線形計画法入門

神山 直之

九州大学 マス・フォア・インダストリ研究所

九州大学
KYUSHU UNIVERSITY

概要

概要

最適化問題とは？

最適化問題 $=$ 「**入力**」$+$「**目的**」

入力：解の候補集合 F と目的関数 $f\colon F \to \mathbb{R}$

目的：$f(x)$ を最小化する $x \in F$ を見つける

- **例１：**

  $F = \{x \in \mathbb{R} \mid -1 \leq x \leq 1\},\ \ f(x) = x^2 - x$

- **例２：**

  $F = \left\{X \subseteq N \mid \sum_{i \in X} c(i) \leq B\right\},\ f(X) = B - \sum_{i \in X} c(i)$

  ただし $N =$ 有限集合, $B \in \mathbb{Z}_+$, $c\colon N \to \mathbb{Z}_+$

最適化問題とは？

- 最適化問題を形式的には以下のように書く

$$\begin{aligned} \text{Minimize} \quad & f(x) \\ \text{subject to} \quad & x \in F \end{aligned}$$

  - “Minimize” は最小化を表しその横に目的関数を書く
  - “subject to” のところに制約を書く

    集合ではなく幾つかの式でかかれることもある

- $f(x)$ の最小値を達成する $x^* \in F$ を**最適解**とよぶ

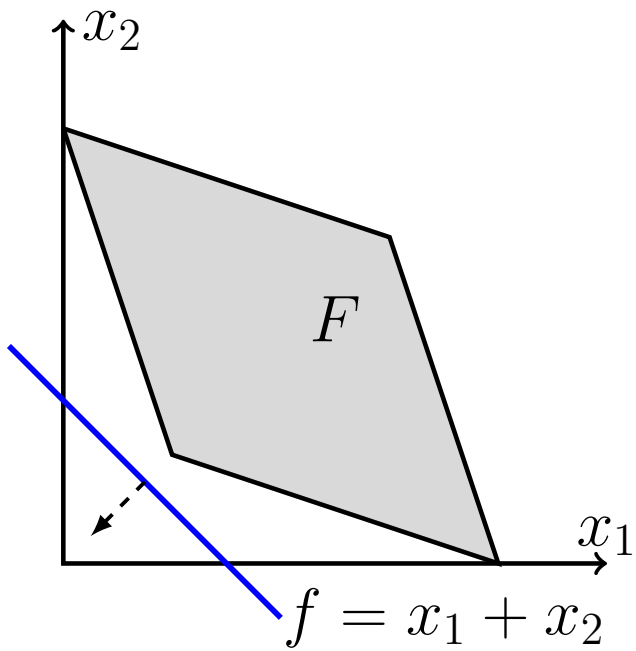
  最適解 x^* に対して $f(x^*)$ を**最適値**とよぶ

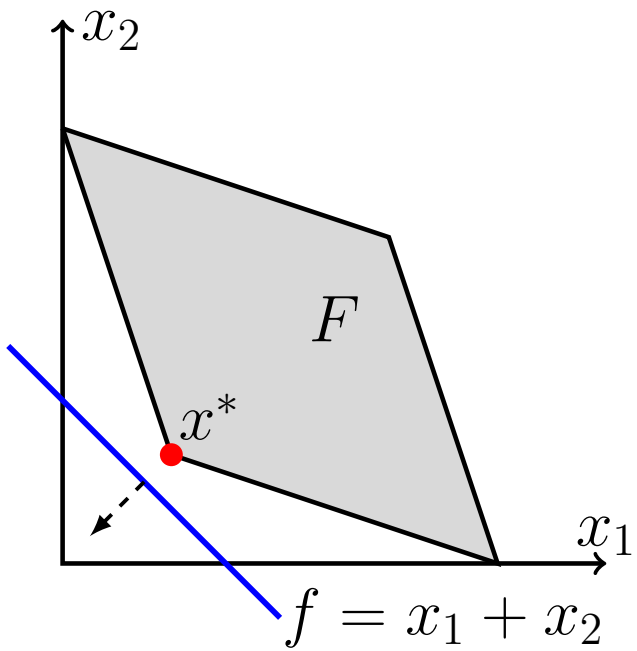
  注意：実は色々細いことがあります

  最適解の存在性などなど

線形計画問題とは？

f や F が「線形」である最適化問題





- **本講義**：線形計画問題の基礎と理論的応用とソルバー

線形計画問題の例

必要な栄養をとれる食事を考える問題

食品 A：価格 $= 2$, 栄養 $1 = 3$, 栄養 $2 = 2$

食品 B：価格 $= 3$, 栄養 $1 = 2$, 栄養 $2 = 6$

必要な栄養量：栄養 $1 = 5$, 栄養 $2 = 8$

$\Longrightarrow$ 最小の価格で必要な栄養を確保するには？

- $x_1 =$ 食品 A の量を表す変数
- $x_2 =$ 食品 B の量を表す変数
- 最小化する価格 $= 2x_1 + 3x_2$
- 制約 $=$ 栄養 1：$3x_1 + 2x_2 \geq 5$,　栄養 2：$2x_1 + 6x_2 \geq 8$

線形計画問題の例

必要な栄養をとれる食事を考える問題

食品 A：価格 $= 2$, 栄養 $1 = 3$, 栄養 $2 = 2$

食品 B：価格 $= 3$, 栄養 $1 = 2$, 栄養 $2 = 6$

必要な栄養量：栄養 $1 = 5$, 栄養 $2 = 8$

$\Longrightarrow$ 最小の価格で必要な栄養を確保するには？

- 以下のような線形計画問題になる

$$
\begin{aligned}
\text{Minimize} \quad & 2x_1 + 3x_2 \\
\text{subject to} \quad & 3x_1 + 2x_2 \geq 5 \\
& 2x_1 + 6x_2 \geq 8 \\
& x_1 \geq 0,\ x_2 \geq 0
\end{aligned}
$$

行列を用いた表現

- 以下の線形計画問題を

$$
\begin{aligned}
\text{Minimize} \quad & 2x_1 + 3x_2 \\
\text{subject to} \quad & 3x_1 + 2x_2 \geq 5 \\
& 2x_1 + 6x_2 \geq 8 \\
& x_1 \geq 0,\ x_2 \geq 0
\end{aligned}
$$

行列を用いて表現すると以下のようになる

$$
\begin{aligned}
\text{Minimize} \quad & \begin{pmatrix} 2 \\ 3 \end{pmatrix}^{\top} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\
\text{subject to} \quad & \begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq \begin{pmatrix} 5 \\ 8 \end{pmatrix} \\
& \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \end{pmatrix}
\end{aligned}
$$

線形計画問題

入力：

- $m \times n$ 行列 $A \in \mathbb{Q}^{m \times n}$
- n 次元ベクトル $c \in \mathbb{Q}^n$
- m 次元ベクトル $b \in \mathbb{Q}^m$

目的：

- 以下の最適化問題の最適解を求める

$$\begin{aligned} \text{Minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax \geq b \\ & x \in \mathbb{R}^n_+ \end{aligned}$$

線形計画問題の基本性質

- $x \in \mathbb{R}^n$ が線形計画問題の実行可能解

  $\overset{\text{def}}{\iff}$ $x \in \mathbb{R}^n_+$ かつ $Ax \geq b$ を満たす

定理

任意の線形計画問題に対して以下の一つが成り立つ

1. 実行可能解が存在しない
2. 実行可能解が存在し目的関数の値に下界がない
3. 実行可能解が存在し目的関数の値に下界がある

   さらにこの下界を達成する実行可能解が存在

- 証明は省略

線形計画問題に対するアルゴリズム

本講演の前提

線形計画法は理論・実用的に効率的に解ける

- 理論的なアルゴリズム
  - 単体法，楕円体法，内点法
- 本講演 $\neq$ これらのアルゴリズムの解説

  本講演 $=$ 問題自体の理論的基礎・理論的応用
- ソフトウェア
  - CPLEX，Gurobi，GLPK
  - **PuLP (Python)**

概要

Python パッケージ PuLP

- 線形計画問題を解くソフトウェア PuLP
  - 正確には Python のパッケージ
- インストールの仕方は例えば

```
sudo pip install pulp
```

- 以下では次の線形計画問題を PuLP で解いてみる

$$
\begin{aligned}
\text{Minimize} \quad & 8x + 19y \\
\text{subject to} \quad & 2x + 7y \geq 40 \\
& 6x + 3y \geq 50 \\
& x \geq 0 \\
& y \geq 0
\end{aligned}
$$

PuLP の使用方法

- PuLP をインポートする

```
import pulp
```

- 問題そのものを作る

```
problem = pulp.LpProblem("P",pulp.LpMinimize)
```

- 変数を定義

```
x = pulp.LpVariable("x")
y = pulp.LpVariable("y")
```

- 目的関数を定義

```
problem += 8*x + 19*y
```

PuLP の使用方法

- 制約を定義

```
problem += 2*x + 7*y >= 40
problem += 6*x + 3*y >= 50
problem += x >= 0
problem += y >= 0
```

- 問題を解いて状況を確認して解を出力

```
status = problem.solve()
print(pulp.LpStatus[status])
print(pulp.value(problem.objective))
print(x.value())
print(y.value())
```

全体のソースファイル

```
import pulp

problem = pulp.LpProblem("P", pulp.LpMinimize)

x = pulp.LpVariable("x")
y = pulp.LpVariable("y")

problem += 8*x + 19*y

problem += 2*x + 7*y >= 40
problem += 6*x + 3*y >= 50
problem += x >= 0
problem += y >= 0

status = problem.solve()
print(pulp.LpStatus[status])
print(pulp.value(problem.objective))
print(x.value())
print(y.value())
```

概要

有向グラフ

- 有向グラフ $D = (V, A)$

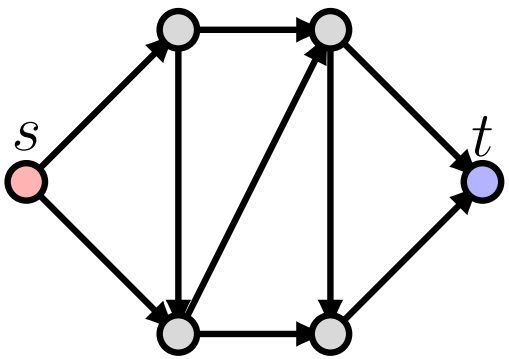
  $V :=$ 点集合と呼ばれる有限集合, $A \subseteq V \times V$

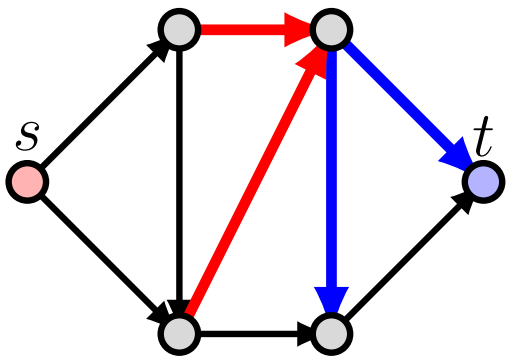
- 特別な点 $s, t \in V$ と関数 $c\colon A \to \mathbb{Q}_+$ に対して

  - 関数 $f\colon A \to \mathbb{R}_+$ が実行可能 s-t フロー $\overset{\text{def}}{\iff}$

    $\forall a \in A\colon f(a) \leq c(a)$

    $\forall v \in V \setminus \{s, t\}\colon \sum_{a \in \varrho(v)} f(a) = \sum_{a \in \delta(v)} f(a)$





最大流問題

$\sum_{a \in \delta(s)} f(a)$ を最大化する s-t 実行可能フローを求める

- 最大流問題は以下の線形計画問題に定式化できる

$$\text{Minimize} \quad -\sum_{a \in \delta(s)} f(a)$$

$$\text{subject to} \quad \sum_{a \in \delta(v)} f(a) = \sum_{a \in \varrho(v)} f(a) \quad (v \in V \setminus \{s, t\})$$

$$f(a) \leq c(a) \quad (a \in A)$$

$$f \in \mathbb{R}_+^A$$

Python による実装例

```
import pulp

problem = pulp.LpProblem("P", pulp.LpMinimize)

x1 = pulp.LpVariable("x1")
x2 = pulp.LpVariable("x2")
x3 = pulp.LpVariable("x3")
x4 = pulp.LpVariable("x4")
x5 = pulp.LpVariable("x5")
x6 = pulp.LpVariable("x6")
x7 = pulp.LpVariable("x7")
x8 = pulp.LpVariable("x8")
x9 = pulp.LpVariable("x9")
```

Python による実装例

```
problem += - x1 - x2

problem += x1 - x3 - x4 == 0
problem += x2 + x3 - x5 - x6 == 0
problem += x4 + x5 - x7 - x8 == 0
problem += x6 + x7 - x9 == 0
problem += x1 <= 1
problem += x2 <= 1
problem += x3 <= 1
problem += x4 <= 1
problem += x5 <= 1
problem += x6 <= 1
problem += x7 <= 1
problem += x8 <= 1
problem += x9 <= 1
```

Python による実装例

```
status = problem.solve()
print(pulp.LpStatus[status])

print(pulp.value(problem.objective))
print(x1.value())
print(x2.value())
print(x3.value())
print(x4.value())
print(x5.value())
print(x6.value())
print(x7.value())
print(x8.value())
print(x9.value())
```

答えとして -2 が出れば正解

概要

NetworkX

- インストールの仕方は例えば

```
sudo pip install networkx
```

- Networkx をインポートする

```
import networkx as nx
```

- 空のグラフを用意する

```
G = nx.DiGraph()
```

- 点を加える

```
G.add_node('s')
G.add_node('1')
G.add_node('2')
G.add_node('3')
G.add_node('4')
G.add_node('t')
```

NetworkX

- 辺を加える

```
G.add_edge('s', '1', capacity=1)
G.add_edge('s', '2', capacity=1)
G.add_edge('1', '2', capacity=1)
G.add_edge('1', '3', capacity=1)
G.add_edge('2', '3', capacity=1)
G.add_edge('2', '4', capacity=1)
G.add_edge('3', '4', capacity=1)
G.add_edge('3', 't', capacity=1)
G.add_edge('4', 't', capacity=1)
```

- 最大流問題を解く

```
flow_value, flows = nx.maximum_flow(G, 's', 't')
```

- 結果を表示する

```
print(flow_value)
```

全体のソースファイル

```
import networkx as nx
G = nx.DiGraph()
G.add_node('s')
G.add_node('1')
G.add_node('2')
G.add_node('3')
G.add_node('4')
G.add_node('t')
G.add_edge('s', '1', capacity=1)
G.add_edge('s', '2', capacity=1)
G.add_edge('1', '2', capacity=1)
G.add_edge('1', '3', capacity=1)
G.add_edge('2', '3', capacity=1)
G.add_edge('2', '4', capacity=1)
G.add_edge('3', '4', capacity=1)
G.add_edge('3', 't', capacity=1)
G.add_edge('4', 't', capacity=1)
flow_value, flows = nx.maximum_flow(G, 's', 't')
print(flow_value)
```

概要

端点解

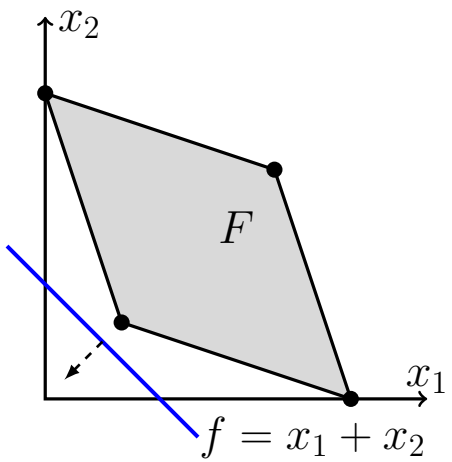
- 線形計画問題の実行可能解 x が端点解 $\overset{\text{def}}{\iff}$

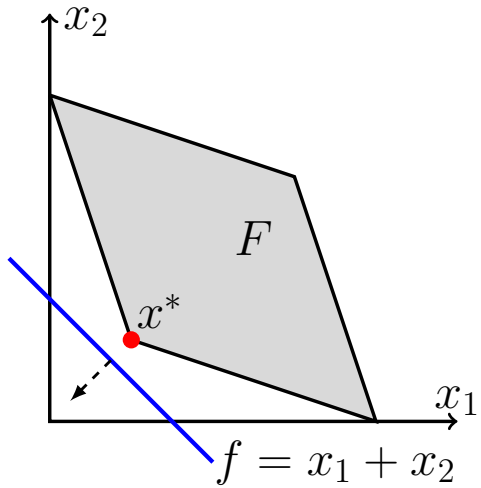
  以下を満たす x と異なる実行可能解 y, z が存在しない

  $$\exists \varepsilon \in \mathbb{R} \text{ s.t } 0 < \varepsilon < 1 : (1-\varepsilon)y + \varepsilon z = x$$

定理（証明略）

最適解が存在すれば端点最適解が常に存在する





概要

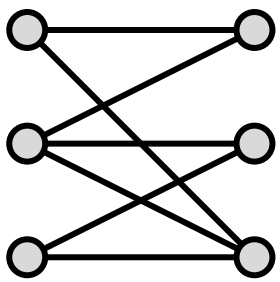
最大マッチング問題への応用

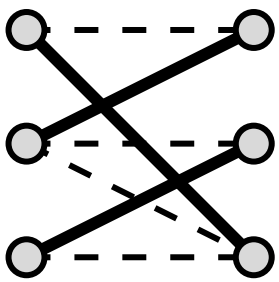
- 二部グラフ $G = (A, B; E)$
  - $A \cap B = \emptyset$ を満たす有限な点集合 A, B
  - 以下を満たす有限な辺集合 E

$$\forall e \in E \colon e \subseteq V,\ |e \cap A| = |e \cap B| = 1$$

- $M \subseteq E$ がマッチング $\overset{\text{def}}{\iff}$

  任意の $v \in A \cup B$ に対して $|\{e \in M \mid v \in e\}| \leq 1$





最大マッチング問題への応用

二部グラフ上の最大マッチング問題

入力：二部グラフ $G = (A, B; E)$

目的：要素数が最大のマッチング

- 線形計画問題っぽく定式化してみる　(整数計画問題)

$$\text{Minimize} \quad -\sum_{e \in E} x(e)$$

$$\text{subject to} \quad \sum_{e \in E:\, v \in e} x(e) \leq 1 \quad (v \in A \cup B)$$

$$x \in \{0, 1\}^E$$

- 注意：$x \in \{0, 1\}$ の制約のある問題は一般には難しい

最大マッチング問題への応用

- $x(e) \in \{0, 1\}$ を $0 \leq x(e) \leq 1$ に緩和

$$\text{Minimize} \quad -\sum_{e \in E} x(e)$$

$$\text{subject to} \quad \sum_{e \in E\colon v \in e} x(e) \leq 1 \quad (v \in A \cup B)$$

$$x(e) \leq 1 \quad (e \in E)$$

$$x \in \mathbb{R}_+^E$$

$\Longrightarrow$ 問題は変わるが線形計画問題になる

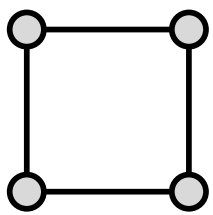
定理

緩和しても最適解の目的関数値は変わらない

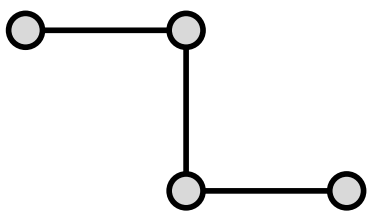
証明

- 任意の端点解の任意の要素が整数であることを示す
- 整数ではない要素を含む端点解 x が存在すると仮定
- $F \stackrel{\text{def}}{=} \{e \in E \mid 0 < x(e) < 1\}$
- 以下の場合に分けて証明
  - F の中に “サイクル” C がある場合

    $\Longrightarrow$ 2 部グラフなので C の長さは偶数
  - F の中に “サイクル” がない場合



サイクル



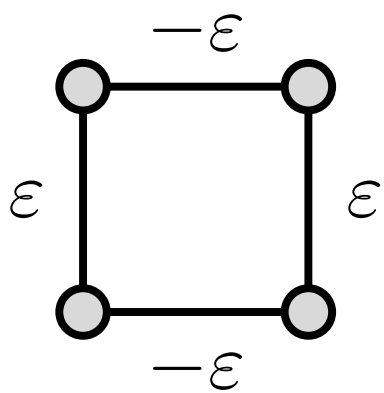
パス

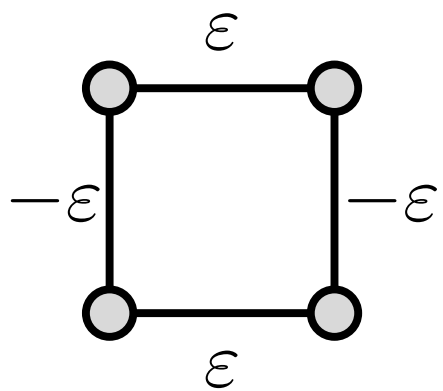
証明：F の中に閉路 C がある場合

- $\alpha \stackrel{\text{def}}{=} \min\{x(e) \mid e \in C\}$
- $\beta \stackrel{\text{def}}{=} \min\{1 - x(e) \mid e \in C\}$
- $\varepsilon \stackrel{\text{def}}{=} \min\{\alpha, \beta\}$
- $x^+, x^- \stackrel{\text{def}}{=} x$ に交互に M を足す・引くしたもの

  $\Longrightarrow$ x^+, x^- は実行可能解かつ $x = \frac{1}{2}(x^+ + x^-)$

  $\Longrightarrow$ x が端点解であることに矛盾





証明：F の中に閉路がない場合

- F の中の任意の極大な “パス” P （u から w へと仮定）
- $v \in \{u, w\}$ にはそれぞれ 1 つの F の辺が入る

  $\Longrightarrow$ この辺を $\delta_F(v)$ と書くと $1 - x(\delta_F(v)) < 1$
- $\alpha \stackrel{\text{def}}{=} \min\{x(e) \mid e \in P\}, \quad \beta \stackrel{\text{def}}{=} \min\{1 - x(e) \mid e \in P\}$
- $\varepsilon \stackrel{\text{def}}{=} \min\{\alpha, \beta\}$
- $x^+, x^- \stackrel{\text{def}}{=} x$ に交互に M を足す・引くしたもの

  $\Longrightarrow$ x^+, x^- は実行可能解かつ $x = \frac{1}{2}(x^+ + x^-)$

  $\Longrightarrow$ x が端点解であることに矛盾

概要

線形計画問題の書き換え

- 行列 A の i 行 j 列を a_{ij} とかく
- ベクトル b を $b \stackrel{\text{def}}{=} (b_1, b_2, \ldots, b_m)^\top$ と定義
- ベクトル c を $c \stackrel{\text{def}}{=} (c_1, c_2, \ldots, c_n)^\top$ と定義
- この記法を使うと線形計画問題は以下のようになる

$$
\begin{aligned}
\text{Minimize} \quad & c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \\
\text{subject to} \quad & a_{1,1} x_1 + a_{1,2} x_2 + \cdots + a_{1,n} x_n \geq b_1 \\
& a_{2,1} x_1 + a_{2,2} x_2 + \cdots + a_{2,n} x_n \geq b_2 \\
& \qquad \vdots \\
& a_{m,1} x_1 + a_{m,2} x_2 + \cdots + a_{m,n} x_n \geq b_m \\
& x_1, x_2, \ldots, x_n \geq 0
\end{aligned}
$$

ラグランジュ緩和

- 先ほどの線形計画問題を緩和してみる
- 制約式 $a_{1,1}x_1 + \cdots + a_{1,n}x_n \geq b_1$ を取り除く
- $b_1 - a_{1,1}x_1 - \cdots - a_{1,n}x_n \leq 0$ なので

  $\Longrightarrow$ $y_1 \geq 0$ を掛けて目的関数に足す

$$
\begin{aligned}
\text{Minimize} \quad & c_1x_1 + \cdots + c_nx_n \\
& \quad + y_1(b_1 - a_{1,1}x_1 - \cdots - a_{1,n}x_n) \\
\text{subject to} \quad & a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n \geq b_2 \\
& \qquad \vdots \\
& a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,n}x_n \geq b_m \\
& x_1, x_2, \ldots, x_n \geq 0
\end{aligned}
$$

ラグランジュ緩和

- 他の制約も緩和してみる (ただし $y_j \geq 0$)

$$\begin{aligned}\text{Minimize} \quad & c_1 x_1 + \cdots + c_n x_n \\ & + \sum_{j=1}^{m} y_j (b_j - a_{j,1} x_1 - \cdots - a_{j,n} x_n) \\ \text{subject to} \quad & x_1, x_2, \ldots, x_n \geq 0\end{aligned}$$

- 項を並べ替えると

$$\begin{aligned}\text{Minimize} \quad & b_1 y_1 + \cdots + b_m y_m \\ & + \sum_{i=1}^{n} x_i (c_i - a_{1,i} y_1 - \cdots - a_{m,i} y_m) \\ \text{subject to} \quad & x_1, x_2, \ldots, x_n \geq 0\end{aligned}$$

双対問題

- この問題は元の問題の「緩和」
- 最適解の目的関数の値は下がっている

  $\Longrightarrow$ 元の問題の最適解の目的関数の値の下界
- 良い下界が欲しいので

  $b_1 y_1 + \cdots + b_m y_m$

  を最大化にする $y_1, y_2, \ldots, y_m$
- しかし $x_i \geq 0$ なので

  $c_i - a_{1,i} y_1 - \cdots - a_{m,i} y_m \geq 0$

  じゃないと意味がない

  $\Longleftarrow$ いくらでも目的関数の値を小さくできる

双対問題

- 以下の問題を元の問題の「双対問題」とよぶ

$$
\begin{aligned}
\text{Maximize} \quad & b_1 y_1 + b_2 y_2 + \cdots + b_m y_m \\
\text{subject to} \quad & a_{1,1} y_1 + a_{2,1} y_2 + \cdots + a_{m,1} y_m \leq c_1 \\
& a_{1,2} y_1 + a_{2,2} y_2 + \cdots + a_{m,2} y_m \leq c_2 \\
& \qquad \vdots \\
& a_{1,n} y_1 + a_{2,n} y_2 + \cdots + a_{m,n} y_m \leq c_n \\
& y_1, y_2, \ldots, y_m \geq 0
\end{aligned}
$$

- 行列を使った表現は以下のようになる

$$
\begin{aligned}
\text{Maximize} \quad & b^\top y \\
\text{subject to} \quad & A^\top y \leq c \\
& y \in \mathbb{R}^m_+
\end{aligned}
$$

双対定理

- $x \in \mathbb{R}^n_+ \stackrel{\text{def}}{=}$ 元の問題の実行可能解 ($x^* \stackrel{\text{def}}{=}$ 最適解)
- $y \in \mathbb{R}^m_+ \stackrel{\text{def}}{=}$ 双対問題の実行可能解 ($y^* \stackrel{\text{def}}{=}$ 最適解)

定理：弱双対定理

$$c^\top x \geq b^\top y$$

証明

- $c^\top x \geq (y^\top A)x = y^\top (Ax) \geq y^\top b = b^\top y$ □

定理：強双対定理（証明は省略）

$$c^\top x^* = b^\top y^*$$

概要

ゼロ和ゲーム

- 入力：
  - 行列 $A = (a_{i,j})_{i=1,\ldots,m,j=1,\ldots,n}$
  - 行プレイヤー R と列プレイヤー C
  - $a_{i,j}$ = C が R に払う金額
- 考えるゲーム：
  - R の戦略

    $\Longrightarrow$ 行集合 $\{1, 2, \ldots, m\}$ 上の確率分布
  - C の戦略

    $\Longrightarrow$ 列集合 $\{1, 2, \ldots, n\}$ 上の確率分布

ゼロ和ゲーム

- 利得の定義：(R の戦略 $\{x_i\}$, C の戦略 $\{y_i\}$)
  - R の利得 $:= \sum_{i=1}^{m} \sum_{j=1}^{n} a_{i,j} x_i y_j$
  - C の損害 $:= \sum_{i=1}^{m} \sum_{j=1}^{n} a_{i,j} x_i y_j$
- 目的：
  - R の目的 $:= \max_{\{x_i\}} \min_{\{y_j\}} \sum_{i=1}^{m} \sum_{j=1}^{n} a_{i,j} x_i y_j$
  - C の目的 $:= \min_{\{y_j\}} \max_{\{x_i\}} \sum_{i=1}^{m} \sum_{j=1}^{n} a_{i,j} x_i y_j$

von Neumann の最大最小定理

定理

$$\max_{\{x_i\}} \min_{\{y_j\}} \sum_{i=1}^{m} \sum_{j=1}^{n} a_{i,j} x_i y_j = \min_{\{y_j\}} \max_{\{x_i\}} \sum_{i=1}^{m} \sum_{j=1}^{n} a_{i,j} x_i y_j$$

- 左辺の問題は以下のように書ける

$$\begin{aligned} \text{Maximize} \quad & z \\ \text{subject to} \quad & \sum_{i=1}^{m} a_{ij} x_i \geq z \quad (j \in \{1, 2, \ldots, n\}) \\ & \sum_{i=1}^{m} x_i = 1 \\ & x_i \geq 0 \quad (i \in \{1, 2, \ldots, m\}) \end{aligned}$$

von Neumann の最大最小定理

- 先ほどの線形計画問題の双対問題は以下のようになる

$$
\begin{aligned}
\text{Minimize} \quad & \alpha \\
\text{subject to} \quad & \sum_{j=1}^{n} a_{ij} y_j \leq \alpha \quad (i \in \{1, 2, \ldots, m\}) \\
& \sum_{j=1}^{n} y_j = 1 \\
& y_j \geq 0 \quad (j \in \{1, 2, \ldots, n\})
\end{aligned}
$$

- これは定理の右辺を求める問題
- 強双対定理より右辺と左辺が一致

□

概要

課題

課題 1

講演で紹介した線形計画問題の例をソルバーで解く

課題 2

以下の問題

$$
\begin{aligned}
\text{Maximize} \quad & z \\
\text{subject to} \quad & \sum_{i=1}^{m} a_{ij} x_i \geq z \quad (j \in \{1, 2, \ldots, n\}) \\
& \sum_{i=1}^{m} x_i = 1 \\
& x_i \geq 0 \quad (i \in \{1, 2, \ldots, m\})
\end{aligned}
$$

の双対問題を導く

課題 2 の解答

- 各 $j = 1, 2, \ldots, n$ に対して変数 $y_j \geq 0$ を用意する
- 変数 β を用意する
- 問題を緩和すると

$$\max \quad z + \sum_{j=1}^{n} y_j (\sum_{i=1}^{m} a_{ij} x_i - z) + \beta (1 - \sum_{i=1}^{m} x_i)$$

- 整理すると

$$\max \quad \beta + z(1 - \sum_{j=1}^{m} y_j) + \sum_{i=1}^{m} x_i (\sum_{j=1}^{n} a_{ij} y_j - \beta)$$

課題 2 の解答

- よい上界を求めたいので　$\min\ \beta$
- z は自由なので　$\sum_{j=1}^{n} y_j - 1 = 0$
- $x_i \geq 0$ なので意味があるためには　$\sum_{j=1}^{n} a_{ij} y_j - \beta \leq 0$
- 以上より

$$
\begin{aligned}
&\text{Minimize} && \beta \\
&\text{subject to} && \sum_{j=1}^{n} a_{ij} y_j \leq \beta \quad (i \in \{1, 2, \ldots, m\}) \\
& && \sum_{j=1}^{n} y_j = 1 \\
& && y_j \geq 0 \quad (j \in \{1, 2, \ldots, n\})
\end{aligned}
$$