Efficient Bayesian Optimization with Automatically Selecting Exploration Policies via Slice Sampling

新領域創成科学研究科複雑理工学専攻　2018 年 3 月修了
指導教員：杉山将　教授　学籍番号：47166119　氏名：渡邉大志

キーワード:Bayesian optimization, slice sampling, information gain

1. 研究背景

近年活躍の場が広がってきた多くの機械学習のアルゴリズムにおいて, ハイパーパラメータの値を適切に設定することはその精度の観点から非常に重要である. しかしハイパーパラメータの値の設定は, ランダムサーチやプログラム作成者の経験や勘に頼ることが多い. ハイパーパラメータの検証 1 回に要する時間が長いとき, ランダムサーチは効率が悪く, 手動で探索を行うことはプログラム作成者に手間と時間がかかり, プログラムやアルゴリズムを汎用化させるうえで大きな障害となる.

近年このような問題を打開すべく, ベイズ最適化という手法が提案され注目を集めている [1]. この手法は, 最適解の予測値と, 予測値に対する不確実性を考慮して次の候補点を逐次的に決定する手法である. 本論文では計算可能なコアが複数ある状況を想定し, その状況において目的の機械学習のハイパーパラメータ推定をより効率的に行う手法を提案する.

2. ベイズ最適化 (Bayesian Optimization)

教師あり機械学習モデルにおいて, 最適化したいハイパーパラメータを入力 $\mathbf{x}$, その結果得られる正答率など評価の基準となる値を出力 y とし, すでに N 個の入力と出力の組 $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ が観測済みであるとする. また, その機械学習モデルにおいて用いられている訓練データを d_{all} とする. 機械学習タスクにおいて行われる訓練, テストなどのすべての過程をブラックボックス関数 f とみなし, 入力と出力との間には $y = f(\mathbf{x}|d_{\mathrm{all}}) + \epsilon$(ただし平均 a, 分散 b^2 の正規分布を $\mathcal{N}(a, b^2)$ として, $\epsilon \sim \mathcal{N}(0, \sigma^2)$) の関係が成り立つとする. これにより, 上記の問題は最適なハイパーパラメータ $\mathbf{x}^* = \arg\max_{\mathbf{x}} f(\mathbf{x}|d_{\mathrm{all}})$ を求める問題に帰着される.

入力空間 $\mathcal{X}$ における任意の 2 点 $(\mathbf{x}, \mathbf{x}')$ 間に対して実数値を出力するカーネルを $k(\mathbf{x}, \mathbf{x}')$ で表す. このとき, 関数 f の事前分布がガウス過程であると仮定し, 尤度が正規分布であることを用いると, その事後分布の平均関数 $\mu(\mathbf{x}|\mathcal{D})$ と分散関数 $v(\mathbf{x}|\mathcal{D})$ は次のように与えられる.

$$\mu(\mathbf{x}|\mathcal{D}) = \mathbf{k}^\top (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_{1:N} \tag{1}$$

$$v^2(\mathbf{x}|\mathcal{D}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}^\top (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{k} \tag{2}$$

ただし $\mathbf{A}^\top$ は行列 $\mathbf{A}$ の転置を表す. また, $\mathbf{k} = [k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_N)]^\top$, $\mathbf{y}_{1:N} = [y_1, \ldots, y_N]^\top$ であり, 行列 $\mathbf{K}$ の (i, j) 成分 $\mathbf{K}_{ij}$(ただし i と j は 1 から N までの整数) について, $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ であり, $\mathbf{I}$ は単位行列である.

式 (1) と式 (2) で定められる平均 $\mu(\mathbf{x}|\mathcal{D})$ と分散 $v(\mathbf{x}|\mathcal{D})$ から獲得関数というものを定義し, この獲得関数を最大化する入力点を選ぶ. このように, すでに結果がわかっている入出力の組から入力点を獲得関数により予測しその点を調べる, という操作を繰り返すことで最適な入力点を求める手法をベイズ最適化という. 獲得関数の選び方にはいくつか種類があるが, 代表的なものとしてガウス過程信頼上限区間 (Gaussian Process Upper Confidence Bound;GP-UCB) が挙げられる [2]. GP-UCB の獲得関数 $a(\mathbf{x}|\mathcal{D})$ は次のように表される.

$$a(\mathbf{x}|\mathcal{D}) = \mu(\mathbf{x}|\mathcal{D}) + \sqrt{\beta_t} v(\mathbf{x}|\mathcal{D}) \tag{3}$$

ただし, β_t は最適化の反復回数 t に依存する係数である.

ベイズ最適化は平均関数 $\mu(\mathbf{x}|\mathcal{D})$ の計算に対して今までに得られている出力値 $\mathbf{y}_{1:N}$ が必要であるため, 仮に複数の使用可能な計算コアがあったとしても並列化できないという欠点がある. 使用可能な計算コアが複数ある状況下で, より効率的な最適化を行うように改善した手法としてマルチタスクベイズ最適化 (Multi-Task Bayesian Optimization;MTBO) [3] と純探索付きガウス過程信頼上限区間 (Gaussian Process Upper Confidence Bound with Pure Exploration;GP-UCB-PE) [4] が挙げられる. MTBO は, 最適化したい関数 $f(\mathbf{x}|d_{\mathrm{all}})$ と, その関数と何らかの関係がある関数が複数ある状況下で, それぞれの情報を共有することで効率的に最適化を行う手法である. また GP-UCB-PE は, 分散関数 (7) が出力結果 $\mathbf{y}_{1:N}$ に依存しないことを利用し, ひとつの計算コアでは GP-UCB 獲得関数 (3) を, 残りの計算コアでは分散関数 (7) を最大化する点を入力点として採用する手法である. MTBO は用意した複数のタスクの情報が最適化したい関数 $f(\mathbf{x}|d_{\mathrm{all}})$ の最適化にとって有用でない場合は通常のベイズ最適化と性能が変わらないという欠点がある. $f(\mathbf{x}|d_{\mathrm{all}})$ の最適化にとって有用かそうでないかということは, 実際にいくつか値を入力してみてその出力応答を調べてみるまでわからないため, MTBO が有用かどうかを事前に見極めることは不可能である.

3. 提案手法

本研究では複数の計算コアが使用可能なときに, 状況に適した探索方策をとることで効率的に最適化を行う手法を提案する. まず最適化したい関数 $f(\mathbf{x}|d_{\mathrm{all}})$ 内で用いられている訓練データ d_{all} のサブサンプリングによって得た d_{sub} を用意する. さらにこの d_{sub} を用いることで, 一点の出力 $f(\mathbf{x}|d_{\mathrm{sub}})$ を返すためにかかる時間は $f(\mathbf{x}|d_{\mathrm{all}})$ の値を得るまでの時間の $1/Q$ 倍になるものとする. 加えて, $f(\mathbf{x}|d_{\mathrm{sub}})$ から $(\mathbf{x}_{\mathrm{sub}}, y_{\mathrm{sub}})$ の点が得られたとして, その入出力間には $y_{\mathrm{sub}} = f(\mathbf{x}_{\mathrm{sub}}|d_{\mathrm{all}}) + \epsilon_{\mathrm{sub}}$(ただし $\epsilon_{\mathrm{sub}} \sim \mathcal{N}(0, \sigma^2_{\mathrm{sub}})$, $\sigma^2_{\mathrm{sub}} > \sigma^2$) の関係が成り立っていると仮定する. 提案手法

においては探索方策として, 純探索とサブサンプル探索を用意する. 純探索は分散関数を最大化する一点を $f(\mathbf{x}|d_{\text{all}})$ に入力することであり, サブサンプル探索は分散関数を最大化する点を順々に Q 点選んでいき, それら Q 点を $f(\mathbf{x}|d_{\text{sub}})$ に入力することを指す.

以上の条件下で, 提案手法はまず一つの計算コアにおいて GP-UCB の獲得関数を最大化する点を入力し, 残りの計算コアにおいては純探索をしたとして得られる情報量 I_{pure} と, サブサンプル探索をしたとして得られる情報量 I_{sub} を比較し, 大きかった方の探索方策を採用する. ここでの情報量とは相互情報量をもとに計算する量であり, 純探索によって入力点 $\mathbf{x}_{\text{pure}}$ を入力したときの I_{pure} とサブサンプル探索によって入力点 $\{\mathbf{x}_{\text{sub},q}\}_{q=1}^{Q}$ を入力したときの I_{sub} を以下のように定義する.

$$I_{\text{pure}} = \frac{1}{2}\log(1+\sigma^{-2}v^2(\mathbf{x}_{\text{pure}}|\mathcal{D})) \tag{4}$$

$$I_{\text{sub}} = \sum_{q=1}^{Q}\frac{1}{2}\log(1+\sigma_{\text{sub}}^{-2}v^2(\mathbf{x}_{\text{sub},q}|\mathcal{D}_q)) \tag{5}$$

ただし $\mathcal{D}_q = \mathcal{D}\cup\{x_{\text{sub},i}\}_{i=1}^{q-1}$ である. さらに式 (1) と式 (2) で用いた平均関数と分散関数は次のように書き直せる.

$$\mu(\mathbf{x}|\mathcal{D}) = \mathbf{k}^\top(\mathbf{K}+\mathbf{A})^{-1}\mathbf{y}_{1:N} \tag{6}$$

$$v^2(\mathbf{x}|\mathcal{D}) = k(\mathbf{x},\mathbf{x}) - \mathbf{k}^\top(\mathbf{K}+\mathbf{A})^{-1}\mathbf{k} \tag{7}$$

ただし $\mathbf{A}$ は対角行列であり, (i,i) 成分は, y_i が $f(\mathbf{x}|d_{\text{all}})$ から得られた点であれば σ が, そうでなければ σ_{sub} が入る.

以上をまとめて, 提案手法では, 一つの計算コアにおいては式 (6) と式 (7) を用いて GP-UCB による最適化を行い, 残りの計算コアでは式 (4) と式 (5) から各探索方策で得られる情報量を計算し, 情報量が大きい方の探索方策を採用することでベイズ最適化を行っていく.

4. 実験

本手法の有用性を検証するため, 三種類のベンチマーク関数, Branin, Ackley, Rosenblock [5] を $f(\mathbf{x}|d_{\text{all}})$ とみなし, これらを最適化する実験を行った. 最適化手法にはランダムサーチ (RS), GP-UCB, MTBO, GP-UCB-PE, 提案手法 (Proposed) を用いた. さらに MTBO と提案手法にでは $f(\mathbf{x}|d_{\text{sub}})$ として, $f(\mathbf{x}|d_{\text{sub}}) = f(\mathbf{x}|d_{\text{all}}) + \mathcal{N}(0, 0.1^2)$ あるいは $f(\mathbf{x}|d_{\text{sub}}) = 0$ を用いた.

$f(\mathbf{x}|d_{\text{sub}}) = f(\mathbf{x}|d_{\text{all}}) + \mathcal{N}(0, 0.1^2)$ のときの各最適化反復における解の最大値の平均と標準誤差は表 1 のようになった. この表の各最適化反復回数において他の手法より精度が高かったものを黄色で示した. $f(\mathbf{x}|d_{\text{sub}}) = f(\mathbf{x}|d_{\text{all}}) + \mathcal{N}(0, 0.1^2)$, つまり $f(\mathbf{x}|d_{\text{sub}})$ が $f(\mathbf{x}|d_{\text{all}})$ と似た出力をする場合, 提案手法の精度が高い結果となった.

同様に $f(\mathbf{x}|d_{\text{sub}}) = 0$ のときの各最適化反復における解の最大値の平均と標準誤差を表 2 に示す. $f(\mathbf{x}|d_{\text{sub}}) = 0$, つまり $f(\mathbf{x}|d_{\text{sub}})$ と $f(\mathbf{x}|d_{\text{all}})$ との出力関係が似ていない場合, 提案手法と GP-UCB-PE の精度が高い結果となった.

以上より, $f(\mathbf{x}|d_{\text{sub}})$ と $f(\mathbf{x}|d_{\text{all}})$ との入出力関係が似ている場合でも似ていない場合でも, 提案法は, それぞれの場合における従来の最善の手法と同程度の性能を達成できることがわかった.

表 1 ：試行回数 100 回における最適化過程で得られた最大値の平均と標準誤差（$f(\mathbf{x}|d_{\text{sub}}) = f(\mathbf{x}|d_{\text{all}}) + \mathcal{N}(0, 0.1^2)$）

function	iteration	RS	GP-UCB	MTBO	GP-UCB-PE	Proposed
Branin	5	0.787(0.089)	0.681(0.201)	**0.959(0.051)**	0.924(0.002)	**0.951(0.110)**
	10	0.843(0.025)	0.701(0.159)	**0.985(0.023)**	0.924(0.002)	**0.961(0.086)**
	20	0.858(0.009)	0.710(0.129)	**0.991(0.013)**	0.925(0.002)	**0.974(0.077)**
	30	0.862(0.006)	0.722(0.139)	**0.996(0.003)**	0.925(0.002)	**0.987(0.061)**
	40	0.865(0.004)	0.762(0.142)	**0.999(0.001)**	0.926(0.002)	**0.988(0.044)**
	50	0.865(0.004)	0.777(0.139)	**0.999(0.001)**	0.926(0.002)	**0.998(0.004)**
Ackley	5	0.539(0.077)	0.421(0.139)	0.621(0.124)	0.401(0.107)	**0.777(0.110)**
	10	0.583(0.092)	0.421(0.139)	0.646(0.139)	0.540(0.025)	**0.821(0.086)**
	20	0.620(0.117)	0.421(0.139)	0.658(0.135)	0.702(0.076)	**0.850(0.077)**
	30	0.675(0.080)	0.421(0.139)	0.677(0.142)	0.702(0.076)	**0.880(0.061)**
	40	0.680(0.079)	0.421(0.139)	0.688(0.141)	0.749(0.076)	**0.902(0.044)**
	50	0.695(0.080)	0.421(0.139)	0.696(0.133)	0.794(0.053)	**0.902(0.044)**
Rosenbrock	5	0.836(0.021)	0.828(0.105)	**0.860(0.026)**	**0.869(0.001)**	**0.859(0.090)**
	10	0.839(0.022)	0.828(0.105)	0.872(0.026)	0.875(0.003)	**0.901(0.067)**
	20	0.849(0.013)	0.839(0.107)	0.898(0.008)	0.881(0.001)	**0.945(0.070)**
	30	0.852(0.011)	0.839(0.107)	0.898(0.008)	0.883(0.006)	**0.962(0.068)**
	40	0.852(0.011)	0.839(0.107)	0.898(0.008)	0.890(0.007)	**0.967(0.070)**
	50	0.857(0.008)	0.839(0.107)	0.898(0.008)	0.893(0.003)	**0.976(0.077)**

表 2 ：試行回数 100 回における最適化過程で得られた最大値の平均と標準誤差（$f(\mathbf{x}|d_{\text{sub}}) = 0$）

function	iteration	RS	GP-UCB	MTBO	GP-UCB-PE	Proposed
Branin	5	0.787(0.089)	0.681(0.201)	0.705(0.031)	**0.924(0.002)**	**0.925(0.002)**
	10	0.843(0.025)	0.701(0.159)	0.705(0.031)	**0.924(0.002)**	**0.925(0.002)**
	20	0.858(0.009)	0.710(0.129)	0.705(0.031)	**0.925(0.002)**	**0.925(0.002)**
	30	0.862(0.006)	0.722(0.139)	0.717(0.028)	**0.925(0.002)**	**0.927(0.003)**
	40	0.865(0.004)	0.762(0.142)	0.738(0.029)	**0.926(0.002)**	**0.927(0.003)**
	50	0.865(0.004)	0.777(0.139)	0.741(0.033)	**0.926(0.002)**	**0.927(0.003)**
Ackley	5	**0.539(0.077)**	0.421(0.139)	0.420(0.187)	0.401(0.107)	0.424(0.183)
	10	**0.583(0.092)**	0.421(0.139)	0.420(0.187)	0.540(0.025)	**0.563(0.117)**
	20	0.620(0.117)	0.421(0.139)	0.422(0.171)	**0.702(0.076)**	**0.701(0.086)**
	30	0.675(0.080)	0.421(0.139)	0.422(0.171)	**0.702(0.076)**	**0.714(0.083)**
	40	0.680(0.079)	0.421(0.139)	0.424(0.177)	**0.749(0.076)**	**0.776(0.083)**
	50	0.695(0.080)	0.421(0.139)	0.424(0.177)	**0.794(0.053)**	**0.792(0.072)**
Rosenbrock	5	0.836(0.021)	0.828(0.105)	0.821(0.106)	**0.869(0.001)**	**0.859(0.002)**
	10	0.839(0.022)	0.828(0.105)	0.831(0.116)	**0.875(0.003)**	**0.870(0.001)**
	20	0.849(0.013)	0.839(0.107)	0.839(0.110)	**0.881(0.001)**	**0.880(0.001)**
	30	0.852(0.011)	0.839(0.107)	0.839(0.110)	**0.883(0.006)**	**0.881(0.002)**
	40	0.852(0.011)	0.839(0.107)	0.839(0.110)	**0.890(0.007)**	**0.887(0.002)**
	50	0.857(0.008)	0.839(0.107)	0.839(0.110)	**0.893(0.003)**	**0.890(0.002)**

5. 結論

本論文では, サブサンプリングにより作成した獲得関数が, ブラックボックス関数の最適化に対して有用か有用でないかを判断し, 探索方策を適応的に選択するベイズ最適化手法を提案した. 数値実験によって, 提案法は, サブサンプリングにより作成した獲得関数が有用である場合でもない場合でも, それぞれの場合における従来の最善の手法と同程度の性能を達成できることがわかった.

参考文献

[1] J. Snoek *et al.*, Practical Bayesian optimization of machine learning algorithm. NIPS, 2012.

[2] N. Srinivas *et al.*, Gaussian process optimization in the bandit setting: No regret and experimental design. ICML, 2010.

[3] K. Swersky *et al.*, Multi-task Bayesian optimization. NIPS, 2013.

[4] E. Contal *et al.*, Parallel gaussian process optimization with upper confidence bound and pure exploration. ECML, 2013.

[5] M. Jamil *et al.*, A literature survey of benchmark functions for global optimisation problems. IJMMNO, 2013.