数理最適化ハンズオン

Powered By Marp

数理最適化とは

制約の中でもっとも良い解を見つけるための手法

最適化問題

公共施設の配置、災害救助、最短路、待ち行列、コンテナ詰め込み、動的価格設定、プロジェクトスケジューリング、シフトスケジューリング、配送、在庫管理、エネルギー供給、生産計画、etc.

「ORを探せ！」ポスター

数理最適化に登場する概念

最適化問題は主に以下の５つの項目によって表現される

- 集合
- 変数
- 制約
- 目的関数
- パラメータ

習うより慣れろ

今日作るもの

シフトスケジューリング最適化問題

ハンズオン準備

Google Colaboratory 実行環境

「ドライブにコピー」ボタンを押すだけ！

Google Colaboratory とは

- Googleが無料で提供する、対話型のPython環境
- Google Drive上で動作・管理する
- データ分析系ライブラリが一通り用意されている
- Jupyter Notebookという種類の形式

Pulp とは

- Pythonの最適化ライブラリ
- 最適化ライブラリは何種類かのソルバーを扱う
- デフォルトで無料のソルバーを使うことができる

Pulpのインストール

```
!pip install pulp
```

Pulpのインポート

```
# Import
from pulp import *
```

集合

モデルを構成する要素・物体（エンティティ）の集合

- 職員
- 日付

```
# Set
Employee = { "emp1", "emp2", "emp3" }
Day      = { 1, 2, 3 }
```

変数

最適化をして求めたい値。最適解が求まると変数が最適値として定まる。

- 保育士eが日付dに出勤するかどうか（0-1変数）

```
# Variable
x = {}
for emp in Employee:
    for day in Day:
        x[emp, day] = LpVariable(name=f"x[{emp}, {day}]", cat="Binary")
```

LpVariable

数理モデルの変数を表すクラス

```
LpVariable(name=[変数名], cat=[変数のカテゴリ])
```

name

変数の名前

cat

変数のカテゴリ

変数のカテゴリ

名前	説明
Continuous	連続変数
Binary	0-1変数
Integer	整数変数

Binary, Integerは計算が重いので、なるべくContinuous

モデルの追加

```
# Create Model
model = LpProblem(sense=LpMinimize)
```

LpProblem

1つのモデルを管理するクラス

```
LpProblem(sense=[LpMinimizeまたはLpMaximize])
```

sense

`LpMinimize` または `LpMaximize` を与えて、最小化を行うか最大化を行うかを指定する。

求解

```
# Solve
status = model.solve()
print(LpStatus[status])
```

- `model.solve()` でstatus codeを返す
- `LpStatus` マップでstatusの名前を取得

求解結果の表示

```
# Display Result
for emp, day in x:
    print(emp, day, x[emp, day].value())
```

制約

求解する上で必ず守る制約条件

- 職員の最低配置人数制約。各日付の配置人数が2人以上になるようにする

```
# Constraint
for day in Day:
    model.addConstraint(
        lpSum(x[emp, day] for emp in Employee)
        >=
        2
    )
```

LpProbrem.addConstraint

```
model.addConstraint([制約式])
```

モデルに制約式の追加を行う

lpSum

```
lpSum([値] for [変数] in [集合])
```

集約関数。標準の `sum()` より高速

目的関数

何を最小化・最大化したいかを表す、最適化の目的となる関数

- 何日以上出勤したらペナルティ
- ペナルティの合計を最小化したい

変数

最適化をして求めたい値。最適解が求まると変数が最適値として定まる。

- 保育士eが超過ペナルティ日数を超過した日数

```
# Variable

...

xp = {}
for emp in Employee:
    xp[emp] = LpVariable(name=f"xp_{emp}", lowBound=0, cat="Continuous")
```

制約

求解する上で必ず守る制約条件

- 変数xpの値を定める制約（コネクト制約）

```
# Constraint

...

for emp in Employee:
    model.addConstraint(
        xp[emp]
        ==
        lpSum(x[emp, day] for day in Day) - 3
    )
```

目的関数

何を最小化・最大化したいかを表す、最適化の目的となる関数

- 何日以上出勤したらペナルティ
- ペナルティの合計を最小化したい

```
model += lpDot([xp[emp] for emp in Employee], [3] * len(Employee))
```

lpDot

```
lpDot([リスト], [リスト])
```

ドット積（内積）を求める。

色々といじってみよう

簡単に中身の話

- 混合整数計画問題（MIP）
- 連続最適化（連続値） → 単体法（シンプレックス法）
- 離散最適化（整数値） → 分枝限定法

今回使用したのは、MIPソルバーという種類のソルバー。他にもたくさんのアプローチ（を実装したソルバー）があるので興味がある人は是非。

MIPソルバーの特徴

- 毎回同じ解が求められる（求解を終えた場合）
- 制約は必ず守られ、守れない場合実行不可能
- 時系列が細かく、期間が長い場合は苦手

モデリング・実装のコツ

- どこまで現実世界を反映させるかを決める（精度と求解時間のトレードオフ）
- ミニマムなデータでテストを繰り返す
- 変数・制約の数をなるべく減らす

このあとの話（残った時間と来週以降）

- 今回作ったモデルの続き（パラメータの定義・モデルの拡張）
- 他の最適化問題をたくさん解いてみる
- Scalaの最適化ライブラリで書き直し、ミニマムアプリに組み込む
- ソルバーの中身の理論を詳しく勉強

リポジトリ

https://github.com/taichi0315/optimization-handson

参考リンク

- PuLPによるモデル作成方法 -Pythonオンライン学習サービス PyQ（パイキュー）ドキュメント
- Pulp公式ドキュメント