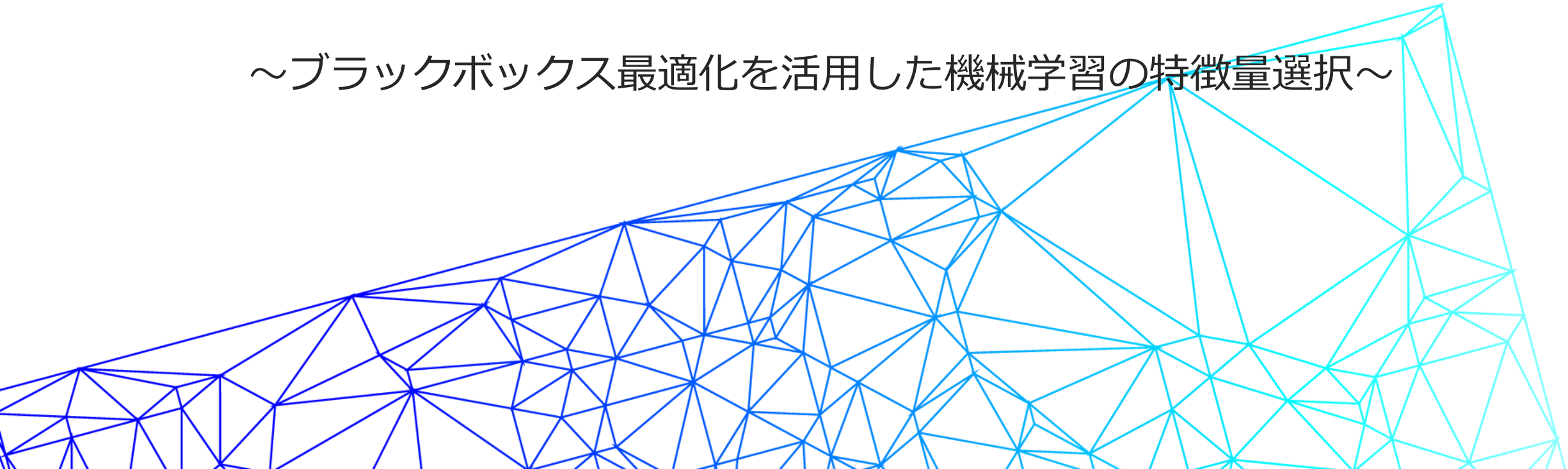
FIXSTARS Amplify

量子コンピュータ時代のプログラミングセミナー

～ブラックボックス最適化を活用した機械学習の特徴量選択～



コンテンツ

Part 1

- Fixstars Amplifyの紹介
- 量子技術・組合せ最適化の概要
- ハンズオンの事前準備

Part 2

- 組合せ最適化ハンズオン

Part 3

- ブラックボックス最適化とは？
- QA-BBO のフロー
- QA-BBOによる最適特徴量抽出
  - 問題設定
  - Amplify-BBOptによる実装
- 次のステップ

ご質問は随時お願いします。

FIXSTARS Amplify

株式会社 Fixstars Amplify (“Amplify”)

- **最適化のための量子コンピューティング・プラットフォームの提供**

1,100+ ユーザー所属組織数（企業、研究所、大学）

1.2億+ 累計ユーザー実行回数 (Amplify AE)

- 2021年10月に設立

- 親会社：株式会社フィックスターズ
  - ソフトウェア高速化のプロフェッショナル集団
  - 日本で初めて D-Wave Systems 社と提携 (2017年)

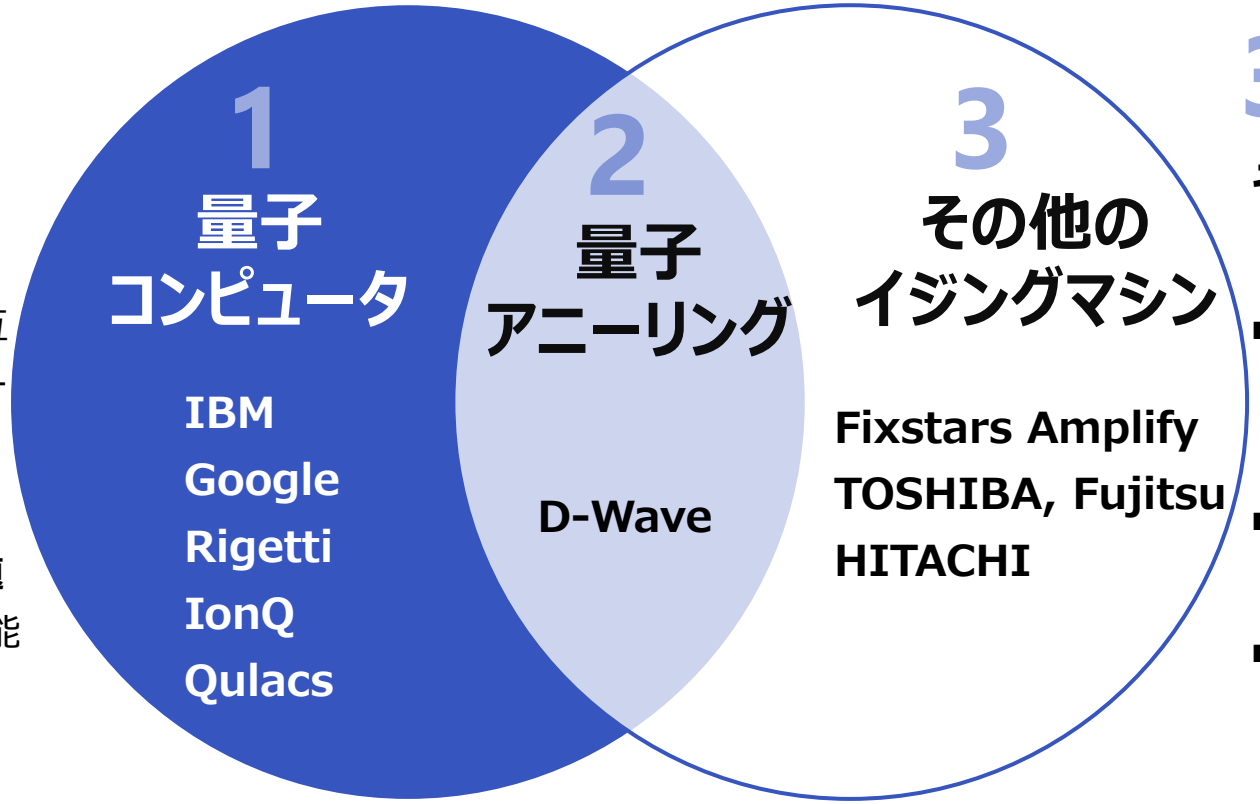
量子技術とFixstars Amplify

量子・量子インスパイアード技術

1 量子コンピュータ

（量子ゲート方式）

- 古典汎用コンピュータの上位互換。量子ゲートを操作。エラー訂正機能の無いNISQ型実機がクラウド利用可能
- QAOAにより**組合せ最適化問題**（**QUBO**）を取り扱うことが可能
- 演算規模：**～数100ビット**



3 その他のイジングマシン

（半導体技術に基づくイジングマシン）

- 二次の多変数多項式で表される目的関数の**組合せ最適化問題**（**QUBO**）専用マシン
- 統計物理学におけるイジング模型に由来。様々な実装により実現。
- 演算規模：
  260,000+ビット（*Amplify AE*）

2 量子アニーリング（量子焼きなまし法式のイジングマシン）

- イジングマシンの一種。量子イジング模型を物理的に搭載したプロセッサで実現。
  量子効果を物理的に調整し、自然計算により低エネルギー状態が出力
- **組合せ最適化問題**（**QUBO**）を扱う専用マシン
- 演算規模：**～数1,000ビット**

FIXSTARS Amplify

最適化問題とは

- 最適化
  - さまざまな制約の下で、数ある選択肢の中から、ある観点で最適な選択を決定
    - 製品の高性能化、製造過程の効率化、コスト削減、歩留まり向上
    - 商品の発注計画、効率的な物流ルート、金融資産の運用
    - 災害復旧スケジュール、公共施設の配置、エネルギー需給バランス
- 数理最適化
  - 対象となる問題を数式で記述し、数理的な計算手法で最善策を求める
    - 問題の数式表現：　　数理モデル
    - 数理モデルの構築：　　定式化
- 数理モデルの構成要素
  - 目的関数：　　目的達成度を表す数式 (最小あるいは最大化)
  - 決定変数：　　選択肢となる変数
  - 制約条件：　　決定変数間の取り得る条件式 (制約関数)

} 制約条件を満たす決定変数は実行可能解と呼ばれる

数理最適化問題の分類

数理最適化

- 連続最適化問題
  - 決定変数が連続値（実数など）
- 離散最適化問題
  - 整数計画問題（決定変数が整数）
  - **0-1 整数計画問題（決定変数が二値）**
    - 線形計画問題（従来の数理最適化ソルバ）
    - **非線形問題（量子・量子インスパイアード）**
      - **QUBO**

Quadratic → **高次項もOK**、線形項もOK（論理変換により次数下げ可能）

Unconstrained → **制約条件もOK**（論理変換・制約のペナルティ化により考慮可能）

Binary → **イジング、整数、実数変数もOK**（論理変換・変数変換により考慮可能）

Optimization

QUBO 目的関数（バイナリ変数・2次多項式）

$$f(q) = \sum_{i<j} Q_{ij} q_i q_j + \sum_i Q_{ii} q_i$$

f: Objective　q: Variables　Q: Factor

$f(q)$ を最小化するような q を求める

Fixstars Amplify プラットフォーム

FIXSTARS Amplify

Fixstars Amplify とは

- **いつでも** 開発環境 と 実行環境 がセット
  すぐにアプリ開発と実行が出来る
- **誰でも** ハードウェアや専門的な知識が不要
  無料で開発がスタート可能
  多くの解説、サンプルコード
- **高速に** 26万ビットクラスの大規模問題の
  高速処理と高速実行が可能
- **あらゆる** 一般に公開されている全てのイジング
  マシンを利用可能

Fixstars Amplify の対応マシンの一例

- **Fixstars Amplify Annealing Engine (Amplify AE)**

FIXSTARS Amplify

STANDARD MACHINE

GPUの優れた並列計算能力を最大限に活用し、複雑な組合せ最適化問題を高速・高精度に解く革新的なアニーリングエンジンです。

- **外部ソルバー**
  - **標準マシン**

Quantum Annealing Ising Machine	Quantum Annealing Ising Machine	Quantum Annealing Ising Machine
D-Wave	TOSHIBA	FUJITSU
STANDARD MACHINE	STANDARD MACHINE	STANDARD MACHINE
D-Wave Systems 2000Q / Advantage	Toshiba Digital Solutions SQBM+	Fujitsu Digital Annealer

…

  - **BYOL マシン**

Quantum Annealing Ising Machine	Mathematical Optimization Solver	Gate-based Quantum Computer	Quantum Circuit Simulator
HITACHI	GUROBI OPTIMIZATION	IBM	
Hitachi, Ltd. CMOS Annealing Machine	Gurobi Gurobi Optimizer	IBM IBM Quantum	Qulacs Qulacs

…

STANDARD MACHINE

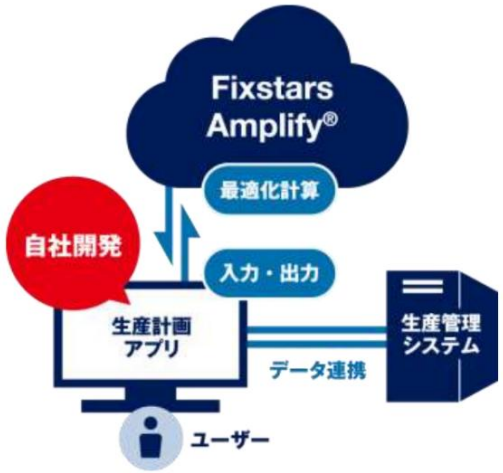
- ベンダ各社と個別マシン利用契約なし、
- 評価・検証用ベーシックプランなら無料、

で利用可能！←**「いつでも」、「誰でも」**

今後も幅広い対応マシンの追加が続々と行われる予定です！←**「あらゆる」**

活用領域とユースケース（PoC・実稼働）

生産計画	• 多品種少量生産、保全計画、設備投資、在庫
従業員割り当て	• 食品、輸送、製造
エネマネ	• エネルギーミックス、装置の運転制御
経路	• 配送、船舶、無人搬送車 (AVG)
メディア	• 最適広告配信
研究開発、設計	• 材料設計 • 物理シミュレーション • **ブラックボックス最適化**



Amplify インタビュー　検索

アニーリングマシンのプログラミング体験

イジングマシンの実行手順

1. 数理モデル検討

解きたい課題の「目的関数」「決定変数」とその「制約条件」を検討する

2. 定式化

「 多項式」で「目的関数」と「決定変数」を記述する
「決定変数」 に対する「制約条件」を Amplify で表現

3. モデル変換

各マシンの仕様や制限に準拠した形式にモデルを変換する
（例: 二次項に制約がある場合は「グラフマイナー埋め込み」問題を解く）

4. 入力データ準備

各マシンのSDKやAPI仕様に合わせてQUBO模型をデータ化する

5. マシン実行

マシンを実行して出力の変数値や目的関数値を解析する。上記の逆の手順を**辿り解きたい課題**の「決定変数」を解釈する

Amplify SDKにより自動で処理

Amplify SDK: 基本的な使用方法 (1/3)

- まずはインポート

```
# Install Amplify SDK to Google Colab
! pip install -q amplify

# Import all functions and classes
from amplify import *
```

- 使用するマシンを選択

```
# Fixstars Amplify AE v1
client = AmplifyAEClient()

# Timeout of 1s
client.parameters.time_limit_ms = 1000  # ms

# API token
client.token = "AE/XXXXXXXXXXXXXXXXX"
```

その他のクライアントを使用する場合はドキュメントを参照
https://amplify.fixstars.com/ja/docs/amplify/v1/clients.html

Amplify SDK: 基本的な使用方法 (2/3)

- 目的関数の定式化（多項式）
  - バイナリ変数の発行

```
# Instantiate a variable generator
g = VariableGenerator()

# Create a binary variable array (length 2)
q = g.array("Binary", 2)
```

バイナリ変数 (**“Binary”**) だけでなく、イジング変数 (**“Ising”**) や、整数変数 (**“Integer”**)、実数変数 (**“Real”**) も指定可能

  - 目的関数の構築

```
f = 2 * q[0] * q[1] + q[0] - q[1] + 1
print(f)
# 2 q_0 q_1 + q_0 - q_1 + 1
```

3次以上の高次多項式も可能。マシンが対応していない場合は Amplify SDK が内部的に次数下げを行う

Amplify SDK: Basic Usage (3/3)

- Amplify数理モデルの作成とマシンの実行

```
model = Model(f)
result = solve(model, client)
```

目的関数と制約条件からモデルを作成し
使用するクライアントと共に求解

- 結果の可視化

```
print(f"objective = {result.best.objective}")
# objective = 0.0

print(f"q = {q.evaluate(result.best.values)}")
# q = [0. 1.]
```

最良解を `best` で指定
目的関数の値を `objective` にて
変数の値を `values` で得る

Amplify SDK によるプログラミング例

```
from amplify import *

# 決定変数を生成
g = VariableGenerator()
q = g.array("Binary", 2)

#目的関数を構築
f = 2 * q[0] * q[1] + q[0] - q[1] + 1

# Amplify モデルを構築
model = Model(f)

#ソルバーの設定
client = AmplifyAEClient()
client.parameters.time_limit_ms = 1000 #ms
client.token = "AE/XXXXXXXXXXXXXXXXXXXx"

# 求解の実行
result = solve(model, client)

# 結果の表示
print(f“objective = {result.best.objective}”)
print(f"q = {q.evaluate(result.best.values)}")
```

1. 定式化
   - 決定変数：スカラーあるいは配列型
   - 目的関数：決定変数による数式処理
   - 制約条件：制約条件の構築及び管理
2. ソルバークライアントの選択
   - ソルバークライアントオブジェクトの構築
   - ほぼ全てのパラメータの設定が可能
3. ソルバーを実行
   - 論理モデルをハードウェアのスペック等に合わせたモデルに変換
   - 適切なモデル変換・定式化手法を選択
4. 解の取得
   - マシンの出力解を逆変換し決定変数の形式で出力

ワークショップ
事前準備（事前メールの内容）

ワークショップの事前準備 (1)

- 【事前メールに記載】ご自身のPC (ブラウザ上) で Python プログラミングを行います。Google Colaboratory を使うので、事前にログイン出来ることを確認をお願いします（要 Google アカウント）

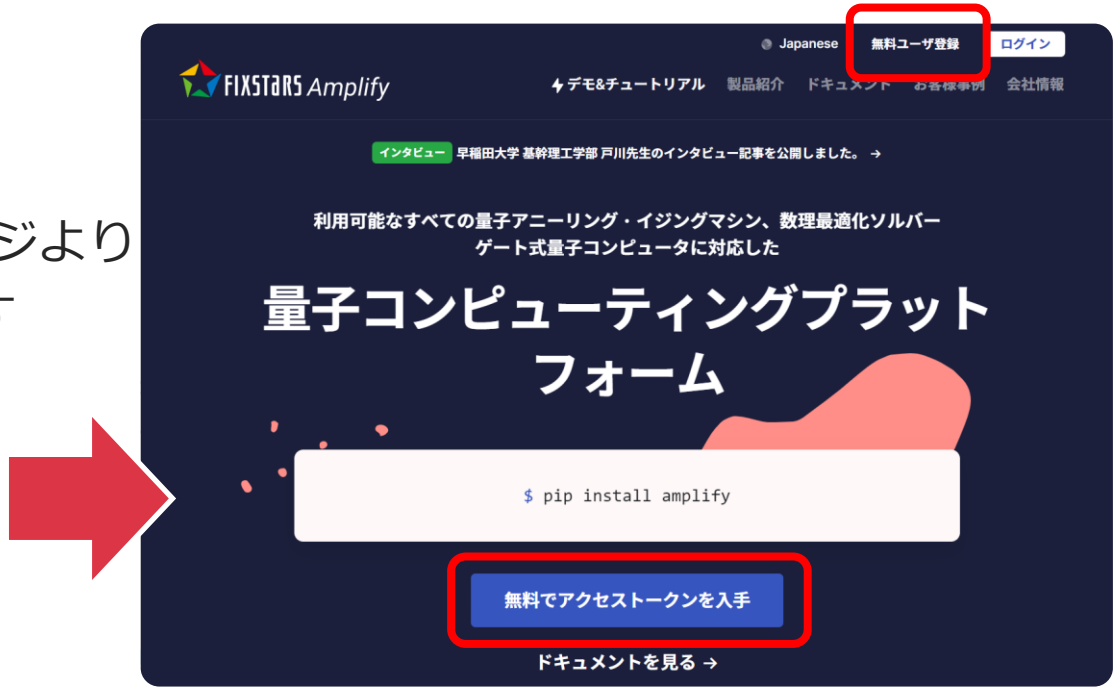
Google Colab 検索

https://colab.research.google.com/

- 【事前メールに記載】 Fixstars Amplify ホームページよりユーザ登録の上、無料トークンの取得をお願いします（1分で終わります）

Fixstars Amplify 検索

https://amplify.fixstars.com/



ワークショップの事前準備 (2)

【事前メールに記載】

- 取得されたトークンを用いて、トークンチェック用サンプルコードが動くか確認をお願いします。

  https://colab.research.google.com/drive/1-Jh2nlhWO97OO96WgBwCSOQmP8BVa6-T （※URLはZoomのチャット欄を参照）

- サンプルコードは閲覧のみ可能な状態です。**「ファイル」→「ドライブにコピーを保存」**の上、ご自身のトークンを入力してください。その後、Shift + Enterで実行下さい。

```
! pip install amplify
token = "AE/**********************************" # ご自身のトークンを入力
```

- ご自身のトークン番号は、Amplifyウェブページよりご確認いただけます。

- 実行後、以下の結果が出力されればOKです。

```
result: [q_0, q_1] = [1. 1.] (f = 0.0)
```

FIXSTARS Amplify

「数の分割問題」のハンズオン

通常の組合せ最適化
（ブラックボックス最適化への導入）

数の分割問題

- 与えられた n 個の整数 $a_0, \cdots, a_{n-1}$ を二つの集合に分ける。集合内の数の和が、もう一方の集合内の数の和と等しくなるようできるか?

NP完全問題：とても難しい問題として知られている → 全通り試すしか方法は無い

数の分割問題（具体例と解法の方針）

具体例

`{2,10,3,8,5,7,9,5,3,2}` の10個の数の完璧な分割は見つけられるか？

答　え

- 存在する
  - `{2,3,5,7,10}` と `{2,3,5,8,9}`
  - どちらも和は 27
- 分割方法は 23 通り存在する (対称を除く)

どうやって解くか？

・ひとつの『数』がどちらの集合に分割されるか全通り試す → $2^{10} = 1024$通り
・効率のよい厳密な方法は知られていない・・・
(もし発見されたら大騒ぎ)

数の分割問題（定式化）

最適化問題：数の分割において最も惜しい組合せは何か？

- **目的関数**

  {集合1の和} – {集合2の和} の絶対値を最小化

- **決定変数**

  数 a_i がどちらの集合に属するかを s_i で表す

  - a_i = { 2,10, 3, 8, 5, 7, 9, 5, 3, 2}
  - s_i = {-1, 1,-1, 1,-1,-1, 1, 1, 1, 1}

数理モデル

- 目的関数 $$f = \left| \sum_{i=0}^{N-1} s_i a_i \right| \quad (s_i \in \{-1, +1\})$$

$\sum s_i a_i$ は、自然と
{『1』の集合の和} – {『-1』の集合の和}
となる！

数の分割問題（バイナリへの式変形）

0-1整数二次計画問題への変換

- Quadratic Unconstrained Binary Optimization (QUBO) 式

$$f = \left| \sum_{i=0}^{N-1} s_i a_i \right| \quad (s_i \in \{-1, +1\})$$

$\sum s_i a_i$ は、自然と
{『1』の集合の和} – {『-1』の集合の和}
となる！

$$\rightarrow \left(\sum_{i=0}^{N-1} s_i a_i \right)^2 \quad (s_i \in \{-1, +1\})$$

絶対値を二次式で表す

$$\rightarrow \left(\sum_{i=0}^{N-1} (2q_i - 1) a_i \right)^2 \quad (q_i \in \{0, +1\})$$

±1をバイナリ変数で表す

数の分割問題（定式化の具体例）

問　題

- a_i ={2,10,3,8,5,7,9,5,3,2} の10個の数の完璧な分割は見つけられるか？

決定変数

- $q_i = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9\} (q_i \in \{0,1\})$ で集合0又は集合1、どちらに所属するかを表す

目的関数

$$f = \left(\sum_{i=1}^{N} (2q_i - 1)a_i \right)^2$$

目的関数を展開

$$f = \begin{pmatrix} 2(2q_0 - 1) + 10(2q_1 - 1) + 3(2q_2 - 1) + 8(2q_3 - 1) + 5(2q_4 - 1) \\ +7(2q_5 - 1) + 9(2q_6 - 1) + 5(2q_7 - 1) + 3(2q_8 - 1) + 2(2q_9 - 1) \end{pmatrix}^2$$

数の分割問題（プログラム）

サンプルコード：
https://colab.research.google.com/drive/11vI2sHbtP_aaZHOTjsGYoGcvw2dHhD1z

1. 定式化

- 問題の定義と決定変数生成器による決定変数の生成

```
a = [2, 10, 3, 8, 5, 7, 9, 5, 3, 2]
q = amplify.VariableGenerator().array("Binary", len(a))
```

- 目的関数、$f = \left(\sum_{i=1}^{N}(2q_i - 1)a_i\right)^2$、の定式化（①②③は同等）

①
```
f = ((2 * q - 1) * a).sum() ** 2
```

②
```
f = 0
for i in range(len(a)):
    f += (2 * q[i] - 1) * a[i]
f **= 2
```

③
```
f = amplify.sum((2 * q - 1) * a) ** 2
```

色々な書き方が出来る

2. 実　行

```
result = amplify.solve(f, client)
```

3. 結　果

```
q = [1, 1, 1, 0, 1, 1, 0, 0, 0, 0], f = 0.0, w = 27
```

得られた目的関数の値0

各集合の合計値27

各数字に対して、集合0か、集合1か

オンラインデモ & チュートリアル

Amplify デモ 検索

https://amplify.fixstars.com/demo

デモアプリケーション

ピクロスパズルの求解

プログラミング難易度 ★★★☆☆

複雑な定式化の例として、数字で与えられるヒントを元にマスを塗り、絵を完成させるパズルゲーム、ピクロスを解くアプリを開発します。

デモアプリ サンプルコード

チュートリアル応用編

ブラックボックス最適化 (1)

プログラミング難易度 ★★★★☆

複雑で未知な目的関数にも適用可能な、機械学習と組み合わせ最適化を組み合わせたブラックボックス最適化手法を紹介し、Amplifyを用いて実装します。

サンプルコード

チュートリアル応用編

ブラックボックス最適化 (2)

プログラミング難易度 ★★★★☆

機械学習と量子アニーリング・イジングマシンを活用するブラックボックス最適化の適用例として、疑似的な高温超電導を実現する材料探索を取り扱います。

サンプルコード

チュートリアル応用編

ブラックボックス最適化 (3)

プログラミング難易度 ★★★★☆

化学プラントにおける生産量を最大化するための運転条件最適化を行います。最適化には、機械学習モデルに基づくブラックボックス最適化と化学反応に関する物理シミュレーションを用います。

サンプルコード

チュートリアル応用編

ブラックボックス最適化 (4)

プログラミング難易度 ★★★★☆

流体機器設計に不可欠な翼型の最適化問題を取り上げます。最適化には、組み合わせ最適化及び機械学習に基づくブラックボックス最適化と流体シミュレーションを用い、翼の揚抗比を最大化するように翼型の探索を行います。

サンプルコード

デモアプリケーション

容量制約つき運搬経路問題 (CVRP)

プログラミング難易度 ★★★★☆

運送業における効率的な配送計画の策定やごみ収集や道路清掃における訪問順序の最適化等での応用が期待される容量制約つき運搬経路問題（CVRP）を取り扱います。

デモアプリ サンプルコード

チュートリアル応用編

ブラックボックス最適化 (5)

プログラミング難易度 ★★★★☆

ブラックボックス最適化により、商業施設による交通集中が発生し得る都市における、交通渋滞を低減するような信号機群の最適制御を実施します。最適化の実施及び実証には、マルチ・エージェント・シミュレーションによる交通シミュレーションを用います。

サンプルコード

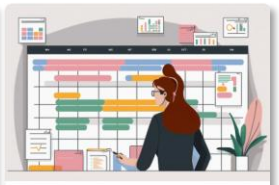
チュートリアル応用編

定式化による交通信号機の最適化

プログラミング難易度 ★★★★☆

都市における渋滞を最小化するために、刻一刻と変化する交通状況に応じ、組合せ最適化を用いてリアルタイムに信号機の最適制御を実施します。また、その様な信号機制御を実施した際の都市の交通量をシミュレーションします。

サンプルコード



チュートリアル応用編

10. 整数長ジョブスケジューリング問題

プログラミング難易度 ★★★☆☆

あらかじめ決まった数のジョブとマシンがあり、それぞれのジョブにかかる時間が分かっているとします。それぞれのジョブをいずれかのマシンに割り当てます。ジョブスケジューリング問題では、最も早く全ジョブが完了するような割り当て方を求めます。

サンプルコード

チュートリアル基礎編

画像のノイズ除去

プログラミング難易度 ★★★☆☆

画像のノイズ除去を行うアプリケーションを開発します。

サンプルコード

チュートリアル応用編

会議室割当問題

プログラミング難易度 ★★★☆☆

制約条件を用いて定式化するアプリケーションの例として会議室割当問題のアプリケーションを開発します。

サンプルコード

チュートリアル応用編

タクシーマッチング問題

プログラミング難易度 ★★★☆☆

目的関数と制約条件を用いて定式化するアプリケーションの例としてタクシーマッチング問題のアプリケーションを開発します。

サンプルコード

デモアプリケーション

グラフ彩色問題

プログラミング難易度 ★★★☆☆

Fixstars Amplifyによる、グラフ彩色問題の定式化を体験します。

デモアプリ サンプルコード

デモアプリケーション

巡回セールスマン問題

プログラミング難易度 ★★★★☆

Fixstars Amplifyによる、巡回セールスマン問題の定式化を体験します。

デモアプリ サンプルコード

デモアプリケーション

数独

プログラミング難易度 ★★★☆☆

Fixstars Amplifyによる、数独の定式化を体験します。

デモアプリ サンプルコード

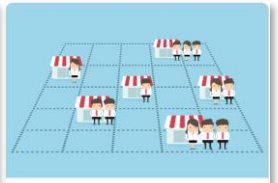
デモアプリケーション

ライドシェア

プログラミング難易度 ★★★★★

集合型ライドシェアの最適化アプリケーションを体験します。

デモアプリ サンプルコード



デモアプリケーション

タスク割当問題

プログラミング難易度 ★★★★★

店舗とタスクに従業員を割り当てる組合せ最適化問題のアプリケーションを体験します。

デモアプリ サンプルコード

デモアプリケーション

ポートフォリオ最適化

プログラミング難易度 ★★★★☆

リスクとリターンを考慮した株式ポートフォリオの最適化アプリケーションを体験します。

デモアプリ サンプルコード

FIXSTARS Amplify

Black-Box Optimization

通常の組合せ最適化とブラックボックス最適化

通常の組合せ最適化

- **目的関数** を定式化可能（例：QUBO）
  - 数の分割問題（**差** を最小化）
    $f = [\Sigma(2q_i - 1)a_i]^2$
  - 経路最適化（**経路距離** を最小化）
    $f = \Sigma\Sigma\Sigma d_{i,j} q_{n,i} q_{n+1,j} \cdots$

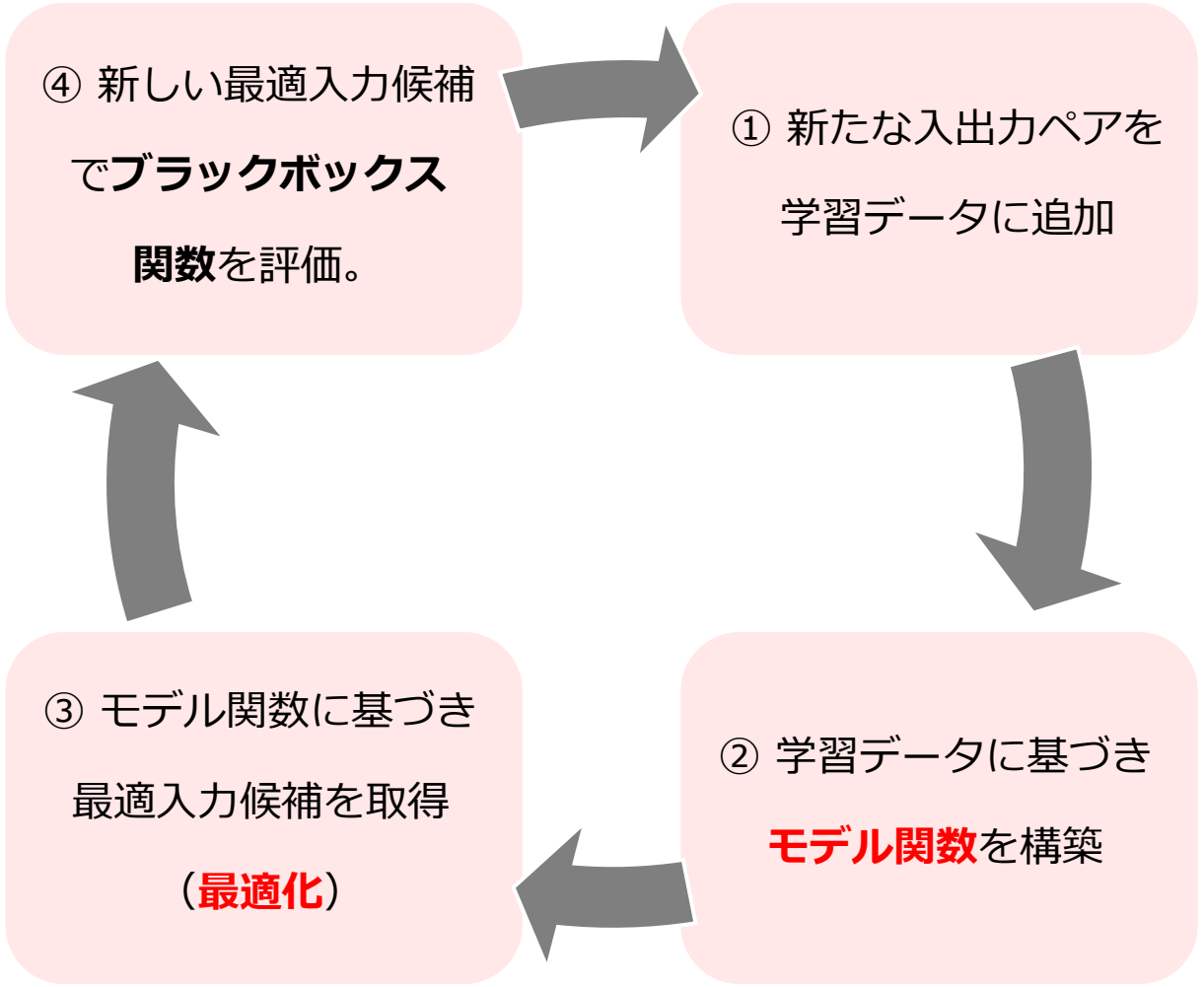
- **最適化の実施**
  - イジングマシンなどより、定式化された目的関数を最小化する

ブラックボックス最適化 (BBO)

- **定式化が困難な** ブラックボックス **目的関数**（解析・実験）
  - 低 **損失** な流体デバイス形状の同定
  - 高 **性能** な材料・構造トポロジーの探索
  - シミュレータやモデルの **誤差** を最小化するハイパーパラメータ最適化

- **最適化の方法**
  - シミュレーションや実験を併用しながら、**試行錯誤**に基づくアプローチ

ブラックボックス最適化のフロー



QA-BBO

モデル関数とQAを使うBBO手法の総称

- **FMQA** (Kitai, et al., *Phys. Rev. Res.*, 2020)
  - **モデル関数 → FM** **F**actorization **M**achine
  - **最適化 → QA** **Q**uadratic-optimization **A**nnealing
- **Kernel-QA**
  - **モデル関数 → Kernel model**
  - **最適化 → QA**

QA-BBO の特徴

- **高次元**の最適化問題にも強い！
  （次元の呪い）
- **制約条件**にも強い！

モデル関数としてのFM

- モデル関数 $g(\boldsymbol{x})$ に機械学習モデルの一種である Factorization Machine (FM) を用いると、次のように変数 $\boldsymbol{x}$ に対する2次式での記述ができる。

$$g(\boldsymbol{x}|\boldsymbol{w},\boldsymbol{v}) = w_0 + \langle \boldsymbol{w},\boldsymbol{x} \rangle + \sum_{i=1}^{n}\sum_{j=i+1}^{n} \langle \boldsymbol{v}_i,\boldsymbol{v}_j \rangle\, x_i\, x_j$$

$$= w_0 + \sum_{i=1}^{n} w_i x_i + \frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n} v_{if}\, x_i\right)^2 - \sum_{i=1}^{n} v_{if}^2\, x_i^2\right)$$

→ QUBO

- k はハイパーパラメータであり、BBOでは通常 **10** 程度でOK。$\boldsymbol{w}$ および $\boldsymbol{v}$ は、FM学習後に取得されるFMパラメータ
  - FM パラメータ数は k に依存。$k = n$ のときは QUBO の相互作用項と同じ自由度がある一方、 k を小さくすることでパラメータ数を減らし過学習を抑制。

K. Kitai, et al., Phys. Rev. Res. (2020).

ブラックボックス最適化 活用例

材料分野に限らず、幅広い分野へ適用可能

QA-BBO: 活用例 (Amplify チュートリアル)

Amplify デモ　検索

チュートリアル応用編

ブラックボックス最適化 (2)

プログラミング難易度 ★★★★☆

機械学習と量子アニーリング・イジングマシンを活用するブラックボックス最適化の適用例として、疑似的な高温超電導を実現する材料探索を取り扱います。

サンプルコード

材料最適化

FMQA × 物理モデル

チュートリアル応用編

ブラックボックス最適化 (3)

プログラミング難易度 ★★★★☆

化学プラントにおける生産量を最大化するための運転条件最適化を行います。最適化には、機械学習モデルに基づくブラックボックス最適化と化学反応に関する物理シミュレーションを用います。

サンプルコード

化学プラント運転条件最適化

FMQA × 化学シミュレーション

チュートリアル応用編

ブラックボックス最適化 (4)

プログラミング難易度 ★★★★☆

流体機器設計に不可欠な翼型の最適化問題を取り上げます。最適化には、組み合わせ最適化及び機械学習に基づくブラックボックス最適化と流体シミュレーションを用い、翼の揚抗比を最大化するように翼型の探索を行います。

サンプルコード

翼形状最適化

FMQA × 流体シミュレーション

チュートリアル応用編

ブラックボックス最適化 (5)

プログラミング難易度 ★★★★☆

ブラックボックス最適化により、商業施設による交通集中が発生し得る都市における、交通渋滞を低減するような信号機群の最適制御を実施します。最適化の実施及び実証には、マルチ・エージェント・シミュレーションによる交通シミュレーションを用います。

サンプルコード

信号制御最適化

FMQA × マルチ・エージェント・シミュレーション

チュートリアル応用編

ブラックボックス最適化 (6)

プログラミング難易度 ★★★★☆

ブラックボックス最適化により、撹拌性能に影響を与える設計パラメータに対して、混合効率が最大化されるような撹拌機の最適設計を実施します。最適化の実施および評価には、濃度分布に基づく簡易的な撹拌シミュレーションを用います。

サンプルコード

機器設計最適化

FMQA × 撹拌シミュレーション

FIXSTARS Amplify

QA-BBO: 活用例 (Amplify ユーザー)

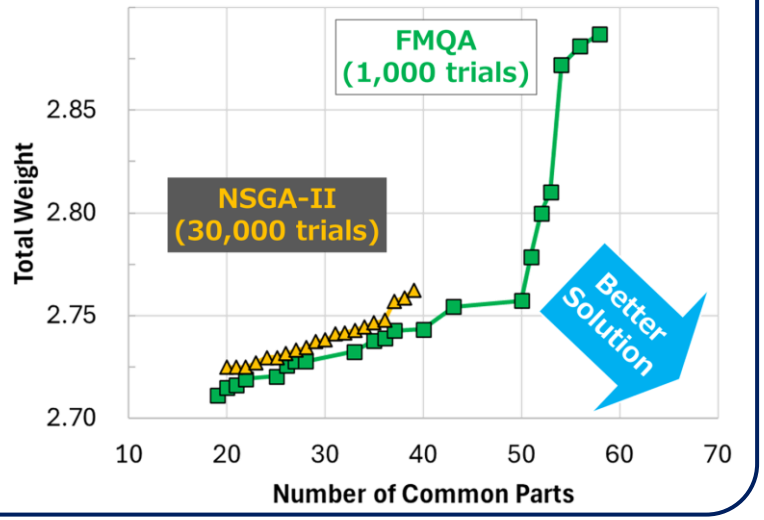
活用領域 化学、 創薬、 食品、 自動車、 電機、 通信、 重工、 エネルギー、ヘルスケア・・・

- 非線形現象の逆問題
- 機械学習：コスト↓精度↑
- 設計開発における部品選定
- 材料配合最適化
- 多目的最適化
- 物理モデルの簡略化

QA-BBO: 実際の応用事例

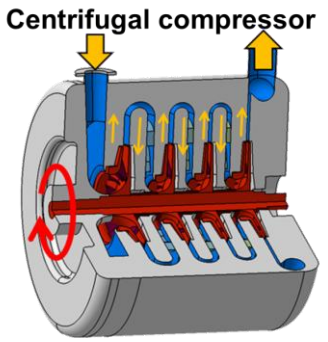
マツダ 様

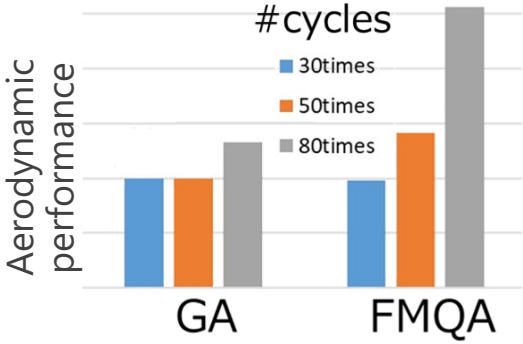
- 市販車両を対象した、複数車種同時設計最適化問題
  実数変数 200 以上の大規模問題
- **多目的最適化：**車体の軽量化と共通部品点数の最大化
- **制約条件：50 以上の制約条件**
  （衝突性能、製造制約、構造制約など）
- 従来手法 (GA/BO) と比較し、3%程度のシミュレーションコストでの最適化を実現。従来手法と同等以上の解を見つけることに成功



川崎重工業 様

- **ターボ機械**の設計最適化問題
- 従来より商用最適化ソフトによる遺伝的アルゴリズム (GA) を使用
- **最適化規模が大きくなると**最適解の求解までに時間がかかり、開発期間が長期化するといった課題
- 従来ツールに比べ、**同じシミュレーション回数**でも**より優れた解**が得られた



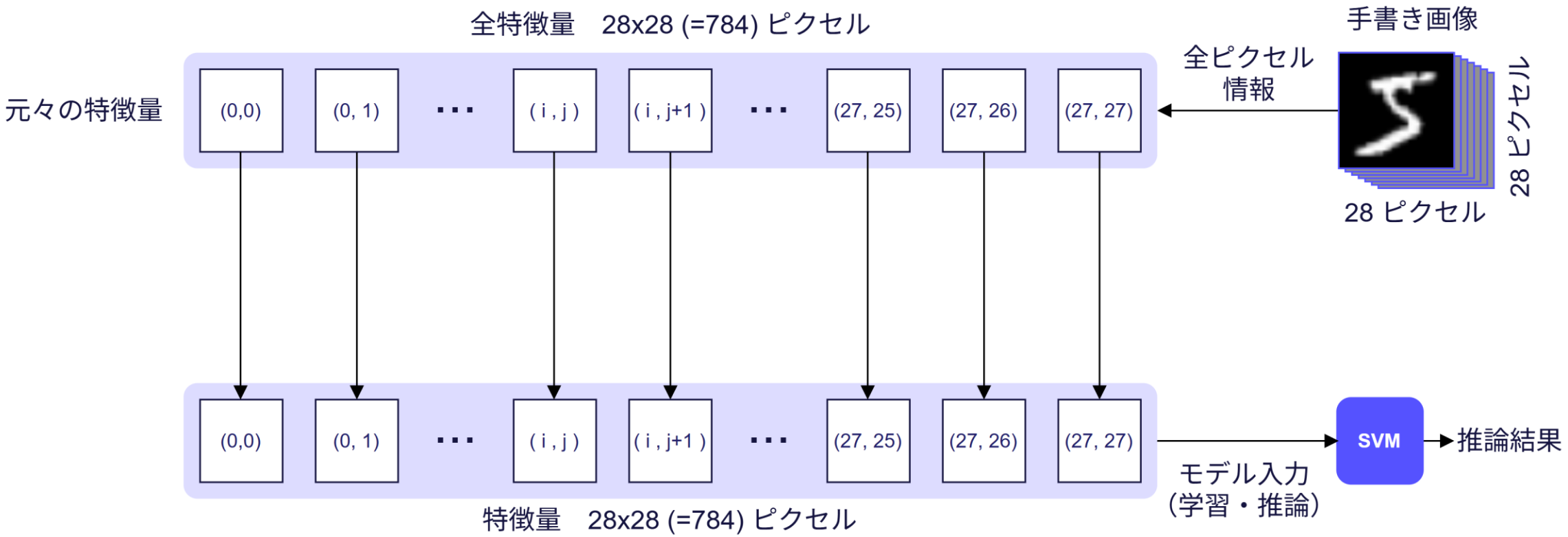


ブラックボックス最適化ハンズオン

問題設定と目的関数

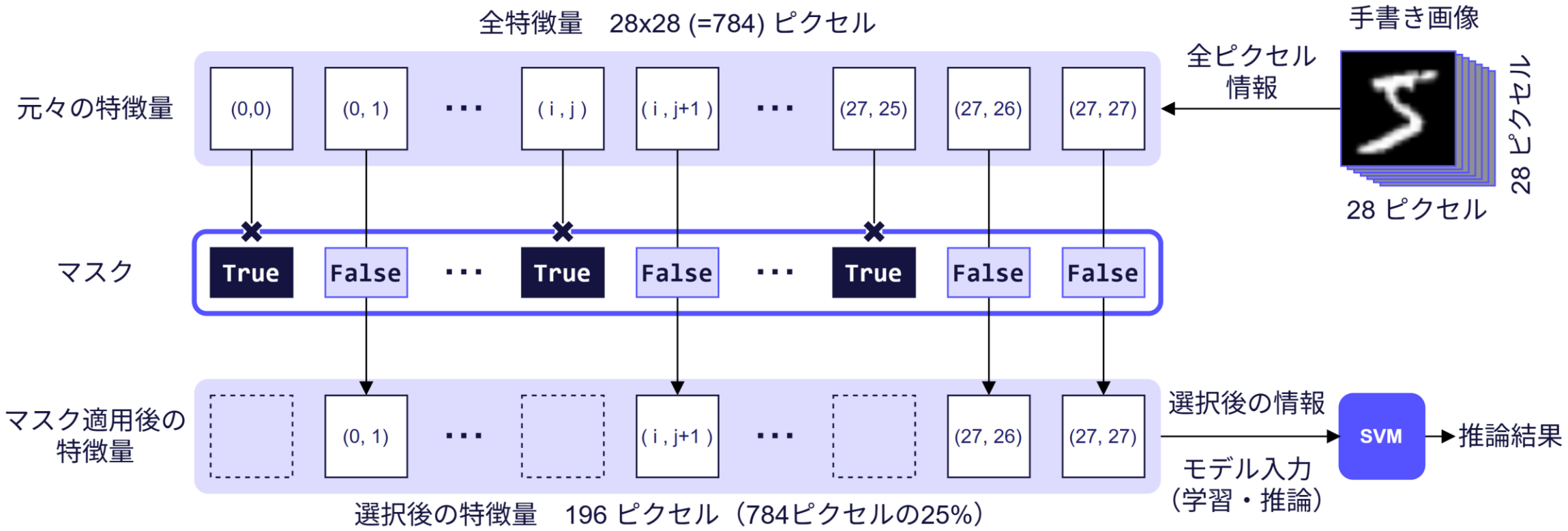
問題設定

- SVMモデルによるMNIST（手書き画像）認識



問題設定

- SVMモデルの最適特徴量選択

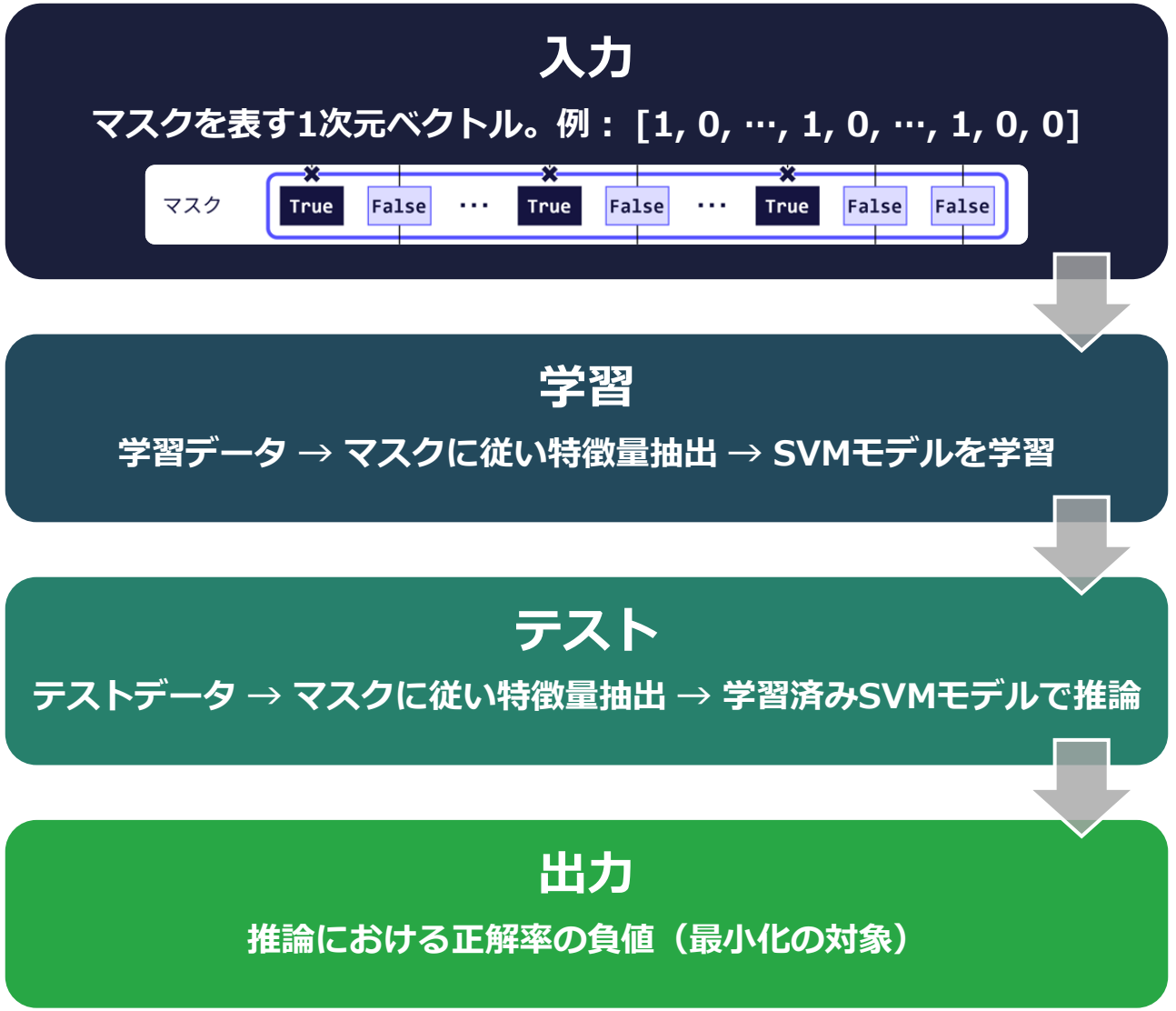


最適なマスクを探索し、軽く、正解率が高いSVMを目指したい！

問題設定

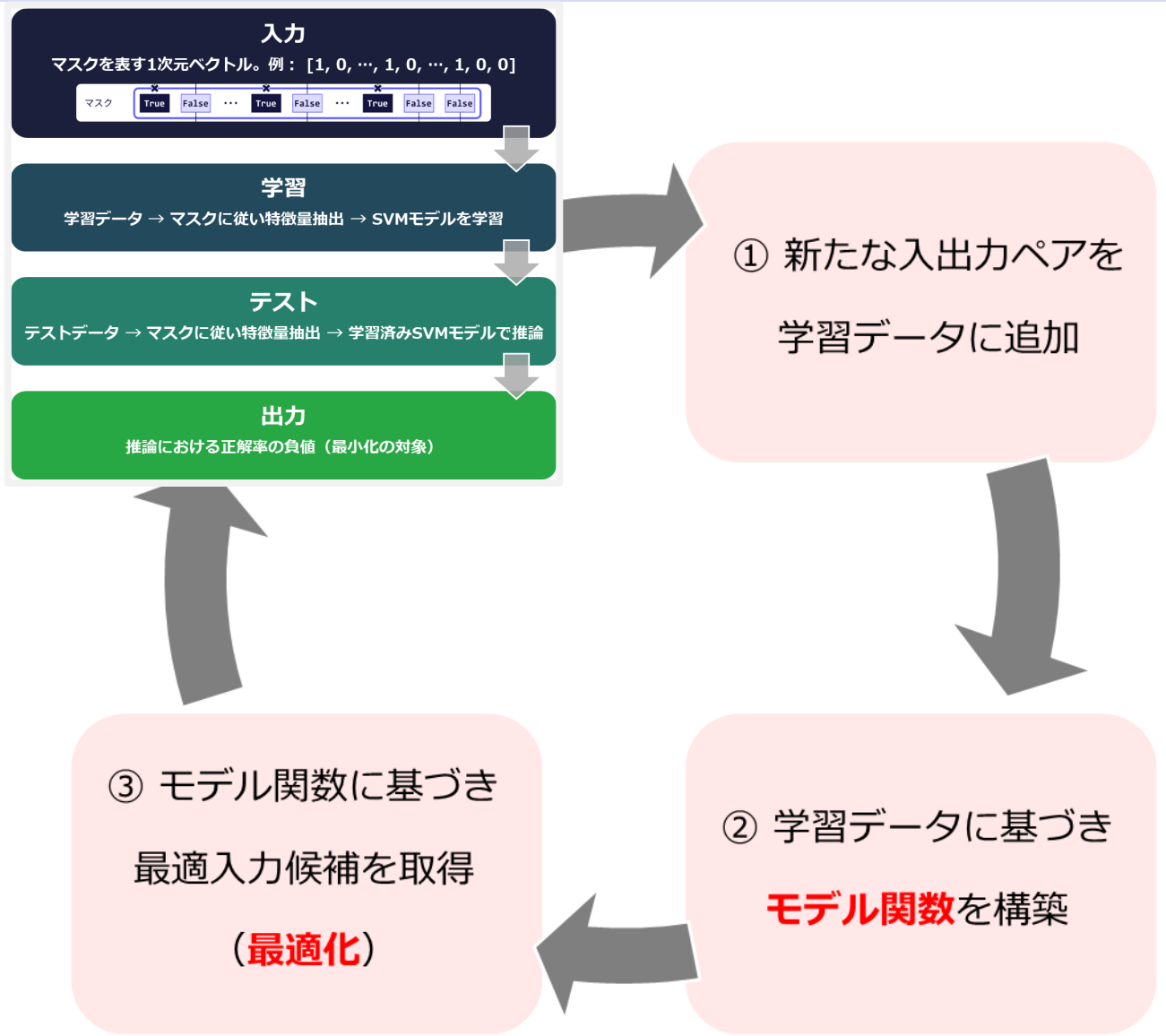
特徴量抽出のためのマスクの最適化

- 推論の正解率の負値が最小となるように、特徴量抽出マスクを最適に決定
- ブラックボックスな目的関数では→の処理を実施。



QA-BBO活用の3方針

- **Amplify SDK + PyTorch**
  - 実装コスト：大
  - 柔軟性：大
- **Amplify-BBOpt ★**
  amplify.fixstars.com/ja/docs/amplify-bbopt/v1/
  - 実装コスト：小～中
  - 柔軟性：中
- **Web App**
  (Amplify-BBOpt backend)

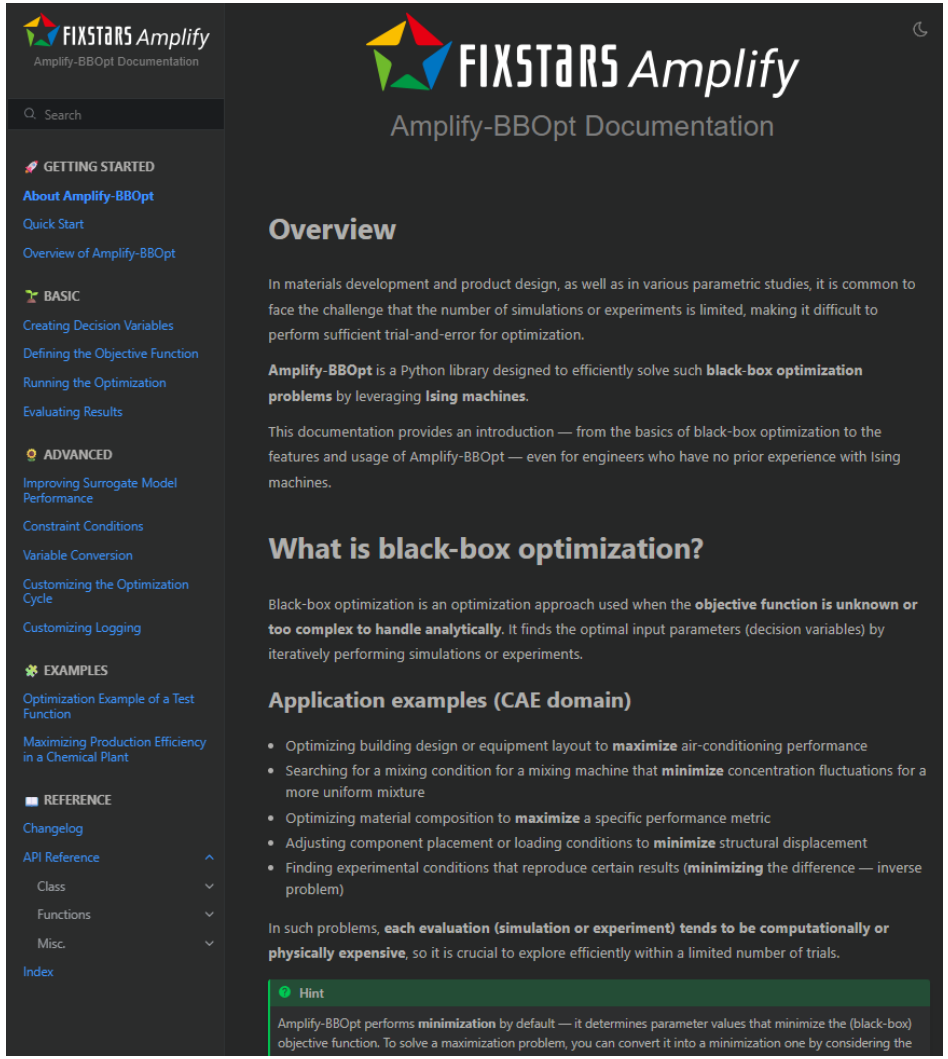


ブラックボックス最適化ハンズオン

Amplify-BBOptによる実装

Amplify-BBOpt: 特徴

- **QA-BBO**[1] の簡単実装：
  - 目的関数の定義（**シミュレーション**や**実験**）
  - イジングマシンによる最適解の**効率的な探索**
  - 過去の結果に基づき、**次に実施するべき**シミュレーション・実験への**入力条件を提示**
- Amplify-BBOpt で使えるアルゴリズム：
  - **F**actorization **M**achine with **Q**uadratic-optimization **A**nnealing (**FMQA**) [2]
  - Polynomial-Based **Kernel** with **Q**uadratic-optimization **A**nnealing (**Kernel-QA**) [3]



amplify.fixstars.com/ja/docs/amplify-bbopt/v1/

[1] QA-BBO: **Q**uadratic-optimization **A**nnealing based **B**lack-**B**ox **O**ptimization

[2] Kitai, K., Guo, J., Ju, S., Tanaka, S., Tsuda, K., Shiomi, J., Tamura, R.: Designing metamaterials with quantum annealing and factorization machines. Phys. Rev. Res. 2, 013319 (2020).

[3] Y. Minamoto and Y. Sakamoto, A black-box optimization method with polynomial-based kernels and quadratic-optimization annealing, arXiv:2501.04225 (2025).

実装 (1/5): ブラックボックスな目的関数

```python
import numpy as np
from amplify_bbopt import blackbox, BinaryVariable


# 問題サイズ（＝最適化対象のマスクの長さ）
problem_size = PIX_SIZE * PIX_SIZE

# マスク率 75% で MnistSvm をインスタンス化
mnist_svm = MnistSvm(mask_ratio=0.75)

# バイナリ決定変数配列を作成
variables = [BinaryVariable() for _ in range(problem_size)]

@blackbox
def my_blackbox_func(x: list[int]=variables):
    # x は要素数 28*28 の一次元配列であることを確認
    assert len(x) == problem_size
    # ブラックボックス関数として、モデルの正解率の負値を返却
    return -mnist_svm.train_eval(x)
```

入力

マスクを表す1次元ベクトル。例： [1, 0, …, 1, 0, …, 1, 0, 0]

学習

学習データ → マスクに従い特徴量抽出 → SVMモデルを学習

テスト

テストデータ → マスクに従い特徴量抽出 → 学習済みSVMモデルで推論

出力

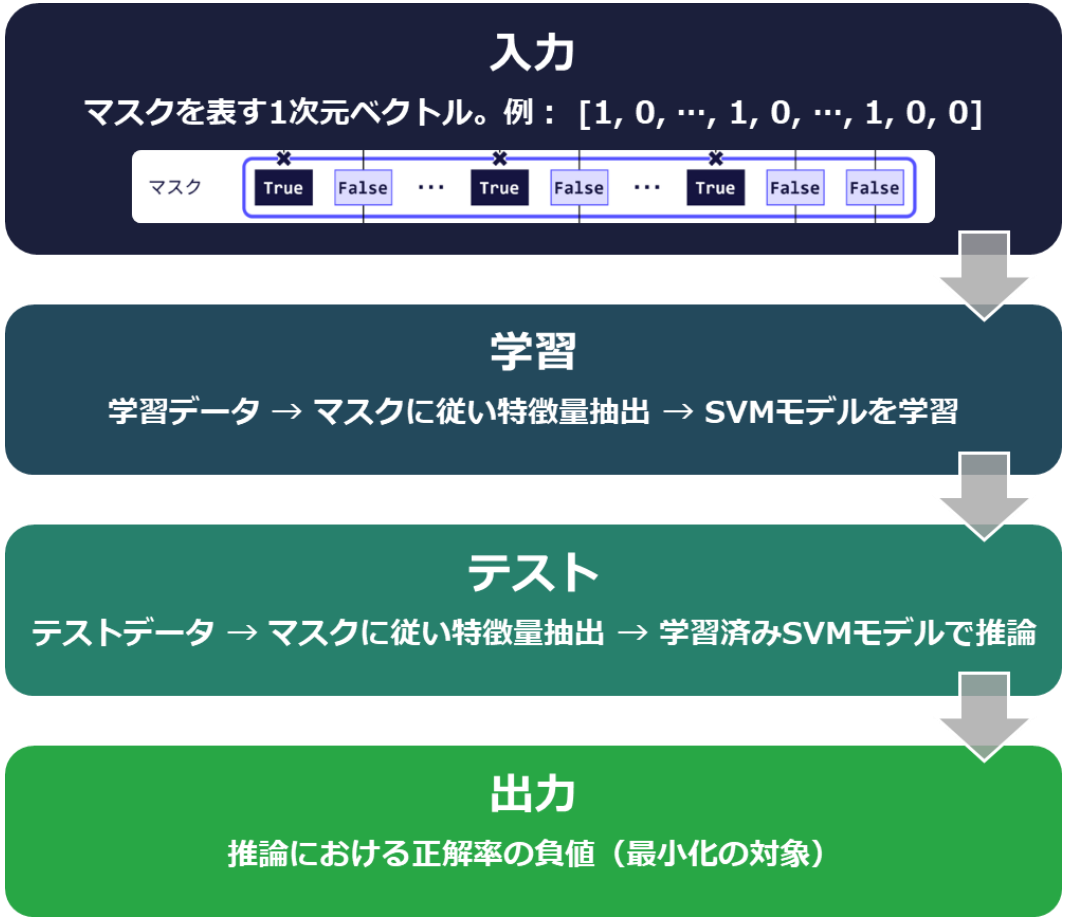
推論における正解率の負値（最小化の対象）

実装 (2/5): マスク率に関する制約条件

```
from amplify import equal_to


# マスクを全て足し合わせると、num_masked_features になるという等式制約
constraint =
     equal_to(sum(my_blackbox_func.variables.x),
              mnist_svm.num_masked_features)
```

- 予め定められた割合の特徴量を取捨するための制約条件
  - 左辺：sum(my_blackbox_func.variables.x)
    - 決定変数配列の合計値
  - 右辺：mnist_svm.num_masked_features
    - マスクされる特徴量の数
      (マスク率 75% * 全特徴量数 28^2)



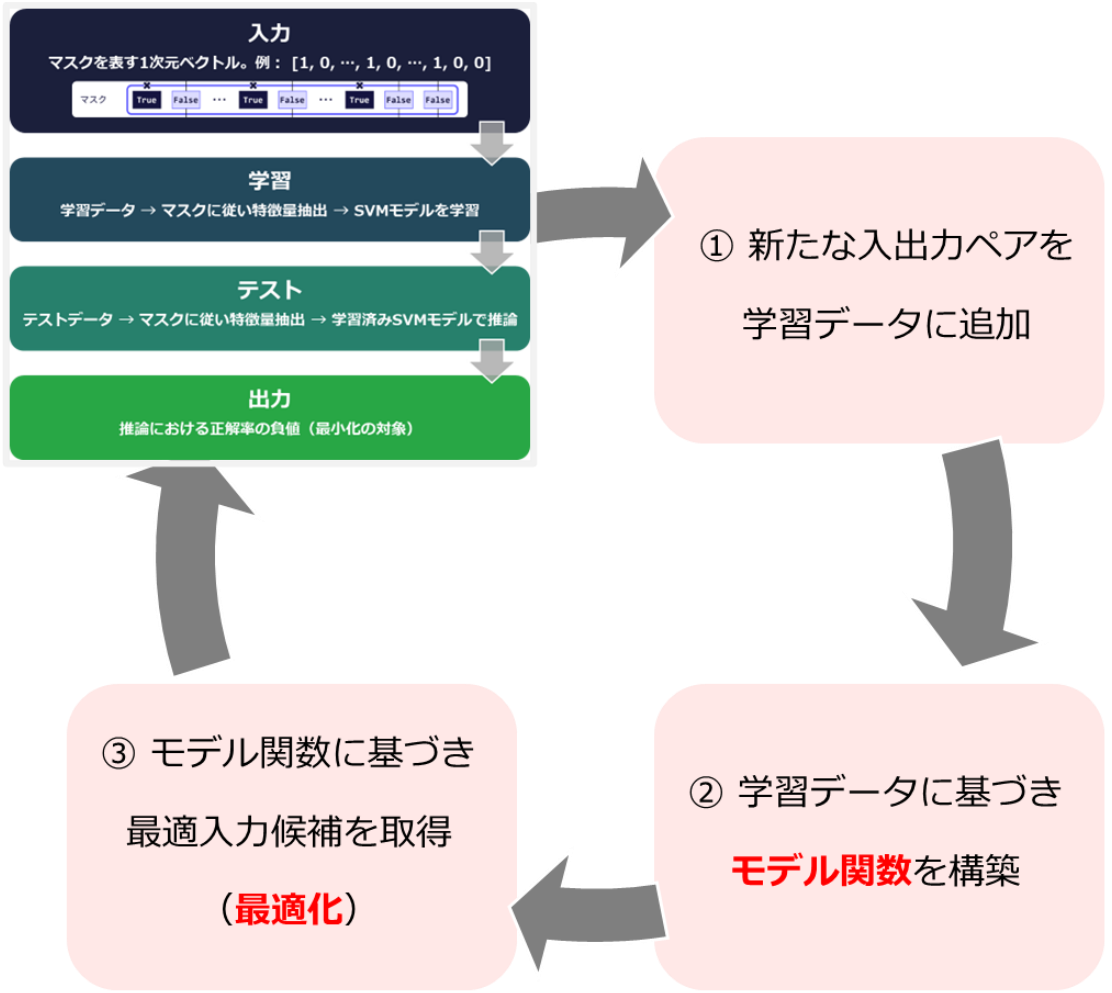
実装 (3/5): 初期学習データ

```python
from amplify_bbopt import Dataset


# 初期学習データのランダム生成および追加（x, y は既存の学習データ）
init_data = Dataset(x, y)


# 初期学習データのサイズ
num_init_data = len(y)


print(num_init_data)
```



ごく少量の学習データサンプルが必要
既存のデータがない場合、Amplify-BBOpt で
初期学習データを生成することも可能。

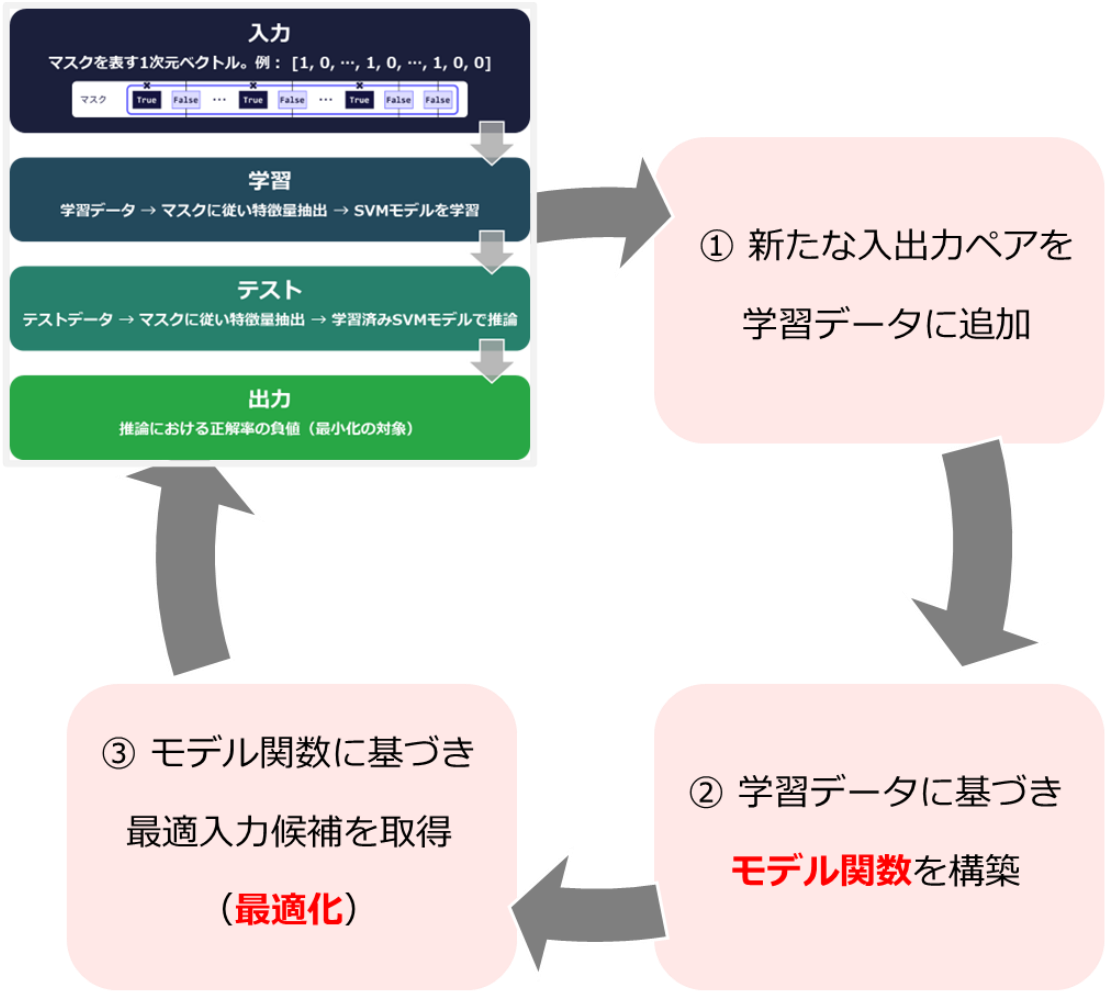
実装 (4/5): ソルバーの設定

```python
from amplify import AmplifyAEClient


# ソルバークライアントを Amplify AE に設定
client = AmplifyAEClient()
client.token = "XXXXXXXXXXXXXXXXXXX"  API トークン
client.parameters.time_limit_ms = 5000
```

その他の対応ソルバークライアントの一例：

- TOSHIBA SQBM+
  - **`ToshibaSQBM2Client()`**
- Fujitsu DA
  - **`FujitsuDA4Client()`**
- D-Wave Systems
  - **`DWaveSamplerClient()`**
  - **`LeapHybridSamplerClient()`**
  - **`LeapHybridCQMSamplerClient()`**

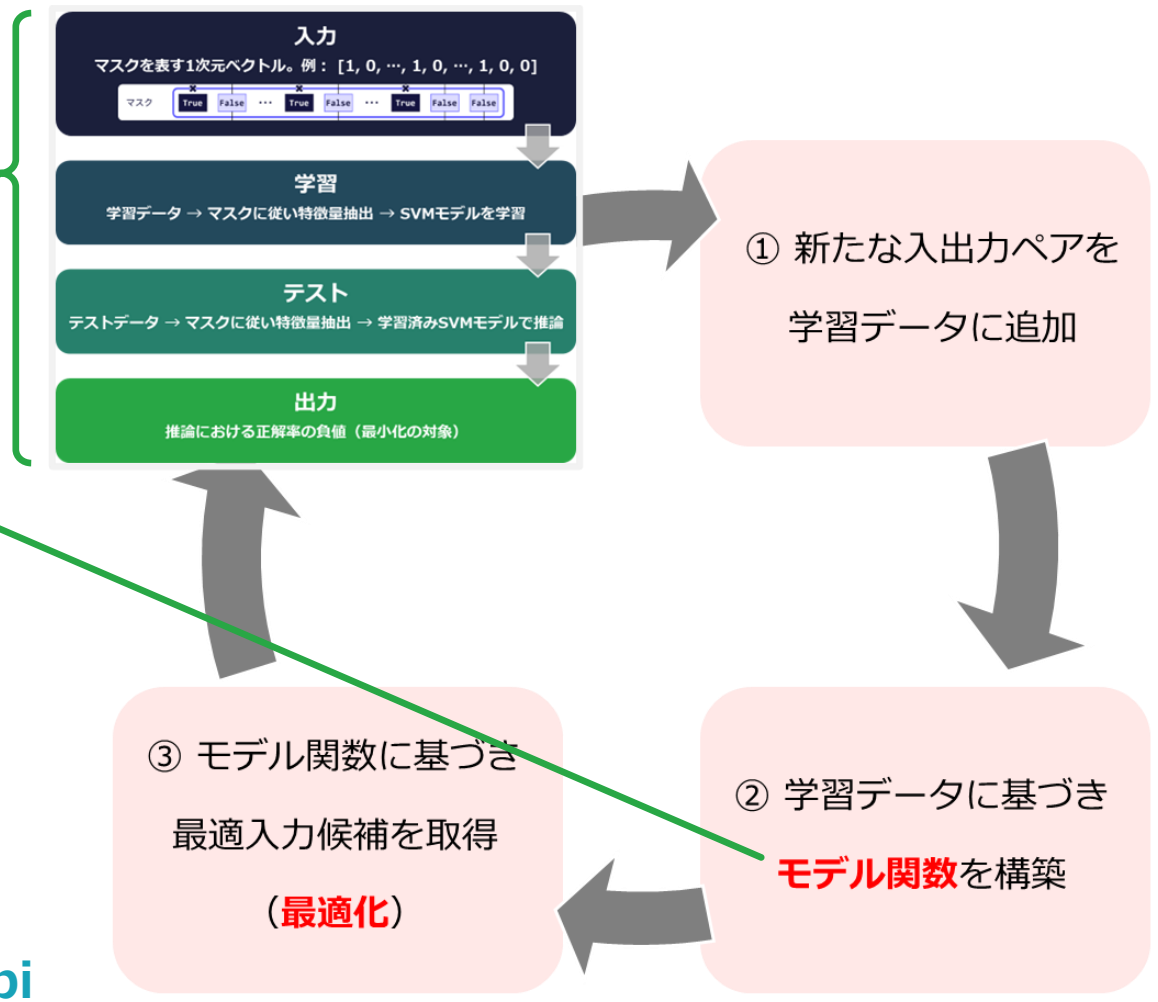


実装 (5/5): 最適化サイクル

```python
import torch
from amplify_bbopt import Optimizer, KMTrainer


# Optimizer のインスタンス化
optimizer = Optimizer(blackbox=my_blackbox_func,
                      trainer=KMTrainer(),
                      client=client,
                      training_data=init_data,
                      constraints=constraint)


# 最適化サイクルの実施
optimizer.optimize(num_iterations=20)
```



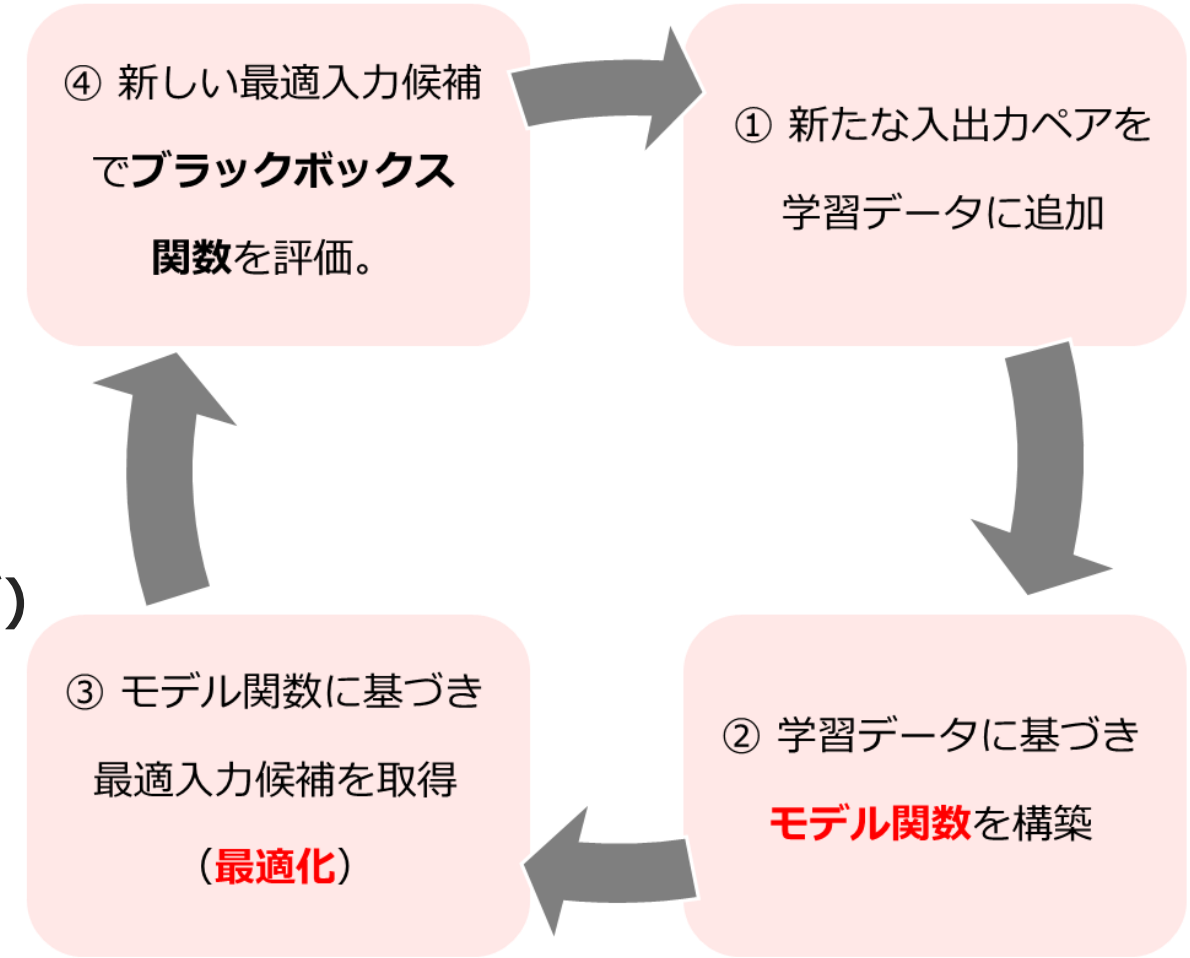
サンプルプログラム：
https://colab.research.google.com/drive/1zbbiLTyb6ieimrtDIgmfF_NYr25bxSSH

今後の進め方

次のステップ

1. **サンプルコードを実行する**
2. **種々の要素を追加実装する**
   - ブラックボックス関数の変更
   - 制約条件を追加
   - アルゴリズムへの工夫の追加

   ➔ 質問などはお気軽にお問い合わせ
3. **AQMCSE 2026 へ参加 (9/14-17 オランダ)**
   - https://aqmcse2026.dryfta.com/
   - 2026年4月30日
     講演アブストラクトの提出締め切り
   - 2026年8月24日
     参加登録締め切り



研究・開発者向けおすすめの進め方

二次・非線形を上手に使いこなせるように、**弊社と一緒に**取り組みを進めていきましょう！

Step 1 ～2ヵ月程度

無料版でチュートリアルをお試し

無料セミナーに参加

Step 2 3ヵ月程度

プライベートトレーニングを受講し、自社課題（簡易版）に挑戦

Step 3 6ヵ月～

プレミアムサポートや Plusオプション を通じて弊社も手厚く伴走します

Sプレミアムプランで自分で手を動かして自社課題を本格的に検証

Step 4 2～6ヵ月程度

弊社が本格的に手を動かします！

研究委託・開発委託を活用して加速！

Step 5 4～6ヵ月程度

必要に応じて → Amplify AE のオンプレ版導入

Step 6

取り組むテーマの拡大

セミナー・トレーニングのご紹介

https://amplify.fixstars.com/ja/news/seminar

お客様の実際の課題解決をご支援するために、**無料セミナー**や**有償トレーニング**を提供しています。

無料セミナー・ワークショップ

ビジネス向け、エンジニア向けに分けて
開催しています！

ビジネス向け

製造業向け量子コンピュータ時代のDXセミナー

見える化、予測・分析、その先の最適化へ

組合せ最適化問題や量子アニーリング・イジングマシンの概要をご紹介したのち、製造業における組合せ最適化を活用したDX推進の一例として、生産計画最適化や生産ラインのシフト最適化などの事例とデモをご紹介いたします。「Fixstars Amplify」を通じて量子アニーリング・イジングマシンを活用することで、どのようなビジネス上の効果が期待できるのかを感じていただきたいと思います。

エンジニア向け

製造業向け量子コンピュータ時代のDXセミナー

最適化の中身を覗いてみよう

製造業における組合せ最適化を活用したDX推進の一例として、生産計画最適化、勤務シフト最適化などの事例を用いて、問題設定の考え方、目的関数や制約条件の定式化、実装のポイントなどを実際のコードを見ながら解説します。また、サンプルコードを用いて、ご自身の環境で実際に量子アニーリング・イジングマシンを動かす体験をしていただきます。

企業向けプライベートトレーニング

お客様が抱える実際の課題やデータを使った
カスタムメイド のトレーニングです！

全4回のレクチャーとお客様に実施いただく「課題」を含む約1.5か月のコースです。コースの前半では、量子アニーリング・イジングマシン専用の開発／実行環境であるFixstars Amplifyを用いてPython言語による組合せ最適化アプリケーション開発方法を学びます。後半では、お客様が抱える実際の課題やデータを使ったトレーニングを実施します。量子アニーリング・イジングマシンを使って実課題の解決に取り組んでみたい方に最適なコースです。

クラウド利用料

	個人単位のプラン ～ 主に研究者・開発者向け ～				組織単位のビジネスプラン ～ 社内システムの利用向け ～		
(金額は税抜)	ベーシック	スタンダード	プレミアム	Sプレミアム	ビジネス スタンダード	ビジネス プレミアム	ビジネス Sプレミアム
月額利用料	**無料**	**10万円** (1名) **30万円** (最大5名)	**20万円** (1名) **60万円** (最大5名)	**30万円** (1名) **90万円** (最大5名)	**20万円** (1アプリ) (同一組織内であれば同一アプリのユーザー数は無制限)	**40万円** (1アプリ)	**60万円** (1アプリ)
計算環境	**スモール**	**ミディアム**	**ラージ**	**スーパーラージ**	**ミディアム**	**ラージ**	**スーパーラージ**
利用GPU (マルチGPUオプションあり)	NVIDIA V100	NVIDIA V100	NVIDIA A100	NVIDIA H100	NVIDIA V100	NVIDIA A100	NVIDIA H100
1ジョブの実行時間 (実行時間延長オプションあり)	10秒	1分	10分	15分	1分	10分	15分
月間実行回数上限 (実行回数追加オプションあり)	制限の可能性あり	無制限			- (制限をかける可能性あり)		
東芝 SQBM+オプション	無料	30万円 (1名)、90万円 (最大5名)			-		
NEC VAオプション	無料	30万円 (1名)、90万円 (最大5名)			-		
富士通 DAオプション	無料	50万円 (1名)、150万円 (最大5名)			-		
D-Wave の利用	無料プログラム (3分/月)				-		
サポート	**ベーシック**	**スタンダード**	**プレミアム**	**プレミアム**	**スタンダード**	**スタンダード**	**スタンダード**
Plus オプション	-	-	月額50万/人		-		

次ページ

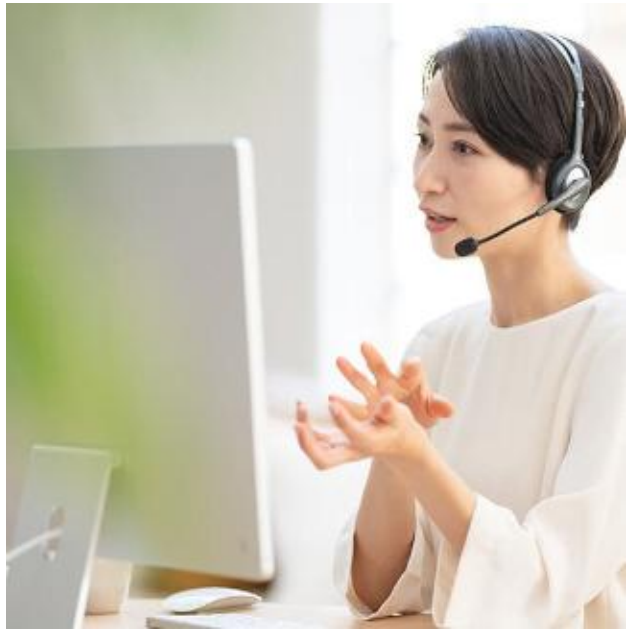
Plusオプション（プレミアムプラン/Sプレミアムプランで追加可能なオプション）

Plusオプション

料金：月額50万円（税込55万円）/ユーザー

- 問い合わせ回数は無制限
- ご質問には翌営業日までに回答（目安）
- 定式化・実装等のご相談
- 特別対応窓口や定例会の設置
- 特別技術支援※

※特別技術支援の内容に応じて期間等は個別にご相談



特別技術支援の例

開発支援

- ユーザー様の実装にお困りの部分に関して、弊社エンジニアがサンプルコードを作って提供します
- 開発支援にかかる期間については個別相談となります

コード最適化レビュー

- 弊社エンジニアがユーザー様が実装したコードを確認し、よりよい実装などがあればサンプルコードを作って提供します

評価支援

- ユーザー様にご提供いただく問題設定で、弊社のエンジニアが様々な計算環境で実験・評価して結果をレポートします
- 複雑な問題になると限られた計算環境では十分な精度の解が得られない可能性があります。本支援では、異なる GPU (V100/A100/H100) や、GPU 数 (1機～4機)、実行時間 (～1時間) で実験・評価し、最適な計算環境の評価・検討のご支援をします
- 問題設定については、ユーザー様にプログラムやデータを送付してもらう、もしくは、問題の概要をテンプレートで回答いただく形になります
- 評価支援にかかる期間については個別相談となります

今後のセミナー予定・情報発信

定期的に無料セミナーを開催しています！

2026/3/11（受付中）「共創パートナー向け最適化セミナー」

最適化とは？実際に収益化するまでの流れは？実稼働例のご紹介？パートナーとしてのビジネス展開にご興味のある方向け

2026/3/18（受付中）「ブラックボックス最適化（機械学習の特徴量抽出）」

ブラックボックス最適化による機械学習の特徴量抽出をハンズオンで実施。

2026/4/9 （予定）「Amplify AE技術解説」

メジャーアップデートされたAmplify独自開発イジングマシンであるAnnealing Engineについて解説。

2026/4/23 （予定）「シフト最適化ハンズオン」

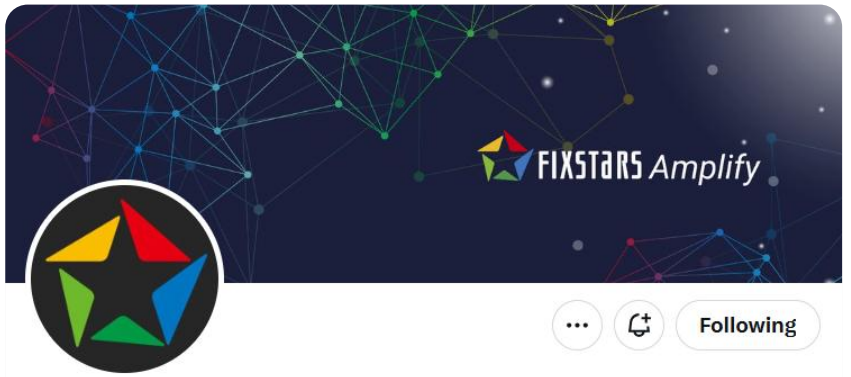
メジャーアップデートされたAmplify独自開発イジングマシンであるAnnealing Engineについて解説。

2026/5/14 （予定）「ブラックボックス最適化技術解説」

量子・量子インスパイアード技術による、ブラックボックス最適化成功のヒントを開設

2026/5/21 （予定）「設計パラメータのブラックボックス最適化」

量子・量子インスパイアード技術による、設計パラメータ最適化をハンズオンで体験！



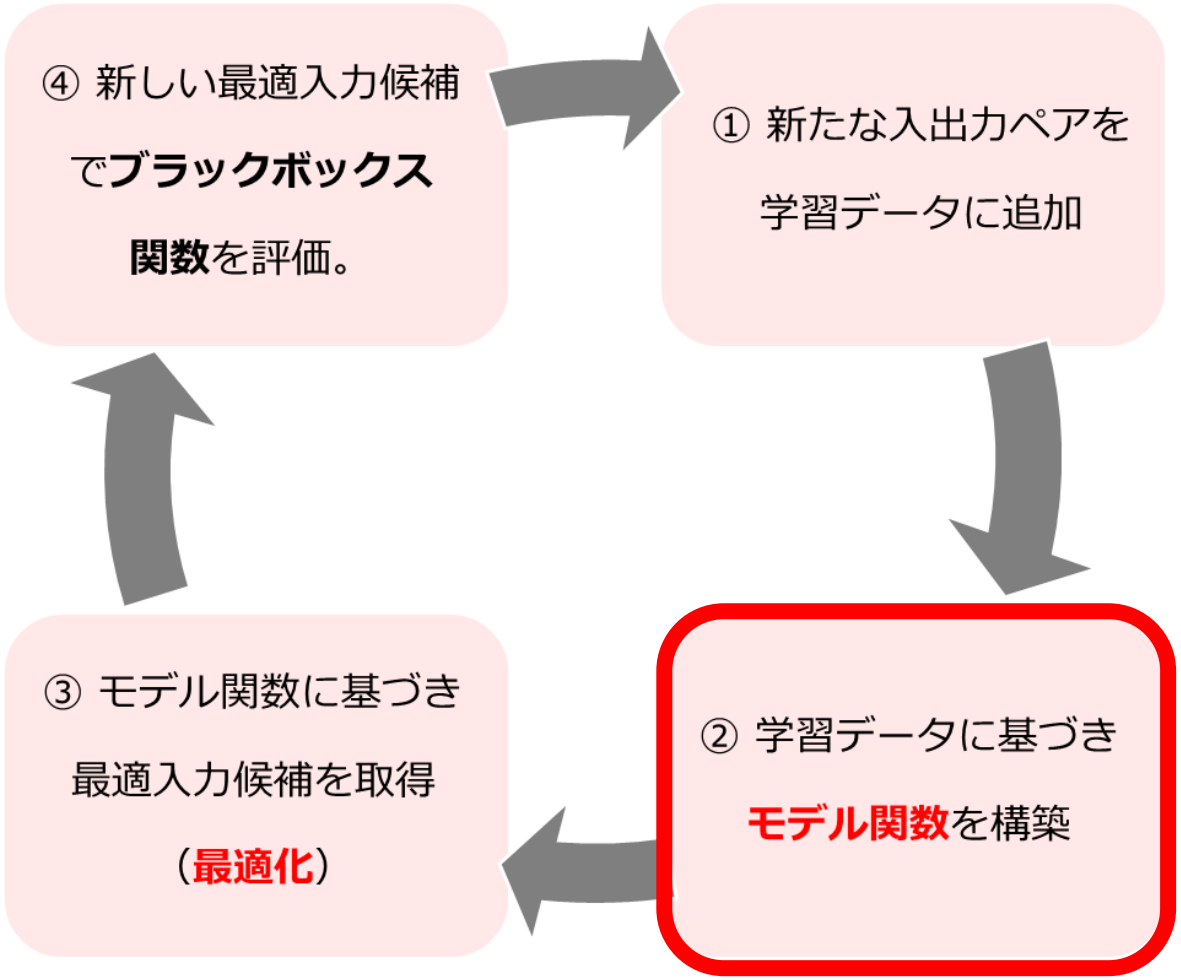
@FixstarsAmplify

ご質問・ご不明点がありましたら、お問い合わせフォームでご連絡下さい
https://amplify.fixstars.com/ja/contact

補足資料

Kernel-QA: もう一つの QA-BBO 手法

- QA-BBO
  - FMQA (Kitai, et al., Phys. Rev. Res., 2020)
    - **FM** モデル + QA
  - **Kernel-QA** (Minamoto & Sakamoto, arXiv:2501.04225, 2025)
    - 2次多項式に基づくカーネルモデル + QA
    - FMモデル構築より速い
    - FMモデル構築よりシンプル
      （ハイパラ無し）
  - ➔ FM/Kernelモデル関数の比較は次スライド



モデル比較 – どちらも QUBO 対応

- **FM** (for FMQA)

  “勾配降下法ベース / 局所最適のリスク”

  $$\hat{f}(\boldsymbol{x}) = w_0 + \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle \, x_i \, x_j$$

  - 潜在ベクトルの次元 k
  - 学習に関する多くのハイパーパラメータ
    （問題によっては、多くのハイパーパラメータがあることが良い場合もあるが、一般的に最適な学習は困難に）
    - エポック数、バッチサイズ、学習率（スケジューラ）、…
  - モデル構築（学習）コスト：$\boldsymbol{O(n \cdot kd)}$
    （機械学習の水準では低コストであるものの、学習は毎サイクル発生するため、低コストほうがよい）

- **Kernel model** (for Kernel-QA)

  “解析的手法 / 数学的に全体最適”

  $$\hat{f}(x) = \sum_j c_j k\left(\boldsymbol{x}_j^*, \boldsymbol{x}\right), \;\; \boldsymbol{c} = (\boldsymbol{K}^* + \lambda \boldsymbol{I})^{-1} \boldsymbol{y}^*$$

  (“ * ” は学習データ)

  - 2次多項式カーネル
    - $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1^T \boldsymbol{x}_2 + \gamma)^2$
  - **Representer 定理**により、数学的に全体最適
  - ⚡ 逆行列とグラム行列の算出に**効率的な計算手法** (Woodbury formula*)を適用
    **Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn., p. 51. Johns Hopkins University Press, Baltimore, MD (1996)*
  - ⚡ モデル構築コスト：$\boldsymbol{O(n)}$

モデル関数の性能向上

- BBOにおけるモデル性能 (FMQA & Kernel-QA)
  - 何を見るべきか？
    - モデル予測値と真値の間の相関係数（特に出力値が比較的小さいサンプルに対しての相関）
    - BBOにおいて、モデルの**RMSE誤差は重要ではない**！
  - 学習データ変換
    - 学習データ内目的関数値の exp 変換により、モデル学習で考慮する必要のあるデータのダイナミックレンジを削減することが可能
      - Y. Minamoto & Y. Sakamoto, arXiv:2501.04225 (2.4節)
      - Amplify-BBOpt ドキュメントページ (https://amplify.fixstars.com/ja/docs/amplify-bbopt/v1/surrogate_performance.html)
    - データ選択
      - 現在進行中（個別にお問い合わせください）