数理解析研究所講究録
第1744巻 2011年 56-59

2010年度冬のLAシンポジウム[11]

確率的評価値をもつゲーム木における最善手探索

奥山 洋平* 畑埜 晃平† 瀧本 英二† 竹田 正幸†

2011年2月2日

1 はじめに

近年，モンテカルロシミュレーションを囲碁に取り入れることにより，コンピュータ囲碁が急速に強くなっている．中でも特にMoGo[2]は，一般的な囲碁の１９×１９という盤面のサイズより小さい９×９の盤面であるが，プロに勝てるほどである．これらモンテカルロシミュレーションに基づく手法は，与えられた局面における自プレイヤーの「勝率」をモンテカルロシミュレーションにより推定し，得られた推定値をその局面の評価値とする．ここで,「勝率」とは，その局面から両プレイヤーがランダムに手を打ち続けたときに，自プレイヤーの勝ち局面に至る確率である．また，現局面から終局面に至る両プレイヤーのランダムな一連の試行をプレイアウトと呼ぶ．モンテカルロシミュレーションにおいては，プレイアウトを何回も繰り返すことにより，勝率を推定するのである．

実際の対戦においては，現局面を根として数手先まで展開したゲーム木（部分ゲーム木）上でミニマックス探索を行い，ミニマックスの意味で最も評価値の高い次局面を次の１手として選択する．このとき選ばれた手は，局面の評価値として勝率そのものを用いた場合のミニマックス解の良い近似解となって欲しい．これを実現する素朴な方法は，部分ゲーム木のすべての葉における勝率を十分高い精度で推定した後，ミニマックス探索を行うことである．しかし，葉の勝率に著しい差がある場合には，いくつかの葉における勝率の推定精度は低くてもよいため，モンテカルロシミュレーションにおけるプレイアウト数を小さくすることができる．

特にゲーム木上での探索手法UCTアルゴリズム[3]は，葉ごとの勝率の推定精度を動的に調整できるという性質を持つ．したがって，葉の勝率に差がある場合，より少ないプレイアウト数でミニマックス解を近似できる．実際，MoGoをはじめ，近年のモンテカルロ囲碁の多くにUCTが採用されている．しかし，UCTの性能の解析において，ε近似の概念を用いた厳密な定式化がされておらず，また，理論保証の部分で強い仮定を用いており，何回プレイアウトすれば求めたい解に収束するのか解析がされていない．

そこで本研究では，問題を簡単にするために２手先まで展開した部分ゲーム木において，最適解であるミニマックス解の近似解を求めるというオフライン問題を設定する．その問題に対し，強い仮定を用いないRUCTアルゴリズム(Robust UCT Algorithm)を提案し，プレイアウト数の評価を行う．結果として，素朴な方法では，$\tilde{O}\left(\frac{1}{\varepsilon^2}\right)$のプレイアウト数（$\varepsilon$は誤差のパラメータ）かかるのに対し，RUCTでは，葉の勝率の差が大きい場合に，$\tilde{O}\left(\frac{1}{\varepsilon}\right)$のプレイアウト数で済むことを示す．

2 問題設定

本節では，本研究の問題設定を行う．

*九州大学大学院システム情報科学府
†九州大学大学院システム情報科学研究院

2.1 ゲーム木

本研究では2手先まで展開した部分ゲーム木を，葉にオラクルがついた深さ2のゲーム木 $\mathcal{T}$ として次のようにモデル化する．ゲーム木の根は K 個の子ノードをもち，それぞれ $1,2,...,K$ のラベルを付ける．ノード $i \in [K]$ には葉が K_i 個ついており，それぞれ $(i,1),(i,2),...,(i,K_i)$ のラベルを付ける．各葉 (i,j), $i \in [K], j \in [K_i]$ には対応するオラクル $O_{i,j}$ が存在する．オラクル $O_{i,j}$ は，(引数なしで) 呼び出されると，アルゴリズムにとっては未知の期待値 $\mu_{i,j}$ のベルヌーイ分布にしたがって0または1の報酬を返す．オラクル $O_{i,j}$ は元のゲーム木において，2手先のノード (i,j) を起点とするモンテカルロシミュレーションのプレイアウトに対応し，期待値 $\mu_{i,j}$ はノード (i,j) の勝率を表す．

2.2 ゲーム木の最善手探索問題

ゲーム木の最善手探索問題を以下のように定義する．

定義 2.1 (ゲーム木の最善手探索問題)．ゲーム木の最善手探索問題とは，入力として葉にオラクルをもったゲーム木 $\mathcal{T}$ と，誤差に関するパラメータ $\varepsilon\,(>0)$ が与えられるとき，以下の不等式を満たすノード $\{1,...,K\}$ 上の分布 $\mathbf{P} = (P_1,...,P_K)$ を出力することである．

$$\max_{i\in[K]} \min_{j\in[K_i]} \mu_{i,j} - \mathbf{E}\left[\sum_{i=1}^{K}\left\{P_i \min_{j\in[K_i]} \mu_{i,j}\right\}\right] \leq \varepsilon$$

ここで，$\mathbf{P}$ はオラクルの返す値によって決まる確率変数であり，不等式中の期待値はその確率空間の下で平均をとったものである．

$\max_{i\in[K]} \min_{j\in[K_i]} \mu_{i,j}$ をゲーム木 $\mathcal{T}$ のミニマックス値と呼び，μ^* と表す．また，ミニマックス値を与えるノード $i = \arg\max_{i\in[K]} \min_{j\in[K_i]} \mu_{i,j}$ をミニマックス解と呼ぶ．

この問題を解く素朴な方法は，各オラクル $O_{i,j}$ を $\tilde{O}\left(\frac{1}{\varepsilon^2}\right)$ 回ずつ呼び出すことにより，$\mu_{i,j}$ の $O(\varepsilon)$ 近似となる値 $\hat{\mu}_{i,j}$ を求め，$\{\hat{\mu}_{i,j}\}$ に基づくミニマックス解 $i^* = \arg\max_{i\in[K]} \min_{j\in[K_i]} \hat{\mu}_{i,j}$ を最善手として出力することである．(厳密には，$P_{i^*} = 1, P_j = 0\,(j \neq i^*)$ となる分布 $\mathbf{P} = (P_1,...,P_K)$ を出力する．) よって，$m \times size(\mathcal{T}) = \tilde{O}\left(\frac{size(\mathcal{T})}{\varepsilon^2}\right)$ 回のオラクル呼び出しで，ゲーム木の最善手探索問題を解くことができる．ただし，$size(\mathcal{T})$ は，ゲーム木の葉の数を表す．

3 提案手法

ゲーム木の最善手探索問題をゲーム木の下段 (ノード $i \in [K]$) と上段 (ゲーム木の根) に分けて考える．

3.1 ゲーム木の下段

ゲーム木の下段では，各ノード $i \in [K]$ において，葉のオラクル $O_{i,1},...,O_{i,K_i}$ の中から報酬の期待値が最小のオラクルと似た振る舞いをする，仮想的なオラクルを作る問題を考える．この問題は，多腕バンディット問題に帰着でき，多腕バンディット問題を解く UCB アルゴリズム [1] を利用することができる．このようにして構成したノード i の疑似オラクル UCB-oracle(i) を Algorithm 1 に示す．

定理 3.1. UCB-oracle(i) を s 回目に呼び出したときに返される値を X_s とする．このとき，

$$\mathbf{E}(X_s) \in \left[\mu_i, \mu_i + (1+o(1))\,4\sqrt{\frac{2K_i \ln s}{s}}\right]$$

が成り立つ．ただし，$\mu_i = \min_{j\in[K_i]} \mu_{i,j}$.

定理 3.1 より，ノード i の疑似オラクル UCB-oracle(i) が s 回目に返す値 X_s は，

$$\mu_i \leq \hat{\mu}_{i,s} \leq \mu_i + O\left(\sqrt{\frac{K_i \ln s}{s}}\right)$$

を満たす期待値 $\hat{\mu}_{i,s}$ のベルヌーイ分布に従うことが分かる．$\hat{\mu}_{i,s} \to \mu_i\,(s \to \infty)$ より，UCB-oracle(i) は十分大きい回数呼び出された後は，報酬の期待値が最小のオラクル $O_{i,j}$ $(\mu_{i,j} = \mu_i = \min_{j\in[K_i]} \mu_{i,j})$ を模倣するとみなすことができる．

Algorithm 1 UCB-oracle(i)

$z_j = 0, s_j = 0, t = 0$ $(j = 1, 2, ..., K_i)$

z_j : オラクル $O_{i,j}$ が返した報酬の和を表す静的変数[1]

s_j : オラクル $O_{i,j}$ を選んだ回数を表す静的変数

t : UCB-oracle(i) が呼び出された回数を表す静的変数

if $\exists j, s_j = 0$ **then**

$s_j = 0$ を満たす j を選ぶ

else

$$j = \underset{j=1,2,...,K_i}{\arg\min} \left(\frac{z_j}{s_j} - \sqrt{\frac{2 \ln t}{s_j}} \right)$$

end if

オラクル $O_{i,j}$ を呼び出し，報酬 r を得る

$z_j \leftarrow z_j + r$

$s_j \leftarrow s_j + 1$

$t \leftarrow t + 1$

$\{1, 2, ..., K_i\}$ 上の確率分布 $\left(\frac{s_1}{t}, ..., \frac{s_{K_i}}{t}\right)$ に従いオラクル j を選ぶ

オラクル $O_{i,j}$ を呼び出し，報酬 r を得る

return r

3.2 ゲーム木の上段

ゲーム木の上段では，K 個の疑似オラクル UCB-oracle(1),...,UCB-oracle(K) の中から，真の期待値が μ_i が最大となるノード i を探す問題を考える．ここで，UCB 用いてこの問題を解くことを考える．UCB の解析には Hoeffding の不等式を用いているが，この不等式は，報酬の系列が独立で，報酬の期待値は固定の場合に用いることができる．しかし，ゲーム木の下段で作った疑似オラクルが返す報酬の系列は独立ではなく，疑似オラクルの持つ報酬の期待値は呼び出しごとに変動する．したがって，UCB の解析結果をそのまま今回の問題の解析に用いることはできない．そこで，新たに報酬の系列が独立でなく，報酬の期待値が変動する場合に用いることができる拡張版の Hoeffding の不等式を提案する．これを用いて，今回の問題のために UCB-oracle を組み込んだ拡張版 UCB の提案と解析を行う．拡張版の Hoeffding の不等式を付録の定理 A.1 に，UCB-oracle を組み込んだ拡張版 UCB である RUCT(Robust UCT) アルゴリズムを Algorithm 2 に示す．

Algorithm 2 RUCT

初期化 : $z_i = 0, s_i = 0$ $(i = 1, 2, ..., K)$

z_i : ノード i での累積報酬

s_i : ノード i を選んだ回数

for $t = 1$ to T **do**

if $\exists j, s_j = 0$ **then**

$s_j = 0$ を満たす j を選ぶ

else

$$j = \underset{j=1,2,...,K}{\arg\max} \left(\frac{z_j}{s_j} + 25\sqrt{\frac{2K_i \ln t}{s_j}} \right)$$

end if

UCB-oracle(j) を呼び出し，報酬 r を得る

$z_j \leftarrow z_j + r$

$s_j \leftarrow s_j + 1$

end for

$\mathbf{P} = \left(\frac{s_1}{T}, ..., \frac{s_K}{T}\right)$

return $\mathbf{P} = (P_1, ..., P_K)$

RUCT において，一回の UCB-oracle の呼び出しに対し，葉のオラクル $\{O_{i,j}\}$ のどれかがちょうど一回呼び出されるので，T がオラクル呼び出し回数を表す．

定理 3.2. $\mu_i = \min_{j \in [K_i]} \mu_{i,j}$，$\mu^* = \max_{i \in [K]} \mu_i$ とすると，*RUCT* の返す分布 $\mathbf{P} = (P_1, ..., P_K)$ は以下の不等式を満たす．

$$\mu^* - \mathbf{E}\left[\sum_{i=1}^{K} P_i \mu_i\right] \leq 2312 \left[\sum_{i:\mu^* > \mu_i} \frac{K_i \ln T}{(\mu^* - \mu_i) T}\right] + o(1)$$

定理 3.2 により，深さ 2 のゲーム木に対して，RUCT を用いたとき，オラクル呼び出し回数

$$T = \tilde{O}\left(\frac{size(T)}{\varepsilon \min_{i:\mu^* > \mu_i} (\mu^* - \mu_i)}\right)$$

[1] アルゴリズムが終了しても値を保持しておく変数

で最善手探索問題を解くことができる．また，$\varepsilon \geq \mu^* - \mu_i$ の場合，RUCT は $\tilde{O}\left(\frac{size(T)}{\varepsilon^2}\right)$ を越えないオラクル呼び出し回数で最善手探索問題を解けることを示せる．

3.3 深さ d のゲーム木への拡張

深さ2のゲーム木と同様の考え方で，ゲーム木の葉から順に疑似オラクルを作っていく．今回の提案した RUCT アルゴリズムは深さ2の部分ゲーム木に対するアルゴリズムであるので，深さが d の場合はアルゴリズム内のオラクルを選ぶ式を，疑似オラクルの報酬の期待値が最大のものを選ぶ場合は，

$$\arg\max_{j \in children(u)} \left(\frac{z_j}{s_j} + (12d+1)\sqrt{\frac{2size\left(\mathcal{T}_j\right)\ln t}{s_j}} \right)$$

最小のオラクルを選ぶ場合は，

$$\arg\min_{j \in children(u)} \left(\frac{z_j}{s_j} - (12d+1)\sqrt{\frac{2size\left(\mathcal{T}_j\right)\ln t}{s_j}} \right)$$

に変更する．ただし，$size\left(\mathcal{T}_j\right)$ はノード j を部分ゲーム木の根としたときの葉の数とする．

定理 3.3. 深さが d の部分ゲーム木に対して *RUCT* アルゴリズムを用いる．*RUCT* の返す分布 $\mathbf{P} = (P_1, ..., P_K)$ は以下の不等式を満たす．

$$\left| \mu^* - \mathbf{E}\left[\sum_{i=1}^{K} P_i \mu_i \right] \right| \leq \frac{8\left(16^d - 1\right)^2}{225} \left[\sum_{i:|\mu^* - \mu_i| > 0} \frac{size\left(\mathcal{T}_i\right)\ln T}{|\mu^* - \mu_i| T} \right] + o(1)$$

4 今後の課題

UCB アルゴリズムの評価の改善により，RUCT アルゴリズムの評価も改善が期待できる．これにより，オラクルの呼び出し回数をさらに減らすことができると考えられる．

参考文献

[1] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, Vol. 47, pp. 235–256, 2002.

[2] Sylvain Gelly, Yizao Wang, Rémi Munos, and Olivier Teytaud. Modification of UCT with Patterns in Monte-Carlo Go. Research Report RR-6062, INRIA, 2006.

[3] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *ECML-06*, 2006.

付 録

A 拡張版 Hoeffding の不等式

定理 A.1 (拡張版 Hoeffding の不等式)**.** $X_1, ..., X_T$ を $\{0,1\}$ を値に持つ T 個の確率変数とする．ただし，任意の $1 \leq t \leq T$ に対し，実数 $\mu_{L,t} < \mu_{H,t}$ が存在して，任意の $a_1 ... a_{t-1} \in \{0,1\}^{t-1}$ に対し，$\mu_{L,t} \leq \mathbb{E}\left[X_t | X_1 = a_1, ..., X_{t-1} = a_{t-1}\right] \leq \mu_{H,t}$ を満たす．このとき，すべての $c > 0$ に対して以下の不等式が成り立つ．

$$\mathbf{P}\left[\frac{1}{T}\sum_{t=1}^{T} X_t > \frac{1}{T}\sum_{t=1}^{T} \mu_{H,t} + c \right] \leq \exp\left(-2c^2 T\right)$$

$$\mathbf{P}\left[\frac{1}{T}\sum_{t=1}^{T} X_t < \frac{1}{T}\sum_{t=1}^{T} \mu_{L,t} - c \right] \leq \exp\left(-2c^2 T\right)$$