

過去記憶を用いたスケーラブルなベイズ最適化

Scalable Bayesian Optimization with Memory Retention

伊藤秀剛^{*1}

Hidetaka Ito

松林達史^{*1}

Tatsushi Matsubayashi

倉島健^{*1}

Takeshi Kurashima

戸田浩之^{*1}

Hiroyuki Toda

^{*1}日本電信電話株式会社 NTT サービスエボリューション研究所

NTT Service Evolution Laboratories, NTT Corporation

Bayesian optimization is a method for the global optimization of black-box functions as few evaluation as possible. It utilizes Gaussian processes to efficiently select parameters to be evaluated. However, it is not scalable because Gaussian processes scale cubically with the number of iterations. In this work, we propose a method for scalable Bayesian optimization by leveraging models used in past iterations, which we call past memory. This technique enables us to fit Gaussian processes to only input-output pairs near the previously selected input parameter. In experiments, we show our proposed method outperforms naive Bayesian optimization in terms of optimization performance with limited time budget.

1. はじめに

ベイズ最適化は探索領域内で関数値が最適となる入力パラメータ $\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathcal{X}} \mathcal{F}(\mathbf{x})$ を求める手法である [Shahriari 16]. ベイズ最適化は、ブラックボックス関数（関数の事前知識がない関数であり、関数にパラメータを入力し評価して得られる関数値のみを利用できる）の最適化に適用可能で、評価する入力パラメータと、評価そのものを交互に繰り返しながら関数を最適化する手法である。この手法はモデルとしてガウス過程を用意し、過去に評価を行った入力パラメータと関数出力値の組を学習する。そして、そのモデルの予測値を基に次に評価を行うべき入力パラメータを決定する。このことによって、関数の評価回数を可能な限り最小限に抑えることができるようにこの手法は設計されており、機械学習のハイパーパラメータの最適化などに幅広く応用されてきた。

しかし、ベイズ最適化は、評価する入力パラメータの決定と関数の評価の繰り返し回数が増えるにつれ、ガウス過程のモデルが学習すべきデータの量が増加するため、入力パラメータの決定に時間がかかるようになる。ガウス過程の計算量は学習するデータ量の3乗に比例するため、学習データ量の増加によりアルゴリズムの実行速度は大幅に低下する。そのため、一定の繰り返し回数以上では入力パラメータ決定が関数の評価より時間がかかるようになる。このことから、ベイズ最適化は繰り返し回数を増やすことが困難であり、通常数百回程度の評価回数に制限される。これの対処方法として、ガウス過程よりも計算量の低い別のモデルへの置き換えを行う手法 [Hutter 11] があるが、これらはモデルの表現力が低下する問題がある。

そこで、本論文では繰り返し回数を増やしても、所要時間の増加を抑えることができる、Bayesian Optimization with Memory Retention (BOMR) を提案する。提案手法のアイデアは、過去の繰り返しにおいて使用したモデルの予測結果、つまり過去記憶を再利用することである。図1に提案手法の概念図を示す。ある繰り返しで左から2番目の赤点（丸い黒枠で囲まれている）が新たな入力パラメータとして選択されたときを考える。ベイズ最適化は、この新たに選択された入力パラメータをデータ点として保持する。ここで、ガウス過程のモデ

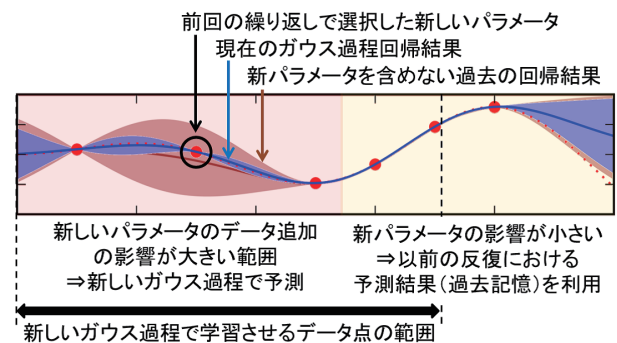


図 1: 提案手法の概要

ルが学習するデータに、新たなデータ点が含まれていないモデルと、含まれているモデルの予測結果を、それぞれ赤茶色と青色で示す。新たなデータ点が学習データに含まれたことによって、そのデータ点の周辺は予測結果が大きく変化することがわかる。一方、新たなデータ点から離れた範囲は、予測結果が似通っており、新しいデータ点を含まないモデルの予測結果を用いても、新しいデータ点を学習したモデルと変わらない十分な予測精度を持つとわかる。

このことに着目し本手法では以下の工夫を行う。まず、直前に選択された新しい入力パラメータの周辺（薄赤の範囲）のみ、この入力パラメータをデータ点として加えて新しく学習したモデルで予測を行う。また、新しく学習するモデルのデータ点は、新しい入力パラメータの周辺を十分に予測できる範囲に存在するデータ（黒矢印の範囲）に限る。このことで、繰り返しの回数の増加に対する学習データの増加を抑え、ガウス過程の計算時間を抑制する。一方、新入力パラメータの周辺以外（黄色の範囲）は、過去記憶として保存した以前の繰り返しにおける予測結果を再利用する。これらの工夫によって、ガウス過程の高い予測精度を維持しながら、繰り返し回数を増やすことが可能である。

ここで、新たに学習したモデルを用いて予測を行う範囲（薄赤の範囲）の決定方法について、本論文では2つの方法を提案する。1つ目が、新たな入力パラメータの影響度合いを表す

数値に閾値を設けて範囲を決める方法、2つ目が新たな入力パラメータの周りの点との相対的な位置関係に基づきボロノイ領域を用いて範囲を決める方法である。

本論文では、人工関数を用いた最適化実験を行う。そして、提案手法が繰り返し回数の増加に対する所要時間の増加を抑えることができ、通常のベイズ最適化より高い精度で最適化を行えることを示す。また、新たなモデルの予測範囲の決定方法は、短時間の最適化ならば1つ目の方法、長時間の最適化ならば2つ目の手法がより有効であることも示す。

2. ベイズ最適化

ベイズ最適化は、評価する入力パラメータの決定とその評価を、試行錯誤的に繰り返す手法である。ベイズ最適化のプロセス実行の前はデータ \mathcal{D}_0 の初期化を行う*1。 t 回目の繰り返しでは、下記の3つのプロセスを通して入力パラメータ \mathbf{x}_t を選択し、評価する。繰り返しは総計 T 回行われる。

1つ目のプロセスがガウス過程を用いたモデルの学習である。ガウス過程は教師あり学習の一種であり、 D 次元のベクトルの入力 $\mathbf{x} = [x_1, \dots, x_D]$ に対してスカラー値の出力 $y = \mathcal{F}(\mathbf{x})$ が対応するデータを用いて学習を行うことで、新たな D 次元のベクトルを入力として、その入力に対応するスカラー値の出力を予測するモデルである[Rasmussen 06]。ガウス過程は、平均的な予測値を決める平均値関数 $m: \mathcal{X} \rightarrow \mathbb{R}$ とデータの相関関係を決める共分散カーネル $k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ によって規定される。本論文では平均値関数は定数 $m(\mathbf{x}) = m_0$ とする。また、共分散カーネルとしてガウスカーネル $k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|_2^2 / (2h^2))$ を用いる。このカーネルは、2つの入力のユークリッド距離が近いと、それらの出力も似ていると予測する性質を持つ。 $h \in \mathbb{R}_+$ はスケールの大きさを規定するハイパーパラメータであり、最尤推定で決定可能である。ガウス過程は $t-1$ 回目までの繰り返しで得たデータ $\mathcal{D}_{t-1} = \{(\mathbf{x}_i, y_i) \mid i = 1, \dots, t-1\}$ を用いて学習を行う。このプロセスの計算量は、ガウス過程の学習のためデータ数を要素数とする正方行列の逆行列計算を行うことで $O(t^3)$ 、これを繰り返すと $O(T^4)$ である。

2つ目のプロセスが、モデルを用いた、次に評価する入力パラメータの選択である。実際の関数に入力して評価を行っていない入力パラメータ \mathbf{x} に対しても、ガウス過程を用いて関数の値の予測を行うことができる。ここで、ガウス過程の利点は予測を平均 $\mu(\mathbf{x})$ 、分散 $\sigma(\mathbf{x})$ のガウス分布の形で行うことで、予測値のみならず、その不確かさも得ることができることである。ガウスカーネルの性質から、入出力データの入力値にユークリッド距離が近い入力パラメータに対しては小さい不確かさを持って、距離が遠い入力パラメータは大きい不確かさを持って予測する。

ベイズ最適化は、このガウス過程の予測結果を用いて獲得関数 $\alpha(\mathbf{x})$ という、次に選択するべき度合いを数値化する関数を計算し、それを最大化する入力

$$\mathbf{x}_t = \arg \max_{\mathbf{x} \in \mathcal{X}} \alpha(\mathbf{x})$$

を、次に評価する入力パラメータとして選択する。本論文では期待改善量 (EI) という獲得関数を用いる [Bull 11]。EI は $\alpha(\mathbf{x}) = (\tau - \mu(\mathbf{x})) \Phi(\gamma) + \sigma(\mathbf{x}) \phi(\gamma)$ と与えられる。ここで、 τ は今まで得られた出力の最適値であり、 $\gamma = (\tau - \mu(\mathbf{x})) / \sigma(\mathbf{x})$

*1 本論文の実験ではランダムサンプリングによって $T_0 = 50$ 点の初期点、 $\mathbf{x}_1^{\text{init}}, \dots, \mathbf{x}_{T_0}^{\text{init}}$ を選択し、その評価を行い $i = 1, \dots, T_0$ に対して $y_i^{\text{init}} = \mathcal{F}(\mathbf{x}_i^{\text{init}})$ を得て、それを \mathcal{D}_0 に加える。

Algorithm 1 Bayesian Optimization with Memory Retention (BOMR)

- 1: **Input:** 目的関数 $\mathcal{F}(\mathbf{x})$, 探索範囲 \mathcal{X} , 獲得関数 $\alpha(\mathbf{x})$, 最大繰り返し回数 T
- 2: \mathcal{D}_0 のランダムサンプリングなどを用いて初期化
- 3: 過去記憶 $\mathcal{M}_0 \leftarrow \emptyset$
- 4: **for** $t = 1, 2, \dots, T$ **do**
- 5: 探索範囲 $\mathcal{X}'_t \subseteq \mathcal{X}$ の絞りこみ
- 6: 学習データ $\mathcal{D}'_t \subseteq \mathcal{D}_{t-1}$ の絞り込み
- 7: ガウス過程による \mathcal{D}_{t-1} へのフィッティング
- 8: $\mathcal{M}_t \leftarrow \{(\mathbf{x}, \mu(\mathbf{x}), \sigma(\mathbf{x})) \in \mathcal{M}_{t-1} \mid \mathbf{x} \notin \mathcal{X}'_t\}$
- 9: $\mathbf{x}_t^{\mathcal{M}} \leftarrow \arg \max_{\mathbf{x} \in \mathcal{M}_t} \alpha(\mathbf{x})$
- 10: $\mathbf{x}'_t \leftarrow \arg \max_{\mathbf{x} \in \mathcal{X}'_t} \alpha(\mathbf{x})$
- 11: $\mathbf{x}_t \leftarrow \arg \max_{\mathbf{x} \in \{\mathbf{x}_t^{\mathcal{M}}, \mathbf{x}'_t\}} \alpha(\mathbf{x})$
- 12: 評価 $y_t \leftarrow \mathcal{F}(\mathbf{x}_t)$
- 13: $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(\mathbf{x}_t, y_t)\}$
- 14: 10行目の最適化中に調べた点のデータを \mathcal{M}_t に追加
- 15: **Return** \mathcal{D}_T 内で最良の入力パラメータ

である。また、 $\Phi(\gamma)$ と $\phi(\gamma)$ はそれぞれ標準正規分布の累積分布関数と確率密度関数である。EI は探索と活用のバランスを取れる性質を持つ。探索とは不確かさの大きい入力パラメータを調べることであり、活用とは良い結果を得た入力パラメータに似ているものを調べることであり、探索だけではランダムな選択が増え効率が悪いが、活用だけでは局所解にトラップされ最適解にたどり着くことができないため、この2つを両立するよう EI は設計されている。本論文では獲得関数の最大化に多スタート局所探索を用いる。多スタート局所探索は、複数の点をランダムで選択し、その場所を起点として局所探索の手法を用いて局所解を探索し、得られた局所解の中から最も最適な点を選択する手法である。局所探索の手法としては L-BFGS を用いる。このプロセスは、獲得関数の計算回数を T_a とすると、ガウス過程の予測のために繰り返し1回当たり $O(T_a t^2)$ 、繰り返し全体で $O(T_a T^3)$ の計算量である。

3つ目のプロセスが選択した入力パラメータの評価である。評価後は結果をデータに追加し、 $t+1$ 回目の反復に移行する。このプロセスの計算量は繰り返し回数に依らないため、 t が小さいときに最も計算時間がかかるのはこのプロセスだが、 t が増加すると1、2つ目のプロセスが計算時間のボトルネックとなる。

3. 提案手法

3.1 過去記憶の利用

提案手法である BOMR をアルゴリズム 1 に示す。BOMR は、過去の繰り返しで用いたガウス過程モデルによる、関数出力値の予測結果を過去記憶として保存する。そして、過去記憶で十分に予測ができる範囲については過去記憶を用いることで、それができない範囲については新たなモデルで予測を行う。探索範囲内の入力パラメータの関数評価値の予測を行い、それを基に次に探索する点を決定する。

t 回目の繰り返しにおいて、BOMR は以下のように過去記憶の利用を行う。まず、前回の繰り返しで選択された入力パラメータ \mathbf{x}_{t-1} の影響が強い範囲が、過去記憶が使えない範囲であるので、これを t 回目の探索範囲 \mathcal{X}'_t とする (5行目)。そして、その探索範囲内の予測を十分に行えるだけの学習データ \mathcal{D}'_t を、過去に評価したデータ \mathcal{D}_{t-1} の中から一部を絞り込む

ことで得て、その D'_t を使ってガウス過程を学習させる (6,7 行目). 次は、 \mathcal{X}'_t を新たに調べなおすため、過去記憶の中から \mathcal{X}'_t を除いた範囲にある入力パラメータのデータだけ保持する (8 行目). その後は獲得関数最大化のプロセスを行う. まず、過去記憶のデータで最も獲得関数の値が大きい点 \mathbf{x}_t^M を得る *2 (9 行目). 次に \mathcal{X}'_t 内で最も獲得関数の値が大きい点 \mathbf{x}'_t を得る (10 行目). そして、 \mathbf{x}_t^M と \mathcal{X}'_t を比較し、獲得関数の値が大きい方を t 回目の繰り返しで評価する入力パラメータ \mathbf{x}_t として選択する (11 行目). 後は \mathbf{x}_t を評価して関数評価値 $\mathcal{F}(\mathbf{x}_t)$ を取得し、評価済のデータに加える (12,13 行目). 最後に 10 行目の最適化の間に調べた点のデータを過去記憶 \mathcal{M}_t に追加する *3 (14 行目).

3.2 探索範囲の絞りこみ

BOMR において、毎回の繰り返しにおける探索範囲 \mathcal{X}'_t をどのように行うかが問題である. 本論文では、これらの決め方について 2 通りの方法を提案する.

3.2.1 カーネルの値の閾値を用いた方法

カーネルの値の閾値を用いた方法を述べる. ある入力パラメータ \mathbf{x} の関数評価値 $\mathcal{F}(\mathbf{x})$ を予測するとき、 \mathbf{x} と過去に選択した入力パラメータ \mathbf{x}_{t-1} とのカーネルの値 $k(\mathbf{x}, \mathbf{x}_{t-1})$ は、 \mathbf{x}_{t-1} が予測結果 $\mu(\mathbf{x}), \sigma(\mathbf{x})$ に与える影響力の強さを表す. そこで、 $k(\mathbf{x}, \mathbf{x}_{t-1})$ がある閾値を上回る範囲を、新しいモデルで予測すべき範囲とみなし、これを含んだ空間に \mathcal{X}'_t を絞り込むことを考える. ガウスカーネルなど一般的なガウス過程のカーネルを用いると、 \mathbf{x} と \mathbf{x}_{t-1} のユークリッド距離が近いときに $k(\mathbf{x}, \mathbf{x}_{t-1})$ が大きくなる. そこで、 \mathbf{x}_{t-1} に近い範囲に \mathcal{X}'_t を絞り込めばよい. そこで、スケールの大きさを規定するハイパーパラメータ h を考慮して、

$$S_c(\mathbf{x}_{t-1}) = \{\mathbf{x} \in \mathcal{X} \mid \|\mathbf{x} - \mathbf{x}_{t-1}\|_2 \leq h \cdot c\}$$

に対し、 $S_c(\mathbf{x}_{t-1}) \subseteq \mathcal{X}'_t$ なる \mathcal{X}'_t を探索範囲とする. ここで、 c はハイパーパラメータであり、 c が大きい程、広い範囲が探索範囲となる. このとき、 h を最尤推定を用いて決定している場合は、新しいガウス過程を学習する前に h の値を得ることができないが、以前の繰り返しにおける h の値で代替することができる *4.

ここで、 $S_c(\mathbf{x}_{t-1})$ は球になるが、獲得関数最大化は L-BFGS が使える立方体 $\{\mathbf{x} \in \mathcal{X} \mid \forall i = 1, \dots, D, x^L \leq x_i \leq x^U\}$ の方が行いやすい. ここで \mathbf{x} の i 番目の要素を x_i と表している. そこで、 \mathcal{X}'_t を、各軸に直交する面を持つ体積最小の立方体とする. このとき \mathcal{X}'_t は $S_c(\mathbf{x}_{t-1})$ に外接する. つまり、 $x^L = x_{(t-1,i)} - h \cdot c$ であり、 $x^U = x_{(t-1,i)} + h \cdot c$ である. ここで、 $x_{(t-1,i)}$ は \mathbf{x}_{t-1} の i 番目の要素である.

3.2.2 ボロノイ領域を用いた方法

次にボロノイ領域を用いて \mathcal{X}'_t を絞り込む方法を述べる. $k(\mathbf{x}, \mathbf{x}_{t-1})$ の観点では \mathbf{x} の関数評価値の予測値に \mathbf{x}_{t-1} が与える影響力の強さは、 $\|\mathbf{x} - \mathbf{x}_{t-1}\|_2$ に対し単調減少する. しかし、 \mathbf{x} の周りにデータ点が少ない場合は、予測値に影響を与える他の要素が少ない為、 \mathbf{x} と \mathbf{x}_{t-1} との距離がある程度遠くて

*2 このとき、獲得関数として EI を用いている場合は、過去の最良値 τ が変化していない場合、同じ平均と分散を持つ予測結果に対して、過去に計算した獲得関数の値と変化がない為、再利用して実行時間を削減することができる.

*3 このとき、有望な結果のみを記憶として保存してもよく、その場合は実行時間を削減できる. 本論文では得られた獲得関数値の局所的な最大値を取る入力パラメータのみを記憶として保存する.

*4 本論文では過去の繰り返し 100 回分の h の中央値を用いた.

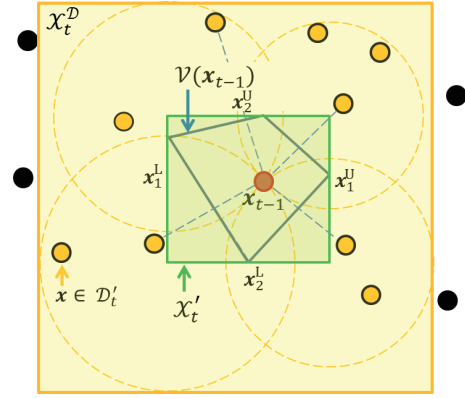


図 2: ボロノイ領域を用いた場合の探索範囲と学習データ点の絞り込み方法の図示. 赤色の点が \mathbf{x}_{t-1} , 深緑色の四角形で囲まれた範囲が $\mathcal{V}(\mathbf{x}_{t-1})$ であり、黄緑色で塗られた範囲が \mathcal{X}'_t である. そして、その外側の黄色い枠で囲まれた領域が \mathcal{D}'_t であり、黄色の点で表されたデータ点が \mathcal{D}'_t に含まれる. 黒い点で表されたデータ点は学習しないデータ点である.

も予測値へ大きな影響を与えたと考えられる. 一方、 \mathbf{x} の周りにデータ点が多い場合は、 \mathbf{x}_{t-1} が与える \mathbf{x} の予測値への影響は小さくなると考えられる. よって、 \mathbf{x}_{t-1} と他のデータ点との相対的な距離関係を考慮して \mathcal{X}'_t を絞り込むべきである.

これを実現する方法としてボロノイ領域の利用がある. この方法では、 \mathbf{x} の関数評価値の予測値に影響を与えるのは、 \mathbf{x} の最近傍点であると考え *5. よって、 \mathbf{x} の最近傍点が \mathbf{x}_{t-1} であれば \mathcal{X}'_t に加える. そこで、 \mathbf{x}_{t-1} のボロノイ領域

$$\mathcal{V}(\mathbf{x}_{t-1}) = \{\mathbf{x} \in \mathcal{X} \mid \forall i = 1, \dots, t-2, \|\mathbf{x}_{t-1} - \mathbf{x}\|_2 \leq \|\mathbf{x}_i - \mathbf{x}\|_2\}$$

を用いて、 $\mathcal{V}(\mathbf{x}_{t-1}) \subseteq \mathcal{X}'_t$ なる \mathcal{X}'_t を探索範囲とする. ボロノイ領域を用いることで、繰り返し回数が増えにつれ探索範囲が狭まっていき、これに応じて学習すべきデータの増大を防げる. ここで $\mathcal{V}(\mathbf{x}_{t-1})$ は複雑な形状であり、また厳密に得るための計算量も大きい. そこで、探索領域は各軸に直交する面を持ち体積が最小となる直方体 $\mathcal{X}'_t = \{\mathbf{x} \in \mathcal{X} \mid \forall i, x_i^L \leq x_i \leq x_i^U\}$ とする. このときも \mathcal{X}'_t は $\mathcal{V}(\mathbf{x}_{t-1})$ に外接する.

\mathcal{X}'_t の計算方法について述べる. $f_i(\mathbf{x}) = x_i$ という関数を定義すると、 $x_i^L = \min_{\mathbf{x} \in \mathcal{V}(\mathbf{x}_{t-1})} f_i(\mathbf{x})$ であり $x_i^U = \max_{\mathbf{x} \in \mathcal{V}(\mathbf{x}_{t-1})} f_i(\mathbf{x})$ である. ここで、 $f_i(\mathbf{x})$ は線型関数である. また、 $\|\mathbf{x}_{t-1} - \mathbf{x}\|_2 \leq \|\mathbf{x}_i - \mathbf{x}\|_2$ という条件は、 \mathbf{x}_{t-1} と \mathbf{x}_i の垂直二等分面で \mathcal{X} を隔てた 2 つの空間のうち、 \mathbf{x}_{t-1} が存在する側に \mathbf{x} が存在する条件、つまり $(\mathbf{x}_{t-1} - \mathbf{x}_i)^\top \mathbf{x} \geq (\|\mathbf{x}_{t-1}\|_2^2 - \|\mathbf{x}_i\|_2^2)/2$ と同値であり、これは線型制約である. これらより、 x_i^L や x_i^U を求める問題は線型計画問題であり、線型計画法で求めることができる *6.

*5 実際 [Bull 11] では、 h が一定の条件にてガウス過程の予測値の不確かさが最近傍点との距離によって上から抑えることができ、これが最適化性能に影響することが示されている.

*6 線型計画法のアルゴリズムの計算量は制約条件の数に応じて増大していく. しかし、 $\mathcal{V}(\mathbf{x}_{t-1})$ の形に影響するデータ点は \mathbf{x}_{t-1} に近いデータ点のみであるので、そのようなデータ点をいくつか選び、それらに関する制約条件のみを考えることで計算量を削減できる. 一部の制約条件のみを考えて得た x_i^L (または x_i^U) が正しいことは、 $f_i(\mathbf{x}) = x_i^L$ (または $f_i(\mathbf{x}) = x_i^U$) となる \mathbf{x} の最近傍点が \mathbf{x}_{t-1} であることを調べればわかる. 最近傍探索は $O(t)$ の計算量で可能であり、kd 木などの手法を用いて高速化することもできる. x_i^L や x_i^U

3.3 学習データ点の絞りこみ

\mathcal{D}'_t の絞り込みは \mathcal{X}'_t に応じて行う。本論文では学習するデータ点の範囲 \mathcal{X}^D_t を決め、 $\mathcal{D}'_t = \{(\mathbf{x}, y) \in \mathcal{D}_{t-1} \mid \mathbf{x} \in \mathcal{X}^D_t\}$ とする。 \mathcal{X}'_t 内の点を十分学習できる範囲を \mathcal{X}^D_t はとる必要があるため、 $\mathcal{X}'_t \subseteq \mathcal{X}^D_t$ であることに注意する。本論文では \mathcal{X}'_t と同様に \mathcal{X}^D_t を各軸に直交する面を持つ直方体とする。

そこでポロノイ領域の考え方をを用いて \mathcal{X}^D_t を決定する。任意の \mathcal{X}'_t 内の点に対し、この点の最近傍データ点が \mathcal{D}'_t に含まれるようにする。つまり、 $\{\hat{\mathbf{x}} \in \mathcal{X} \mid \exists \mathbf{x} \in \mathcal{X}'_t, \mathbf{x} \in \mathcal{V}(\hat{\mathbf{x}})\} \subseteq \mathcal{X}^D_t$ である。ここで \mathcal{X}'_t のある頂点 \mathbf{x}_c を考える。 \mathbf{x}_c にとって \mathbf{x}_{t-1} より速いデータ点は最近傍になりえないため、 \mathbf{x}_c が中心で $\|\mathbf{x}_{t-1} - \mathbf{x}_c\|_2$ を半径とする球の内側に、 \mathbf{x}_c の最近傍データ点が含まれる。よって、全ての角に対しこのような球を考え、これらの球を全て内包する最小体積の直方体を \mathcal{D}'_t とする。角のみ考えれば、その他の点に対しても条件が満たされる。図2に2次元におけるポロノイ領域を用いた場合の探索範囲と学習データ点の絞り込みの模式図を示す。

4. 実験

本章では人工関数の最適化実験を通して提案手法の有効性を実証する。最適化の目的関数として次元が $D = 3$ の Rosenbrock 関数 $\mathcal{F}(\mathbf{x}_t) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$ を用いる。探索範囲は $\mathcal{X} \in [-5, 10]^D$ である。

提案手法として、 $c = 1$ としてカーネルの値の閾値を用いた手法 (BOMR S) と、ポロノイ領域を用いた手法 (BOMR V)、そしてこれら2つを両方使用してより狭い方の探索範囲に限定した手法 (BOMR SV) の3手法を用いる。また、ベースラインとして通常のベイズ最適化 (BO) を用いる。提案手法の最初の繰り返しにおいては過去記憶がないため、 $\mathcal{X}'_1 = \mathcal{X}$ 、 $\mathcal{D}'_1 = \mathcal{D}_0$ とする。獲得関数の計算回数は、ベースラインでは $T_a = 1000D$ 回に制限する。提案手法では \mathcal{X}'_t が狭く同じ計算回数は必要ないため、 $1000D$ 回に \mathcal{X}'_t の対角線の長さ \mathcal{X} の対角線の長さの比をかけた回数に制限する。CPU に Xeon E7-4870 を用いて計算を行い、全ての最適化は1スレッドのみを使用して実行する。乱数の影響を抑えるため、10種類のランダムシードを用いて最適化を全手法10回ずつ行い、実線が平均値であり、標準偏差の大きさだけ上下双方を塗りつぶしたグラフで結果を示す。スケールの違いに対する影響を取り除くため、平均と分散は対数をとった値に対して計算している。

図3の左に、横軸が繰り返しの回数であり、縦軸が所要時間である、両対数グラフを各手法に対して示す。横軸の値が大きい方が高速である。ベースラインのBOは 10^5 秒経過時点で繰り返し回数は 10^3 回に到達しない。一方、どの提案手法もBOよりも高速であり、 10^5 秒の時点で繰り返し回数は10倍以上の差がつく。BOMR Sは繰り返し回数が小さいときに特に高速である。しかし時間がたつとBOと近い計算量のオーダーで計算時間が増大する。これは、閾値が一定だと閾値内に存在するデータ点の数が繰り返し回数の増加に伴い増えるからである。BOMR Vは最初はBOMR Sより低速であるが、最終的にはこれより高速になる。BOMR Vは、繰り返し回数が増えるに従いポロノイ領域が縮小し、学習すべきデータ点の数を抑える効果により、計算量のオーダーがBOより小さい。BOMR SVは全ての繰り返し回数で、BOMR SとBOMR Vの高速である方と同程度である。

が正しくない場合、制約条件を増やしてもう一度線型計画問題を解き、得られた x^L_i や x^U_i が正しいことを調べなおせば良い。

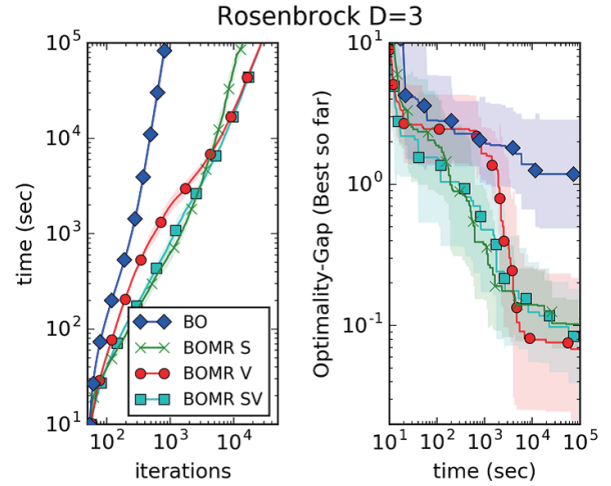


図3: 各手法の所要時間と最適化精度

図3の右に、横軸が所要時間であり、縦軸が t 回目の繰り返し終了時点で得た関数評価値のベスト値 ($\min_{t' \leq t} \mathcal{F}(\mathbf{x}_{t'})$) である、両対数グラフを示す。この関数の最適値は0であるので、関数値そのものの値が、最適値との差 (Optimality-Gap) を示す。得た入力パラメータのうち最良のものを最終的に取得することを想定するため、ベスト値を記載している。縦軸の値が小さい程、最適化の精度が高い。提案手法はどれもBOよりも高い最適化性能を示している。過去記憶の利用により、モデルの表現力を維持したまま高速化が出来ているとわかる。

5. 結論

本論文では繰り返し回数が増えるにつれ急激に低速化してしまうベイズ最適化の弱点を克服する、新しい手法BOMRを提案した。BOMRは以前の繰り返しで得たモデルの予測結果を過去記憶として保存し、それを再利用することで、繰り返し回数の増加に対してモデルの学習データ量が増加することを抑え、スケーラブルな最適化を実現した。実験では、高速化の結果で提案手法が通常のベイズ最適化より、同じ計算時間でより良い入力パラメータを発見できることを示した。

参考文献

- [Bull 11] Bull, A. D.: Convergence rates of efficient global optimization algorithms, *Journal of Machine Learning Research*, Vol. 12, pp. 2879–2904 (2011)
- [Hutter 11] Hutter, F., Hoos, H. H., and Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration, *In LION*, pp. 507–523 (2011)
- [Rasmussen 06] Rasmussen, C. E. and Williams, C. K. I.: *Gaussian processes for machine learning.*, MIT Press (2006)
- [Shahriari 16] Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and Freitas, de N.: Taking the human out of the loop: A review of bayesian optimization, *Proceedings of the IEEE*, Vol. 104, No. 1, pp. 148–175 (2016)