

行列計算による機械学習： 入門と応用

今倉 暁

筑波大学 システム情報系

imakura@cs.tsukubai.ac.jp



データ解析

● データ解析のフロー（教師あり学習）

➤ 入力（教師データセット）

- ✓ 教師データ: x_i , 対応する正解値: y_i , データ数: n

$$\{x_i, y_i\}_{i=1,2,\dots,n}$$

➤ 目的

- ✓ 教師データセットを再現するモデルを構築

$$y_i \approx f(x_i, w), \quad i = 1, 2, \dots, n$$

- ✓ テストデータ x_{test} に対する正解値を予測する

データ解析

データ解析のフロー（教師あり学習）

➤ Step 1: モデルの選択

$$f(x_i, w)$$

➤ Step 2: 誤差の定義

$$D(\{f(x_i, w), y_i\}_{i=1,2,\dots,n})$$

➤ Step 3: モデルの最適化

$$w_{\text{opt}} = \arg \min_w D(\{f(x_i, w), y_i\}_{i=1,2,\dots,n})$$

➤ Step 4: 未知のデータの解析

$$y_{\text{test}} = f(x_{\text{test}}, w_{\text{opt}})$$

データ解析

データ解析のフロー（教師あり学習）

➤ Step 1: モデルの選択

- ✓ 線形/非線形回帰モデル
- ✓ (ディープ) ニューラルネットワーク
- ✓ 木構造
- ✓ ...

➤ Step 2: 誤差の定義

- ✓ 二乗誤差
- ✓ Kullback-Leibler divergence
- ✓ 正則化
- ✓ ...

データ解析

● データ解析のフロー（教師あり学習）

➤ Step 3: モデル最適化

- ✓ 行列分解
- ✓ 固有値計算
- ✓ (確率的) 勾配降下法
- ✓ ...

特にこの部分は
「行列計算」
がいっぱい

➤ Step 4: 未知のデータの解析

- ✓ モデルの適用

● もくじ

● はじめに

● 行列計算による機械学習：入門

- 最小二乗法とその発展
- 次元削減法
- 行列計算の観点での計算の工夫

● 行列計算による機械学習：発展

- プライバシー保護機械学習
- データコラボレーション解析

行列計算による機械学習：入門

最小二乗法とその発展

最小二乗法

問題設定

➤ 学習データ

✓ データ (m:特徴量数、n:サンプル数)

$$X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n]^T \in \mathbb{R}^{n \times m}$$

✓ 正解ラベル (もしくは正解データ)

$$\boldsymbol{y} = [y_1, y_2, \dots, y_n]^T \in \mathbb{R}^n$$

最小二乗法

概要

➤ 最も単純な機械学習法

✓ モデル: 線形回帰

$$\mathbf{y} \approx f(\mathbf{X}, \mathbf{w}) = \mathbf{X}\mathbf{w}, \quad \mathbf{w} \in \mathbb{R}^m$$

✓ 誤差: 二乗誤差

$$D(f(\mathbf{X}, \mathbf{w}), \mathbf{y}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 = \sum_{i=1}^n (y_i - \mathbf{x}_i^T \mathbf{w})^2$$

➤ 定式化

$$\min_{\mathbf{w} \in \mathbb{R}^m} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$$

● 最小二乗法

● 計算法

$$\min_{w \in \mathbb{R}^m} \|y - Xw\|_2^2$$

- 厳密な解法
 - ✓ QR分解
 - ✓ 特異値分解
 - ✓ 正規方程式
- 近似解法（機械学習では近似解で十分な場合が多い）
 - ✓ 反復法（例：Krylov部分空間法）
 - ✓ 近似QR分解（例：不完全QR分解）
 - ✓ 近似特異値分解（例：ランダムイズドSVD）
 - ✓ ...

正則化

● 正則化とは

- 過学習（学習データへの過適合）を防ぐために、誤差項に加えてモデルの複雑さに対するペナルティを課す
- オッカムの剃刀
 - ✓ ある事柄を説明するためには、必要以上に多くを仮定するべきでない [Occam, 14世紀]

● 代表的な正則化項

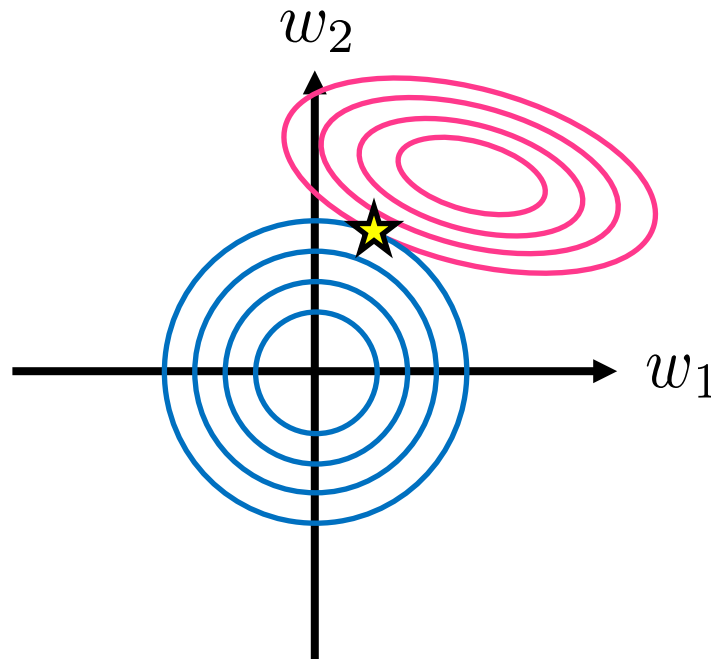
- L2正則化：滑らかさ $\|w\|_2^2 = \sum_i w_i^2$
- L1正則化：スパース性 $\|w\|_1 = \sum_i |w_i|$
- L0正則化：スパース性 $\|w\|_0 = w$ の非零要素数
- 組み合わせ（例：L1正則化とL2正則化）

計算コストが大きく
あまり使われない

● L1正則化とL2正則化のイメージ

● L2正則化




- ◎: 誤差項の等高線
- ⊙: L2正則化項の等高線
- ☆: 最適解

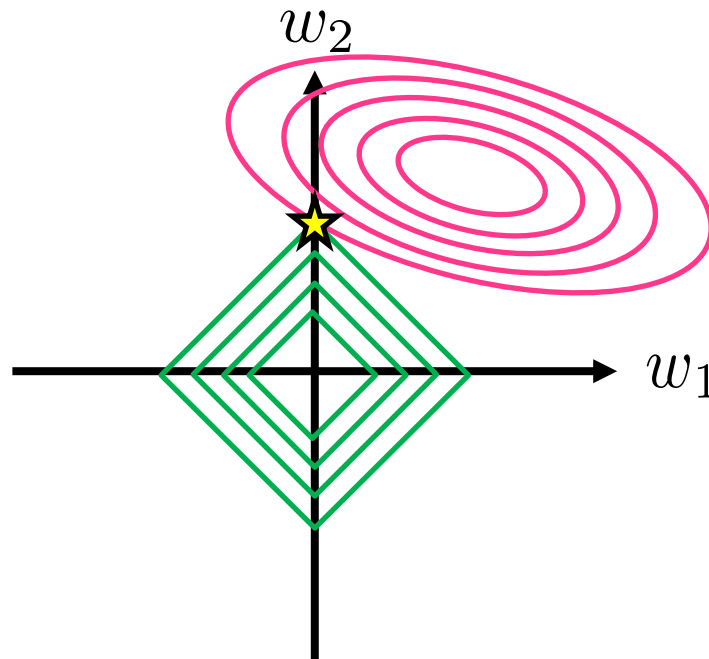


重みが過剰に大きな値を持ちにくくなる

● L1正則化とL2正則化のイメージ

● L1正則化

- : 誤差項の等高線
- : L1正則化項の等高線
- : 最適解



重みがスパースになる

● 最小二乘法 + 正則化

● Ridge回帰

➤ 線形回帰 + L2正則化

$$\min_w \|y - Xw\|_2^2 + \lambda \|w\|_2^2, \quad \|w\|_2^2 = \sum_i w_i^2$$

● Lasso

➤ 線形回帰 + L1正則化

$$\min_w \|y - Xw\|_2^2 + \lambda \|w\|_1, \quad \|w\|_1 = \sum_i |w_i|$$

● Elastic Net

➤ 線形回帰 + L1正則化 + L2正則化

$$\min_w \|y - Xw\|_2^2 + \lambda \|W\|_1 + (1 - \lambda) \|W\|_2^2$$

Ridge回帰

概要

線形回帰 + L2正則化

$$\min_w \|y - Xw\|_2^2 + \lambda \|w\|_2^2, \quad \|w\|_2^2 = \sum_i w_i^2$$

計算法

線形最小二乗問題に帰着

$$\|y - Xw\|_2^2 + \lambda \|w\|_2^2 = \left\| \begin{bmatrix} y \\ 0 \end{bmatrix} - \begin{bmatrix} X \\ \sqrt{\lambda} I_m \end{bmatrix} w \right\|_2^2$$
$$\left(\because \|a\|_2^2 + \|b\|_2^2 = \left\| \begin{bmatrix} a \\ b \end{bmatrix} \right\|_2^2 \right)$$

線形最小二乗問題 → QR分解、特異値分解、反復法、、

Lasso

概要

➤ 線形回帰 + L1正則化

$$\min_w \|y - Xw\|_2^2 + \lambda \|w\|_1, \quad \|w\|_1 = \sum_i |w_i|$$

計算法

- 直接厳密解を求めることができないため、反復法で近似解を計算
 - ✓ 座標降下法 (Coordinate Descent: CD法)
 - ✓ 交互方向乗数法 (Alternating Direction Method of Multipliers : ADMM法)
 - MATLABで使われている

Elastic Net

概要

- 線形回帰 + L1正則化 + L2正則化

$$\min_w \|y - Xw\|_2^2 + \lambda \|W\|_1 + (1 - \lambda) \|W\|_2^2$$

計算法

- L2正則化項を誤差項と結合する (Ridge回帰と同様)
- 得られた最小二乗問題 + L1正則化に対してLassoと同様に反復法で近似解を計算
 - ✓ 座標降下法 (Coordinate Descent: CD法)
 - ✓ 交互方向乗数法 (Alternating Direction Method of Multipliers : ADMM法)
 - MATLABで使われている

カーネル法

● 基本的アイデア

- 解析性能の改善のため、入力データを非線形変換することで特徴量を増やす

$$\boldsymbol{x} \in \mathbb{R}^m \rightarrow \hat{\boldsymbol{x}} = \phi(\boldsymbol{x}) \in \mathbb{R}^{\hat{m}}, \quad m < \hat{m}$$

- 線形変換のみでは判別・分類できない問題に特に有効

● 例：2次多項式カーネル

- 元データ

$$\boldsymbol{x} = [x_1, x_2]^T$$

- 変換後のデータ

$$\hat{\boldsymbol{x}} = [1, x_1, x_2, x_1x_2, x_1^2, x_2^2]^T$$

カーネル法 + 線形回帰

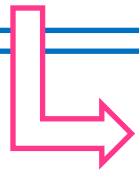
● 定式化

$$\min_{\hat{\mathbf{w}} \in \mathbb{R}^{\hat{m}}} \|\mathbf{y} - \hat{X} \hat{\mathbf{w}}\|_2^2$$

● 効率的な計算法:カーネルトリック

➤ 重み行列に対する制約

$$\hat{\mathbf{w}} = \hat{X}^T \tilde{\mathbf{w}}$$



$$\min_{\tilde{\mathbf{w}} \in \mathbb{R}^n} \|\mathbf{y} - K \tilde{\mathbf{w}}\|_2^2, \quad K = \hat{X} \hat{X}^T$$

- $\hat{X} = \phi(X)$ を計算することなく直接グラム行列 K を計算
- ✓ 計算の効率化
 - ✓ 無限次元の非線形変換も考慮可能

カーネル法

カーネルトリック

- 「非線形関数の設定→グラム行列の計算」ではなく、「グラム行列を直接計算」(グラム行列は対称半正定値)

$$K = \hat{X} \hat{X}^T, \quad k_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j)$$

代表的なカーネル

- 多項式カーネル ($c \geq 0$, d : 自然数)

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + c)^d$$

- ガウスカーネル (RBFカーネル)

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp \left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\sigma^2} \right)$$

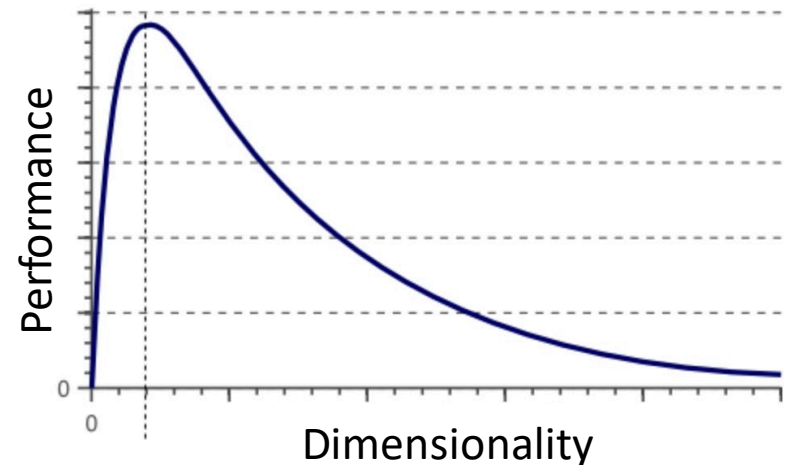
行列計算による機械学習：入門

次元削減法

次元削減法

データの次元

- 特徴量が多いほど最適化問題の最小値は小さくなる
- 実用上は特徴量が大きすぎると性能が低下する

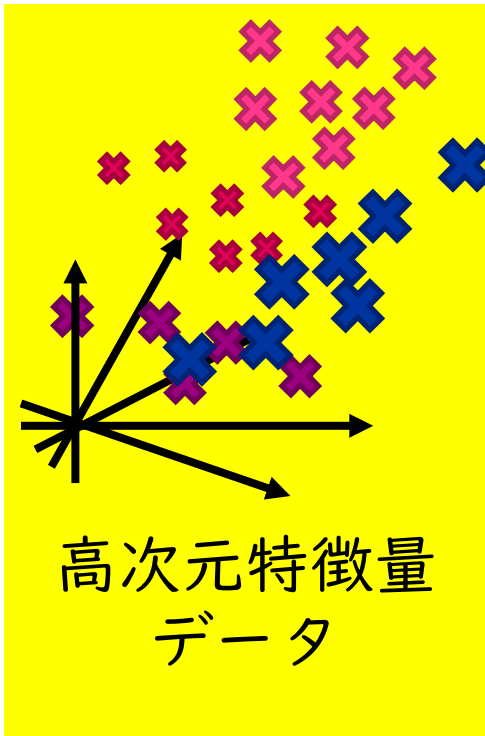


次元削減

- 高次元データの特徴量を削減する方法
 - ✓ 解析手法の計算時間の削減、解析性能の改善
- 手法の分類
 - ✓ 特徴量選択: 特徴量の一部をそのまま用いる
 - ✓ 写像による次元削減: 高次元特徴量を持つデータを低次元空間へ射影する

次元削減法

- 行列トレースの最小化/最大化に基づく次元削減法
 - 高次元特徴量を持つデータ点の低次元空間への写像
 - ✓ 教師なし: 主成分分析 (PCA)、LPP、など
 - ✓ 教師あり: 線形判別分析 (LDA)、LFDA、LADA、など



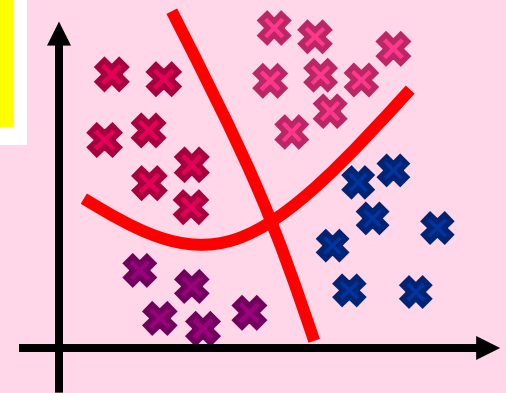
m 次元データ

x

削減後の次元

$$\tilde{m} \tilde{x} = B^T x$$

低次元空間へ
の写像



局所性保存射影 (LPP)

概要

- 教師なし次元削減法
- 局所構造を保存する低次元空間を構築
 - ✓ 元データで近いものは低次元空間でも近くに

定式化

$$\min_B \sum_{ij} w_{ij} \underbrace{\|B^T(\mathbf{x}_i - \mathbf{x}_j)\|_2^2}_{\text{低次元空間での距離}} \quad + B \text{ に対する直交条件 (詳細は後述)}$$

類似度 (元データで近いものは大きな値を持つ)

$$w_{ij} = \exp \left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\sigma^2} \right)$$

*実用上はk近傍法によりスパース化する (元データで十分遠いものは $w=0$ とする)

局所性保存射影 (LPP)

最小化問題の変形

$$\begin{aligned}
 & \sum_{ij} w_{ij} \|B^T(\mathbf{x}_i - \mathbf{x}_j)\|_2^2 \\
 &= \sum_{ij} w_{ij} (B^T \mathbf{x}_i - B^T \mathbf{x}_j)^T (B^T \mathbf{x}_i - B^T \mathbf{x}_j) \\
 &= 2 \sum_i \left(\sum_j w_{ij} \right) (B^T \mathbf{x}_i)^T (B^T \mathbf{x}_i) - 2 \sum_i (B^T \mathbf{x}_i)^T \left(\sum_j w_{ij} (B^T \mathbf{x}_j) \right) \\
 &= 2 \sum_i \underline{d_i} (B^T \mathbf{x}_i)^T (B^T \mathbf{x}_i) - 2 \sum_i (B^T \mathbf{x}_i)^T \underline{(B^T X \mathbf{w}_i)} \\
 &= 2 \underline{\text{Tr}(B^T X^T D X B)} - 2 \underline{\text{Tr}(B^T X^T W X B)} \\
 &= 2 \underline{\text{Tr}(B^T X^T L X B)}
 \end{aligned}$$

$$\sum_i \mathbf{x}_i^T \mathbf{y}_i = \text{Tr}(X Y^T)$$

$$L = D - W$$

局所性保存射影 (LPP)

概要 (再掲)

- 教師なし次元削減法
- 局所構造を保存する低次元空間を構築
 - ✓ 元データで近いものは低次元空間でも近くに

定式化

- 行列トレースの最小化問題

$$\min_B \text{Tr}(B^T X^T L X B) \quad \text{s.t.} \quad B^T X^T D X B = I$$

$$L = D - W, \quad D = \text{diag}(d_i), \quad d_i = \sum_j w_{ij}$$

局所性保存射影 (LPP)

● パラメータ設定

➤ 類似度行列の σ のヒューリスティックな設定法

✓ 類似度行列の定義

$$w_{ij} = \exp \left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\sigma^2} \right) \rightarrow w_{ij} = \exp \left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\sigma_i \sigma_j} \right)$$

✓ 各サンプル周りのスケーリング
($\mathbf{x}_i^{(k)}$: サンプル \mathbf{x}_i 周りの k 近傍)

$$\sigma_i = \|\mathbf{x}_i - \mathbf{x}_i^{(k)}\|_2$$

✓ k の設定

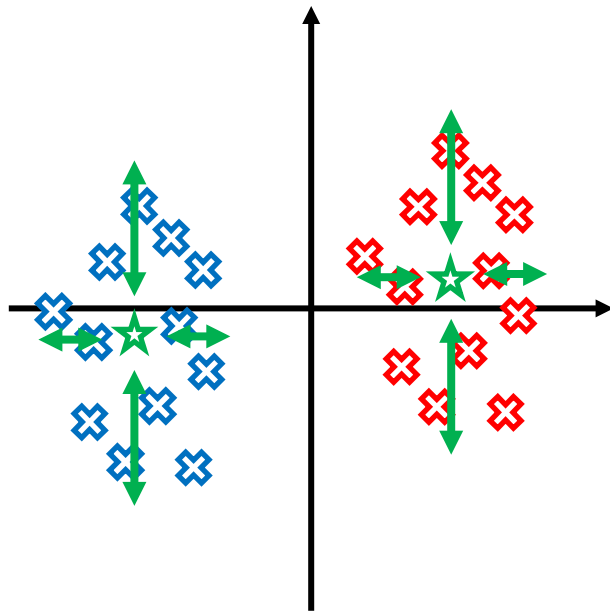
— 経験的に $k=7$ がよいとされる

線形判別分析 (LDA)

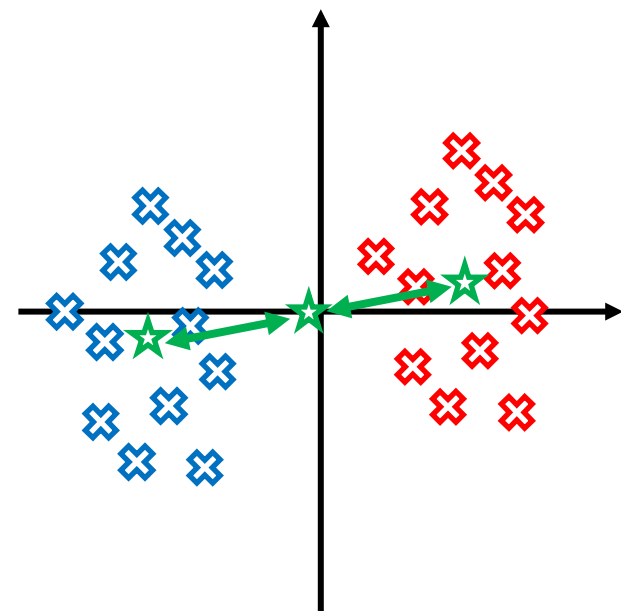
概要

- 教師あり次元削減法（分類問題のクラスラベルを利用）
- クラス内分散の最小化 & クラス間分散の最大化
 - ✓ 同じラベルのデータは近く、違うラベルは遠く

クラス内分散



クラス間分散



線形判別分析 (LDA)

● 定式化

➤ クラス内分散の最小化

$$\min_B \sum_{y=1}^c \sum_{i, y_i=y} \|B^T(x_i - \mu_y)\|_2^2$$

クラス数

クラス内平均値

$$\mu_y = \frac{1}{n_y} \sum_{i, y_i=y} x_i$$

➤ クラス間分散の最大化

$$\max_B \sum_{y=1}^c n_y \|B^T(\mu_y - \mu)\|_2^2$$

全データの平均値

$$\mu = \frac{1}{n} \sum_i x_i$$

注意

$$B^T \mu = \frac{1}{n} \sum_i B^T x_i$$

➤ 統合

✓ クラス内分散を1に正規化し、クラス間分散を最大化

線形判別分析 (LDA)

最適化問題の変形

クラス内分散

$$\sum_{y=1}^c \sum_{i, y_i=y} \|B^T(\mathbf{x}_i - \boldsymbol{\mu}_y)\|_2^2$$

$$= \sum_i (B^T(\mathbf{x}_i - \boldsymbol{\mu}_{y_i}))^T (B^T(\mathbf{x}_i - \boldsymbol{\mu}_{y_i}))$$

$$\sum_i \mathbf{x}_i^T \mathbf{y}_i = \text{Tr} \left(\sum_i \mathbf{x}_i \mathbf{y}_i^T \right) = \text{Tr}(XY^T)$$

$$= \text{Tr} \left(B^T \left(\sum_i (\mathbf{x}_i - \boldsymbol{\mu}_{y_i})(\mathbf{x}_i - \boldsymbol{\mu}_{y_i})^T \right) B \right)$$

$$= \text{Tr}(B^T S_{\text{in}} B)$$

$$S_{\text{in}} = \tilde{X}^T \tilde{X}$$

$$\tilde{X} = X - [\boldsymbol{\mu}_{y_1}, \boldsymbol{\mu}_{y_2}, \dots, \boldsymbol{\mu}_{y_n}]^T$$

線形判別分析 (LDA)

最適化問題の変形

クラス間分散

$$\sum_{y=1}^c n_y \|B^T(\boldsymbol{\mu}_y - \boldsymbol{\mu})\|_2^2$$

$$= \sum_{y=1}^c n_y (B^T(\boldsymbol{\mu}_y - \boldsymbol{\mu}))^T (B^T(\boldsymbol{\mu}_y - \boldsymbol{\mu}))$$

$$\sum_i \mathbf{x}_i^T \mathbf{y}_i = \text{Tr} \left(\sum_i \mathbf{x}_i \mathbf{y}_i^T \right) = \text{Tr}(XY^T)$$

$$= \text{Tr} \left(B^T \left(\sum_{y=1}^c n_y (\boldsymbol{\mu}_y - \boldsymbol{\mu}) (\boldsymbol{\mu}_y - \boldsymbol{\mu})^T \right) B \right)$$

$$= \text{Tr}(B^T \underline{S_{\text{all}}} B)$$

● 線形判別分析 (LDA)

● 概要 (再掲)

- 教師あり次元削減法 (分類問題のクラスラベルを利用)
- クラス内分散の最小化 & クラス間分散の最大化
 - ✓ 同じラベルのデータは近く、違うラベルは遠く

● 定式化

- 行列トレースの最大化問題

$$\max_B \text{Tr}(B^T S_{\text{all}} B) \quad \text{s.t.} \quad B^T S_{\text{in}} B = I$$

● トレース最適化問題の求解

● 定式化

- 行列トレースの最大/最小化 (A_1 :対称、 A_2 :SPD)

$$\max_{B \in \mathbb{R}^{m \times \ell}} \text{Tr}(B^T A_1 B) \quad \text{or} \quad \min_{B \in \mathbb{R}^{m \times \ell}} \text{Tr}(B^T A_1 B) \quad \text{s.t.} \quad B^T A_2 B = I$$

● 計算法

- 一般化固有値問題に帰着

$$A_1 \mathbf{u}_i = \lambda_i A_2 \mathbf{u}_i, \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$$

- 解は固有ベクトルを並べた行列となる

$$B = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_\ell] \quad \text{or} \quad B = [\mathbf{u}_n, \mathbf{u}_{n-1}, \dots, \mathbf{u}_{n-\ell+1}]$$

行列計算による機械学習：入門

行列計算の観点での計算の工夫

● 背景

● 計算時間、計算量、計算効率（実行性能）

$$\text{計算時間} = \text{計算量} / \text{実行性能}$$

- 計算時間は計算量だけでなく、実行性能にも強く依存
- 実行性能は下記項目に依存する
 - ✓ マシン性能
 - 理論実行性能: CPU性能のカatalogスペック
(CPU数、クロック周波数など)
 - データ転送性能: CPU-メモリ間などの転送性能
 - ✓ 計算対象
 - 「データ量 vs 計算量」の比

● 背景

● 計算時間、計算量、計算効率（実行性能）

$$\text{計算時間} = \text{計算量} / \text{実行性能}$$

- 計算時間は実行性能の下記項目に依存する
 - ✓ 計算対象：「データ量 vs 計算量」の比
 - 「データ量 = 計算量」
 - 計算時間はデータ転送がボトルネック
 - 遅い！
 - 「データ量 << 計算量」
 - 計算時間は計算性能に依存
 - 理論性能に近い計算速度を実現（速い！）

実行性能

行列行列積 >> 行列ベクトル積 > ベクトル計算

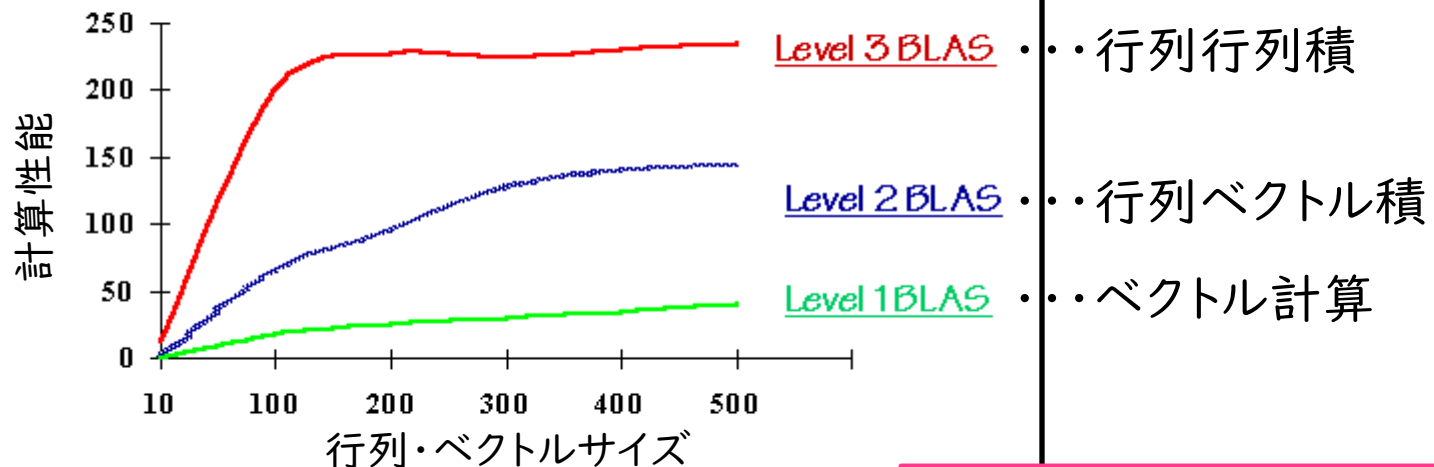
背景

● 行列行列積と行列ベクトル積の性能差

- BLAS (Basic Linear Algorithms、基本行列計算ライブラリ)の性能 [Dongara et al., SC97]

BLAS for Performance

IBM RS/6000-590 (66 MHz, 264 Mflop/s Peak)



◆ Development of blocked algorithms important for performance

ブロックアルゴリズム (行列積ベースの方法) が重要

● ガウスカーネルでの例

● 背景(再掲)

- 行列積は計算効率が高い
→ 可能な限り行列積ベースで実装すべき

● ガウスカーネル

- 定義

$$K = [k_{ij}], \quad k_{ij} = \exp \left(-\frac{\|x_i - x_j\|_2^2}{\sigma^2} \right)$$

- ナイーブな実装 (MATLAB)

```
for j = 1:n
    for i = 1:n
        K(i,j) = exp(-norm(X(:,i) - X(:,j))^2/sigma^2);
    end
end
```

計算の主要部: n^2 回のノルム計算

● ガウスカーネルでの例

● 背景(再掲)

- 行列積は計算効率が高い
→ 可能な限り行列積ベースで実装すべき

● ガウスカーネル

- 定義

$$K = [k_{ij}], \quad k_{ij} = \exp \left(-\frac{\|x_i - x_j\|_2^2}{\sigma^2} \right)$$

- 対称性を利用した実装 (MATLAB)

```
for j = 1:n
    K(j,j) = 1;
    for i = 1:j-1
        K(i,j) = exp(-norm(X(:,i) - X(:,j))^2/sigma^2);
        K(j,i) = K(i,j);
    end
end
end
```

計算の主要部: $n(n-1)/2$ 回のノルム計算

● ガウスカーネルでの例

● ガウスカーネル

➤ 式変形

$$K = [k_{ij}], \quad k_{ij} = \exp\left(-\frac{g_{ij}}{\sigma^2}\right), \quad g_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2$$

$$G = [g_{ij}], \quad g_{ij} = \mathbf{x}_i^T \mathbf{x}_i + \mathbf{x}_j^T \mathbf{x}_j - 2\mathbf{x}_i^T \mathbf{x}_j$$

$$\mathbf{g} = [\|\mathbf{x}_1\|_2^2, \|\mathbf{x}_2\|_2^2, \dots, \|\mathbf{x}_n\|_2^2]^T$$

$$= \mathbf{g}\mathbf{1}^T + \mathbf{1}\mathbf{g}^T - 2\mathbf{X}\mathbf{X}^T$$

➤ 行列積ベースの実装 (MATLAB)

計算量の主要部: 行列積

```
g = sum(X.^2, 1);
G = repmat(g, n, 1) + repmat(g', 1, n) - 2*X*X';
K = exp(- G / sigma^2);
```


● ガウスカーネルでの例

● ガウスカーネル

➤ MATLABでの計算例

- ✓ $m = 1000, n = 1000, 2000, 4000$
- ✓ $\text{sigma} = 1$;
- ✓ $X = \text{rand}(n,m)$;

➤ 実行結果（ノートPCを利用）

| | $n = 1000$ | $n = 2000$ | $n = 4000$ |
|---------|-------------|--------------|---------------|
| ナイーブな実装 | 7.78 [sec.] | 40.04 [sec.] | 202.41 [sec.] |
| 対称性を利用 | 3.58 [sec.] | 19.95 [sec.] | 99.35 [sec.] |
| 行列積を利用 | 0.03 [sec.] | 0.11 [sec.] | 0.43 [sec.] |

→ 特に n が大きい場合に大幅な高速化を実現
(計算法の工夫が重要)

400x
faster

● 各種の分散行列の計算での例

● 背景(再掲)

- 行列積は計算効率が高い
→ 可能な限り行列積ベースで実装すべき

● 各種の分散行列の計算

- 定義(W:対称)

$$S = \sum_{ij} w_{ij} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T$$

- ナイーブな実装 (MATLAB)

```
S = zeros(m);  
for j = 1:n  
    for i = 1:n  
        xd = X(i,:) - X(j,:);  
        S = S + W(i,j) * xd' * xd;  
    end  
end
```

計算の主要部: n^2 回の行列和

● 各種の分散行列の計算での例

● 背景(再掲)

- 行列積は計算効率が高い
→ 可能な限り行列積ベースで実装すべき

● 各種の分散行列の計算

- 定義(W:対称)

$$S = \sum_{ij} w_{ij} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T$$

- 対称性を利用した実装 (MATLAB)

```
S = zeros(m);  
for j = 1:n  
    for i = 1:j-1  
        xd = X(i,:) - X(j,:);  
        S = S + W(i,j) * xd' * xd;  
    end  
end  
S = 2 * S;
```

計算の主要部: $n(n-1)/2$ 回の行列和

● 各種の分散行列の計算での例

● 各種の分散行列の計算

➤ 式変形

$$\begin{aligned} S &= \sum_{ij} w_{ij} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T \\ &= 2 \sum_i \left(\sum_j w_{ij} \right) \mathbf{x}_i \mathbf{x}_i^T - 2 \sum_{ij} w_{ij} \mathbf{x}_i \mathbf{x}_j^T \\ &= 2X^T(D - W)X, \quad D = \text{diag}(d_i), \quad d_i = \sum_j w_{ij} \end{aligned}$$

➤ 行列積ベースの実装 (MATLAB)

```
D = diag(sum(W));  
S = 2 * X * (D - W) * X';
```

計算の主要部: 行列行列積

✓ (計算量が $1/m$ 程度に削減されている)

● 各種の分散行列の計算での例

● 各種の分散行列の計算

➤ MATLABでの計算例

✓ $m = 100, n = 1000, 2000, 4000$

✓ $X = \text{rand}(n,m); W = \text{rand}(n); W = (W + W')/2;$

➤ 実行結果

| | n = 1000 | n = 2000 | n = 4000 | 3000x faster |
|---------|--------------|--------------|---------------|-----------------|
| ナイーブな実装 | 15.29 [sec.] | 72.88 [sec.] | 300.67 [sec.] | |
| 対称性を利用 | 8.31 [sec.] | 36.11 [sec.] | 150.72 [sec.] | |
| 行列積を利用 | 0.01 [sec.] | 0.03 [sec.] | 0.10 [sec.] | |

→ 特にnが大きい場合に計算量削減効果以上の大幅な高速化を実現(計算法の工夫が重要)

行列計算による機械学習：発展 プライバシー保護機械学習

ニーズ


データ解析

➤ 様々な分野でデータが蓄積され、解析ニーズが高まっている

➤  医療分野

✓ 病気リスク予測

✓ リスク因子推定

➤  ものづくり分野

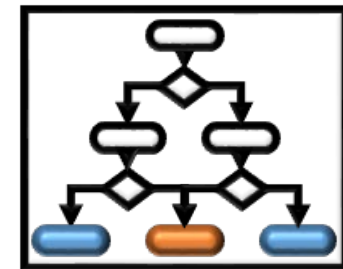
✓ 製品開発の最適化

✓ 故障検知・予測

➤  金融分野

✓ 需要/リスク予測

| ID | Risk | Age | Gender | high | weight | ... |
|----|------|-----|--------|-------|--------|-----|
| 1 | 1 | 45 | Male | 164.3 | 65.4 | ... |
| 2 | 0 | 25 | Female | 144.6 | 46.4 | ... |
| 3 | 1 | 36 | Female | 154.7 | 43.3 | ... |
| 4 | 0 | 62 | Male | 174.5 | 73.2 | ... |



解析モデル



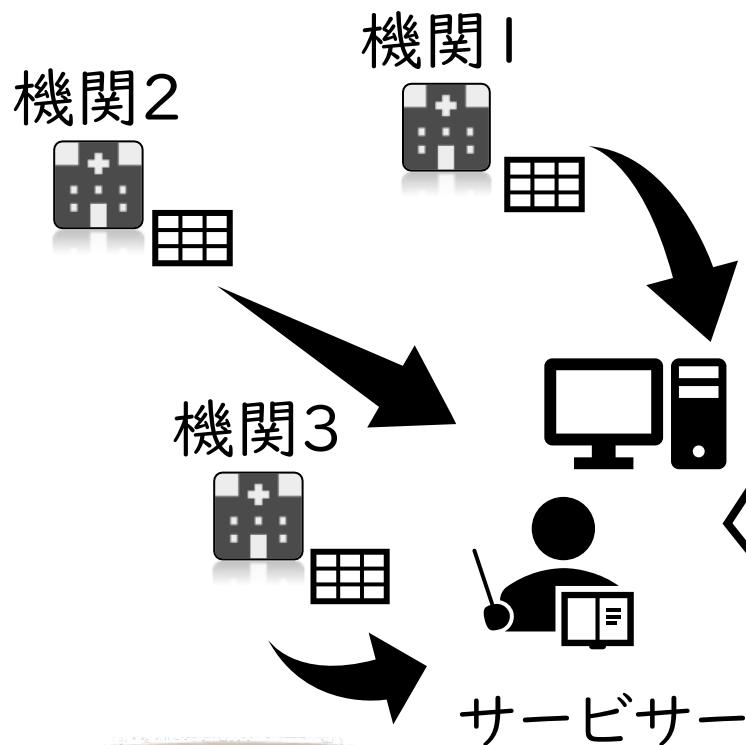
➤ 予測
➤ 因子推定

ニーズ

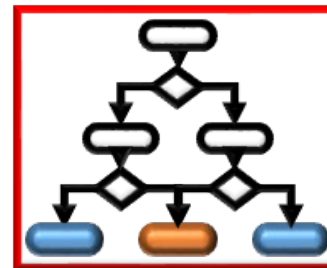
● 多機関分散データ統合解析(理想)

➤ 単独機関のデータは必ずしも十分ではない

✓ 多機関のデータを統合し解析することで、よりよい解析結果が期待される



| ID | Risk | Age | Gender | high | weight | ... |
|-------|------|-----|--------|-------|--------|-----|
| 機関1-1 | 1 | 45 | Male | 164.3 | 65.4 | ... |
| 機関1-2 | 0 | 25 | Female | 144.6 | 46.4 | ... |
| 機関1-3 | 1 | 36 | Female | 154.7 | 43.3 | ... |
| 機関2-1 | 0 | 62 | Male | 174.5 | 73.2 | ... |
| 機関2-2 | 1 | 12 | Male | 153.1 | 76.2 | ... |
| 機関3-1 | 1 | 62 | Female | 174.1 | 42.5 | ... |
| 機関3-2 | 0 | 78 | Female | 156.8 | 63.2 | ... |

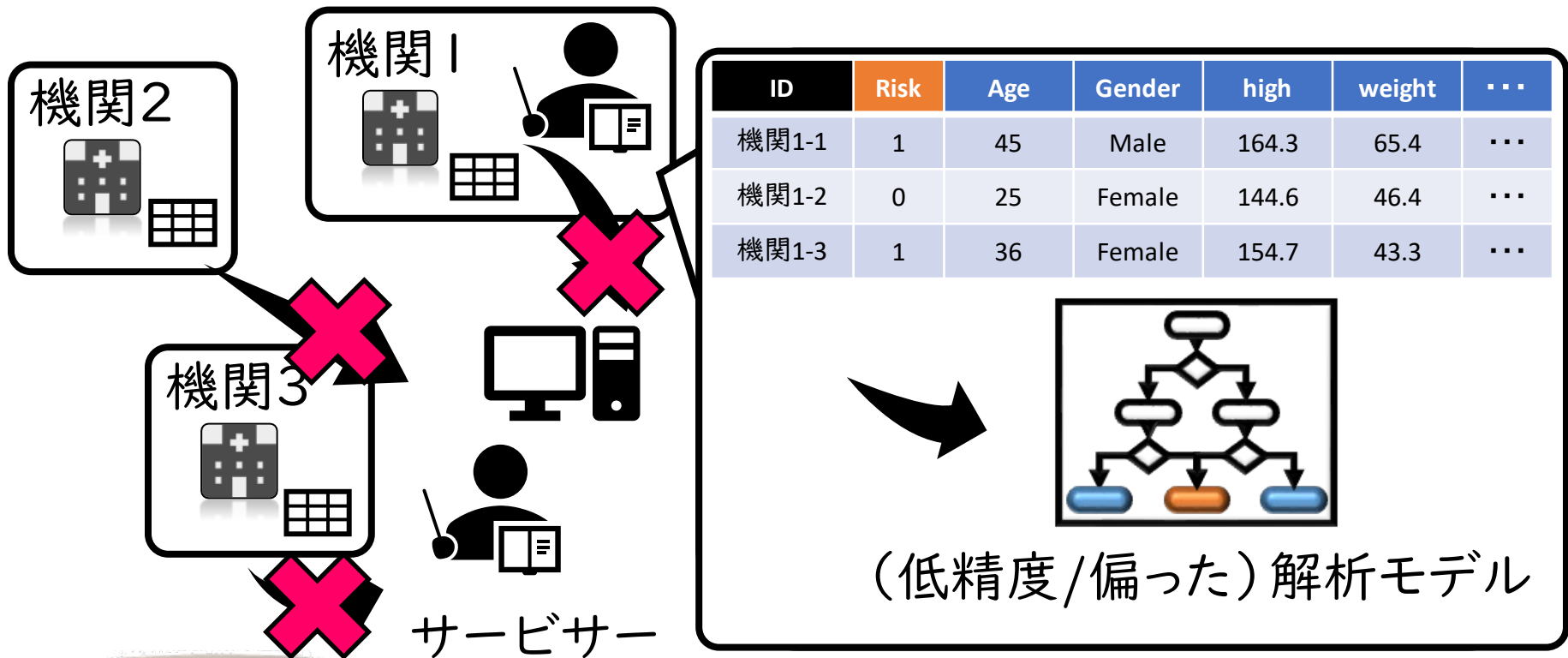


高性能
解析モデル

ニーズ

多機関分散データ統合解析（現実）

- ▶ 個人情報保護や企業秘密などの観点から、データを共有することは困難な場合がある
 - 単独機関のデータのみを用いて解析が行われる



● プライバシー保護機械学習法

● 狙い

- 複数機関が分散保持するデータを安全に統合解析する技術

● 代表的な技術

- 秘密計算（例、2023年にNTTが国際標準規格化）
 - ✓ 生データを四則演算可能暗号化方式で暗号化し、暗号化したまま解析を実施する方法
 - ◎安全性:大 △計算コスト:大
- 連合学習 [Google]
 - ✓ AIモデルを各機関で共有し、各機関の個別データのみでのAIモデル学習を順次反復的に行う方法
 - ◎計算コスト:小 △機関間通信コスト:大

行列計算による機械学習：発展 データコラボレーション解析

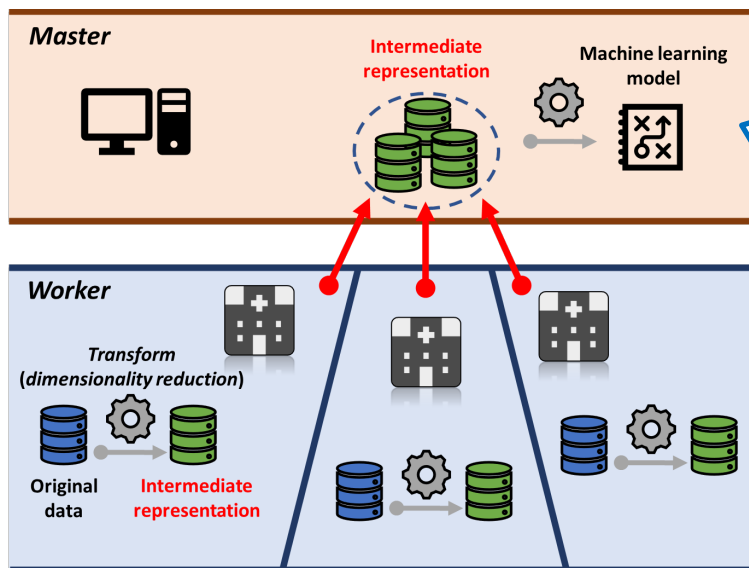
プライバシー保護機械学習

データコラボレーション(DC)解析 [I+,2020] [I+,2021]

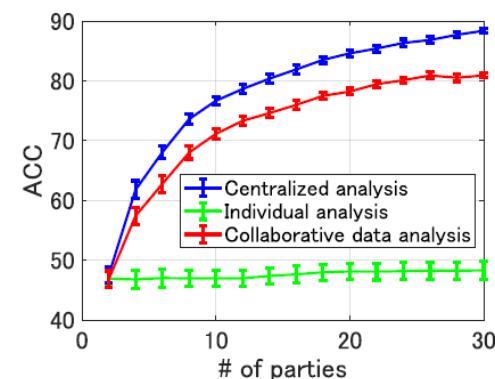
- 複数機関が分散保持するデータを安全に統合解析する技術
- 各機関が独自に抽象化した「中間表現」を共有し解析
- 特徴
 - ✓ 暗号化を行わない

→ 計算コスト小

✓ 機関をまたいだ反復通信が不要 → 通信コスト小



DC解析(●)は、生データを共有することなく、単独機関での解析(●)より高精度かつ、生データ共有時(●)に近い解析性能を実現



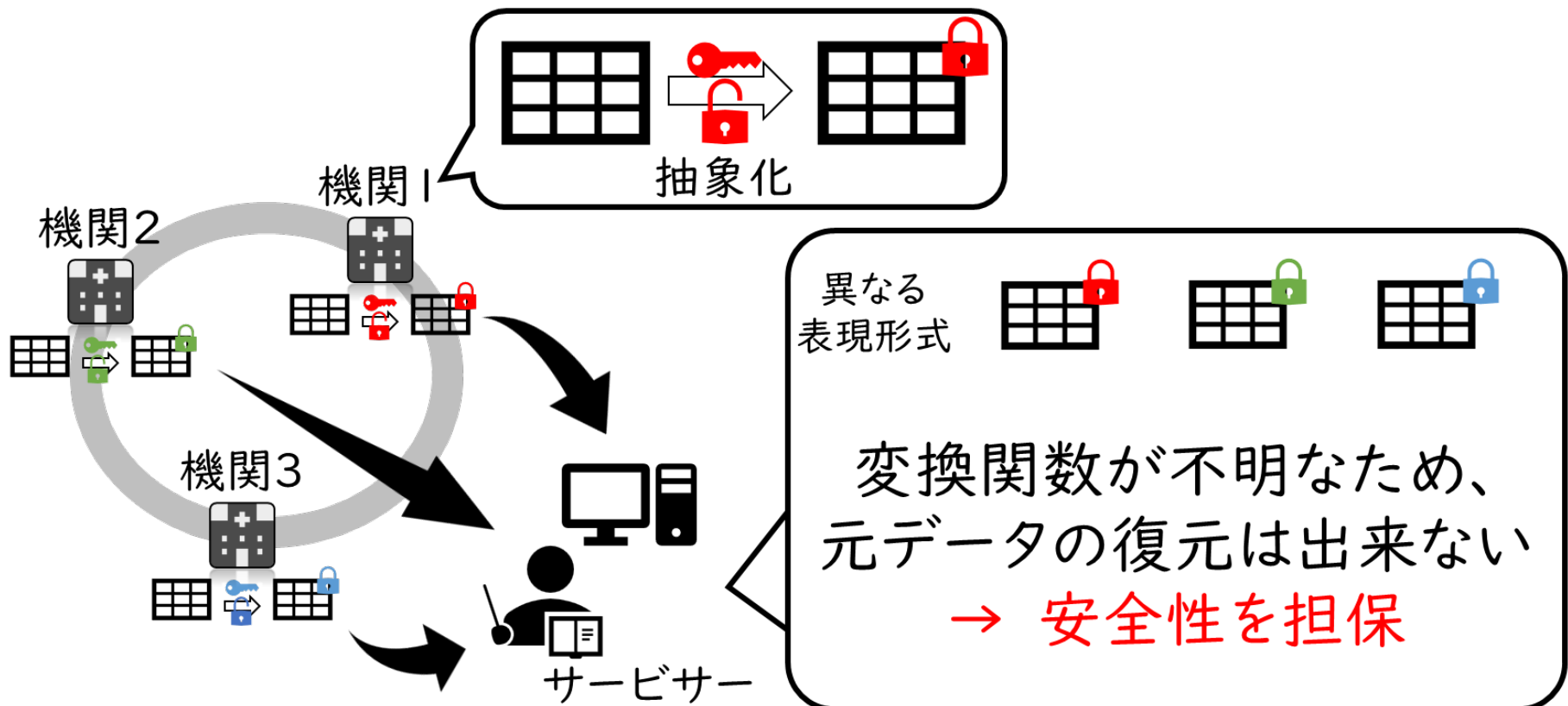
生データ共有
DC解析

単独解析

基本コンセプト



Step 1 : 中間表現の共有

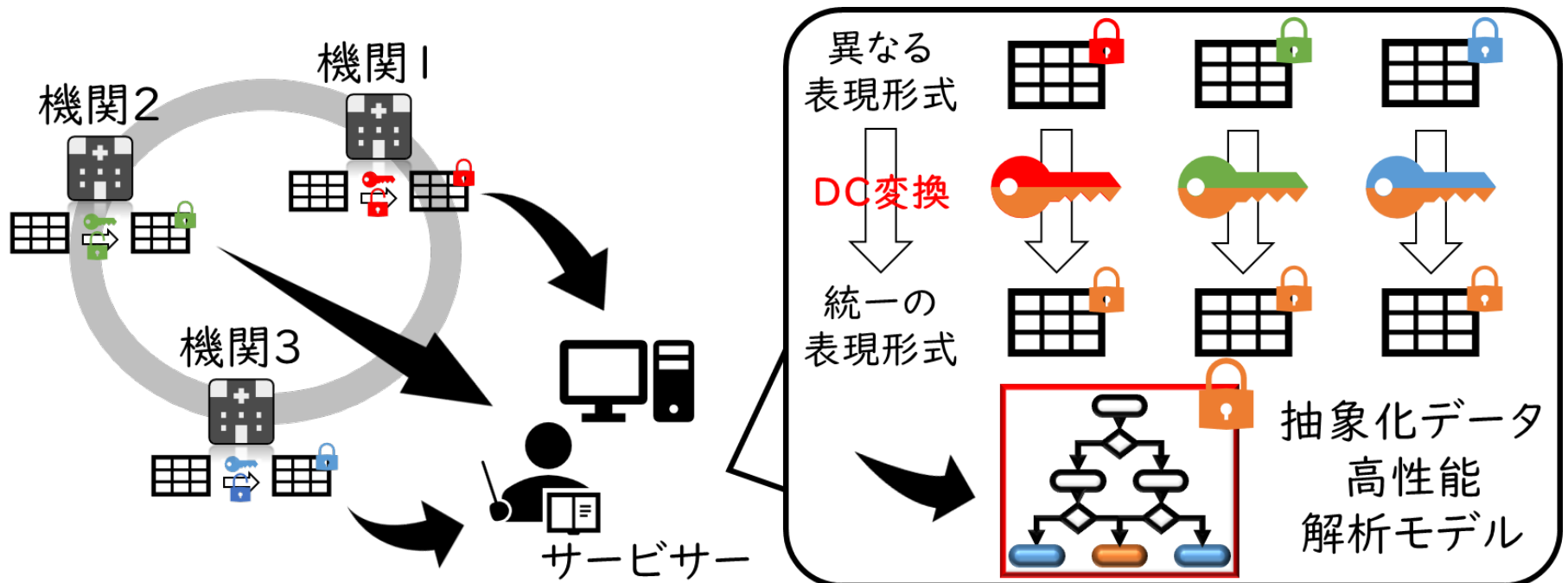
- 秘匿情報を含む「生データ」ではなく、各機関が独自に抽象化を行った「中間表現」() を共有



基本コンセプト

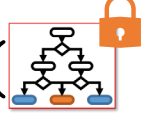

Step 2 : DC表現への変換・解析

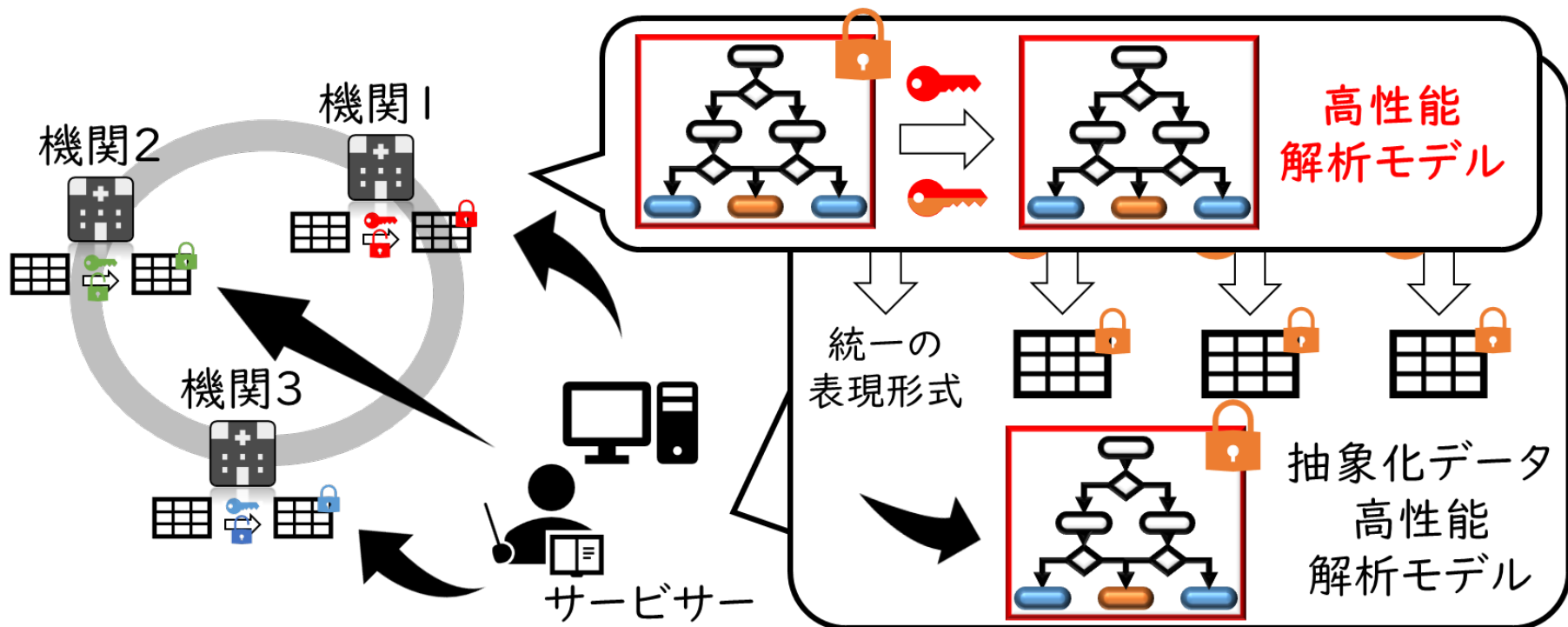
- 各中間表現 () を統合解析可能な統一の表現形式「DC表現」 () に変換し、解析
- 抽象化データに対する高性能解析モデルを得る



基本コンセプト

Step 3 : 解析モデルの送付

- 抽象化データに対する高性能モデル () およびDC変換 () を各機関に送付
- 各機関は高性能解析モデルを得る



問題設定

● テーブルデータの回帰・分類問題

- 学習データセットからモデルを学習する

n : サンプル数
 m : 特徴量次元数

$$t(X) \approx Y$$

$$X = [x_1, x_2, \dots, x_n]^T \in R^{n \times m}, \quad Y = [y_1, y_2, \dots, y_n]^T \in R^{n \times \ell}$$

説明変数

目的変数

- データ分散: 水平 (サンプル) 分散

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_c \end{bmatrix}, Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_c \end{bmatrix}$$

- 各機関は X_i, Y_i を保持し、 X_i を秘匿としたい

統計量ではなくデータそのものを秘匿としたい

● 中間表現の構築

$$m > \tilde{m}_i$$

→ これを共有する

- 関数 f を開示しないことでデータの安全性を担保

● アルゴリズム

● DC表現の構築

- **難しさ**: 中間表現は1つのデータセットとして解析できない
(\because 関数 f が i に依存するため)
- **対策**: 中間表現を統合可能な「**DC表現**」に変換する

$$\tilde{X}_i \rightarrow \hat{X}_i = g_i(\tilde{X}_i)$$

行列の行ごとに作用する関数

- ✓理想的には、**任意**のデータに対して**一致**

$$g_i(f_i(X)) = g_j(f_j(X)) \text{ for } \textbf{any } X$$

- ✓現実的には、**特定の(アンカー)**データに対して**近似**

$$g_i(f_i(X^{anc})) \approx g_j(f_j(X^{anc})) \text{ for } \textbf{X}^{anc}$$

共有可能なデータ
(例:乱数行列)

● アルゴリズム

● DC表現の構築

➤ 関数 g の具体的な設定法

✓ 関数を線形とする: $\tilde{X}_i \rightarrow \hat{X}_i = g_i(\tilde{X}_i) = \tilde{X}_i G_i$

✓ 摂動最小化問題として求解

$$\min_{E_i, G_i, \|Z\|_F=1} \sum_i \|E_i\|_F^2 \quad s.t. \quad (\tilde{X}_i^{anc} + E_i)G_i = Z$$

→ 特異値分解に基づき計算

$$\tilde{X}_{1:c}^{anc} = [\tilde{X}_1^{anc}, \tilde{X}_2^{anc}, \dots, \tilde{X}_c^{anc}] = [U_1, U_2] \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \rightarrow G_i = (\tilde{X}_i^{anc})^\dagger U_1$$

アンカーデータの間接表現を並べた行列

● アルゴリズム

● DC表現を統合して解析

➤ DC表現を1つのデータセットとして解析

$$h(\hat{X}) \approx Y$$

ここで、

$$\hat{X} = \begin{bmatrix} \hat{X}_1 \\ \hat{X}_2 \\ \vdots \\ \hat{X}_c \end{bmatrix} \left(= \begin{bmatrix} f_1(X_1)G_1 \\ f_2(X_2)G_2 \\ \vdots \\ f_c(X_c)G_c \end{bmatrix} \right), Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_c \end{bmatrix}$$

✓ 任意の教師あり学習が利用可

— リッジ回帰、LASSO、ロジスティック回帰、DNNなど

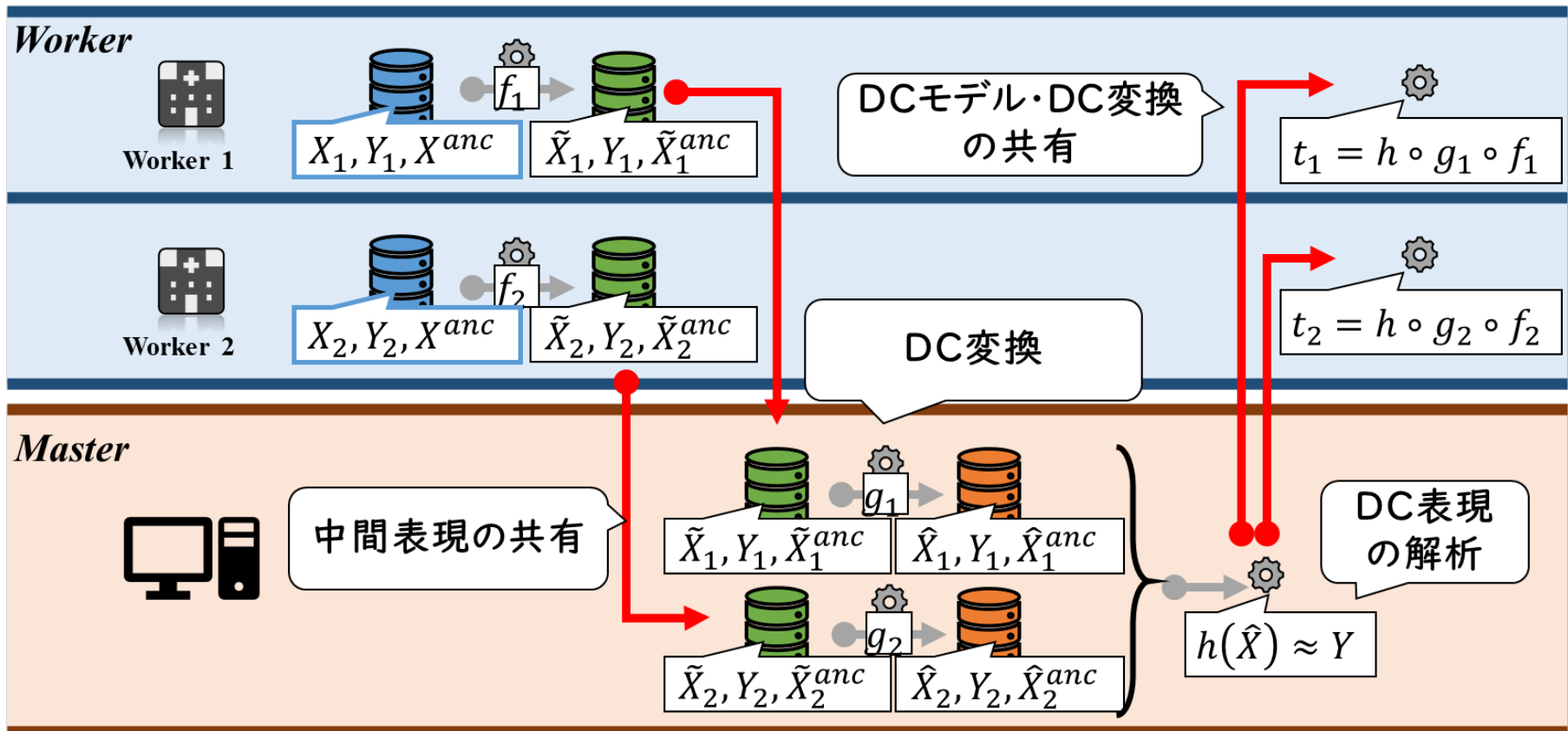
● 関数 g, h をユーザーに送付・ユーザーサイドでテスト

$$Y^{pred} = h(g_i(f_i(X^{test})))$$

アルゴリズム

全体像 (2機関の場合)

- ▶ 生データではなく、中間表現を共有
- ▶ 機関をまたいだ通信は2回のみ



実施例

● データセット

- batterysmall.mat [MATLAB Statistic and Machine Learning]
 - ✓ リチウムイオンバッテリーのセンサーデータからバッテリーの充電状態 (SOC) を予測するタスク
 - 説明変数: リチウムイオンバッテリーのセンサーデータ
 - » 電圧、電流、温度、平均電圧、平均電流
 - 目的変数: バッテリーの充電状態 (SOC)
 - 学習用サンプル数: 6,773
 - » 各機関が20サンプル保持
 - » 機関数: 1-10
 - テスト用サンプル数: 1,319

● 実施例

● 実験方法

- 学習データをランダムに変更して50回施行
- テストデータのRMSEの平均・95%信頼区間を評価
- 比較解法
 - ✓ 単独解析: 1機関20サンプルのみで学習
 - ✓ 集中解析: 生データを共有して学習
 - ✓ DC解析:
 - 中間表現生成: PCA→ランダム射影
 - アンカーデータ:
 - » 特徴量ごとの値域に合わせた一様乱数で生成
 - アンカーデータ数: 1,000

実施例

● 結果：中間表現の比較

➤ 機関1:

生データ

| 電圧 | 電流 | 温度 | 平均電圧 | 平均電流 |
|-------|-------|-------|-------|-------|
| 0.978 | 0.754 | 0.921 | 0.978 | 0.755 |
| 0.978 | 0.756 | 0.918 | 0.978 | 0.759 |
| 0.386 | 0.751 | 0.492 | 0.385 | 0.751 |
| 0.978 | 0.759 | 0.921 | 0.978 | 0.765 |
| 0.762 | 0.684 | 0.284 | 0.893 | 0.713 |



中間表現

| f1 | f2 | f3 | f4 |
|-------|--------|-------|-------|
| 1.808 | -0.741 | 1.070 | 1.321 |
| 1.812 | -0.739 | 1.073 | 1.324 |
| 1.376 | 0.013 | 1.024 | 0.948 |
| 1.819 | -0.737 | 1.080 | 1.328 |
| 1.556 | -0.547 | 0.845 | 1.228 |

中間表現は生データの特徴量をそのまま保存しない

➤ 機関2:

生データ

| 電圧 | 電流 | 温度 | 平均電圧 | 平均電流 |
|-------|-------|-------|-------|-------|
| 0.970 | 0.751 | 0.053 | 0.971 | 0.751 |
| 0.633 | 0.687 | 0.037 | 0.685 | 0.691 |
| 0.659 | 0.747 | 0.552 | 0.603 | 0.670 |
| 0.606 | 0.876 | 0.940 | 0.527 | 0.862 |
| 0.629 | 0.876 | 0.933 | 0.533 | 0.798 |



中間表現

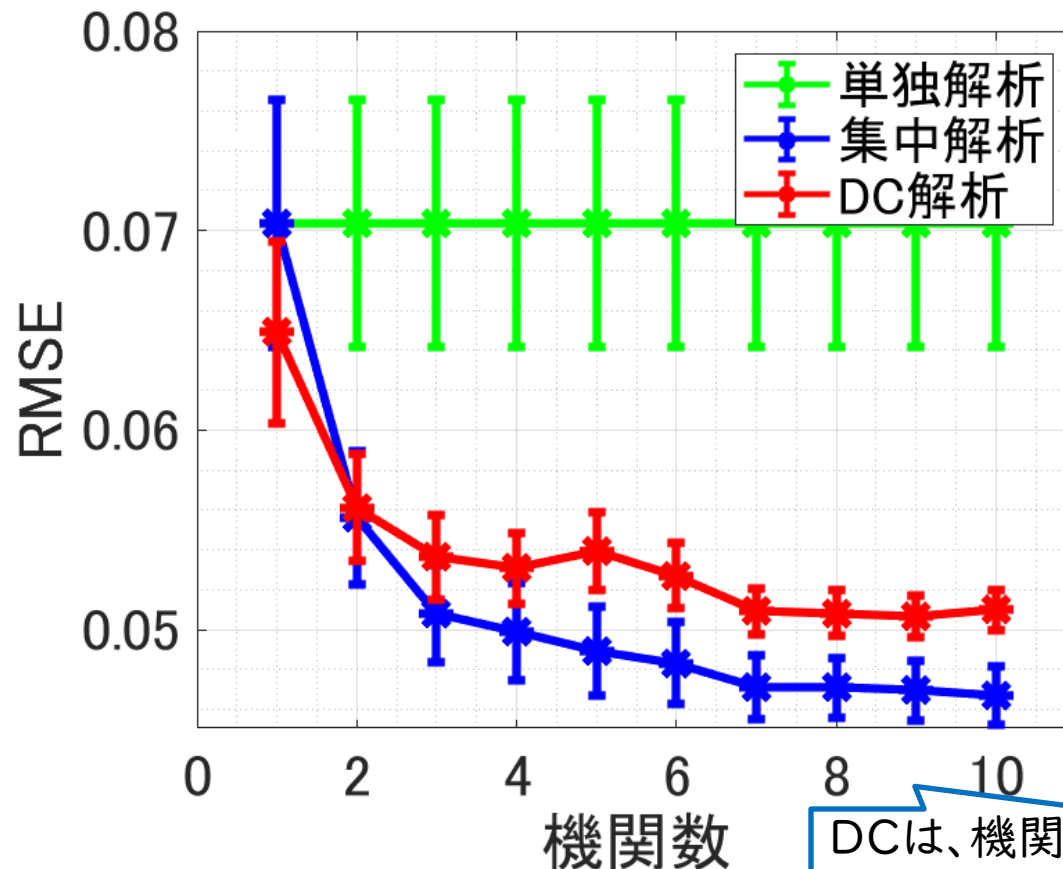
| F1 | F2 | F3 | F4 |
|--------|--------|-------|-------|
| -3.456 | -0.616 | 2.746 | 0.611 |
| -2.575 | -0.449 | 2.107 | 0.425 |
| -3.540 | -1.522 | 2.216 | 0.776 |
| -4.343 | -2.268 | 2.506 | 0.978 |
| -4.325 | -2.292 | 2.438 | 1.005 |

機関ごとに値域も異なる

実施例

● 結果：解析精度の比較

➤ RMSE (値が小さい方が精度がよい)



DCは、機関数の増加に伴って、高い性能を発揮

まとめ



● まとめ

● 行列計算による機械学習について紹介した

➤ 入門: 行列計算に基づく手法

✓ 行列計算に基づく基礎的な機械学習法を紹介

— 2乗誤差の最小化に基づく手法

— 行列トレースの最適化に基づく次元削減法

✓ これらのアルゴリズムの実装の際に行列計算の観点から「行列積ベースの実装の重要性」を紹介

➤ 発展: プライバシー保護機械学習

✓ データコラボレーション解析の概要について紹介