Vol.2011-GI-25 No.2
2011/3/5

ガウス過程を用いたモンテカルロ碁における戦略獲得

荒井　光[†1]　福永修一[†1]

近年，Gaussian Process Bandits (GPB) をモンテカルロ木探索へ適用したアルゴリズムであるガウス過程木探索が提案された．ガウス過程木探索は従来のアルゴリズムよりも性能のよい手法である．そこで本研究では，ガウス過程木探索に基づくモンテカルロ碁を提案する．提案手法は，ガウス過程における共分散行列とカーネル関数を用いて，信頼上限関数を最大化する戦略を選択する．そして提案手法の有効性を数値シミュレーションにより確認する．

Strategy Acquisition in Monte-Carlo Go Using a Gaussian Process

Hikaru Arai[†1] and Shuichi Fukunaga[†1]

Recently, a tree search algorithm using a Gaussian process which applied Gaussian Process Bandits (GPB) to a Monte-Carlo tree search was proposed. It has a better performance than former methods. This paper proposes a Monte-Carlo go based on the tree search algorithm using the Gaussian process. The proposed method employs a strategy which maximizes a upper confidence function using a covariance matrix and a kernel function in the Gaussian process. Numerical simulations show the effectiveness of the proposed method.

†1 東京都立産業技術高等専門学校
Tokyo Metropolitan College of Industrial Technology

1. はじめに

囲碁はボードゲームの中で難しいゲームとされ，探索空間が非常に大きいことから，盤面の価値を評価することが難しいとされている．しかし近年のコンピュータ囲碁の実力はモンテカルロ法を用いることにより，大幅に上昇している．モンテカルロ法を用いた研究は1993 年からされている[1]．考え方としては，乱数を用いて次の手を決定し，ゲーム終了まで行う．ゲーム終了後の盤面により局面の評価を行い勝率を計算し，勝率が高い局面ほど良い局面とし次の手を決定するものである．このような考え方を取り組んだコンピュータ囲碁のことをモンテカルロ碁と呼ぶ．当然，このような考え方のみではモンテカルロ碁はあまり強くはないが，UCT (UCB applied to trees) と呼ばれる木探索アルゴリズム[2] を用いることで，大幅に強くすることに成功している．UCT とは UCB (Upper Confidence Bound) と呼ばれる戦略[3] を用いた木探索アルゴリズムである．UCB は Multi-Armed Bandit 問題にいおいて，計算量を少なく抑えつつ，高い報酬を得ることができるものである．

また，最近 UCB より良い性能を示す Gauusian Process Bandits (GPB)[4),5)] と呼ばれる手法が考案された．この手法は UCB にガウス過程を用いたものである．UCB でははじめの選択と終わり状態のみにより評価を行い次の選択を決定するが，この手法でははじめの選択から終わりまでの選択を採取することにより評価を行い，次の選択を決定する違いがある．また，GPB を木探索アルゴリズムに適用した手法が Dorard らにより提案されている[6]．この手法はダミノードと呼ばれるノードを用いることで,GPB を木探索に有効に適用している．

本研究では，ガウス過程を用いたモンテカルロ碁を提案する．ガウス過程を用いた木探索アルゴルリズムをそのままモンテカルロ碁に適用するのは計算量などで困難なため, アルゴリズムをモンテカルロ碁が適用できるように変更を行った．また, 有効性を検証するためにをランダムな戦略を持つプログラムと対戦を行った．

本論文の構成は以下の通りである. 第 2 節ではモンテカルロ碁について述べる．第 3 節では，本研究で提案するガウス過程を用いたモンテカルロ碁ついて説明する．そして，第 4 節では実験方法ならびに実験結果について示す．第 5 節でまとめと今後の課題について述べる．

2. モンテカルロ碁

最初にモンテカルロ碁の基本となっている UCB について説明する. そして,UCB を木探索へ適用した UCT について説明する．

2.1 UCB

機械学習の分野で扱うわれる問題の一つとして Multi-Armed Bandit 問題と呼ばれるものがある．この問題は腕が複数あるスロットマシーンを対象とする．コインをスロットマシーンにいれ, 複数ある腕のなかから一つ選択すると, 未知の確率分布にしたがって報酬が返される. 試行中, 腕の報酬に関する確率分布は変化しないものとする．この問題の目的は, 与えられたコインを用いて, 最大の報酬を得ることである．どのように腕を選択すれば最大の報酬を得ることができるか, その戦略を考えるのが Multi-Armed Bandit 問題である．

Multi-Armed Bandit 問題の一つの戦略として UCB と呼ばれる戦略が考案されている．この戦略は UCB 値と呼ばれる値を計算し, その値にしたがって腕を選択するものである．UCB 値の計算式はいくつか考えられているが, 手 j の UCB 値は次式でよく表される．また，一度も選択されていない腕の UCB 値は最大とする．

$$UCB(j) = \bar{x}_j + \sqrt{\frac{2\ln n}{n_j}} \tag{1}$$

ここで，$\bar{x}_j$ は j 番目の腕における現時点での報酬期待値，n_j は腕 j を選択したした回数, n はそれまでにスロットマシーンに腕を選択した総回数である．UCB 値は期待値とバイアスの形からなり，期待値のみではグリーディ戦略となるが，バイアスにより探索と搾取を効率的に行うことが可能となる．したがって，UCB 値を用いることで有望そうな手に多くの搾取を割くことができ, また, まだあまり選択されていない腕に対する探索も効率的に行うことが可能となる．

2.2 UCT

UCT は UCB を木探索に適用したものである.UCB では，はじめの選択のみを UCB 値によって決定していたため, 深さ 2 以上の探索については探索回数を多くした場合でも最適な戦略を得ることができなかった．そこで, はじめの選択以降にも UCB 値により選択を行い, そのような探索においても対応できる手法として UCT が考案された.

はじめに,UCB 値を用いてたどるノードを決定する．ここで, たどるノードが一定回数選択されていた場合は, そのノードの子ノードに対しても UCB 値を用いてたどるノードを決定する．そうでない場合は, モンテカルロ法を用いてたどるノードを木の最深にあるリーフノードまで選択する. この動作をプレイアウトと呼ぶ．リーフノードまでたどれたら, そのときの報酬を計算し, 一定回数選択されているたどってきたノードの UCB 値の更新を行う. 以上の動作を繰り返し, 最後にはじめに選択できるノードにおいて,UCB 値が最大のものを次の選択として決定する.

3. ガウス過程を用いたモンテカルロ碁

本節では, 提案手法であるガウス過程を用いたモンテカルロ碁について説明する．最初に,UCB にガウス過程を適用した GPB について説明する．次に,GPB を木探索に適用したアルゴリズムを示す．そして, その木探索アルゴリズムを適用したモンテカルロ碁について説明する．

3.1 Gaussian Process Bandits(GPB)

集合 $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ を考える．$\mathbf{x}$ の報酬を $f(\mathbf{x})$ とし, それにノイズ ϵ が加えられた値 $y = f(\mathbf{x}) + \epsilon$ が観測される．ここで $\epsilon \sim \mathcal{N}(0, \sigma^2_{noise})$ である．

また, ガウス過程は平均関数と共分散関数 $k(\mathbf{x}, \mathbf{x}')$ で定義される．ただし, 平均関数は簡単化のため 0 をとる関数とする．k は $\mathbf{x}$ の要素とのカーネルを意味する．

$(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)$ が与えられた場合, ガウス過程は次式で表される平均 $\mu_t(\mathbf{x})$ と分散 $\sigma^2_t(\mathbf{x})$ を持つ.

$$\mu_t(\mathbf{x}) = \mathbf{k}_t(\mathbf{x})^T C_t^{-1} \mathbf{y} \tag{2}$$

$$\sigma^2_t(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_t(\mathbf{x})^T C_t^{-1} \mathbf{k}_t(\mathbf{x}) \tag{3}$$

ここで,C_t と $\mathbf{k}_t$ は次式で定義される.

$$(C_t)_{i,j} = k(\mathbf{x_i}, \mathbf{x_j}) + \sigma^2_{noise}\delta_{i,j} \tag{4}$$

$$(\mathbf{k}_t(\mathbf{x}))_i = k(\mathbf{x}, \mathbf{x}_i) \tag{5}$$

UCB とは訓練集合 $\mathbf{x} \in \mathcal{X}$ の中から最大の UCB 値をもつものを次の選択とする戦略であった. したがって, 以下のように定式化できる.

$$x_{t+1} = \arg\max_{\mathbf{x} \in \mathcal{X}} \{f_t(\mathbf{x}) = \mu_t(\mathbf{x}) + B(t)\sigma_t(\mathbf{x})\} \tag{6}$$

ここで,$B(t)$ は探索と搾取のバランスを取る. また,$B(t) = 0$ である場合, グリーディアルゴリズムとなる．オリジナルの UCB では $B(t) \sim \sqrt{\log t}$ で定義されている. また,$f_t(\mathbf{x})$ は式 (2),(3) より以下の式で表される.

$$f_t(\mathbf{x}) = \mu_t(\mathbf{x}) + B(t)\sigma_t(\mathbf{x}) \tag{7}$$

$$= \mathbf{k}_t(\mathbf{x})^T C_t^{-1} \mathbf{y} + B(t)\sqrt{k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_t(\mathbf{x})^T C_t^{-1} \mathbf{k}_t(\mathbf{x})} \tag{8}$$

また, 共分散行列は次式で表される.

$$C_t = \begin{pmatrix} C_{t-1} & \mathbf{k}_{t-1}(\mathbf{x}_t) \\ \mathbf{k}_{t-1}(\mathbf{x}_t^T & k(\mathbf{k}_t, \mathbf{k}_t) + \sigma^2_{noise} \end{pmatrix} \tag{9}$$

Gaussian Process Bandits (GPB) とは以上の式から UCB 値を計算し UCB を行う. そして最後にそのなかで UCB 値が最大のものを選択する戦略である. UCB では選択後の結果のみを用いて, はじめに選択されたノードの UCB 値を更新していく. 一方 GPB では, プレイアウトの UCB 値を更新していく違いがある. 似たようなプレイアウトの結果を大きく反映し, そうでないプレイアウトの結果は小さく反映される特徴がある.

3.2 ガウス過程を用いた木探索アルゴリズム

GPB を木探索に適用するためのアルゴリズムを示す. GPB と同様にプレイアウトを行い, そのプレイアウトに対する UCB 値をガウス過程を用いて計算する. ガウス過程を用いた木探索アルゴリズムは GPB と違い, ダミーノードと呼ばれるノードをアルゴリズム中に用いる. ダミーノードとは実際に存在しないノードであるが, ランダムウォーク中に選択されるノードと同じ深さに作成されるノードである. 一度以前に行われたプレイアウトの一部から始まるランダムウォークを効率よく行うことが可能となる.

はじめに, 一番最初にたどられるノードとしてルートノードを作成する. またルートノードのダミーノードを作成し, 集合 $\mathbf{S}$ に加える. また, 訓練集合の最後に選択されたノードであるリーフノードからなる集合と報酬ベクトル, 訓練集合の要素におけるカーネル行列をそれぞれ $\mathbf{X}_t, \mathbf{y}_t, \mathbf{K}_t$ として宣言しておく. また, 繰り返し回数を数える変数 t も 0 で初期化して宣言をしておく. また,UCB 値の計算に必要な共分散行列を , $\mathbf{C}_t$ も宣言しておく. 以上に宣言したものを用いて, ノードの UCB 値の計算が行われる.

今まで行われたプレイアウトにおいて最後に選択されたリーフノードとダミーノードからなる集合 $\mathbf{S}$ から,UCB 値が最大のものを選択する. もしダミノードを選択した場合は, 深さからランダムウォークを行う. ランダムウォークは木の最深 D まで行う. たどりり方は, 現在たどっているノードの一つ深さが深い子ノードである子ノードとダミーノードを作成し. その作成した子ノードをたどっていく. 深さ D に達したら, そのとき選択しているノードを集合 $\mathbf{S}$ に加える.

選択されたノードかならなる訓練集合と集合 $\mathbf{X}_t$ に関するカーネルを計算する. その後, 集合 $\mathbf{X}_t$ に選択された末端ノード x を追加する. また, 報酬を計算し集合 $\mathbf{y}_t$ に追加する. そして,, 訓練集合の要素におけるカーネル行列の $\mathbf{K}_t$ と共分散行列 , $\mathbf{C}_t$ を計算する. 最後にそれらを用いて, 集合 $\mathbf{S}$ の全てのノードに対して,UCB 値の更新を行う. 以上の動作を t が停止基準になるまで行い, 最後に集合 $\mathbf{X}_t$ から UCB 値が最大のリーフノードかなる集合を出力する.

アルゴリズムの詳細は以下の通りである.

[GPB を用いた木探索アルゴリズム]

Step1 初期化

ルートノードとダミーチャイルドを作成する. また, 選択されるノード集合 $\mathbf{S}$ 先ほど作成したダミーたチャイルドを加える. 訓練集合でのリーフノード集合 $\mathbf{X}_t$, 訓練集合の報酬集合を $\mathbf{y}_t$, カーネルマトリクスを $\mathbf{K}_t$, 共分散行列 $\mathbf{C}_t$ をそれぞれ空集合として作成する. そして, 繰り返し回数 t は 0 で初期化を行う.

Step2 パスの選択

選択ノードを格納する x を作成する. t が 0 である場合はルートノード作成時に作成したダミーチャイルドを選択し x に格納する. そうでなければ,x には集合 $\mathbf{S}$ の中で UCB 値が最大のものを選択し x に格納し Step5 へ.

Step3 ランダムウォーク

x の兄弟ノード x' を作成する. また,x の兄弟ノードが全て作成されていたら,x を木と集合 S から取り除く. そして x に x' を格納する.x の深さが D 以上であれば Step5 へ.

Step4 パスの作成

x の子ノード x' とダミーチャイルドを作成する. ダミーチャイルドを S に追加し x に x' を格納する.x の深さが木の最大長 D 未満であれば Step4 へ.

Step5 報酬の獲得訓練集合の追加

x と X_t のカーネルベクトルを計算する. その後に x を X_t に追加する. また,x の報酬を計算し, 計算結果を y に追加する.K_t と C_t を計算する.

Step6 UCB 値の更新

S の中から Step6 に入ってからまだ UCB 値を更新していないノード n を選択する. n と X_t のカーネルベクトルを計算する. そして,$\mathbf{k}$,C_t,$\mathbf{y}_t$ と式 (8) を用いて UCB 値を計算し更新する. S の中でまだ更新していないノードがあれば Step6 へ.

Step7 繰り返し判断

t の値を 1 追加する.t がまだ終了回数に到達していなければ Step2 へ.

Step8 出力

X_t のなかで UCB 値が最大のものを検索し, そのパスを出力する.

上記のアルゴリズムにより,GPB を用いた木探索を行うことができる.

3.3 ガウス過程を用いたモンテカルロ碁

前節で記したアルゴリズムをモンテカルロ碁へ適用する. アルゴリズム中の x は手の選択によるプレイアウトに相当する. また,x の報酬はプレイアウトで作成される局面での勝敗を

利用する.

ダミーノードと今まで行われたプレイアウトの中で最大の UCB 値を持つノードを選択する. 選択したノードがダミーノードであった場合は, その親ノードから始まるランダムウォークを行い, ゲーム終了か深さ D まで手を選択していく. その後, プレイアウトの結果から報酬を得る. 本研究では勝利であれば 1 を負けであれば 0 とした. 今まで行われたプレイアウトと今行われたプレイアウトとのカーネルを計算する. また, グラム行列と共分散行列を計算し今まで行われたプレイアウトの UCB 値を更新する. 以上の選択と計算を十分行い, 最後に今まで行われたプレイアウトの UCB 値が最大のものを選択し, そのプレイアウトのはじめに選択されたノードが保持している位置が次の手となる.

本研究では, ゲームはじめからおわりまでの全てのパスを保持することが困難であったため, アルゴリズム中の深さ D をある一定の数値とし, それ以降のパスは,UCT における未展開ノードであるときと同様にして, モンテカルロ法を用いて作成した.

本研究で提案するアルゴリズムを以下に示す.

[ガウス過程を用いたモンテカルロ碁]

Step1　初期化

前節で示したアルゴリズムと同様にして, ルートノードとそのダミーチャイルドを作成する. また繰り返し回数 t は 0 で初期化を行う.

Step2　初手の選択

選択する手を格納する x を作成する. t が 0 である場合はルートノード作成時に作成したダミーチャイルドを選択し x に格納する. そうでなければ,x には集合 **S** の中で UCB 値が最大の手を選択し x に格納し Step5 へ.

Step3　ランダムウォーク

x の親ノードからルートノードまで遡り, 親ノードまでにできる局面を作成する. そして, その局面からまだ子ノードとして作成されたいない合法手のひとつを子ノードとして x' を作成し,x に x' を格納する. また, 全て合法手 (ルール違反でない手) の子ノード作成した場合は,x を木と集合 S から取り除く.x の深さが D 以上であれば Step5 へ.

Step4　深さ D までの手を選択

x の合法手の中から子ノード x' を作成し, ダミーチャイルドも作成する. ダミーチャイルドを S に追加し x に x' を格納する.x の深さが木の最大長 D 未満であれば Step4 へ.

Step5　報酬の獲得訓練集合の追加

選択された手の系列から局面を作成する. まだ, ゲームが終了していない場合はモンテカルロ法を用いて合法手を選択していきゲームを終了させる. x と X_t のカーネルベクトルを計算し, その後に x を X_t に追加する. また,x の報酬を計算し, 計算結果を y に追加する.K_t と C_t を計算する.

Step6　UCB 値の更新

S の中から Step6 に入ってからまだ UCB 値を更新していない手 n を選択する. n と X_t のカーネルベクトルを計算する. そして,$\mathbf{k}$,C_t,$\mathbf{y}_t$ と式 (8) を用いて UCB 値を計算し更新する. S の中でまだ更新していない手があれば Step6 へ.

Step7　繰り返し判断

t の値を 1 追加する.t がまだ終了回数に到達していなければ Step2 へ.

Step8　出力

X_t のなかで UCB 値が最大のものを検索し, その系列のはじめの手を次に選択するてとして出力する.

上記のアルゴリズムにより,GPB を用いた木探索をモンテカルロ法に適用を行った.

4. 実　　験

GPB を用いた木探索アルゴリズムを適用したモンテカルロ碁の有効性を示すため，ランダムな戦略をもつプログラムとの対戦を行った.

対戦は 9 路盤で先手と後手それぞれ 10 回行った．コミは 7 目半とした．GPB の 1 回の手の選択に行うプレイアウト数はそれぞれ 100 回とした．GPB において, アルゴリズ中の D は 5 と設定した. 実験結果を**表 1** に示す.

実験結果から，GPB を用いた木探索アルゴリズムを適用したモンテカルロ碁が，ランダムな戦略をもつプログラムより勝率が上まっていることがわかる．したがって，GPB を用いた木探索アルゴリズムの方が有効であると考えられる.

表 1　適用手法とランダムな戦略との対戦結果
Table 1　The result of the game between the proposed method and the player with random strategy

	適用手法	ランダムな戦略
先手での勝率 [%]	50	50
後手での勝率 [%]	90	10

Vol.2011-GI-25 No.2
2011/3/5

5. おわりに

本研究では，GPB を用いた木探索アルゴリズムをモンテカルロ碁に適用した．また，ランダムな戦略をもつプログラムと対戦させ，勝率が上まり，その手法の有効性を示すことができた．

今後は，事前知識を追加することによって探索するノードをあらかじめ絞るなどの探索の効率化を図るとともに，UCT で行われている改良を取り組むなどして改良を行っていく．これにより，さらに強い戦略を持つモンテカルロ碁が期待できる．

参 考 文 献

1) B. Brügman，Monte Carlo Go，Technical report, Physics Department，Syracuse University ,1993
2) L. Kocsis and C. Szepesvári, Bandit-based Monte-Carlo planning, Proc. of 15th European Conference on Machine Learning, pp.282 - 293 , 2006
3) P. Auer, N. Cesa-Bianchi and P. Fischer, Finite-time Analysis of the Multiarmed Bandit Problem, Machine Learning, 47(2 - 3), pp.159 - 174 ,2002
4) L. Dorard, D. Glowacka and J. Shawe-Tyalor, Gaussian Process Modelling of Dependencies in Multi-Armed Bandit Problems, In proceedings of te 10th International Symposium on Operational Research (SOR'09) ,2009
5) N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. Proceedings of the International Conference on Machine Learning (ICML2010),2010
6) L. Dorard, J. Shawe-Taylor, Gaussian Process Bandits for Tree Search: Theory and Application to Planning in Discounted MDPs. http://arxiv.org/abs/1009.0605