2017/6/28 PyData.Tokyo

# 数理モデリングからはじめる Python 数理最適化

Iwanaga Jiro
Retty Inc.
Mail : iwanaga@retty.me
Tiwtter : @pseudo_finite

# まえおき


本発表は、2017年4月23日の

PyCon mini Kumamoto

にて発表した内容を加筆・修正したものになります

# コンテンツ

1. 自己紹介
2. 数理モデルとは
3. PuLPを使って数理モデリング入門
4. ケーススタディ

# 1. 自己紹介

# 自己紹介

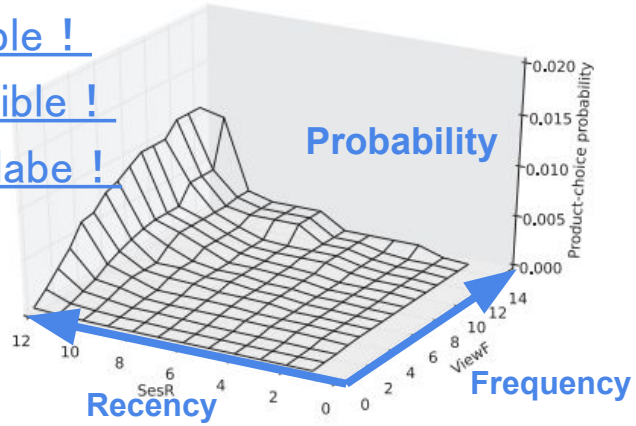
- 岩永二郎
- 経歴
  2008年4月～　数理システム
  2012年9月～　NTTデータ数理システム
  **2016年9月～　Retty**

#データサイエンティスト　#コンサルタント　#プログラマー
#数理最適化　#機械学習　#自然言語処理　#統計学
#数理論理学　#マーケティングサイエンス　#教育心理学
#Python　#R　#SQL　#C/C++
#Tableau

# 活動

- 2013年 2012年度データ解析コンペティション課題設定部門最優秀賞
  経営科学系研究部会連合協議会
- 2014年 日経情報ストラテジー主催データサイエンティストジャパン登壇
- 2014年 データサイエンスフェスティバルレコメンドコンテスト優勝
  START TODAY CO.,LTD.;株式会社スタートトゥディ
- 2015年 情報オリンピック夏季セミナー講師
- 2015年～2017年 経営科学系連合協議会データ解析コンペティション 審査員
- 2017年 PyCon mini Kumamoto 登壇
- Iwanaga, J., Nishimura, N., Sukegawa, N., & Takano, Y. (2016). Estimating product-choice probabilities from recency and frequency of page views. *Knowledge-Based Systems*, *99*, 157-167.

Stable ！
Flexible ！
Scalabe ！




$$\text{maximize} \quad \sum_{(i,j)\in R\times F} \left(q_{ij}\log x_{ij} + \left(n_{ij}-q_{ij}\right)\log\left(1-x_{ij}\right)\right)$$

$$\begin{aligned}
\text{subject to} \quad & x_{ij} \le x_{i+1,j} \quad ((i,j)\in R\times F,\ i \le |R|-1),\\
& x_{ij} \le x_{i,j+1} \quad ((i,j)\in R\times F,\ j \le |F|-1),\\
& x_{i+1,j}-x_{ij} \le x_{i+2,j}-x_{i+1,j} \quad ((i,j)\in R\times F, i\le |R|-2),\\
& x_{i,j+1}-x_{ij} \ge x_{i,j+2}-x_{i,j+1} \quad ((i,j)\in R\times F,\ j \le |F|-2),\\
& 0 < x_{ij} < 1 \quad ((i,j)\in R\times F).
\end{aligned}$$

# 2. 数理モデルとは

# モデルと言えば？

# ファッションモデル


出典元：集英社「Seventeen」

**ファッションモデル**とは、ファッションブランドの衣服や装飾品を身に付け、**ブランドのイメージ**として広告やファッション雑誌の被写体、あるいはファッションショーなどに出演することを職業としているモデルのことを言う。
【出典：Wikipedia】

# プラモデル

出典元URL：http://www.modelers-g.jp/modules/myalbum/photo.php?lid=22711

**プラモデル**とは、組み立て式の**模型**の一種。適度に分割して成形されたプラスチック製の部品群と、組み立て説明書などをセットにしたキットの形で販売される。
【出典：Wikipedia】

# モデルの2つの意味

## 手本・模範


## 型・模型

出典元：集英社「Seventeen」

出典元URL：http://www.modelers-g.jp/modules/myalbum/photo.php?lid=22711

# モデルの2つの意味

## 手本・模範


出典元：集英社「Seventeen」

## 型・模型

## 数理モデル

出典元URL：http://www.modelers-g.jp/modules/myalbum/photo.php?lid=22711

# 数理モデル

数理モデルとは、通常は、時間変化する現象の計測可能な主要な指標の動きを模倣する、微分方程式などの「数学の言葉で記述した系」のことを言う。モデルは「模型」と訳され「数理模型」と呼ばれることもある。**元の現象を表現される複雑な現実とすれば、モデル（模型）はそれの特別な一面を簡略化した形で表現した「言語」（いまの場合は数学）**で、より人間に理解しやすいものとして構築される。構築されたモデルが、元の現象を適切に記述しているか否かは、数学の外の問題で、原理的には論理的には真偽は判定不可能である。人間の直観によって判定するしかない。どこまで精緻にモデル化を行ったとしても、得た観察を近似する論理的な説明に過ぎない。
【出典：Wikipedia】

**数理モデルは現象の模型**

# 数理モデリングをするエンジニアへ

**現象**を、**仮定**と**近似**と**捨象**をした結果が

**数理モデル**であることを忘れてはならない

| | |
|---|---|
| **仮定** | **取り扱う問題**は、**数理モデルで説明できる**<br>数理最適化 / 機械学習 / 統計モデル / 微分方程式 / etc. |
| **近似** | ある**要件**を**特定の数式**で**表現**する<br>1次式 or 2次式 / 目的関数 or 制約 / etc. |
| **捨象** | ある**要件**を**無視**する<br>優先順位が低い / 賛否両論 / 数式で表現が難しい / etc. |

# ３．PuLPを使って数理モデリング入門

# 数理最適化とは

**数理最適化**とは、ある条件に関して最もよい元を、利用可能な集合から選択することをいう。　【出典：Wikipedia】

**集合表現**　$min.\ f(x)\quad s.t.\quad x \in X$

**条件表現**　$min.\ f(x)\quad s.t.\quad \phi(x)$

**行列表現**　$min.\ f(x)\quad s.t.\quad Ax \leq b$

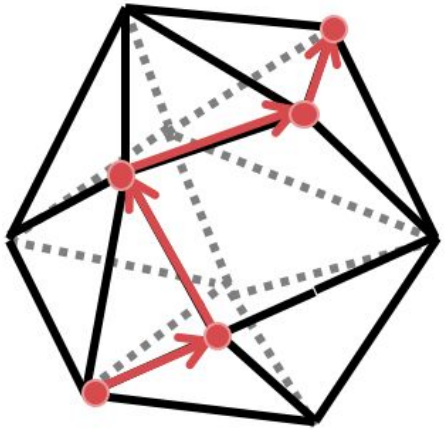
**例）線形計画法**

【幾何的解釈】多面体の内部・周が解

【定理】　最適解は多面体の頂点に存在

【単体法[1947 Danzig]】

適当な頂点から探索を開始、目的関数が改善される頂点に移動を繰り返すといずれ最適解にたどり着く



# 数理最適化問題を解くための仕組み

## 基本的な仕組み

**MPS/LPファイル**
線形計画問題を
汎用的に記述する
形式のファイル

```
NAME          SAMPLE
ROWS
 E  R1
 L  R2
 G  R3
 N  C
COLUMNS
    X1        R1                1.   R2                1.
    X1        C                 4.
    X2        R2                2.   R3                1.
    X3        R2               -1.   R3                1.
    X3        C                -1.
    X4
RHS
    B                                                    10
    B                           2.
BOUNDS
 LO BND1      X1                3.
 LO BND1      X2                1.
 UP BND1      X2                4.
HESSIAN
    X2        X2                2.
    R2
    X1        X2                1.
INITIAL
    X1                          4.
    X2                          2.
ENDATA
```

MPSファイル → ソルバー

MPSファイル作成はツライので
モデリング言語の出番

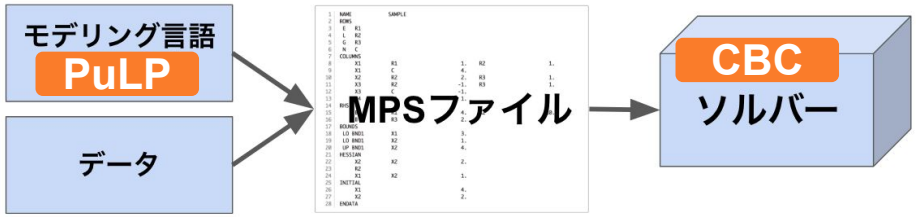
モデリング言語による実装
データ
→ MPSファイル → ソルバー

MPSファイル : ttp://www.msi.co.jp/nuopt/docs/v18/manual/html/16-04-00.html

# Pythonライブラリ PuLP とは

PuLP is an **LP/IP modeler** written in Python. It can generate MPS or LP files and call GLPK, CLP/CBC, CPLEX, and Gurobi to solve linear problems.

- PuLPはCOIN-OR（※1）で開発しているPythonライブラリ
- **LP / IP**（※2）を記述する**モデリング言語**で**MPSファイル**を作成する
- PuLPのデフォルトの**ソルバー**は**CBC**（※3）




MPSファイル：ttp://www.msi.co.jp/nuopt/docs/v18/manual/html/16-04-00.html

- GitHub URL：https://github.com/coin-or/pulp
- インストール

```
$pip install pulp
```

※1 COIN-OR：Open Source for the Operations Research Community
※2 LP/IP：Linear Programming/Integer Programming
※3 CBC：Coin-or branch and cut

# 数理最適化問題を構成する部品

- 数理最適化問題は制約プログラミングで記述する

**制約プログラミング**はプログラミングパラダイムの一つである。制約プログラミングにおいては、**変数間の関係を制約という形で記述**することによりプログラムを記述する。　【出典：Wikipedia】

- 数理最適化問題を構成する**4つの部品**

| 関数 | 部品 | 説明 |
|---|---|---|
| 入力 | パラメータ | モデルを構成する係数・定数 |
| 出力 | 変数 | 決定したい値 |
| 関数定義 | 目的関数 | 最適化したい指標の記述 |
| | 制約式 | 変数間の条件・範囲の記述 |

# 連立1次方程式をPuLPで解く(1/2)

【問題】
1個**120円**のりんごと1個**150円**のなしを合わせて**10個**買ったら、代金の合計が**1440円**だった。りんごとなしは**それぞれ何個**買ったのか。

【解答】
りんごの個数を x、なしの個数を y と置いて定式化すると

$$120x + 150y = 1440$$
$$x + y = 10$$

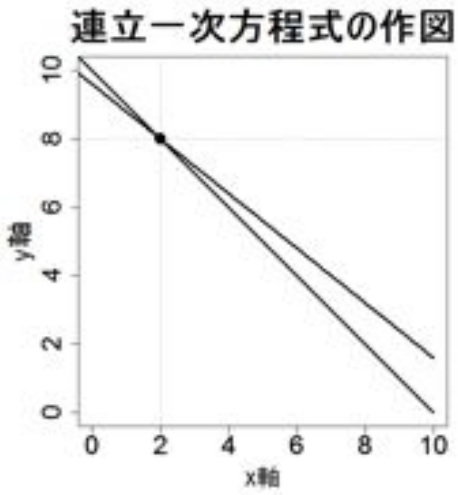
数理モデリング

これを解くと

$x = 2$

$y = 8$

以上より、りんご**2個**、なし**8個**を買った。



# 連立１次方程式をPuLPで解く(2/2)

```
from pulp import *
# 連立１次方程式は目的関数が無いが、最小化問題として仮定義
prob = LpProblem('LE', LpMinimize)

# 変数の定義
x = LpVariable('x', cat='Continious')
y = LpVariable('y', cat='Continious')

# 制約式の定義
prob += 120*x + 150*y == 1440
prob += x + y == 10

# 求解
status = prob.solve()

# 結果の閲覧
print('status:', LpStatus[status])
print('x:%.1f  y:%.1f' % (value(x),  value(y)))
```

```
status: Optimal
x:2.0  y:8.0
```

# 線形計画問題をPuLPで解く(1/2)

【問題】
利益が最大になる製品 A、B の生産量を求めたい。
ただし、以下の条件がある。
・製品Aの生産には原料Pを**1単位**、原料Qを**1単位**使う
・製品Bの生産には原料Pを**1単位**、原料Qを**3単位**使う
・原料Pは**4単位**、
　原料Qは**6単位**ある
・製品Aは1個**2ドル**、
　製品Bは1個**3ドル**の利益がある

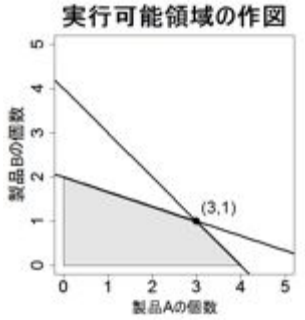
| | 製品A | 製品B | 原料制約 |
|---|---|---|---|
| 原料P | 1 | 1 | 4 |
| 原料Q | 1 | 3 | 6 |
| 製品利益 | 2 | 3 | |

【解答】
製品A、製品Bの生産個数をx、yと置いて定式化すると

$$
\begin{aligned}
&\text{最大化} \quad 2x + 3y \\
&\text{条件} \quad x + y \leqq 4 \\
&\qquad x + 3y \leqq 6 \\
&\qquad x \geqq 0,\ y \geqq 0
\end{aligned}
$$

数理モデリング

これを解くと、x=3(個)、y=1(個) であり、利益は 9(ドル)



# 線形計画問題をPuLPで解く(2/2)

```
from pulp import *
# LPの定義
prob = LpProblem('LP', LpMaximize)

# 変数の定義
x = LpVariable('x', lowBound=0, cat='Continious')
y = LpVariable('y', lowBound=0, cat='Continious')

# 目的関数の定義
prob += 2*x + 3*y

# 制約式の定義
prob += x + y <= 4
prob += x + 3*y <= 6

# 求解
status = prob.solve()
print('status:', LpStatus[status])
print('x:%.1f  y:%.1f' % (value(x),  value(y)))
```

```
status: Optimal
x:3.0  y:1.0
```

# 4. ケーススタディ

# 最適化プロジェクトの1サイクル

| プロセス | 内容 |
| --- | --- |
| サービス理解 | 問題設定をするための前提を理解する |
| 問題設定 | 前提を考慮した上で問題を設定する |
| 数理モデリング | 問題を数理最適化の言葉に置き換える |
| モデリング言語で実装 | 数理モデルをモデリング言語で実装 |
| 数値実験 & 解の検証 | 数値実験をして得られた解を検証・評価 |

# Retty のサービスの理解

## ビジョン

食を通じて世界中の人々をHappyに

## サービス


## 特徴

Rettyの3つの特徴～実名でおすすめのお店を紹介できる～

**信頼の実名制**


**not 評価 but 推薦**

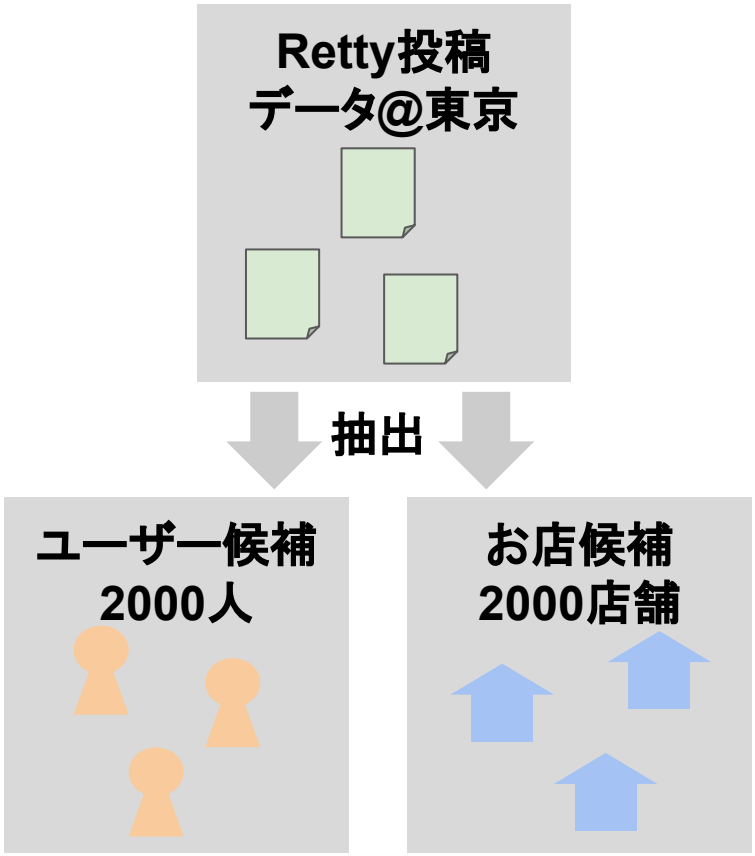

**人の繋がり**


**人からオススメのお店を探す体験を創る**

# 問題設定

**グルメな人をフォローして東京のお店を網羅せよ！**

**STEP1**
ユーザー候補・お店候補の抽出




**STEP2**
ユーザーの選抜とお店の網羅

# 数理モデリング–最大被覆問題–

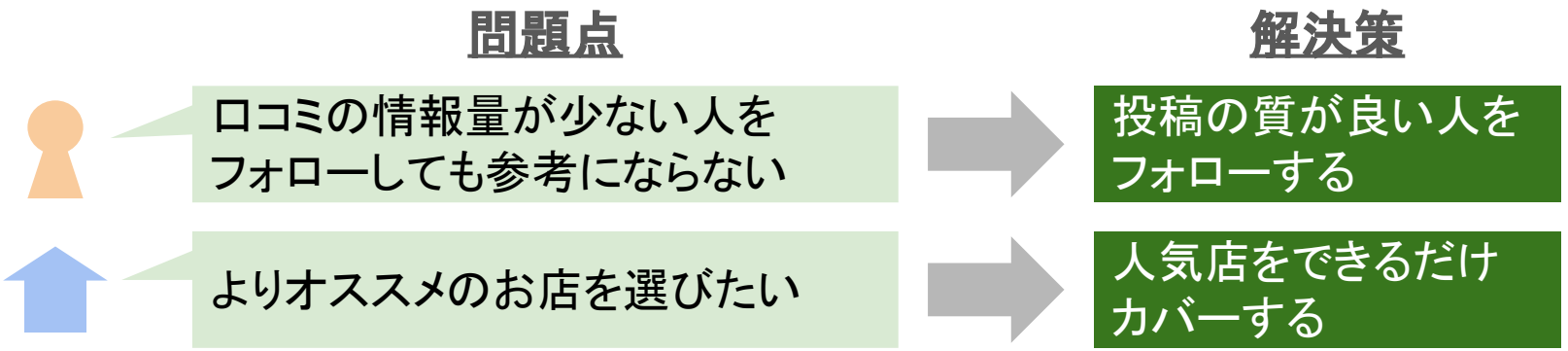
- 集合
  $I$: ユーザーの集合
  $J$: お店の集合
  $I_j$: お店$j$ $(\in J)$に投稿したことがあるユーザーの集合
- パラメータ
  $N$: 選択するユーザー数
- 変数
  $x_i \in \{0,1\}$ $(i \in I)$: ユーザー$i$ $(\in I)$を選択する場合に*1*を取る*Binary*変数
  $y_j \in \{0,1\}$ $(j \in J)$: お店$j$$(\in J)$がカバーされた場合に*1*を取る*Binary*変数
- 制約式
  $\sum_{i \in I} x_i = N$: ユーザーを$N$人選択する
  $y_j \leq \sum_{i \in I_j} x_i$ $(j \in J)$: ユーザー$i$$(\in I)$が選択された場合に
  そのユーザーが投稿したお店にフラグが立つことを許す制約
- 目的関数
  $maximize$ $\sum_{j \in J} y_j$ : カバーするお店数を最大化する

# PuLP で実装 −大事なところを抜粋−

```
import pulp
class restaurant_covering:
    # データの定義

    def preprocess(self):pass

    def build_model(self):
        # LPの定義
        self.prob = pulp.LpProblem('RC', pulp.LpMaximize)
        # 変数の定義
        self.x = pulp.LpVariable.dicts('x', self.Is, cat='Binary')
        self.y = pulp.LpVariable.dicts('y', self.Js, cat='Binary')
        # 目的関数の定義
        self.prob.objective = sum([self.y[j] for j in self.Js])
        # 制約式の定義
        self.prob += sum([self.x[i] for i in self.Is]) == self.N
        for j in self.Js:
            self.prob += self.y[j] <= pulp.lpSum(
                                [self.x[i] for i in self.J2Is[j]])

    def optimize_model(self)
        self.status = pulp.LpStatus[self.prob.solve()]

    def postprocess(self):pass
```

# 数値実験 & 解の検証

- 数値実験
  - ✓ N = 5（フォローすべき5人を探索）で実験
  - ✓ 解の組合せ：2000ユーザから5ユーザを選択する組合せ
    ⇒ 約265兆通り
  - ✓ 実験環境：MacBook Pro (Retina, 13-inch, Early 2015)
    CPU：2.9 GHz Intel Core i5
    メモリ：16 GB 1867 MHz DDR3
  - ✓ 計算時間：モデリング：6.81[s] / 求解：192.37[s]
- 解の検証
  - ✓ 5人で1309/2000店舗をカバー



# 数理モデルへのフィードバック

- 集合
  $I$: ユーザーの集合
  $J$: お店の集合
  $I_j$: お店$j$ $(\in J)$に投稿したことがあるユーザーの集合
- パラメータ
  $N$: 選択するユーザー数
  $uscore_i \quad (i \in I)$: ユーザー$i(\in I)$スコア
  $rscore_j \quad (j \in J)$: お店$j(\in J)$スコア
  $\alpha$: ユーザーとお店の重要性を調整するパラメータ

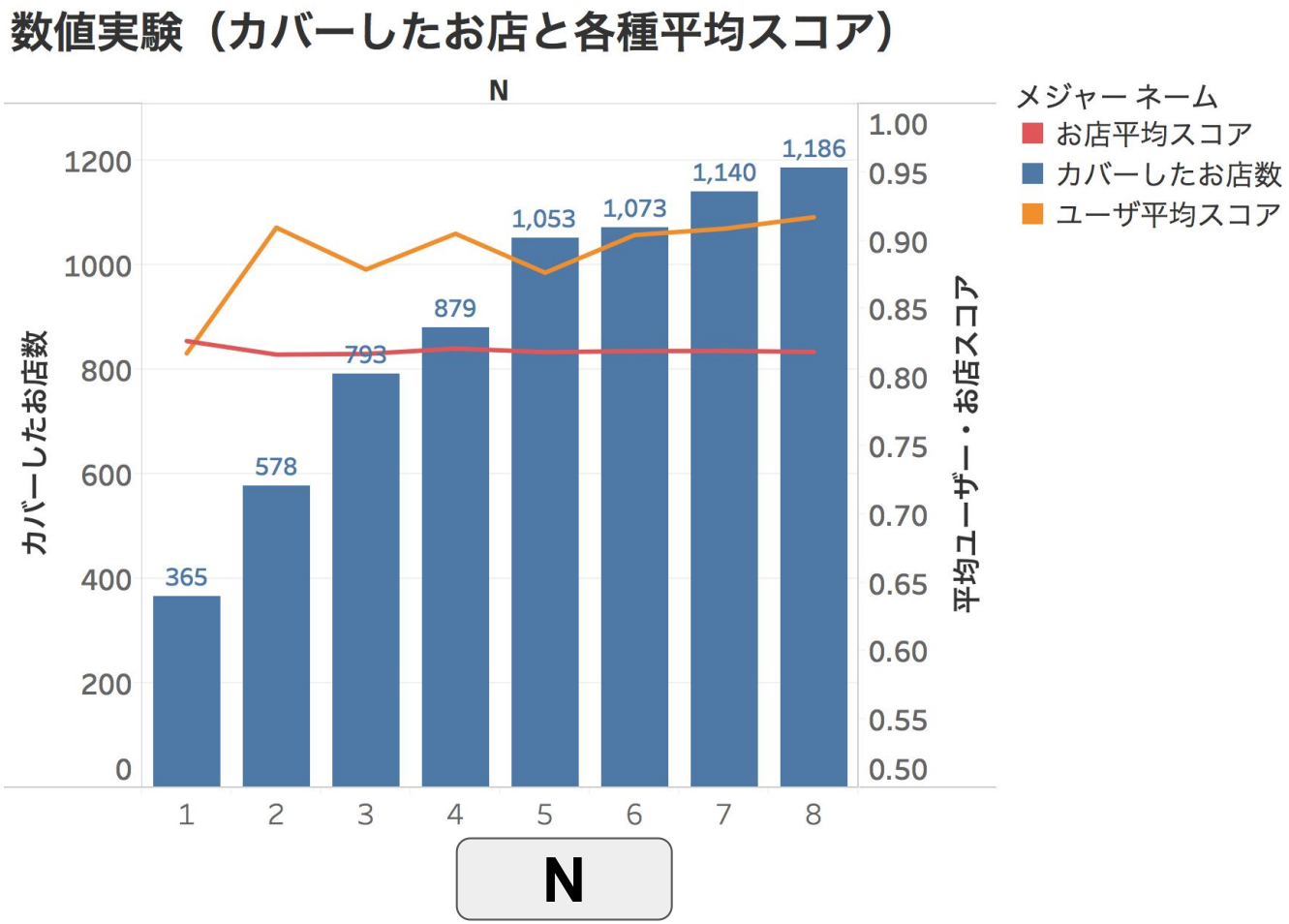
  **追加パラメータ**
- 変数
  $x_i \in \{0,1\} \quad (i \in I)$: ユーザー$i$ $(\in I)$を選択する場合に*1*を取る*Binary*変数
  $y_j \in \{0,1\} \quad (j \in J)$: お店$j(\in J)$がカバーされた場合に*1*を取る*Binary*変数
- 制約式
  $\sum_{i \in I} x_i = N$: ユーザーを$N$人選択する
  $y_j \leq \sum_{i \in I_j} x_i \quad (j \in J)$: ユーザー$i(\in I)$が選択された場合に
  そのユーザーが投稿したお店にフラグが立つことを許す制約
- 目的関数

$$maximize \quad \alpha \cdot \sum_{j \in J} y_j \cdot rscore_j + (1-\alpha) \cdot \sum_{i \in I} x_i \cdot uscore_i$$

: カバーするお店スコア総和と選抜するユーザースコア総和を重み付きで最大化

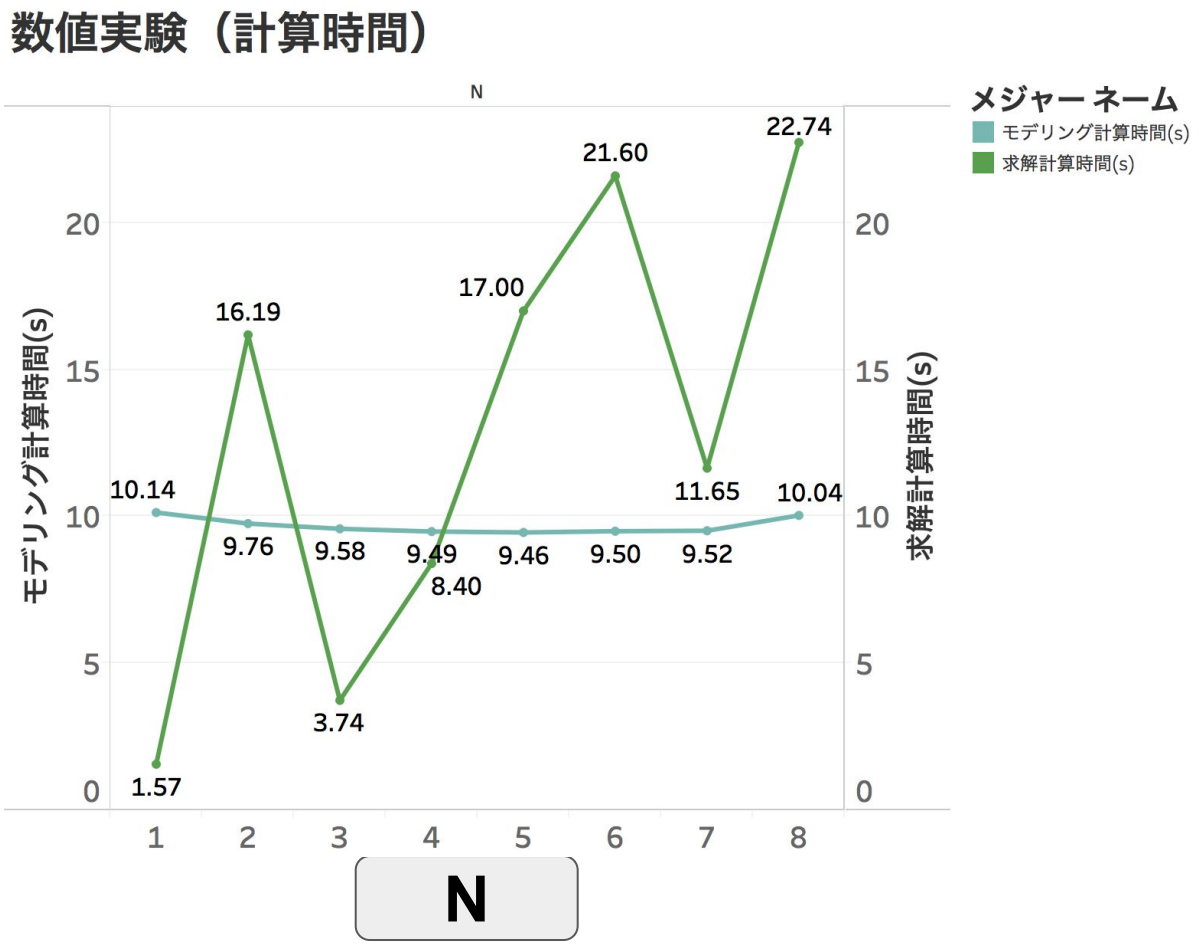
**投稿の質と人気店を考慮**

# 数値実験(N=1～8) & 評価(1/3)




**カバーするお店が増えても**
**ユーザ平均スコア･お店平均スコアが安定**

# 数値実験(N=1～8) & 評価(2/3)




**モデリング時間は安定**
**求解計算時間は徐々に大きくなる傾向があるが不安定**

# 今後の課題

サービスへの落とし込み

- 要件の観点
  - ✓ 偏った意見を回避するために2人以上のユーザーでお店をカバーしたい
  - ✓ エリア・料理カテゴリを指定してお店をカバーしたい
  - ✓ パーソナライズしてフォローするユーザを選びたい
- デザイナーとの連携、UX・UIへの落とし込み
- 大規模0-1整数計画問題を解きたい

# 参考資料

# 【参考】MPSファイル

```
NAME            SAMPLE
ROWS
 E   R1
 L   R2
 G   R3
 N   C
COLUMNS
      X1        R1                  1.    R2                  1.
      X1        C                   4.
      X2        R2                  2.    R3                  1.
      X3        R2                 -1.    R3                  1.
      X3        C                  -1.
      X4        R1                  1.
RHS
      B         R1                  4.    R2                 10.
      B         R3                  2.
BOUNDS
 LO BND1        X1                  3.
 LO BND1        X2                  1.
 UP BND1        X2                  4.
HESSIAN
      X2        X2                  2.
      R2
      X1        X2                  1.
INITIAL
      X1                            4.
      X2                            2.
ENDATA
```

MPSファイル：ttp://www.msi.co.jp/nuopt/docs/v18/manual/html/16-04-00.html