

「技術スタッフ交流会プログラム」
データ構造化ワークショップ2024
Python中級者向け

ハイパーパラメータ 入門編



Smart Solutions株式会社

はじめに

機械学習において非常に重要な要素である「**ハイパーパラメータ**」およびハイパーパラメータの最適化について学びます。

ハイパーパラメータの最適化には「**ブラックボックス最適化**」という手法が多く用いられます。この講義では、ブラックボックス最適化を行うフレームワークである「**Optuna**」をご紹介します。

なお本セミナーの内容は、以下の書籍を参考に事例等を引用して作成しています。

Optunaによるブラックボックス最適化 (オーム社)

著者：佐野 正太郎, 秋葉 拓哉, 今村 秀明, 太田 健, 水野 尚人, 柳瀬 利彦

ブラックボックス最適化とは

入門編では「機械学習」も「ハイパーパラメータ」も登場しません(!?)

- 以下のような問題を解くための手法



問題

出力 y の値が最大(or 最小)となる、
入力 x_1, x_2, x_3 の値は何でしょうか？

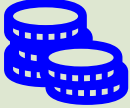



- 機械学習の分野に限らず、様々な分野に適用可能

ポイント

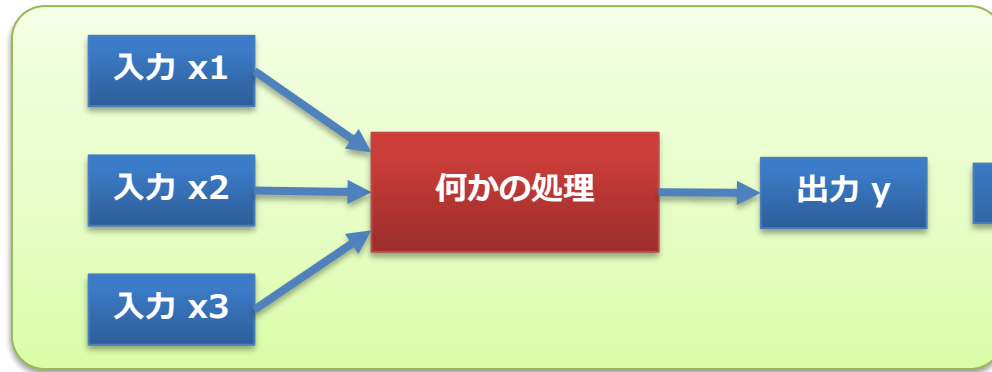
- 「何かの処理」の中身は・・・不明 or 難しい (=ブラックボックス)
- **試しに色々入力してみて、イイ感じの出力を見つけるしかない！！**
 - 代表的な手法：グリッドサーチ、ランダムサーチ、ベイズ最適化

ブラックボックス最適化の適用事例

機械学習以外の分野における様々な適用事例

分野	概要	入力	出力
金融ポートフォリオ最適化 	資産の配分を調整し、リスクを抑えつつリターンを最大化	各資産への投資比率 (例：株式、債券、不動産)	リスクに対するリターンの妥当性 (例：シャープレシオ)
新薬開発の配合最適化 	有効成分や投与量の組み合わせを調整し、治療効果を最大化しつつ副作用を最小化	各成分の割合や投与量	治療効果と副作用のバランス (例：病状改善率、副作用発現頻度)
製品設計の最適化 	部品設計を調整し、強度や性能を維持しつつ製造コストを最小化	設計パラメータ (例：材料の厚み、形状、配置)	性能とコストのバランス (例：強度／耐久性、製造コスト)
料理レシピの最適化 	レシピの様々な要素を調整し、味わいや香りの総合評価を最大化	レシピ (例：材料、分量、温度、調理時間)	実験参加者による評価値 (例：味わい／香りの採点)

用語の定義



問題

出力 y の値が最大(or 最小)となる、
入力 x_1, x_2, x_3 の値は何でしょうか？

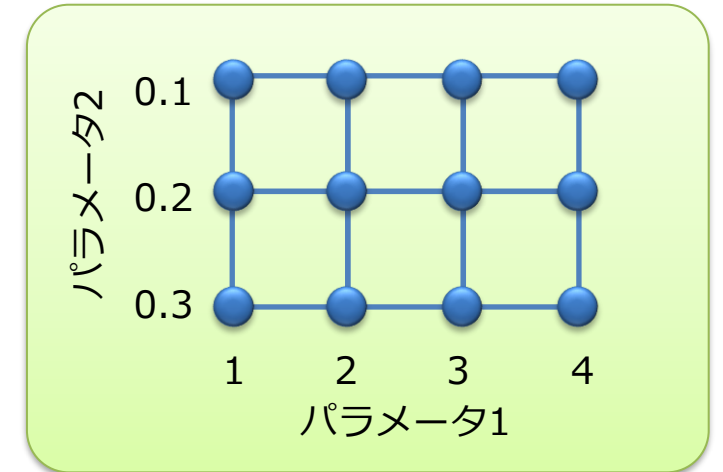
- 今回の講義では、以下のとおり用語を定義
 - 一般的によく使われる用語だが、文献によっては異なる場合もあり

上図中の表現	用語
何かの処理	目的関数
入力(x_1, x_2, x_3)	パラメータ
入力(x_1, x_2, x_3)の値	パラメータ値
出力(y)の値	評価値

手法1：グリッドサーチ

- パラメータ値のすべての組み合わせを試す総当たり方式

例：2つのパラメータに対し、パラメータ値1の候補が{1, 2, 3, 4}、
パラメータ値2の候補が{0.1, 0.2, 0.3}であれば、計12通りを試行



メリット

- 最適解を確実に求めることができる

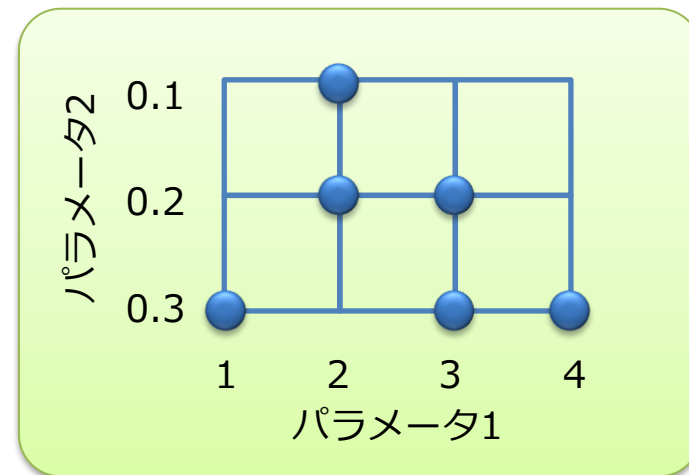
デメリット

- パラメータ数やパラメータ値の候補数が増えると、計算コストが爆発的に増加（次元の呪い）

手法2：ランダムサーチ

- パラメータ値をランダムにサンプリングして評価する手法
 - 事前に試行回数の上限や、評価値の目標などを設定し、そこに達するまで試行

例：前ページのパラメータ値の候補から、ランダムで6回試行

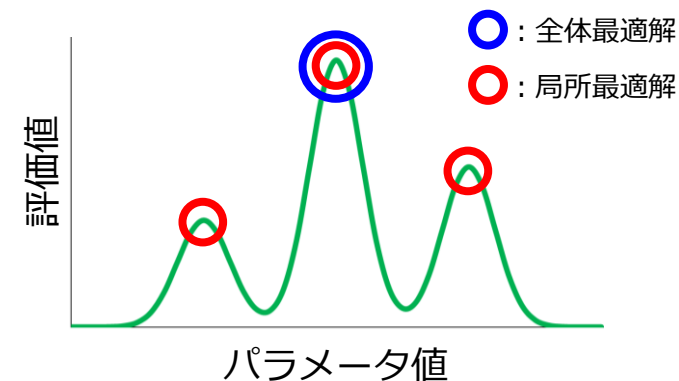


メリット

- 次元の呪いを軽減（ランダム性により、少ない試行回数でもある程度は均一に試行）
- 計算コストを調整可能（試行回数を制限できる）
- 完全にランダムな試行であるため、**局所最適解**に引きずられない

デメリット

- 必ずしも最適解が得られるわけではない
- パラメータ値の選び方に根拠がないため効率的とは言えない



手法3：ベイズ最適化

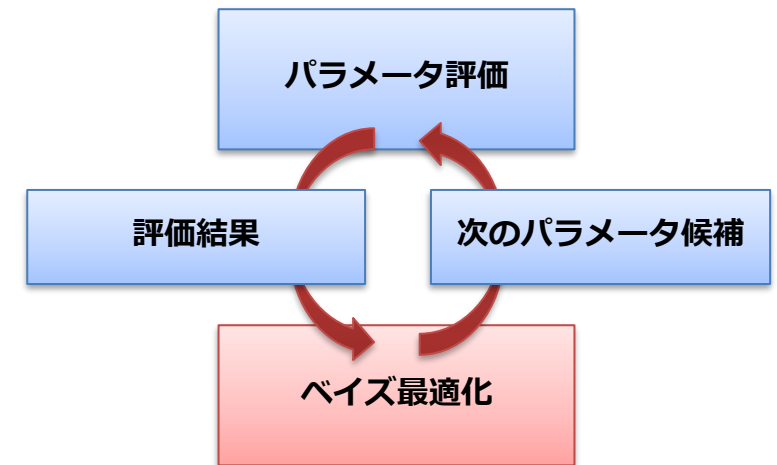
- 以前に試行したパラメータ値とその評価値を元に、次に試すパラメータ値を確率的に選ぶ方法
 - ランダムサーチと同様に、事前に試行回数の上限や、評価値の目標などを設定し、そこに達するまで試行
 - SMBO (Sequential Model-Based Optimization)と呼ばれるアプローチのひとつ

メリット

- 次元の呪いを軽減 (高い評価値が期待できる候補を優先して試行)
- 計算コストを調整可能 (試行回数を制限できる)
- 過去の評価結果を活用するため、効率的な探索が可能

デメリット

- 必ずしも最適解が得られるわけではない
- 過去の評価結果を活用する(=影響を受ける)ため、局所最適解に陥る可能性がある
 - ただし、ある程度のランダム性を持たせるなどの対策はされている



[参考] ガウス過程によるベイズ最適化の原理

1. 初期データから予測モデルを構築

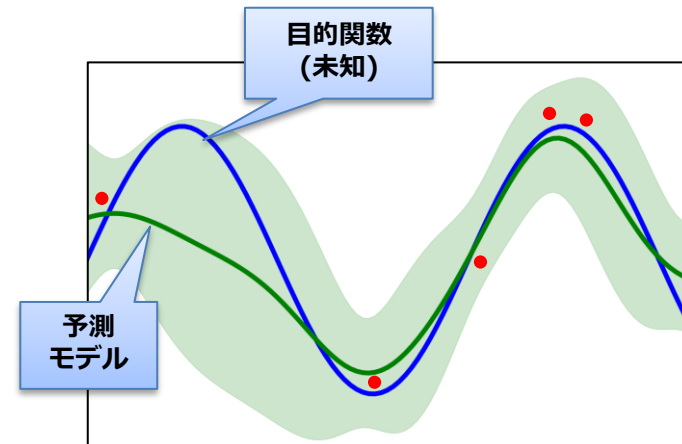
- いくつかの初期データ(ランダムサンプリングなど)を用いて、目的関数を近似する予測モデルを構築

2. 次に試すパラメータ値を決定

- **探索**と**活用**のバランスを考慮して、次に試すべきパラメータを決定
 - 探索：分散が大きい(不確実性が高い)領域を試して、情報を増やす
 - 活用：予測値が高い領域を試して、良い結果を狙う
- このバランスは獲得関数(例：EI, PI, UCBなど)というもので評価される

3. 決定したパラメータを試行した結果に基づき、予測モデルを更新

4. ステップ2～3を繰り返し、目的関数の最適解を目指す



ポイント

効率的な探索が行われるためには、**選択したモデルが、実際の評価値の分布を十分に表現できている**ことが大前提

※ ガウス過程以外にTPE(Tree-structured Parzen Estimator)による実装などもあり

各手法の比較

特徴	グリッドサーチ	ランダムサーチ	ベイズ最適化
試行範囲	全範囲を網羅	ランダムに選択	効率的に重要領域を探索
計算コスト	高い	低くできる (解の精度とのトレードオフ)	低くできる (解の精度とのトレードオフ)
実装の容易さ	簡単	簡単	やや複雑 ※本講義で解決！
解の精度	確実 (試行範囲に最適解が存在することが大前提)	中程度 (計算コストとのトレードオフ、 運任せの面あり)	高い (計算コストとのトレードオフ、 局所最適解、モデル選択に注意)

問題の性質や計算資源に応じて、手法を選ぶことが重要

- 小規模な問題であったり、潤沢な計算資源がある場合はグリッドサーチ
- 均一なサンプリングで、目的関数の大まかな傾向を分析したい場合はランダムサーチ
- 大規模な問題において、できる限り高精度な解を求めるならベイズ最適化 など

Optuna

- Pythonで利用できる、オープンソースのブラックボックス最適化フレームワーク
- グリッドサーチ、ランダムサーチ、ベイズ最適化を含む、様々な手法を簡単に利用可能
- 最適化プロセスの可視化や、最適化の並列実行など、高度な機能も簡単に利用可能

ハイパーパラメータ 入門編 [Google Colaboratory]

<https://colab.research.google.com/drive/1WTCTGqqkXhT3fjjoQqHDqnGemVqs4fy8?usp=sharing>