ベイズ最適化

（株）すうがくぶんか

内容

1. ベイズ最適化とはなにか？
    a. ベイズ最適化で解きたい問題
    b. ベイズ最適化の概要
2. ベイズ最適化の仕組み
    a. ガウス過程回帰モデル
    b. 獲得関数
3. Python言語のパッケージ

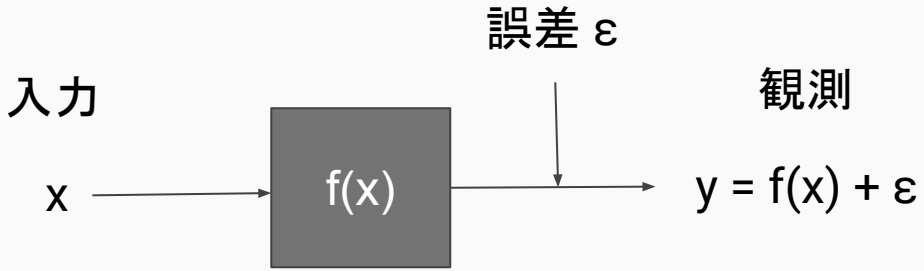
1. ベイズ最適化とはなにか？

- ベイズ最適化で解きたい問題
- ベイズ最適化の概要

ベイズ最適化で解きたい問題(1/4)

ブラックボックス関数の最適化

入力xと観測yの関係を数式で陽に表すことができないような関数



問題　この関数の最小値(or 最大値)を与える入力の値x*を求めたい。
なお、この最小値のことを最適解という。

ベイズ最適化で解きたい問題（2/4）

従来の方法
・ラテン超立方体サンプリング（Latin Hypercube Sampling）

入力　D個の変数
出力　N個の実験候補の入力値

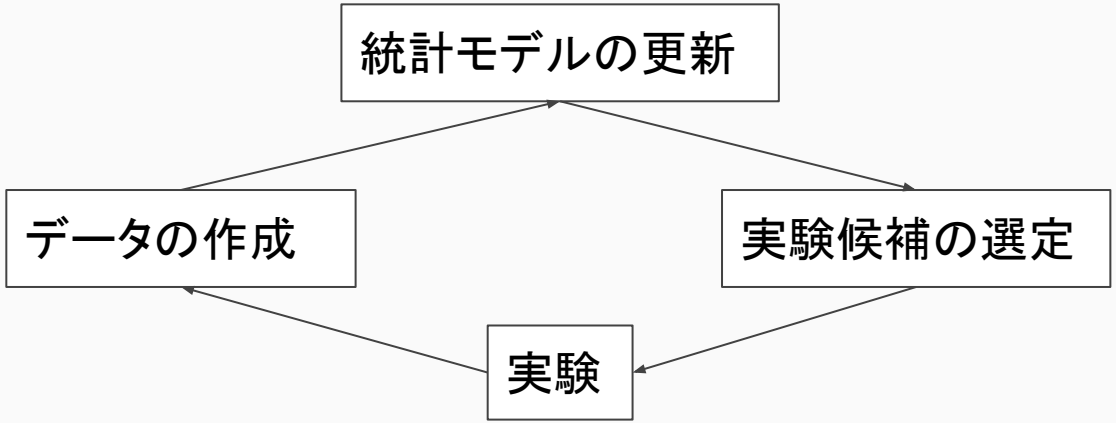
1. 各入力変数をN個の領域に分割する。
2. 各入力変数dごとに1からNまでの整数を並べ替えて順列a[d]を作る。
3. 2で作ったD個の順列の1番目の値a[1](1),...,a[D](1)に対応する分割領域から一様分布に従って一つ値x[1]を抽出する。順列の2番目、...、N番目についても同様の操作を行い、x[2],...,x[N]を得る。

→　得られたx[1],...,x[N]が実験候補の入力値

ベイズ最適化で解きたい問題(3/4)

疑問　各実験で得られる値を次の実験を行う入力値の探索に用いたほうが、固定された実験計画より効率的に最適解を探索できるのではないか？

統計的実験計画



十分に繰り返したら一番小さな観測値が得られた入力値を最適解とする。

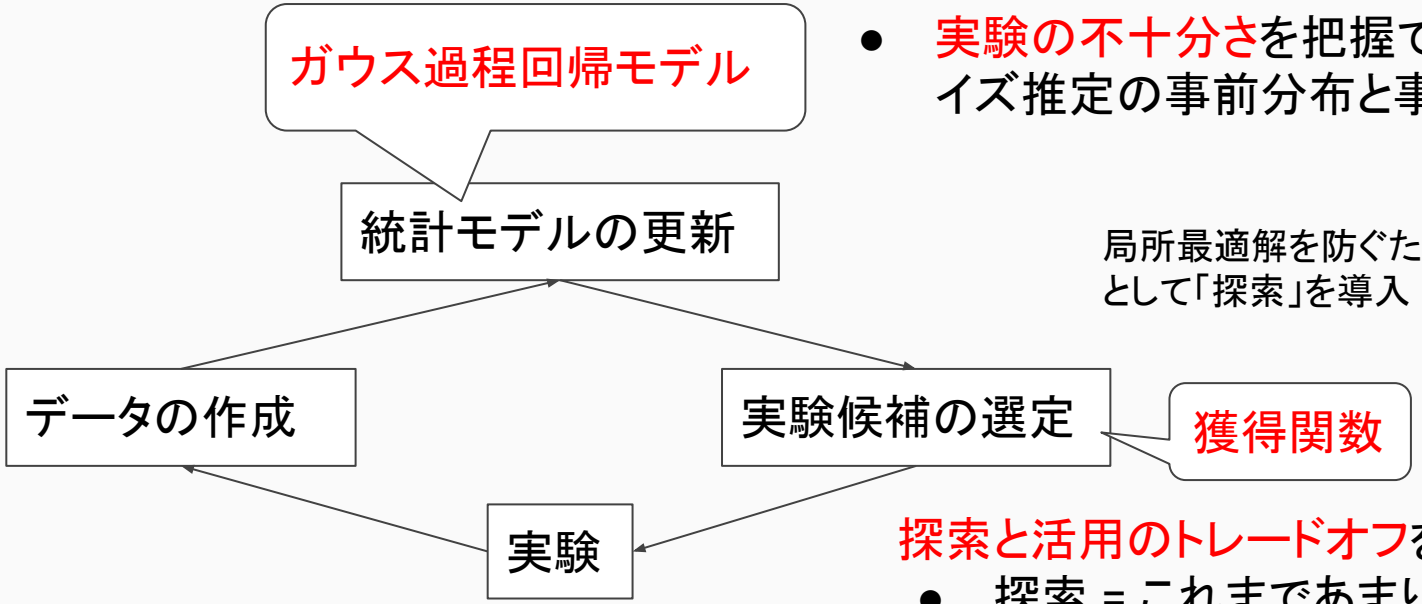
ベイズ最適化で解きたい問題（4/4）

従来の方法
・応答曲面法（response surface method, RSM）

以下を繰り返す。
1. ブラックボックス関数を近似する2次曲面をデータから推定する。
2. 2次曲面の最小解がある方向に実験の中心点を取り替え、回転可能性をみたすような実験計画を立て直す。（回転可能性 = 予測誤差が実験の中心点からの距離にのみ依存）

欠点　ブラックボックス関数が複雑な場合に、局所最適解に陥りやすい。
→　ベイズ最適化（Baysian Optimization）

ベイズ最適化の概要(1/1)



- 実験の不十分さを把握できるようにする。(ベイズ推定の事前分布と事後分布)

局所最適解を防ぐための工夫として「探索」を導入

探索と活用のトレードオフを設計

- 探索 = これまであまり実験していない候補の入力値を実験したい。
- 活用 = すでに良い結果を得た候補の入力値をさらに改善したい。

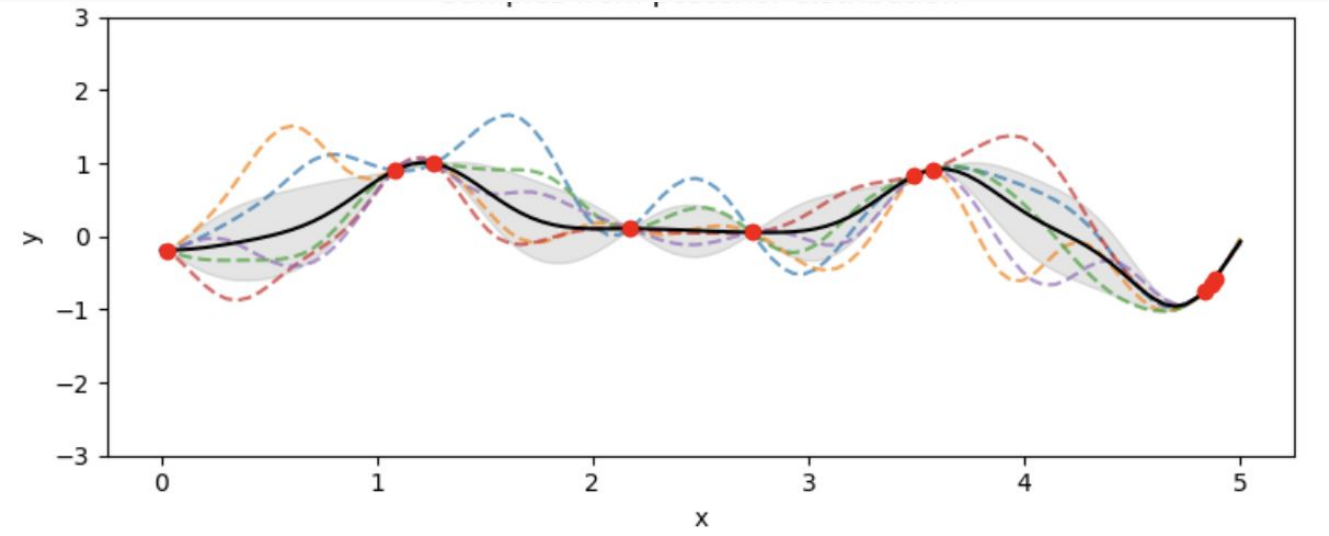
2. ガウス最適化の仕組み

- ガウス過程回帰モデル
- 獲得関数

ガウス過程回帰モデル(1/3)

関数f(x)がガウス過程に従っていると仮定する。
⇔ (f(x[1]), ..., f(x[n]))は多変量正規分布に従っている。nは任意でよい。

- 今回は、「関数f(x)はきっとこれであろう」という自信の度合いを示すために用いる。(ベイズ推定の事前分布・事後分布)
- ガウス過程の分散を関数の推定の自信のなさとみなす



ガウス過程回帰モデル(2/3)

ベイズ推定
・ガウス過程を関数の事前分布に設定すると、事後分布もガウス過程になる。

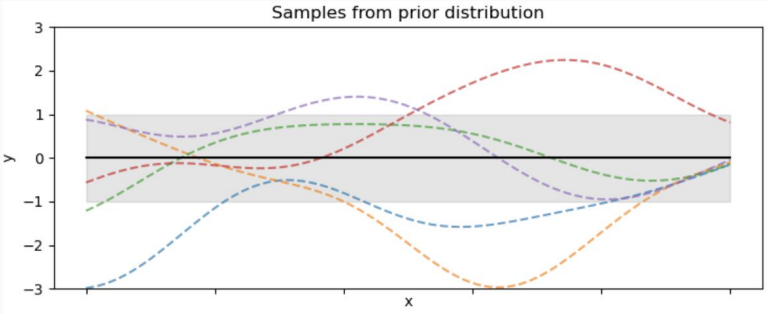
$$f(x) \sim N(\mu(x), k(x,x))$$

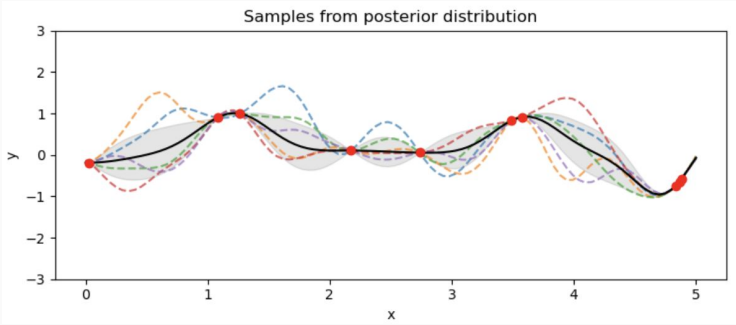
データD

$$f(x) \mid D \sim N(\mu_{post}(x), \sigma^2_{post}(x))$$

$$\mu_{post}(x) = \mu(x) + k(x_D, x)^T (K_n + \sigma^2 I_n)^{-1} (y_D - \mu(x_D))$$

$$\sigma^2_{post}(x) = k(x,x) - k(x_D, x)^T (K_D + \sigma^2 I_n)^{-1} k(x_D, x)$$

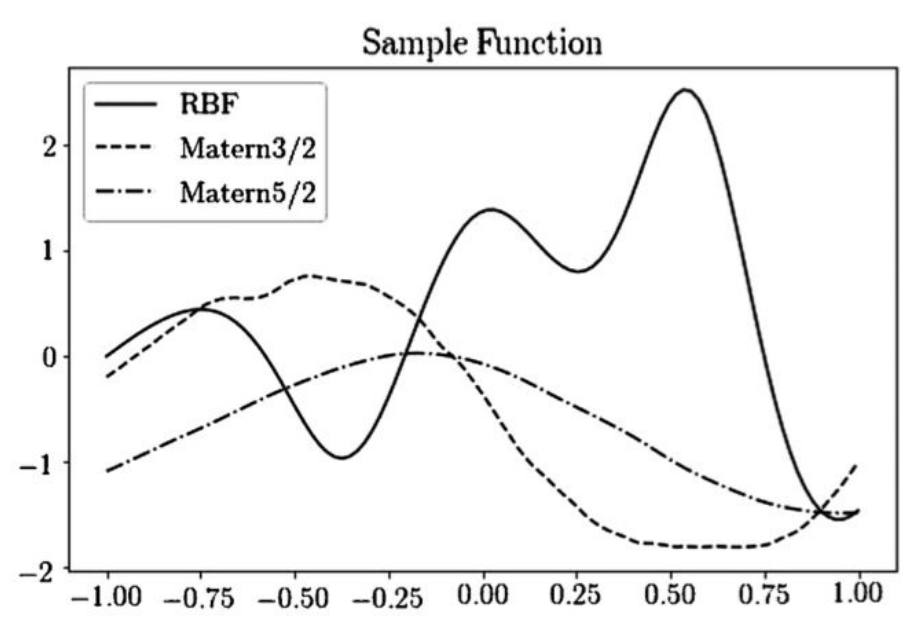




ガウス過程回帰モデル(3/3)

カーネル関数k(x, x')の選択(重要なハイパーパラメータ)

- RBFカーネル
- Matern 3/2カーネルとMatern 5/2カーネル



カーネル関数の選択は、関数f(x)の滑らかさの想定を決めることに対応する。RBF, Matern 5/2, Matern 3/2の順になめらかではなくなっていく。

(経験則)ブラックボックス関数がなめらかはわからないため、RBFよりMaternらを用いるほうが多い。

獲得関数(1/5)

その1　もっと小さな観測値が存在する確信度に基づく獲得関数

- PI(Probability of Improvement)
- 入力xにおける関数の値f(x)が、これまでに観測した結果の最小値τに比べて小さくなる確信度PI(x)を事後分布で計算する。
- PI(x)が最も大きな入力xを次の実験候補にする。

$$\mathrm{PI}(x) = \mathbb{P}[f(x) < \tau] = \Phi\left(\frac{\tau - \mu_{post}(x)}{\sigma_{post}(x)}\right)$$

Φは標準正規分布の累積分布関数

獲得関数(2/5)

その2　改善度の期待値に基づく獲得関数

- EI(Expected Improvement)
- 関数f(x)の値とこれまでに観測した結果の最小値τとの値の差(改善度)が最も大きくなると期待される入力xを次の実験候補にする。

$$\mathrm{EI}(x) = \mathbb{E}\left[[\tau - f(x)]_+\right] = \int_{-\infty}^{\tau} (\tau - f)\phi(f; \mu_{post}(x), \sigma^2_{post}(x))df$$

φは正規分布の確率密度関数

獲得関数(3/5)

その3　関数の値を小さく見積もったときの値を獲得関数にする

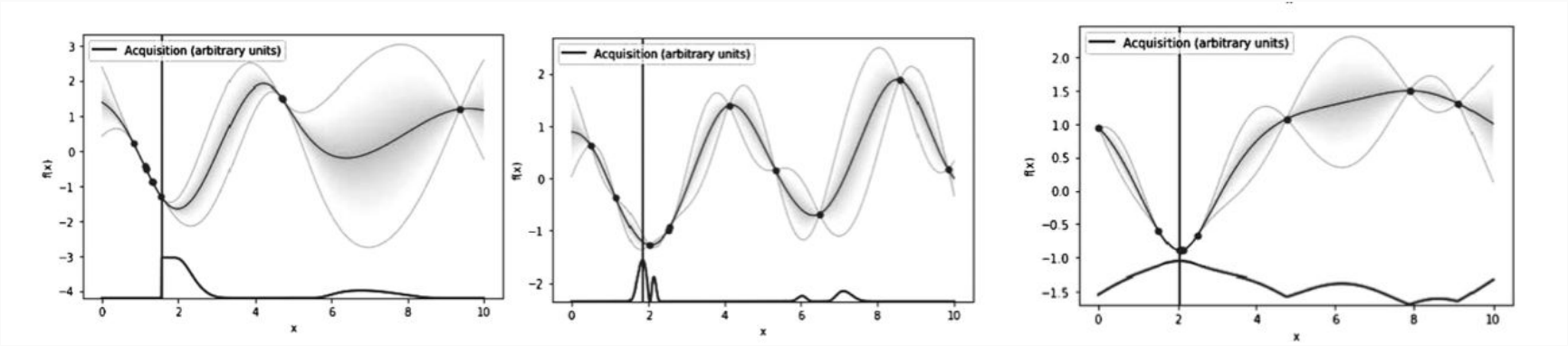
- LCB(Lower Confidence Bound)
- 関数f(x)の事後分布から、入力xにおける関数f(x)の値を小さく見積もったときいくらになるかを計算できる。
- どれくらい小さく見積もるかは分析者がコントロールする:ハイパーパラメータβ

$$\mathrm{LCB}(x) = -(\mu_{post}(x) - \beta\sigma_{post}(x))$$

獲得関数(4/5)

「良い」獲得関数とは？

- 収束するか　大域最適解に収束するか？
- 収束の速度　早く大域最適解に収束する。
- 解きたい問題によりけりで、<u>各手法の明確な優位性はわかっていない。</u>



PI　　EI　　LCB

獲得関数（5/5）

コメント

- 初手にはEIが使われる傾向にある。理由は以下の2通り。
  - PIが過度に活用を優先してしまうような問題でも、EIが適切に動く例が知られている。
  - LCBはハイパーパラメータβを持つが、その設定の仕方はわからない。
  - （注）EIが獲得関数として優れているという意味ではない。
- 獲得関数はベイズ最適化の精度に大きく関わる。
  - 研究レベルでは問題に応じて獲得関数を設計している。
  - 一般的に使える獲得関数も、トンプソンサンプリングや情報量を用いるものまで、まだまだたくさんの提案が行われている。

3. Python言語のパッケージ

Python言語のパッケージ

既成の獲得関数でお手軽にベイズ最適化する場合

- GPyOpt：以下で紹介するGPyをベースにしたパッケージ
- Ax：以下で紹介するBoTorchをベースにしたパッケージ
- 最近提案された獲得関数は、まだまだ実装されていない。

獲得関数を自分で設計する場合（主に研究用途）

- ガウス過程回帰モデルはパッケージで簡単に済ますことが多い。
- ガウス過程回帰モデルの実装：GPy, GPyTorch, BoTorch
  - sklearnにも実装されているが、事後分布からサンプリングできないなど、必要な機能が不足している。
  - BoTorchはGPyTorchをベイズ最適化を実装するときに使いやすくしたもの。