囲碁AIで使う補正ありUCBアルゴリズムの性能改善

楊 晨[1,a)]　金子 知適[1]

概要：近年多くの囲碁プログラムはUCT（UCB for Tree）を使っているが、UCB1値の代わりにRaveなどの実用上の補正ありアルゴリズムを利用する囲碁プログラムは少なくない。元々UCBは多腕バンディット問題から出た解決法であり、Regretという指標が重視されている。解決法の中にKLUCBアルゴリズムはKL情報量から、Regret理論的な下限を満たしている。PGAMEなどの実験で、実際にRegret減少に有効だということが証明された。本研究ではKLUCB法で補正ありUCBアルゴリズムを改良し、UCTでよりRegretの小さいUCB値を求めることによって、囲碁AI性能とRegretの関係を議論する。

An Improved Algorithm for Go Program with Adjusted UCB Value

Yang Chen[1,a)]　Kaneko Tomoyuki[1]

Abstract: Most of the recent Go computer programs use Monto-Carlo Tree Search(UCT), but many of those programs actually use adjusted UCB value like Rave instead of UCB1 value. As a solution of the multi-armed bandit problem, "Regret" is very important to performance evaluation of a ucb algorism. KLUCB algorithm achieved the lower bound of "Regret", by using Kullback-Leibler divergence. In experiments such as the PGAME, it's actually effective to reduce the Regret. By improving the adjusted ucb algorisom with KLUCB ,in this paper we discussed relationship between the AI performance of GO and the "Regret".

1. 初めに

現在のコンピュータ囲碁 [1] では、ほとんどのプログラムがUCT [2] と呼ばれるモンテカルロ木探索の手法を使っている。UCTは、多腕バンディット問題で使われているUCB1値に基づく行動選択手法とゲーム木探索の組み合わせで、ゲーム木の葉の評価としてはランダムシミレーションの結果を利用する。

多腕バンディット問題はRegretという指標で手法を評価し、Regretが小さいほど良い手法とされる。一方、UCB1という手法のRegret上限はバンディット問題の理論的な下限より大きいので、tightではない。また、実際の囲碁プログラムはUCB1を補正して使っている場合が多いが、それらのRegret上限は明らかではない。

本研究では囲碁プログラムが使っているRAVE [3] などの実用上の補正を加味したUCBから、よりRegretの小さいUCB値を求めることによって、囲碁AI性能とRegretの関係を議論する。

2. 関連研究

2.1 Mini-Max 法

チェス、将棋などの完全情報ゲームでよく用いられている探索アルゴリズムの１つである。プレイヤーAが自分に対して有利で結果がMaxとし、相手プレイヤーBに有利の結果がMiniとなる。すべての局面を展開した後、プレーヤーに対する有利なルートを選べていく。

しかし、大体のゲームの探索空間が大き過ぎ、すべての局面を展開することが不可能である。よって実際の探索は時間など制限で有限のDepthを探索し、末端ノードの局面の優劣を評価する。

GnuGOなどの囲碁プログラムも局面の評価関数を使っているが、囲碁の正確な局面評価が困難であり、後で説明するUCTアルゴリズムに超えられていた。

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

2.2 多腕バンディット問題と Regret

多腕バンディット問題は、各アームの報酬は $[0,1]$ で事前に確率分布が決められているがプレイヤにはわからない状況で、プレイヤの利得を最大化する問題である。様々な戦略の良さを議論する指標の一つが Regret であり、それは、ある回数最善のアームを引き続けた場合の報酬から、同じ回数その戦略に従ってアームを引いて得られる報酬の期待値の差を表す。Regret が小さいほど報酬の多い戦略となる。

任意の有効アルゴリズムに対して、バンディット問題の Regret 理論上の下限は 1985 年に Lai と Robbins が求めている [4]。

一方 Regret の上限は手法により異なる。UCB1 は以下で定義される値が最大になるアームを毎回選ぶ戦略である。 [5]

$$\mathrm{UCB1} = \frac{\mathrm{S}_t(a)}{N_t(a)} + \sqrt{\frac{\log(t)}{2N_t(a)}} \tag{1}$$

ここで $N_t(a)$ はアーム a のこれまでの試行回数、$\frac{\mathrm{S}_t(a)}{N_t(a)}$ はアーム a のこれまでの試行の平均利得、t が全アームの試行回数である。

Regret の上限は式 (2.2) で与えられる。

$$\lim_{n\to\infty} \sup \frac{E[R_n]}{\log(n)} \leq \sum_{a:\theta_a<\theta^*} \frac{1}{2(\theta^* - \theta_a)} \tag{2}$$

ここで θ^* は最適なアームの利得の期待値を、θ_a は他のアームを表す。

2.3 UCT

UCT(Upper bound Confidence for Tree) とは、2006 年 CrazyStone の優勝で注目され始め、Mini-Max 探索とランダムシミレーションで構成されたアルゴリズムである。先述した Mini-Max 法の末端ノードの局面は、評価関数の代わりにランダムプレイの勝率で評価する。

探索の流れは以下に示す：

1. 一番有望な手（UCB 値最も高いノード）を選択して降りる

2. 子ノードが存在する場合、1 に戻る

3.1 回ランダムシミレーションする

4.2 回目のランダムシミレーションであれば、子ノードを展開する

5. 得った結果で経路上のノードを更新する

2.4 KL-UCB

KL-UCB [7] は KL 情報量を含む以下の式を用いてアームを選ぶ:

$$\arg \max_{1\leq a\leq K} \max_{q} \left\{ q \in (\theta, 1] : KL(\theta, q) \leq \frac{\log(t)}{N(a)} \right\} \tag{3}$$

ここで a はアームの種類で K 本あるとする。また $KL(,)$ は、KL 情報量である。θ はアーム a のこれまで試行での平均利得で $KL(\theta, q)$ が閾値を越えない q の最大値を書くアームに対して求め、それが最大であるようなアームを選択する。この手法の Regret の下限は以下のように求められている:

$$\lim_{n\to\infty} \inf \frac{R_n(\theta)}{\log(n)} \geq \sum_{a:\theta_a<\theta^*} \frac{\theta^* - \theta_a}{KL(\theta_a, \theta^*)} \tag{4}$$

KLUCB は、UCB1 より Regret の上限が小さいという点で優れるだけでなく、PGAME という仮想ゲームの実験でも、Regret が小さいという実験結果が報告されている。ただし、計算時間は一般には UCB より増える。この点はゲーム木探索に応用する場合においては、試行回数が少ない節点が大多数であることを活かして、事前計算による表引きなどの手法で軽減可能である。

2.5 修正 UCB 値：RAVE

囲碁では多くの場合に着手の順序を入れ替えて手順前後があっても似たような結果になることから、Rapid Action Value Estimation (RAVE) という手法が提案されている [3]。

これは一回のランダムシミレーションで 自分が着手した着手可能点の回数分の勝率の更新が可能であり、手早く着手の良さを推測することができる。Fuego などの囲碁プログラムに実装した結果、有効なアルゴリズムだと思われる。一方、石の取りがあった時、違う順番で同じ所を打っても完全に同じ局面にはならないので、Regret を含めて厳密な解析はなされていないない。

3. 提案手法

本研究では、KL-UCB と囲碁 AI で使っている補正あり UCB 式を組み合わせることを提案する。これにより、Regret が減少し、モンテカルロ木探索の性能が向上することが期待される。

評価には、オープンソースの囲碁プログラムである Fuego [6] を用いている。Fuego の場合、デフォルトの設定で着手選択は

$$(weight \cdot \text{実際の平均勝率} + (1 - weight) \cdot \text{RAVE の平均勝率})$$

が最大になるような着手が選ばれる。ここで「RAVE の平均勝率」は、平均勝率を補正し UCB1 の第二項に似た効果を発揮していると考えられる。

この Fuego の着手選択に KL-UCB の考え方をどのように導入するかについて様々な選択肢が考えられるが、まず RAVE の効果をそのまま残し、第一項で平均勝率の最大値の代わりに一定の信頼性での勝率の上限 (KL-UCB の q) を用いる次式で着手を選ぶ方法を実装した:

$$\arg\max_{1\leq a\leq K}\{ \\ weight * \max_q\{q \in (\theta, 1] : KL(\theta, q) \leq \frac{\log(t)}{N(a)}\} \quad (5) \\ +(1 - weight) * \theta_{Rave}\}$$

4. 実験と評価

4.1 実験条件

効果を測定するために、オープンソースソフトウェアFuegoに実装した。実験は9路盤（コミ六目半、中国ルール）で同じランダムプレイ回数で、違う着手を選ぶアルゴリズムを実装したfuegoバージョンとオリジナルバーションを5000回ゲーム対戦される。

デフォルトではFuegoが定石を使っているが、今回の実験では利用しなかった。また実験は全部1スレッドで、fuegoの並列化（VirtualLoss）も使っていなかった。

4.2 実験結果

表 1 500回シミレーション、白の勝率

	黒 UCB	黒 RAVE
白 UCB	47.0%	/
白 KLUCB	46.2%	/
白 RAVE	/	49.6%
白 KLRAVE	/	48.7%

表 2 1000回シミレーション、白の勝率

	黒 UCB	黒 RAVE
白 UCB	48.4%	/
白 KLUCB	47.1%	/
白 RAVE	/	49.2%
白 KLRAVE	/	49.5%

表 3 3000回シミレーション、白の勝率

	黒 UCB	黒 RAVE
白 UCB	47.2%	/
白 KLUCB	48.2%	/
白 RAVE	/	46.8%
白 KLRAVE	/	47.0%

4.3 考察

多腕バンディット問題ではRegret減少に有効なKLUCBが実験したところ、囲碁にはほとんど効果がない。囲碁と多腕バンディット問題の違いは、囲碁は最後に最善手を選ぶのが目的で、多腕バンディット問題は全体の報酬が大きくするのが目的である。

囲碁にとってRegretの減少は収束スピードに関わると考えていたが、実際の囲碁プログラムの探索中にはRegretの納まりによって「無駄なランダムシミレーション」の数が非常に有限であるのが原因だと考えられる。

5. まとめ

本研究ではKLUCBの方法で補正ありUCBアルゴリズムを改良し、UCTでよりRegretの小さいUCB値を求めることによって、囲碁AI性能とRegretの関係を議論する。

多腕バンディット問題ではKLUCBはRegretの減少に効果があるが、実際囲碁プログラムで場合、実験したところ効果が殆ど無い。実験データに不十分なところもあるが、Regretの減少は囲碁プログラムに対して、重要性が低いと考えられる。

参考文献

[1] Gelly, S., Kocsis, L., Schoenauer, M., Sebag, M., Silver, D., Szepesv ari, C. and Teytaud, O.: The grand challenge of computer Go: Monte Carlo tree search and extensions, Commun. ACM, Vol. 55, No. 3, pp.106-113 (2012).

[2] Kocsis, L. and Szepesvari, C Bandit Based Monte-Carlo Planning, Machine Learning: ECML 2006,Vol. 4212, Springer, pp. 282-293 (2006).

[3] 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).

[4] Lai, T. and Robbins, H.: Asymptotically efficient adaptive allocation rules, Advances in Applied Mathematics, Vol. 6, No. 1, pp. 4-22 (1985).

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

[6] Enzenberger, M. and Muller, M.: Fuego - An Open-source Framework for Board Games and Go Engine Based on Monte-Carlo Tree Search, Technical report, University of Alberta, Dept. of Computing Science,TR09-08 (2009).

[7] Garivier, A. and Cappe, O.: The kl-ucb algorithm for bounded stochastic bandits and beyond, COLT(Kakade, S. M. and von Luxburg, U., eds.), JMLR Proceedings, Vol. 19, pp. 359-376 (2011).