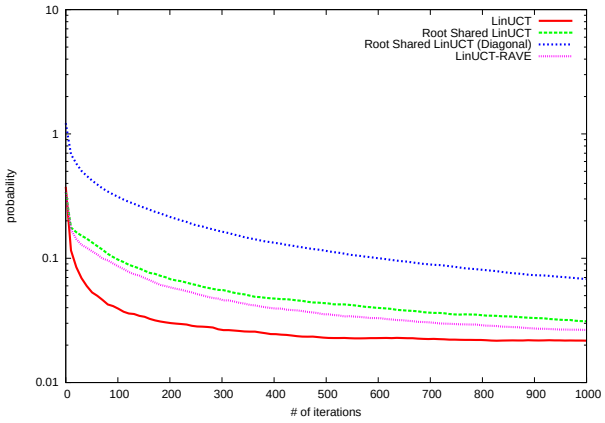

図 3 イテレーション毎の $|(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*)(\boldsymbol{x}_m - \boldsymbol{x}_o)|$ (log-scale)

期待値の差である図 3 から, どの手法も真の期待値を正しく回帰できることが分かり, その中でも各ノードで独立に回帰を行う LinUCT は最も誤差が少なくなった. しかし L^2 ノルムを比較すると, 回帰係数を共有させた手法ではイテレーションを重ねるごとに減少する傾向が見られるが, 独立に回帰を行う LinUCT ではイテレーションの早い段階で値の収束が起きており, 回帰係数が正しく学習できていない.

最後に, 図 5 は UCT, $\mathrm{LinUCT}_{\mathrm{Shared}}$, そして分散共分散

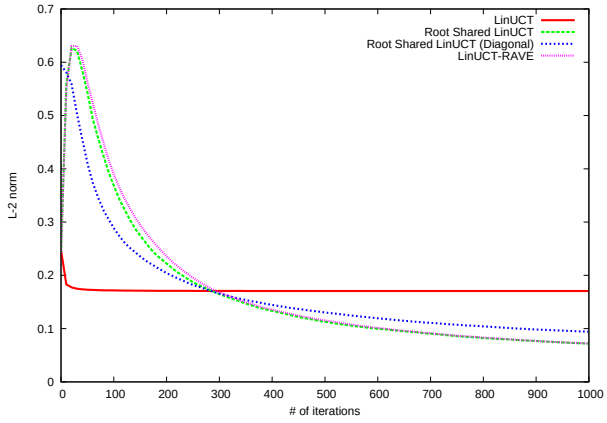

図 4 イテレーション毎の $\|\boldsymbol{\theta}^* - \hat{\boldsymbol{\theta}}\|_2$

行列の対角成分のみを用いる $\text{LinUCT}_{\text{Shared}}$ (Diagonal) の三つについて, 特徴ベクトルの次元 d を 10, 30, 50, 70 に設定した際の, 1000 イテレーションにかかった時間の平均である.

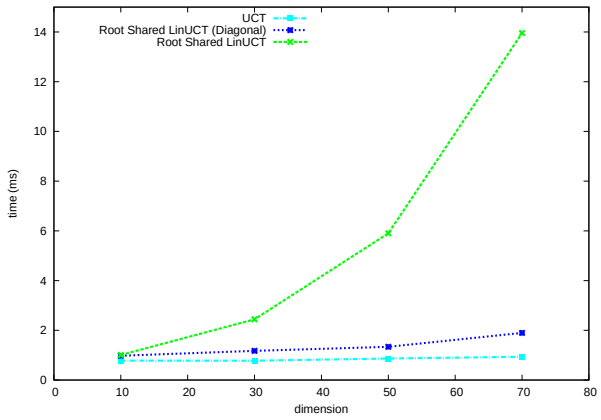

図 5 1000 イテレーションの平均実行時間 (ミリ秒)

UCT, $\text{LinUCT}_{\text{Shared}}$ (Diagonal) と比較して逆行列計算の計算量が大きい $\text{LinUCT}_{\text{Shared}}$ はやはり次元の増加につれ計算時間も増大してしまっているが, 対角成分のみを保存した $\text{LinUCT}_{\text{Shared}}$ (Diagonal) では逆行列の計算が $o(d)$ で行えるため, UCT と比較してほとんど差がない. 図 2 の結果も踏まえると分散共分散行列の非対角成分を考慮する必要はほとんどないと考えてもよいと思われる.

4.2 オセロによる評価

具体的なゲームとして提案手法をオセロを題材に評価を行った. オセロを題材とした理由は盤面の特徴がわかりやすいため特徴ベクトルが直感的に設計しやすいことと, 先行研究において UCT の有効性が示されているためである [7].

以下の実験において, 特徴ベクトルは,

- 確定石の数 (自分/相手)
- C square (a2, b1 など) の数 (自分/相手)
- X square (b2, b7 など) の数 (自分/相手)
- 定数 (1)

の 7 次元のベクトルを用いた.

はじめに, 回帰係数の値がプレイアウトを行うごとにどのような傾向で変化するかを観察した. 図 7 は図 6 の局面から探索を行った際の, $\text{LinUCT}_{\text{RootShared}}$ のそれぞれの特徴ベクトルの成分に対応する回帰係数の変化の様子である.

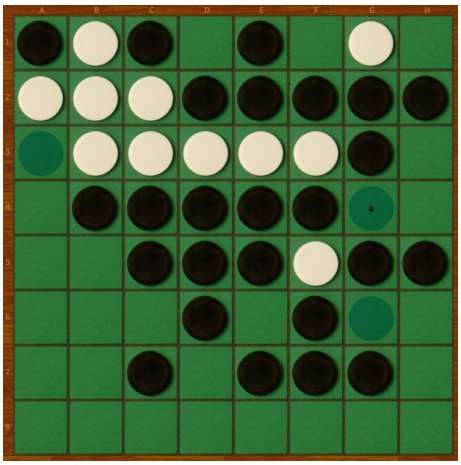

図 6 探索開始局面

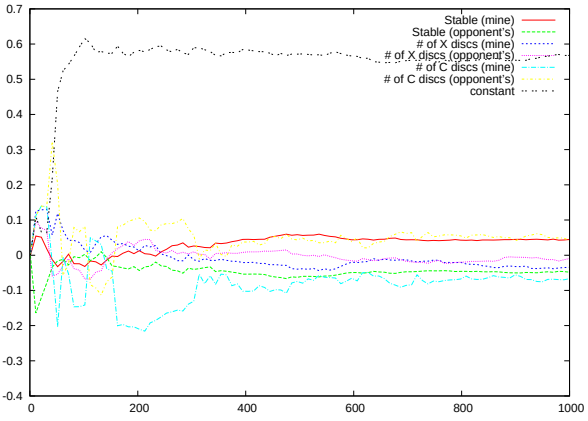

図 7 回帰係数の変化の傾向

この局面においては候補手が a3, g4, g6 の三つ存在し, それぞれの特徴ベクトルの要素は図 7 凡例上から順に

- a3: (3, 0, 2, 0, 1, 1, 1)
- g4: (1, 0, 2, 0, 1, 1, 1)
- g6: (1, 0, 2, 0, 1, 1, 1)

となる. すなわち LinUCB は勝率が最も高いのは a3 であると判断する. 定数に対応する係数が 0.5 程度に収束し, 自分に有利となる特徴 (自分の確定石や相手の X, C square) は正, 不利となる特徴は負の値になり, 概ね妥当な値であると考えられる.

次に UCT と対戦を行い, 性能の比較をした. 対戦は先手後手交代で計 500 試合行った. プレイアウト数は一手ご

とに 3000 回とした.

また今回の実験では特徴空間が 7 次元と狭く, (3) 式をそのまま用いると特徴ベクトルが全く同じノード同士の第二項が全く同じになってしまう問題が生じ, 選択がうまく機能しなかったため, 以下の式に変更した.

$$\mathrm{LinUCB}(t,a) = \hat{\boldsymbol{\theta}}_a \boldsymbol{x}_a^T + \alpha\sqrt{\frac{T_{a,s}}{T_a}\,\boldsymbol{x}_a^T \boldsymbol{A}_a^{-1}\boldsymbol{x}_a} \qquad (9)$$

ここで $T_{a,s}$ は値を共有している局面すべての訪問回数の和, T_a はその局面の訪問回数である. これによって訪問回数が相対的に少ないノードにもある程度プレイアウトを割くことができる. $\mathrm{LinUCT}_{\mathrm{RootShared}}$-Rave で用いる式 (4) においてもこの式を用いた.

式 (4) 中の α は 1.0, 式 (4) 中の k は 1000 として実験を行った. 表 1 に対戦結果の勝率と勝率 0.5 とした場合の両側二項検定の p-value を示す.

表 1 UCT との比較実験

	勝率	p-value
LinUCT	0.464	0.117
$\mathrm{LinUCT}_{\mathrm{RootShared}}$	0.132	2.01E-67
$\mathrm{LinUCT}_{\mathrm{RootShared}}$-Rave	0.480	0.396
$\mathrm{LinUCB}_{\mathrm{RootShared}}$-Policy	0.558	8.26E-3

またそれぞれの手法について, 一手ごとの思考時間の平均をグラフにした図を図 8 に示す.

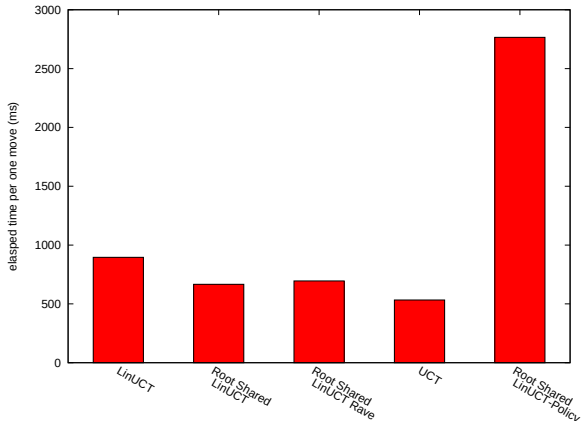

図 8 一手ごとの平均思考時間 (ms)

LinUCT は予想通り UCT と比較的近い成績となった. $\mathrm{LinUCT}_{\mathrm{RootShared}}$ は UCT を大きく下回る結果となった. 原因として, 今回用いた特徴空間は 7 次元と狭く, また LinUCB の性質上全く同じ特徴ベクトルを持つ腕同士は LinUCB の値も同じになってしまう. 故に今回の実験では選択する際に各ノード間の比較がうまく行かなかったためであると考えられる. $\mathrm{LinUCB}_{\mathrm{RootShared}}$-Policy は 7 次元という少ない特徴を用いたにもかかわらず有意に勝ち越しているが, シミュレーションの方策を決定する際にはある程度の計算速度も求められるため, あまり大きな次元を持つ特徴ベクトルになると手の特徴抽出にも期待値の計算にも時間がかかりすぎてしまうため, どのような特徴を用いるかについては調整が難しいと考えられる. 思考時間については, 今回実装したものは 7 次元という少ない特徴ベクトルの次元であったため, プレイアウト中の行動決定をランダムに行っているものは大きな差ができなかった. 各腕が独立に回帰を行う LinUCT は, 値を共有して回帰を行うものとくらべてわずかだが遅くなることが確認できる. またプレイアウト中に逐次局面の特徴を抽出して期待値を計算している $\mathrm{LinUCB}_{\mathrm{RootShared}}$-Policy は一番早い UCT と比較して約 5 倍程度遅くなっている. 特徴ベクトルの次元が 7 次元という少ない場合であるため, これ以上特徴空間の次元を増やして実行するには困難であることが予想される.

5. おわりに

本研究ではモンテカルロ木探索に特徴を用いた LinUCB を用いる新たな手法を提案し, 仮想的なゲーム木を用いてその基礎的な評価を行った. 評価からは, ゲームにおいてさらなる展望が期待できる結果が得られた.

また具体的なゲームの例としてオセロを題材に実装して既存手法である UCT との比較を行った. 今回の結果からは, まだ十分に用いる特徴等を吟味することができなかったため, UCT と比較して大きく劣る結果となったが, 今後のための様々な知見を得ることができた.

LinUCB が推定する勝率の期待値を用いることで, プレイアウト中の方策を向上させることができたが, プレイアウトにかかる時間について課題が残る.

今後はオセロにおいてより多くの特徴を用いることや, その他のゲームを題材に LinUCT などの性能評価を行う.

参考文献

[1] Auer, P., Cesa-Bianchi, N. and Fischer, P.: Finite-time analysis of the multiarmed bandit problem, *Machine learning*, Vol. 47, No. 2-3, pp. 235–256 (2002).

[2] Bubeck, S. and Cesa-Bianchi, N.: Regret analysis of stochastic and nonstochastic multi-armed bandit problems, *arXiv preprint arXiv:1204.5721* (2012).

[3] Li, L., Chu, W., Langford, J. and Schapire, R. E.: A contextual-bandit approach to personalized news article recommendation, *Proceedings of the 19th international conference on World wide web*, ACM, pp. 661–670 (2010).

[4] 中張遼太郎，水上直紀，　浦晃，　三輪誠，鶴岡慶雅，近山隆：LinUCB の 1 人麻雀への適用，ゲームプログラミングワークショップ 2013 論文集，pp. 114–117 (2013).

[5] Gelly, S. and Silver, D.: Monte-Carlo tree search and rapid action value estimation in computer Go, *Artificial Intelligence*, Vol. 175, No. 11, pp. 1856–1875 (2011).

[6] Crammer, K. and Gentile, C.: Multiclass classification with bandit feedback using adaptive regularization, *Machine learning*, Vol. 90, No. 3, pp. 347–383 (2013).

[7] Hingston, P. and Masek, M.: Experiments with Monte Carlo Othello (2007).