

Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design

Niranjan Srinivas, Andreas Krause[†], Sham Kakade^{††}, Matthias Seeger^{†††}

[†]: California Institute of Technology

^{††}: University of Pennsylvania

^{†††}: Saarland University

January 18, 2023

Presenter 石倉 雅紀

目次

- 1 はじめに
- 2 ベイズ最適化
- 3 GP-UCB
- 4 実験
- 5 まとめ

1 はじめに

背景

ブラックボックス関数

ベイズ最適化

本論文の貢献

2 ベイズ最適化

3 GP-UCB

4 実験

5 まとめ

- 目的関数に対する最適変数探索問題として定式化できるものも多い
 - ▶ 耐久性の高いロボットの開発

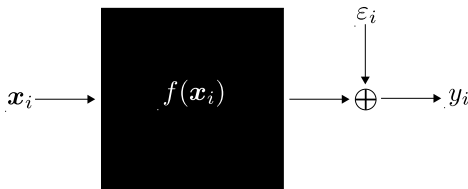


ブラックボックス関数

- ブラックボックス関数 f

$$y_i = f(x_i) + \varepsilon_i$$

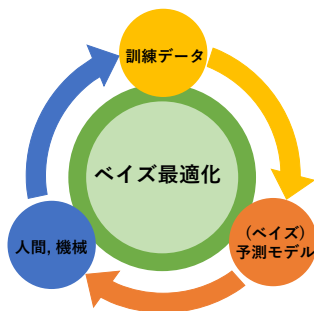
入力： x_i , 出力： y_i , 誤差： ε_i



- ブラックボックス関数 f
 - ▶ 具体的な形状が不明
- ブラックボックス関数を扱う実問題も多く存在する

ベイズ最適化

- 多くの実問題はブラックボックス関数最適化と等価
- できるだけ少ない関数評価回数で最適値を見つけたい
 - ▶ 特に関数値を得るコストが高い場合
- 有効な手法として**ベイズ最適化**がある



モデリング

訓練データから
予測モデルを構築



決定

予測モデルを最も改良する点を
モデルに基づき決定する



観測

選ばれた点を
実際に観測

貢献

- ガウス過程を用いた最適化問題を解決するためのアルゴリズムを提案
- 実データ実験において優れた性能を発揮

1 はじめに

2 ベイズ最適化

- 問題設定

- ガウス過程

- ガウス過程を用いたベイズ最適化

- 獲得関数

- 従来の獲得関数

3 GP-UCB

4 実験

5 まとめ

- 候補入力 $\mathcal{X} = \{x_1, \dots, x_n\}$ が与えられている
- 関数 f を評価して出力 $y_i = f(x_i)$ を得るにはコストがかかる
- できるだけ少ないコストで関数 f を最大化するパラメータ x を求めたい

$$x^* = \arg \max_{x \in \mathcal{X}} f(x)$$

- ▶ ガウス過程は確率分布を用いた複雑で非線形な関数を扱うためのモデルである
- ▶ このような複雑で非線形な関数を扱う場合には一般的な最適化手法では解決が困難になることがある

ガウス過程

任意の入力 $\{x_1, \dots, x_n\}$ に対して $\{f(x_1), \dots, f(x_n)\}$ が n 次元正規分布に従うなら f はガウス過程に従う

- 関数 f がガウス過程に従うことを $f(x) \sim \mathcal{GP}(\mu(x), k(x, x'))$
- 平均関数: $\mu(x) = \mathbb{E}[f(x)]$
- 共分散関数: $k(x, x') = \mathbb{E}[(f(x) - \mu(x))(f(x') - \mu(x'))]$

ガウス過程回帰

- f の事前分布がガウス過程であると仮定: $f(x) \sim \mathcal{GP}(0, k(x, x'))$
- 観測データ $(X, y) = \{(x_i, y_i)\}_{i=1}^t = \mathcal{D}$ が与えられた下での事後分布

$$f(x_*) \mid \mathcal{D} \sim N\left(k_{\mathcal{D}, x_*}^\top K_{\mathcal{D}\mathcal{D}}^{-1} y, k_{x_* x_*} - k_{\mathcal{D}, x_*}^\top K_{\mathcal{D}\mathcal{D}}^{-1} k_{\mathcal{D}, x_*}\right)$$

$$K_{\mathcal{D}\mathcal{D}} = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_t) \\ \vdots & \ddots & \vdots \\ k(x_t, x_1) & \cdots & k(x_t, x_t) \end{pmatrix}, k_{\mathcal{D}, x_*} = \begin{pmatrix} k(x_1, x_*) \\ \vdots \\ k(x_t, x_*) \end{pmatrix}, k_{x_*, x_*} = k(x_*, x_*)$$

- f にガウス過程事前分布 $\mathcal{GP}(0, k(x, x'))$ を仮定する
- 訓練データ $(X, y) = \{(x_i, y_i)\}_{i=1}^t$ に基づき f の事後分布を求める
- 事後分布に基づき**最適点**を次に観測する
- 観測した (x_{next}, y_{next}) を訓練データに追加し再び事後分布を求める

獲得関数

探索する点を選択するために使用される関数

- 獲得関数を $\alpha(x_{t-1})$ とすると次に観測する点 x_t は以下のように決定

$$x_t = \arg \max_x \alpha(x_{t-1})$$

獲得関数の設計指針

活用と探索のバランスを考慮して設計する

- 活用: 現在の予測から最も高い確率で最適解が存在する点を探索
 - ▶ 活用重視のとき局所解に陥りやすい
- 探索: 未知の領域を探索し新しい知見を得ること
 - ▶ 最適解にたどり着くまでに時間を要する

mean only

$$x_t = \arg \max_x \mu_{t-1}(x)$$

- 予測平均が最大となる x を選択
⇒ 活用中心

variance only

$$x_t = \arg \max_x \sigma_{t-1}^2(x)$$

- 予測分散が最大となる x を選択
⇒ 探索中心

1 はじめに

2 ベイズ最適化

3 GP-UCB

提案手法

GP-UCB のアルゴリズム

4 実験

5 まとめ

- 探索・活用に特化した手法では最適化がうまくいかない場合が多い
⇒ 探索と活用のバランスが取れるような手法を提案

GP-UCB: Gaussian Process Upper Confidence Bound

$$\boldsymbol{x}_t = \arg \max_{\boldsymbol{x} \in \mathcal{D}} \mu_{t-1}(\boldsymbol{x}) + \beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x}).$$

- β_t は，探索の度合いを表す定数
 - ▶ β_t が小さい \Rightarrow 予測平均の比重が大きくなる \Rightarrow 活用重視
 - ▶ β_t が大きい \Rightarrow 予測分散の比重が大きくなる \Rightarrow 探索重視

Algorithm 1 The GP-UCB algorithm.

Input: Input space \mathcal{D} ; GP Prior $\mu_0 = 0, \sigma_0, k$

for $t = 1, 2, \dots$ **do**

 Choose $\mathbf{x}_t = \arg \max_{\mathbf{x} \in \mathcal{D}} \mu_{t-1}(\mathbf{x}) + \sqrt{\beta_t} \sigma_{t-1}(\mathbf{x})$

 Sample $y_t \leftarrow f(\mathbf{x}_t) + \varepsilon_t$

 Perform Bayesian update to obtain μ_t and σ_t

end for

1 はじめに

2 ベイズ最適化

3 GP-UCB

4 実験

実験手法

実験対象データ

実験結果

5 まとめ

- 実験手法

- ▶ variance only
- ▶ maximum mean
- ▶ Expected Improvement(EI)
- ▶ Most Probable Improvement(MPI)
- ▶ GP-UCB

- Mean Average Regret を用いた評価

- ▶ $\text{Regret} = (\text{真の関数の最大値}) - (\text{観測済みデータの最大値})$
- ▶ Average Regret : 1 回の試行中の Regret の平均
- ▶ Mean Average Regret : 全試行の Average Regret の平均

1. 合成データ

- ▶ $\mathcal{D} \in [0, 1]$ の範囲を一様に 1000 点で区切った点を候補点とする
- ▶ $T = 1000, \sigma^2 = 0.025, \delta = 0.1$
- ▶ 30 回試行

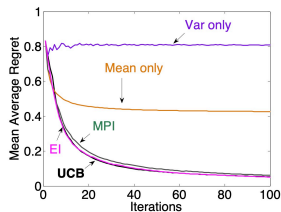
2. 温度データ

- ▶ 46 個のセンサーから 1 分間隔で 5 日以上計測された気温
- ▶ $T = 46, \sigma^2 = 0.5, \delta = 0.1$
- ▶ 2000 回試行

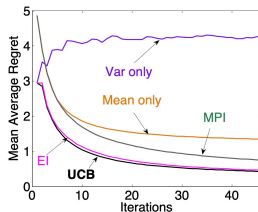
3. 交通データ

- ▶ 357 個のセンサーから 1 ヶ月間午前 6 時から午前までに通過する車の速度
- ▶ $T = 357, \sigma^2 = 4.78, \delta = 0.1$

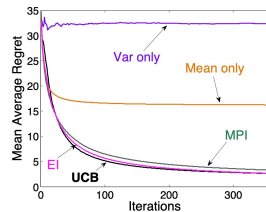
実験結果



(a) Squared exponential



(b) Temperature data



(c) Traffic data

- すべての実験結果において GP-UCB が損失少なく最適化を行なっている
- 既存手法の EI, MP と比較して同等以上の性能を示した

- 1 はじめに
- 2 ベイズ最適化
- 3 GP-UCB
- 4 実験
- 5 **まとめ**
結論

- GP-UCB というベイズ最適化の獲得関数を新たに提案
 - ▶ GP-UCB は探索と活用を両立する獲得関数
- GP-UCB は既存の手法と同等以上の性能を示した
 - ▶ Mean Average Regret の観点で実データと合成データにおける性能を発揮