2020.02.07 2019年度修士論文審査会

非負行列因子分解の学習による高速化

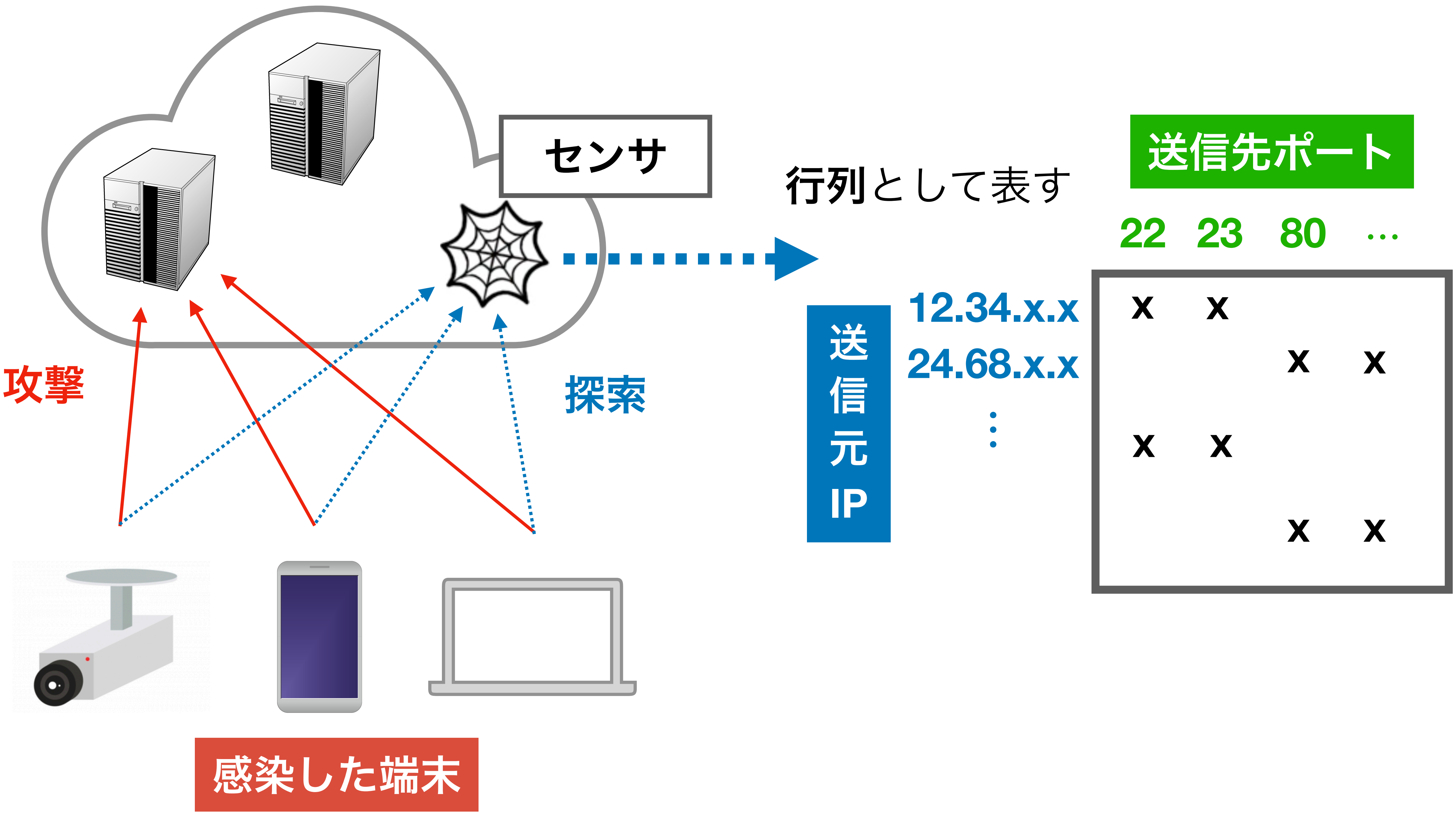
行列分解・テンソル分解の学習による高速化

村田研究室

5318E032　金原秀明

サイバー攻撃の予兆を検知したい

感染を拡大させるための**探索行動**を捉える

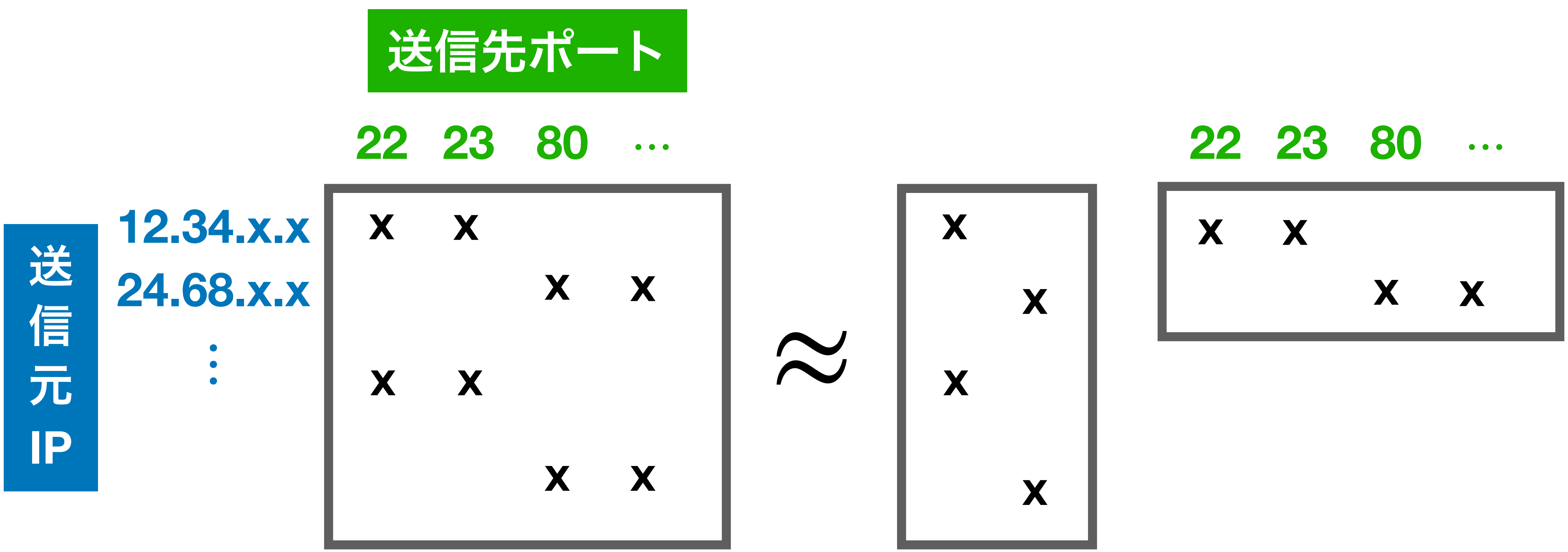


背景 | 非負行列因子分解の大規模な問題に対する応用

サイバー攻撃の予兆を検知したい

感染を拡大させるための**探索行動**を捉える

非負行列因子分解 (NMF) を適用すると通信パターンの傾向がわかる

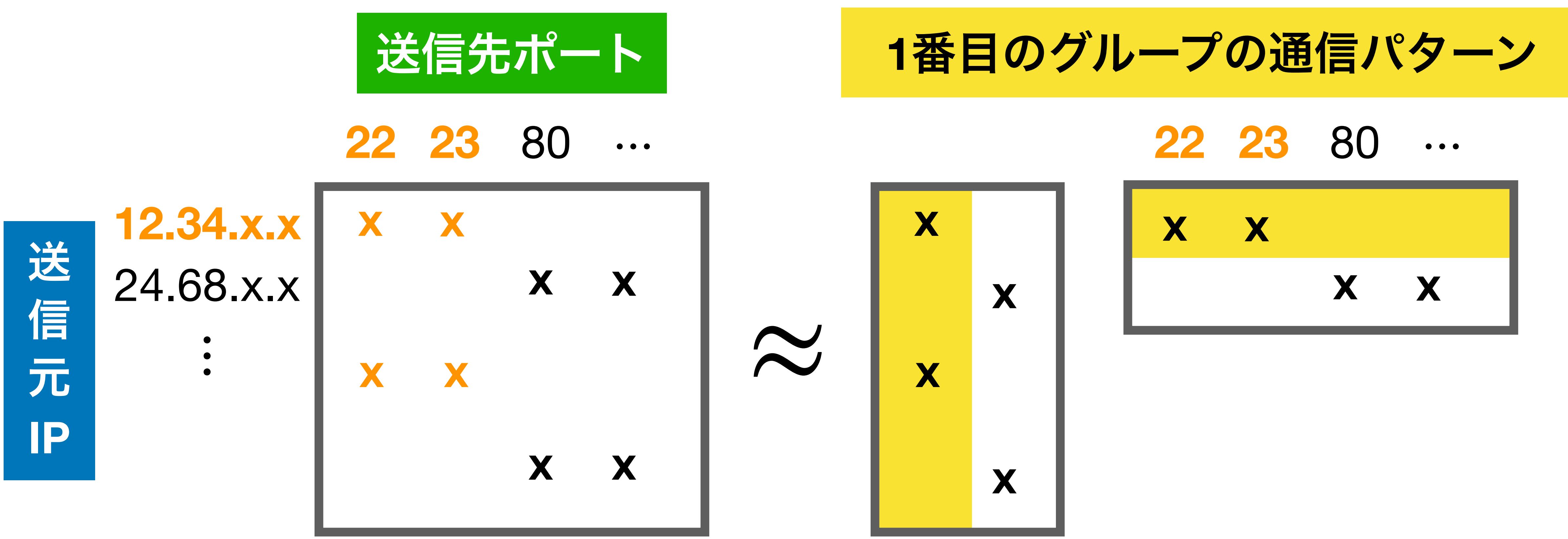


背景 | 非負行列因子分解の大規模な問題に対する応用

サイバー攻撃の予兆を検知したい

感染を拡大させるための**探索行動**を捉える

非負行列因子分解 (NMF) を適用すると**通信パターンの傾向**がわかる

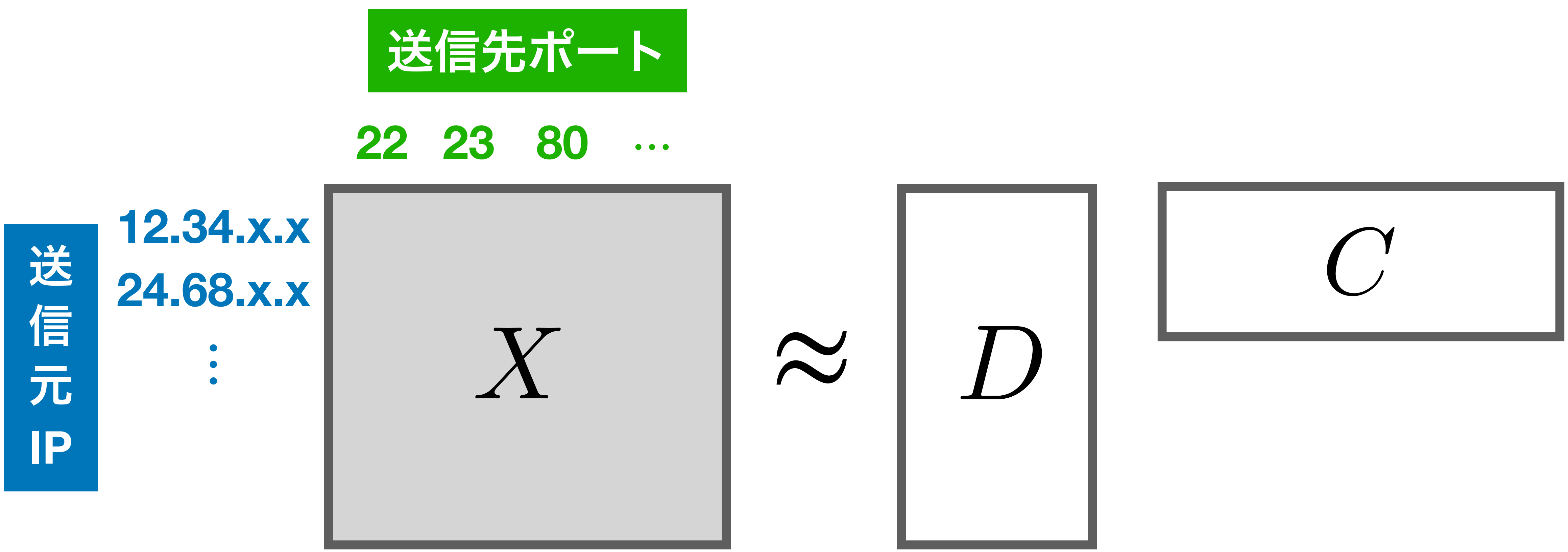


背景 | 非負行列因子分解の大規模な問題に対する応用

サイバー攻撃の予兆を検知したい

感染を拡大させるための**探索行動**を捉える

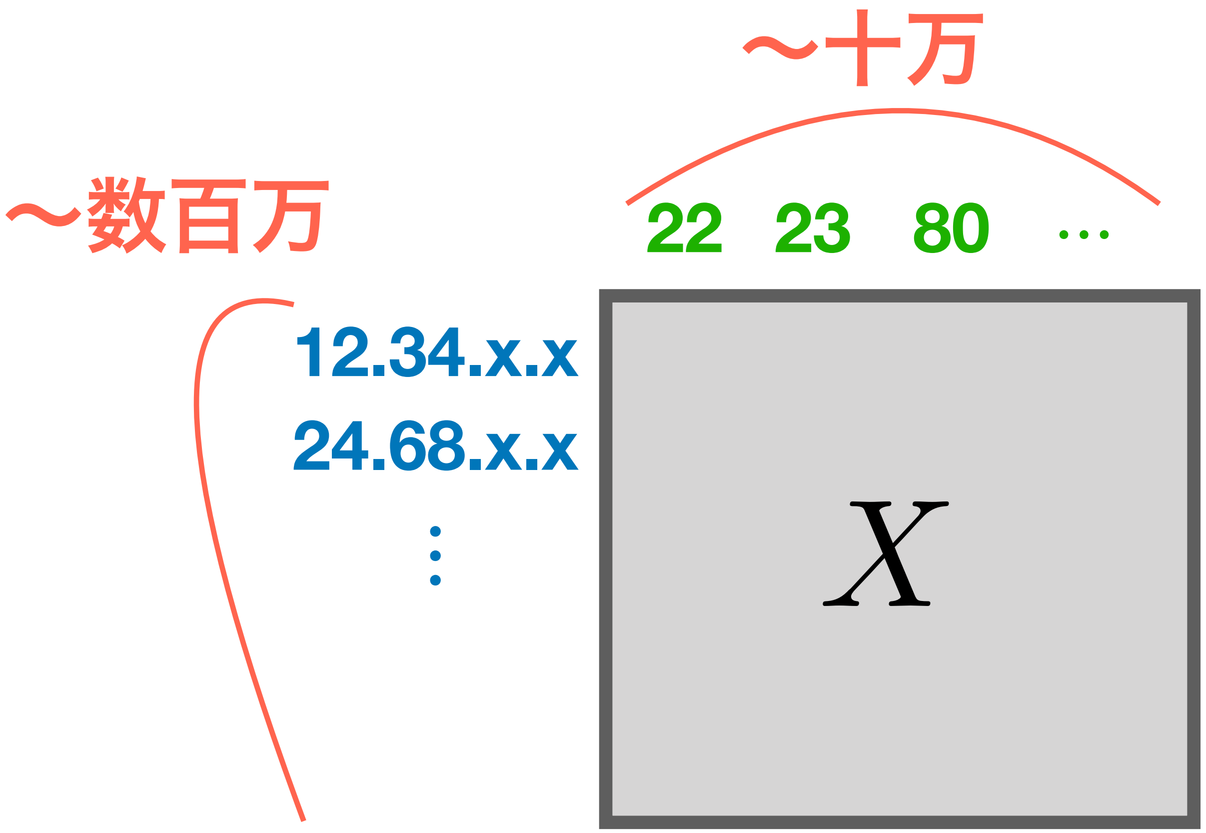
非負行列因子分解 (NMF) を適用すると通信パターンの傾向がわかる



データ行列Xをできるだけ表現するようなDとCを求める

背景 | 非負行列因子分解の大規模な問題に対する応用

非負行列因子分解 (NMF) を適用すると通信パターンの傾向がわかる



問題点

- リアルタイムで**高速に分解**したい
- 行列が**大規模で全て扱うことが不可能**

更新回数を削減することに注目

研究概要

NMFの学習による高速化を行った

- 行列分解を学習によって高速化した例がない
- 更新に学習器を用いることで必要な更新回数を削減

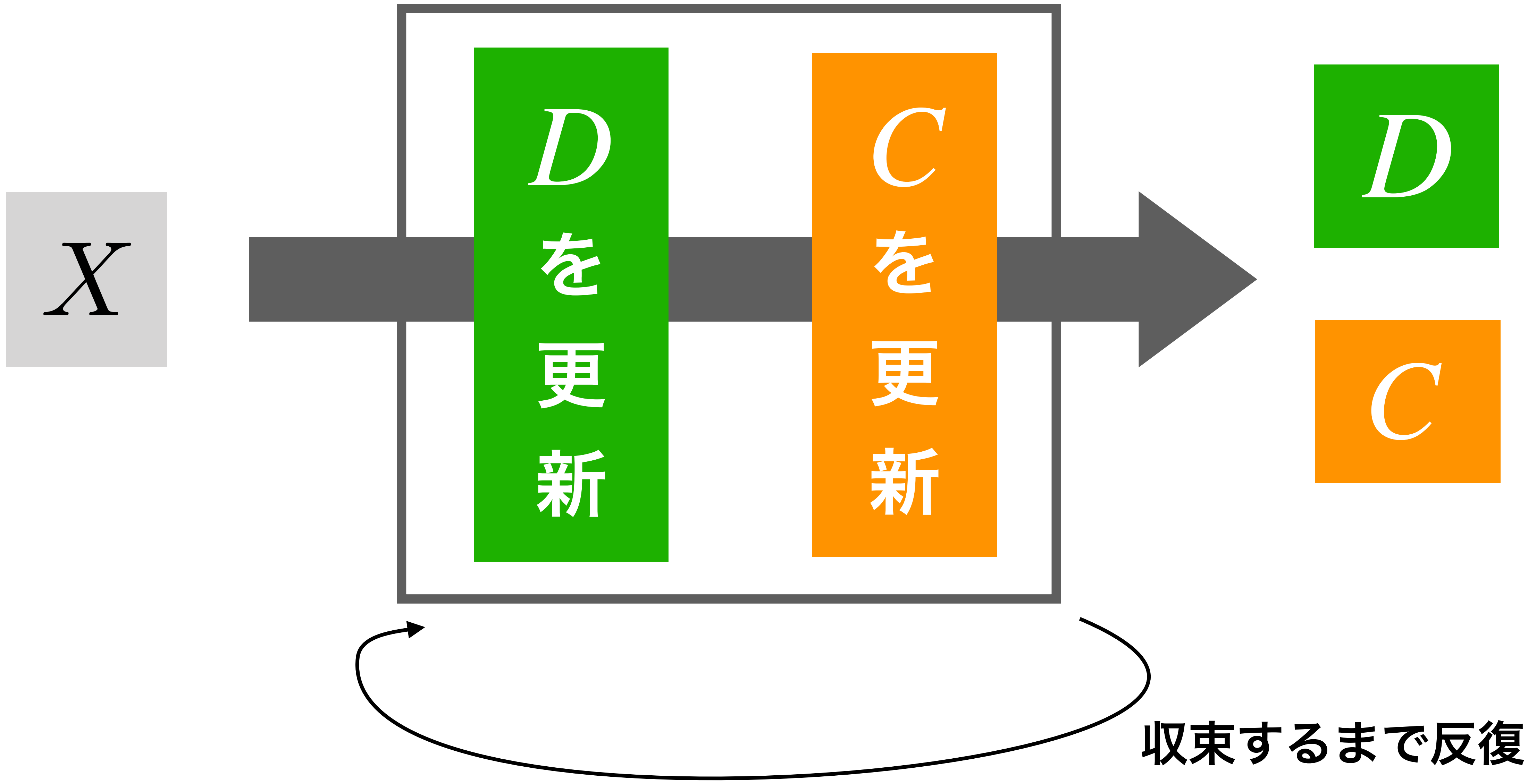
行列が大規模でも効率的に計算を行う実装

- 学習器への入力を工夫して行列を部分的に更新するように
- サンプリングによって行列サイズが可変でも同じモデルで分解可能

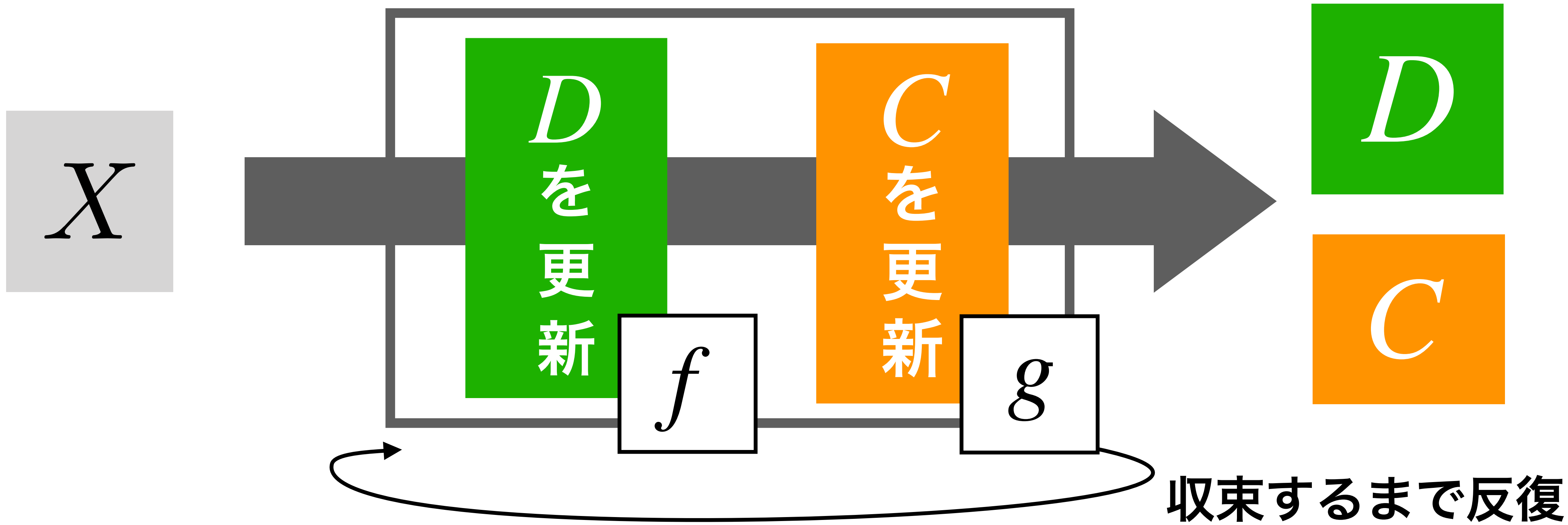
NMFのアルゴリズム

データ行列Xをできるだけ表現するようなDとCを求める

- **交互にD, Cを最適化する**
- **反復計算を何度も行う**



NMFのアルゴリズム | 乗法更新則 (MU)



Dを更新する関数の例

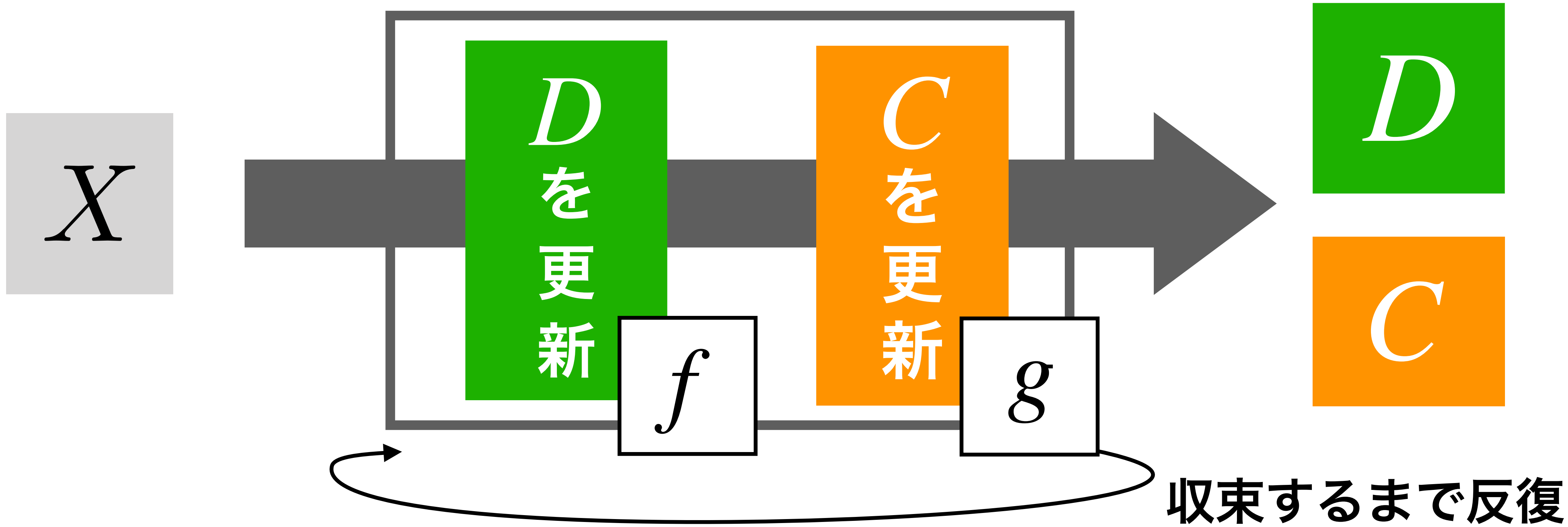
- 乗法更新則 (MU)

$$f(X, D, C) = D \circledast \frac{XC^T}{DCC^T}$$

パラメータの設定が不要だが収束が遅い

$\circledast$：要素積

NMFのアルゴリズム | 勾配降下法



Dを更新する関数の例

- 勾配降下法

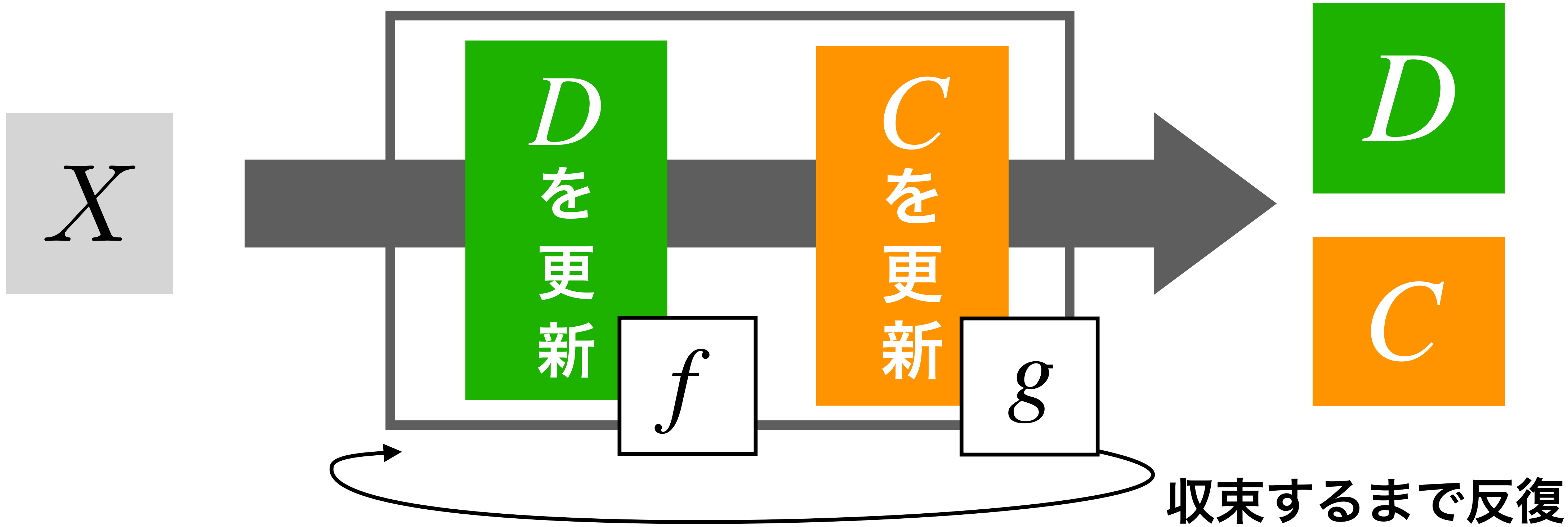
$$f(X, D, C) = [D - \eta \nabla_D \mathcal{L}]_+$$

学習率ηを決める必要あり

損失関数 $\mathcal{L}_{\mathrm{euc}} = \frac{1}{2}||X - DC||_F^2$

$[\,\cdot\,]_+$: 負の要素を0にする

NMFのアルゴリズム | 提案手法 (LNMF)



Dを更新する関数の例

- 提案手法 (Learned NMF; LNMF)

$$f(X, D, C) = [D - u(\nabla_D \mathcal{L}, X)]_+$$

学習器によって更新分を推定

u は中間層1層のニューラルネット

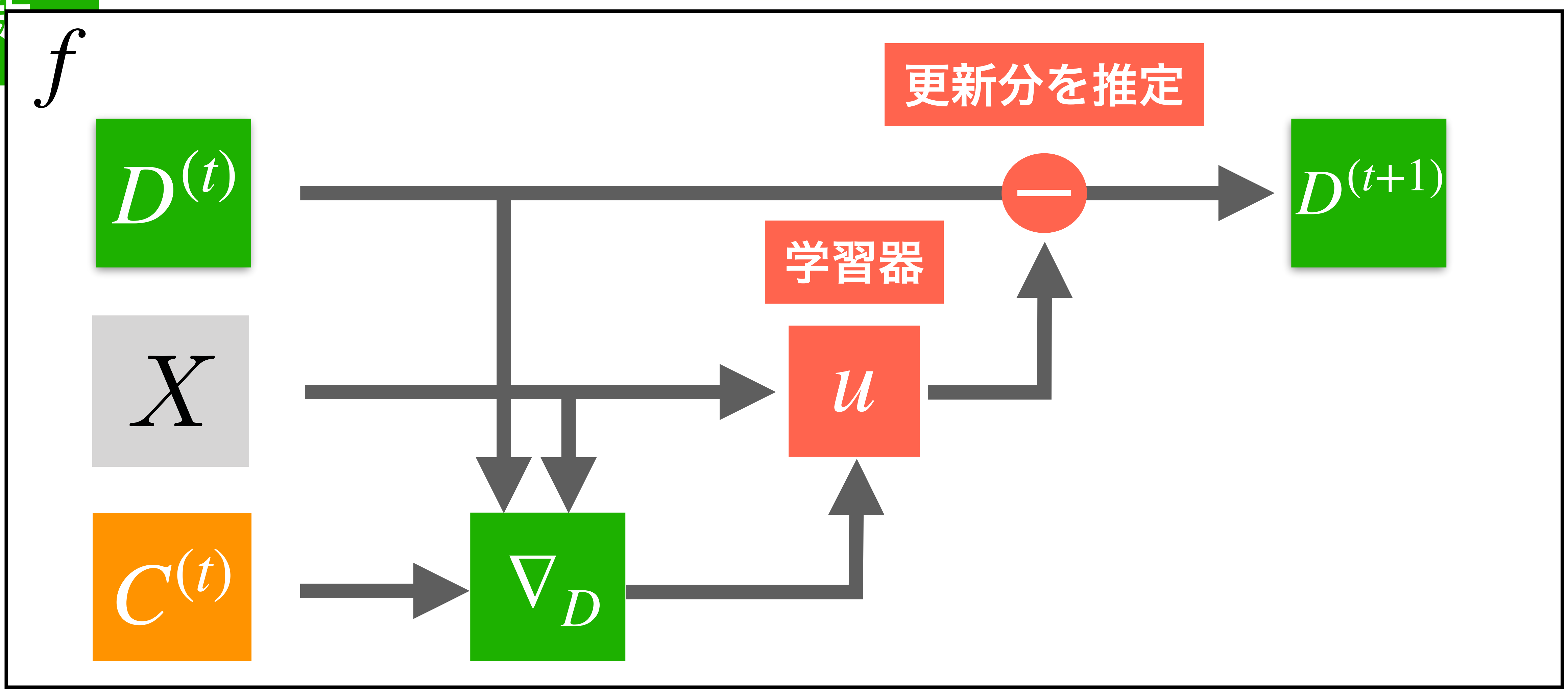
$$u(x) = W_2 \tanh(W_1 x)$$

提案手法 (LNMF) によるDの更新

- 学習器によって更新分を推定

Dを更新

$$f(X, D, C) = [D - u(\nabla_D \mathcal{L}, X)]_+$$

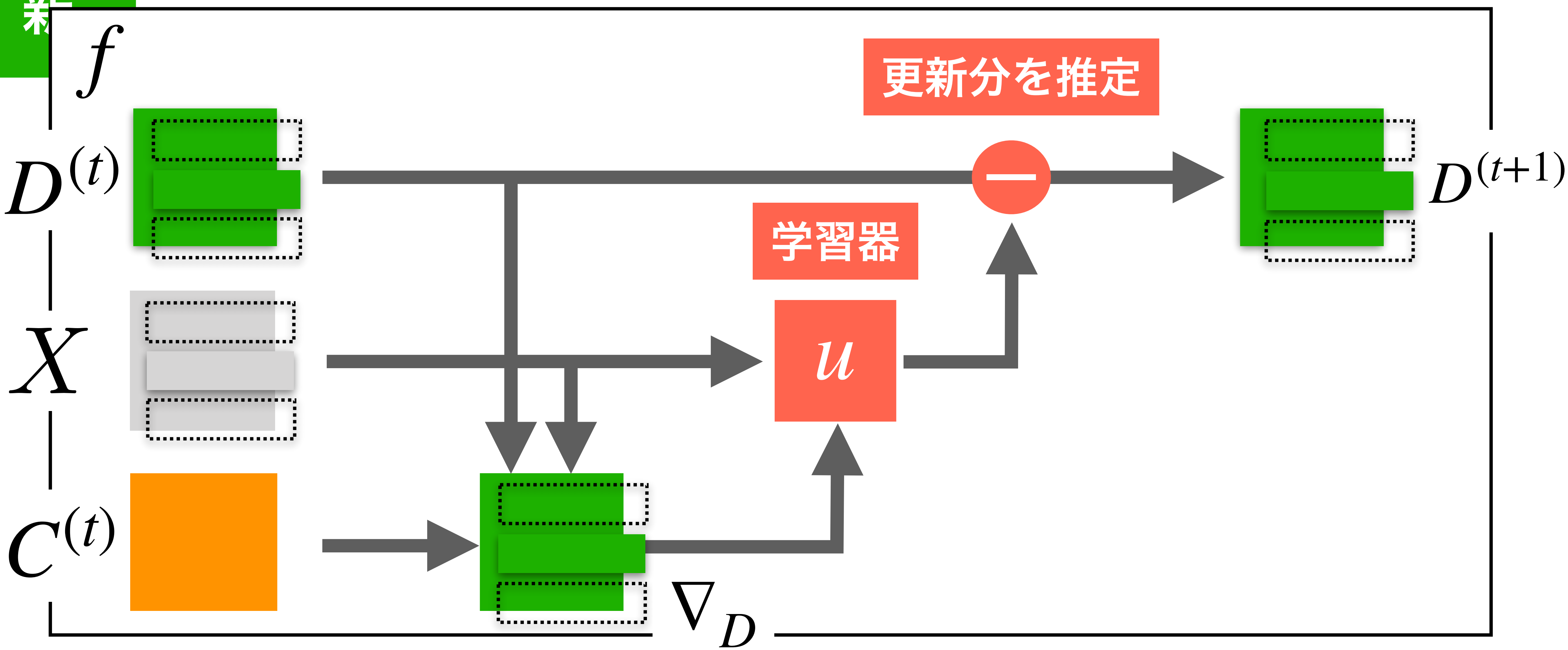


提案手法 (LNMF) によるDの更新

- 学習器によって更新分を推定
- **ベクトル毎に計算する**

Dを更新

$$f(X, D, C) = [D - u(\nabla_D \mathcal{L}, X)]_+$$

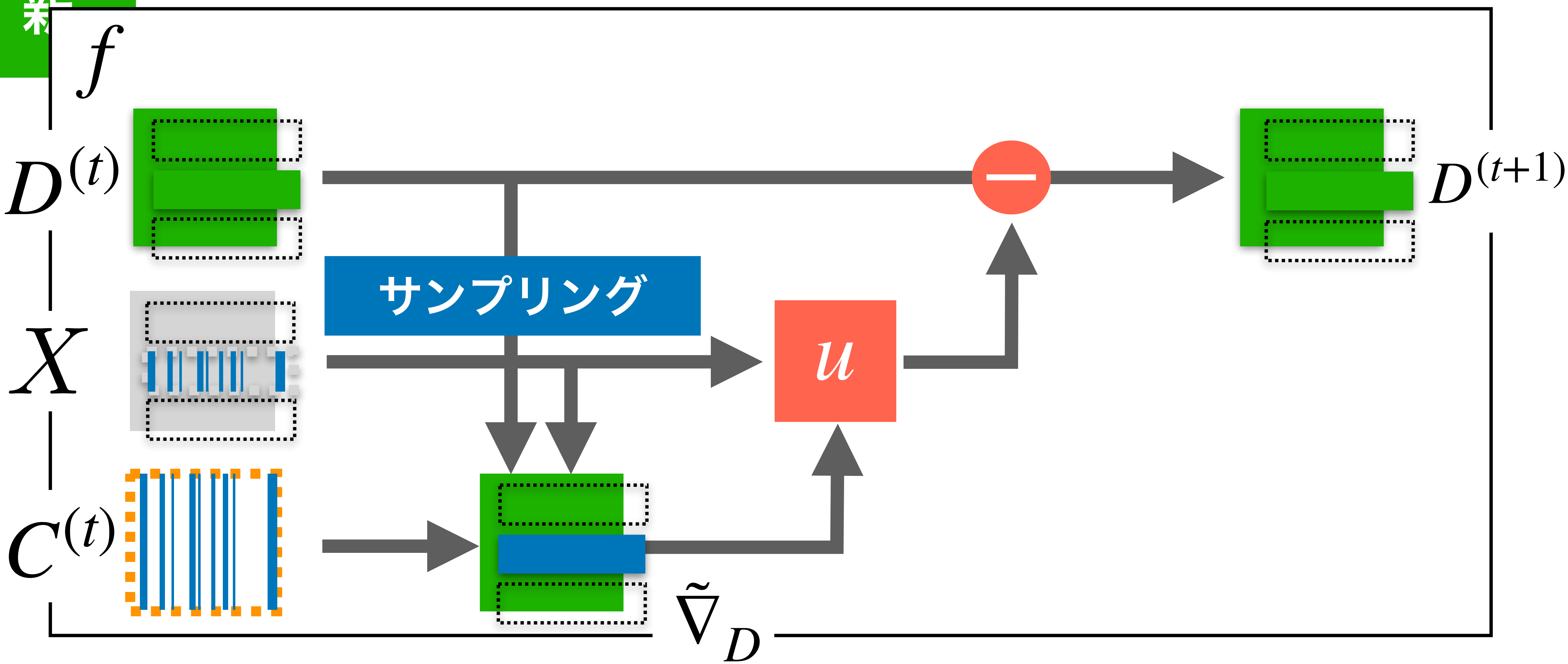


提案手法 (LNMF) によるDの更新

- 学習器によって更新分を推定
- ベクトル毎に計算する
- **サンプリング結果を用いて近似計算**

Dを更新

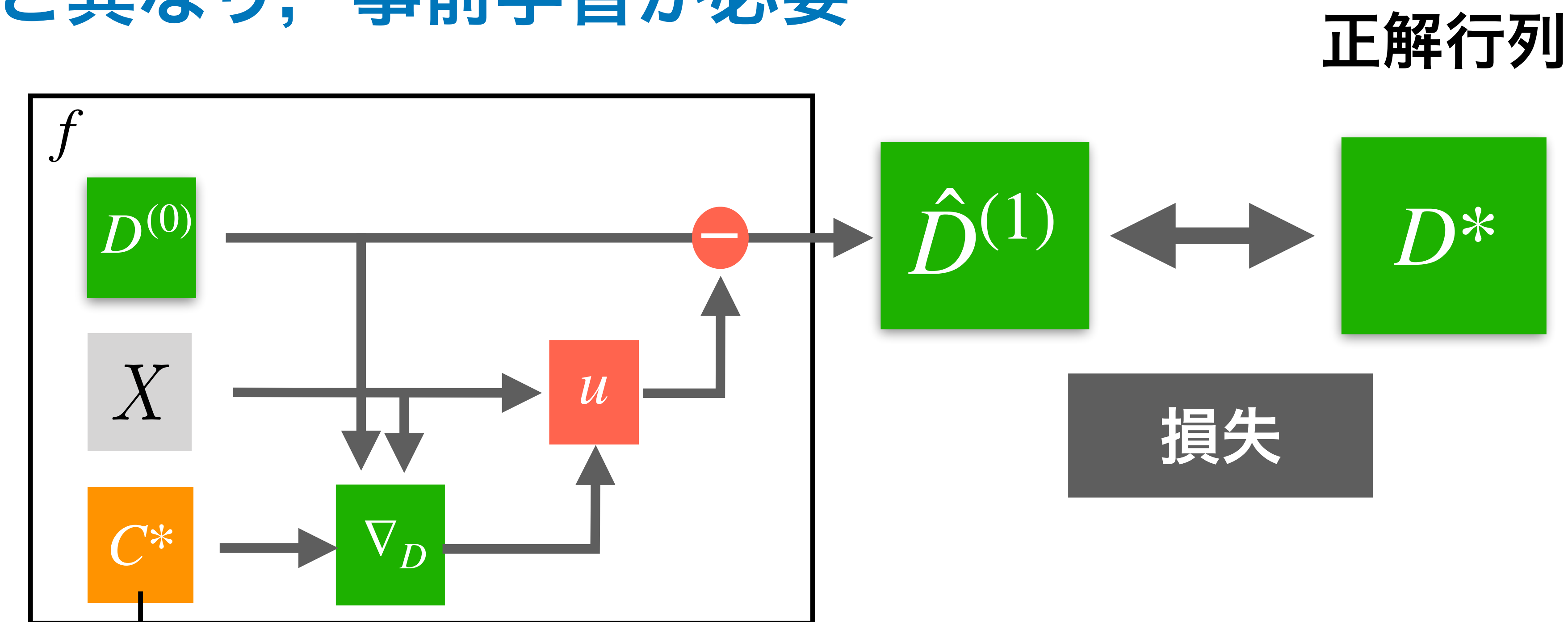
$$f(X, D, C) = [D - u(\nabla_D \mathcal{L}, X)]_+$$



提案手法の学習方法

通常のNMFと異なり，事前学習が必要

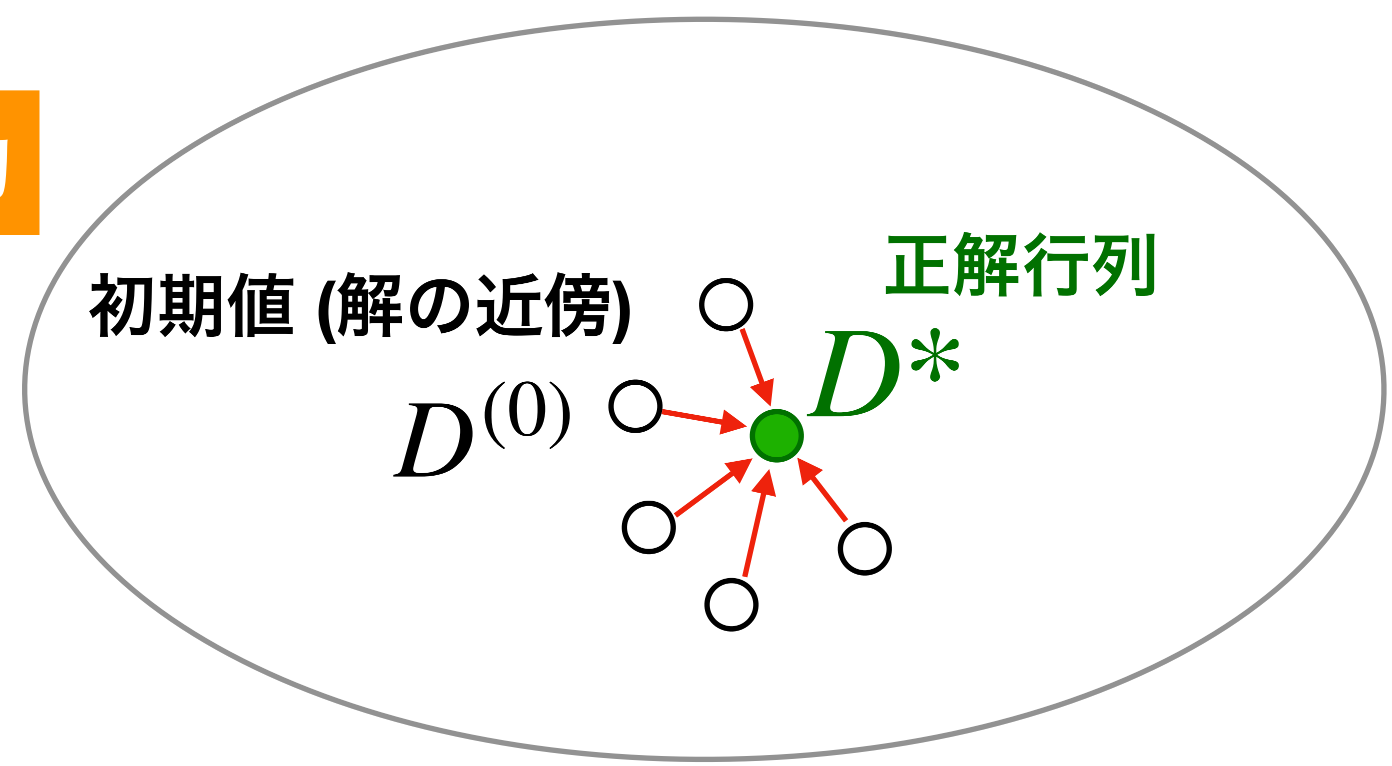
学習時



(Dの更新のみ)

Cは正解行列を入力

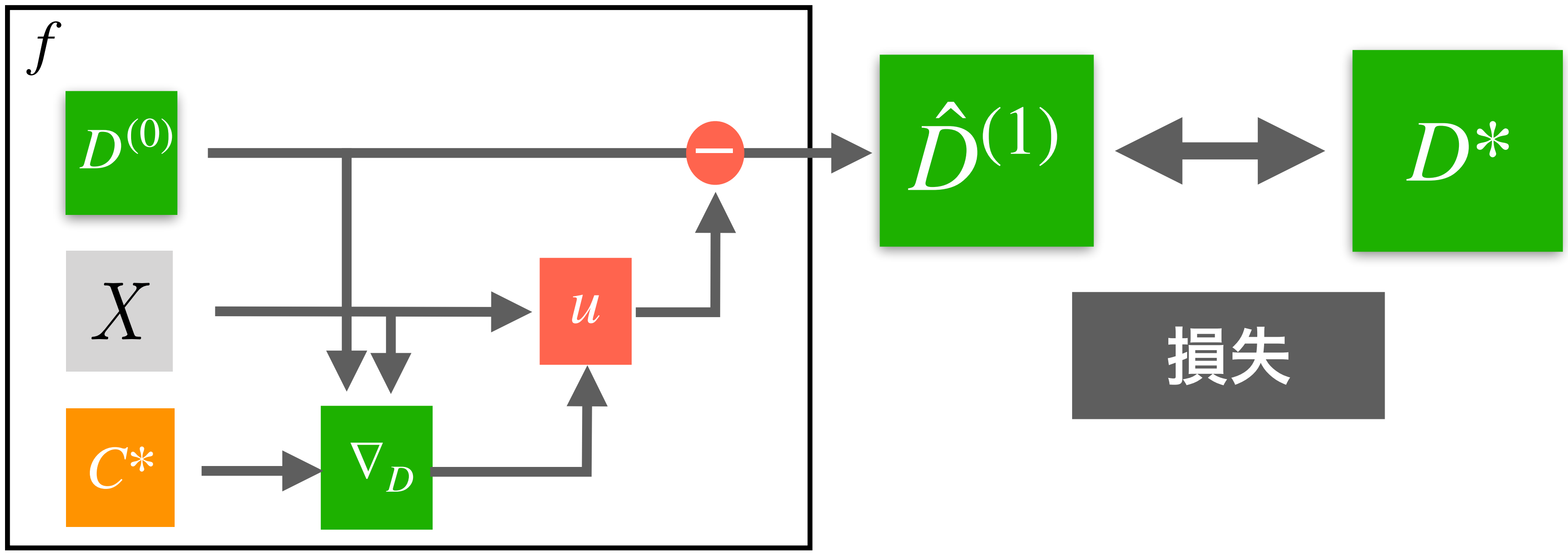
解近傍からの更新方向を学習



提案手法の学習時と推論時

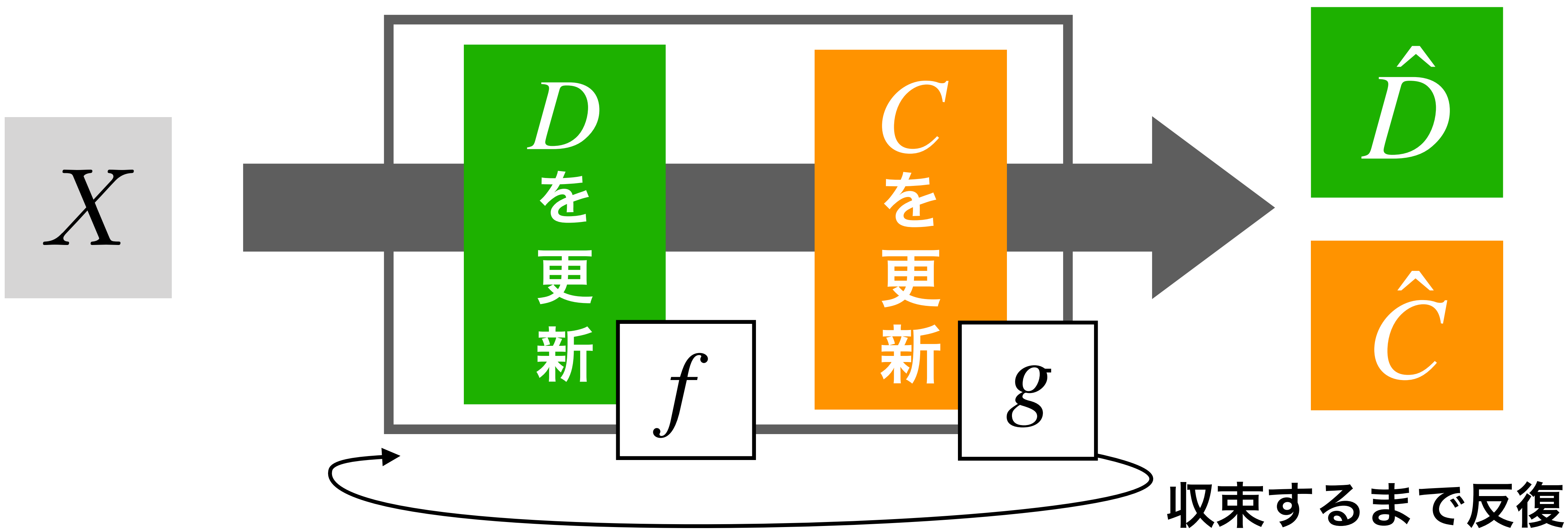
推論時は通常のNMFのアルゴリズムと同様に反復計算を行う

学習時



(Dの更新のみ)

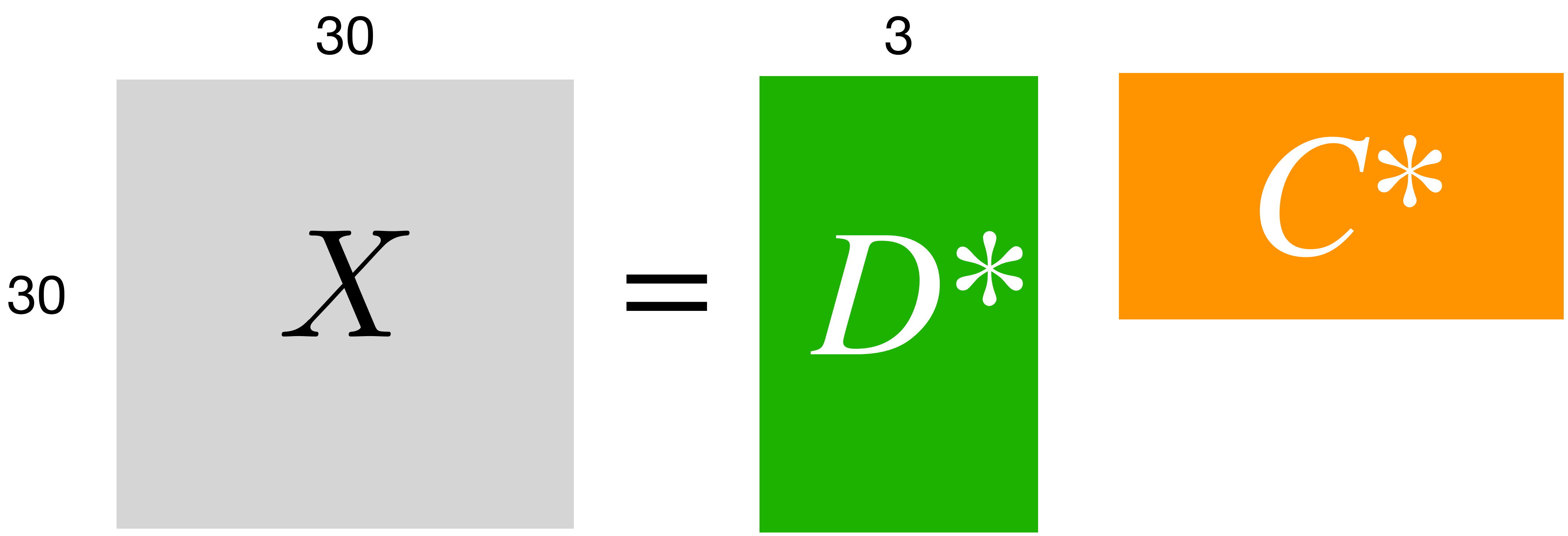
推論時



実験設定 | LNMFの学習

- 人工データ実験 (以下のセットを1つのサンプルとする)
  - 正解行列 D^*, C^*を事前に生成 $d^*_{ij},\ c^*_{ij} \sim \mathcal{U}(0,1)$
  - $X = D^*C^*$

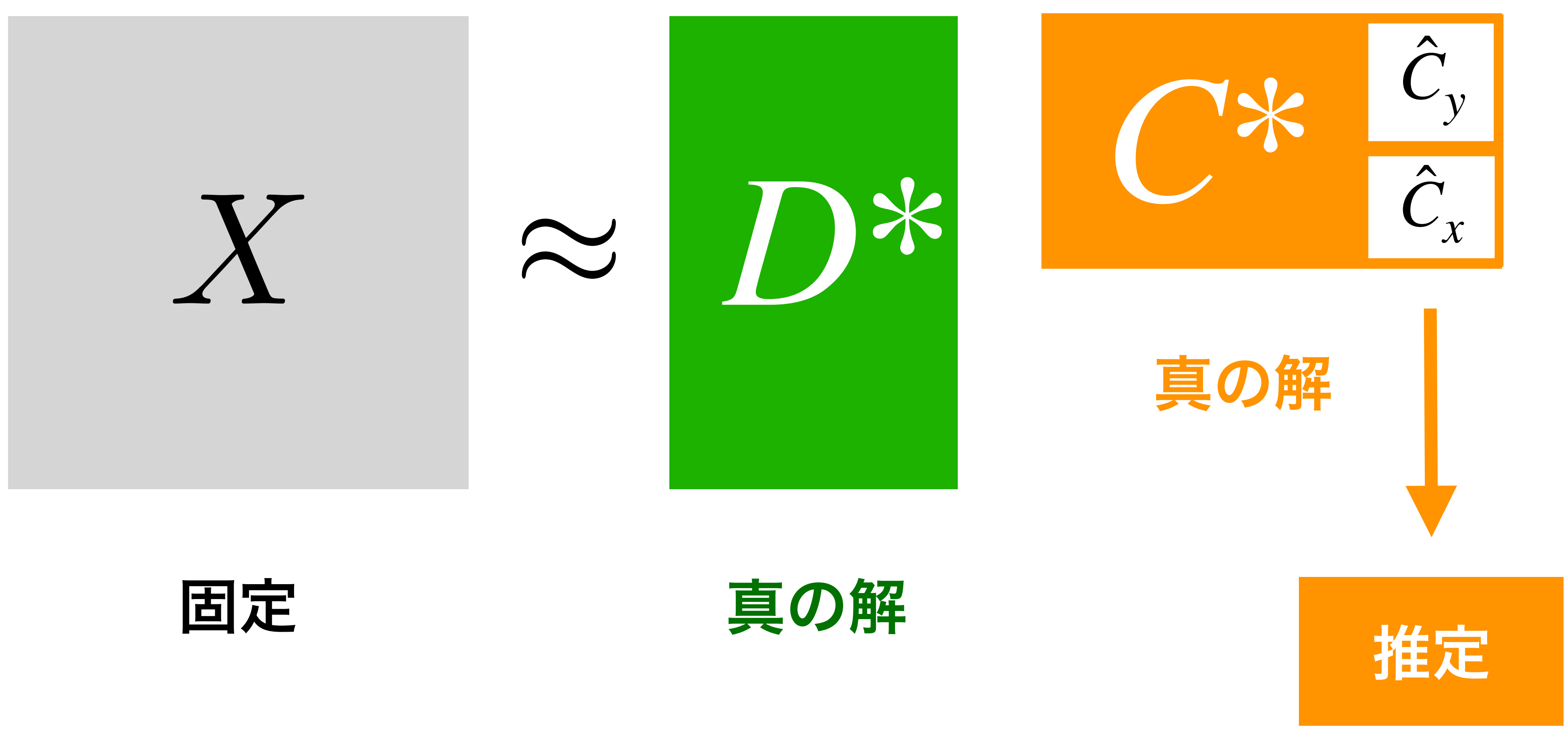
学習データ数：1500万



実験結果 | 解近傍での収束状況

収束状況を可視化するために2要素のみ推定を行う

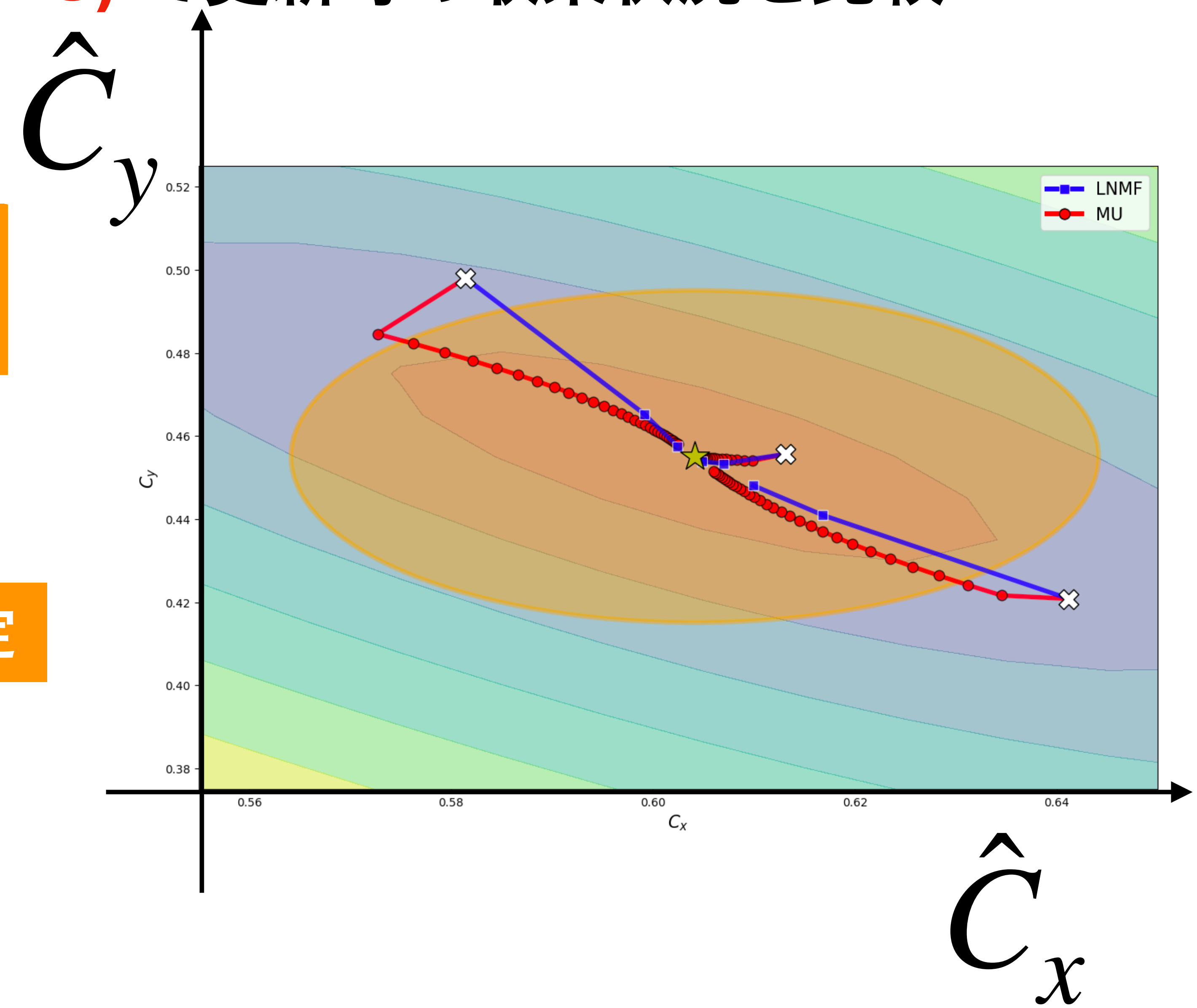
提案手法 (LNMF) と既存手法 (MU) で更新毎の収束状況を比較



実験結果 | 解近傍での収束状況

収束状況を可視化するために2要素のみ推定を行う

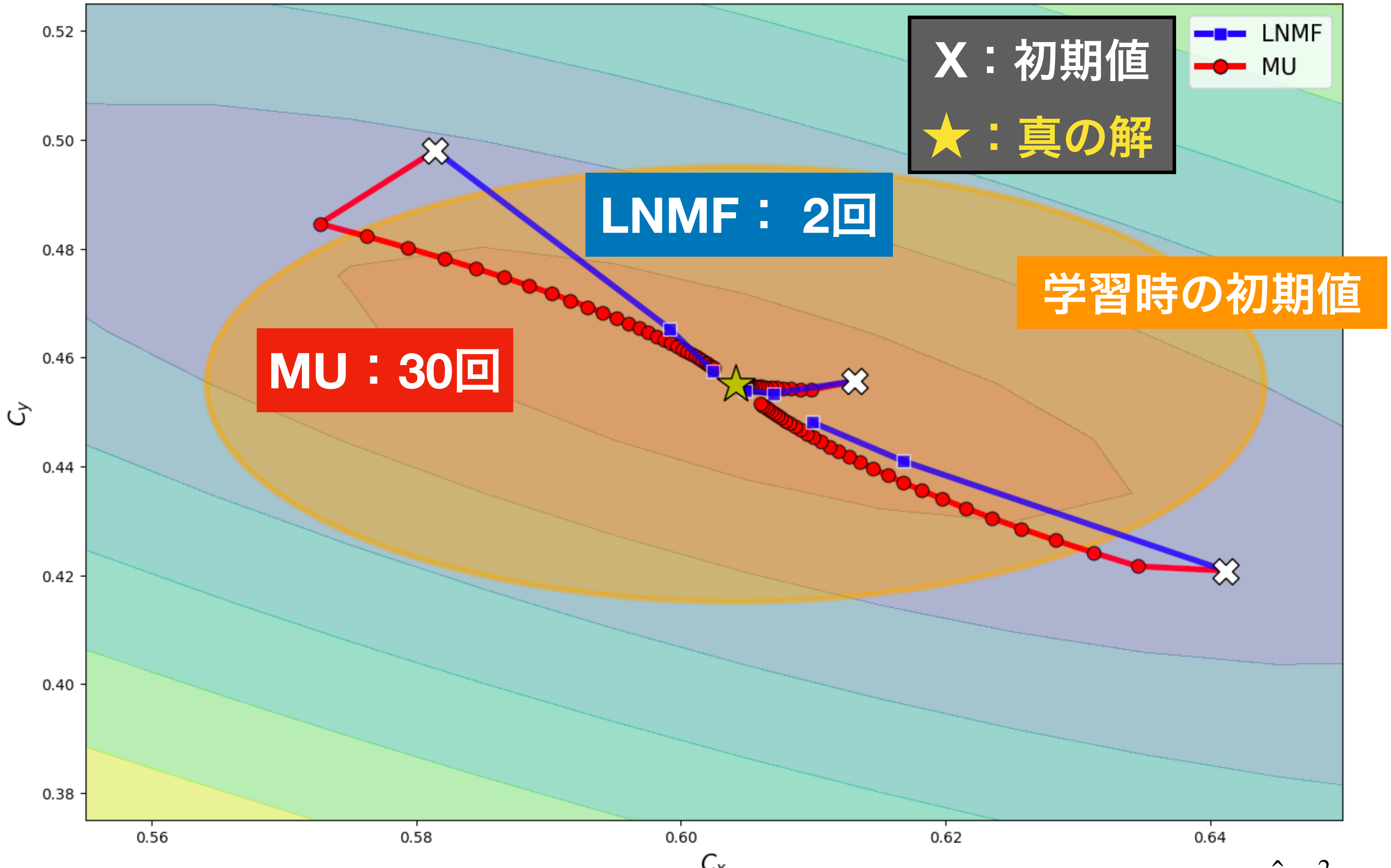
提案手法 (LNMF) と既存手法 (MU) で更新毎の収束状況を比較



実験結果 | 解近傍での収束状況

解近傍に初期値を設定したとき (学習時と同じ設定)

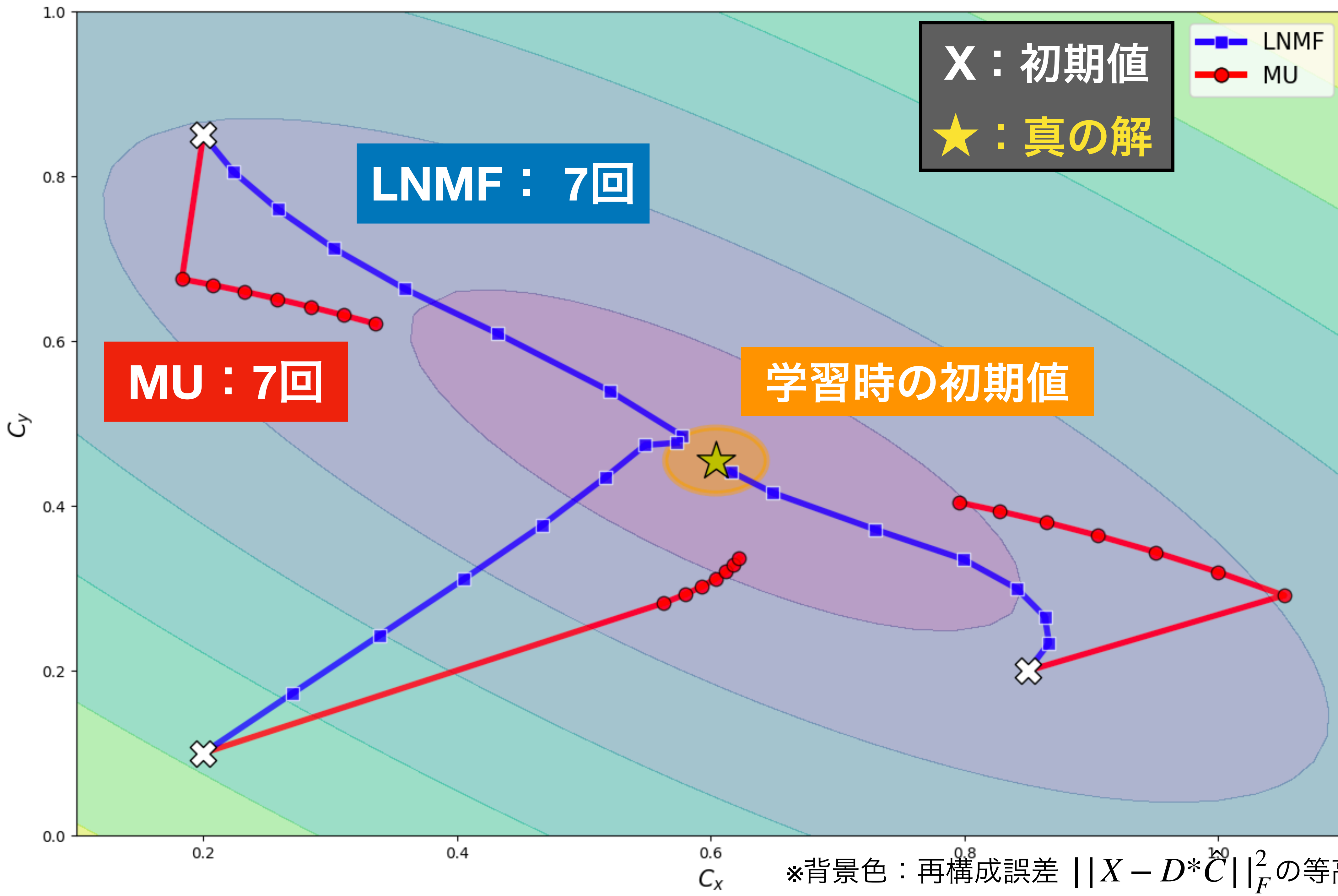
- MUよりも大きい更新幅，良い更新方向になっている



※背景色：再構成誤差 $||X - D^*\hat{C}||_F^2$ の等高線プロット

実験結果 | ランダムな初期値からの収束状況

ランダムな初期値から更新を行ってもうまく更新が行えた

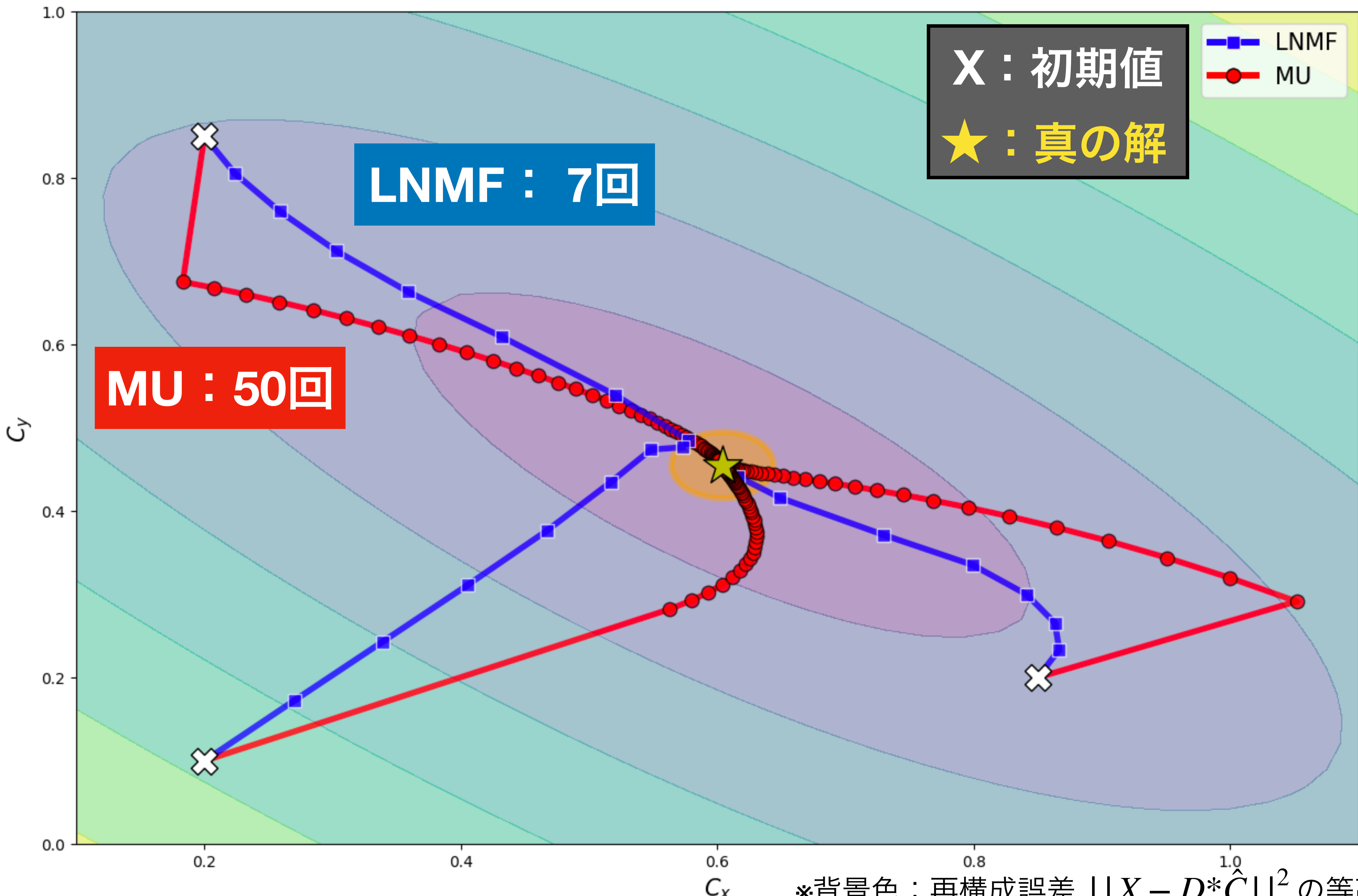


※背景色：再構成誤差 $||X - D^*C||_F^2$ の等高線プロット

実験結果 | ランダムな初期値からの収束状況

ランダムな初期値から更新を行ってもうまく更新が行えた

→定量的に比較

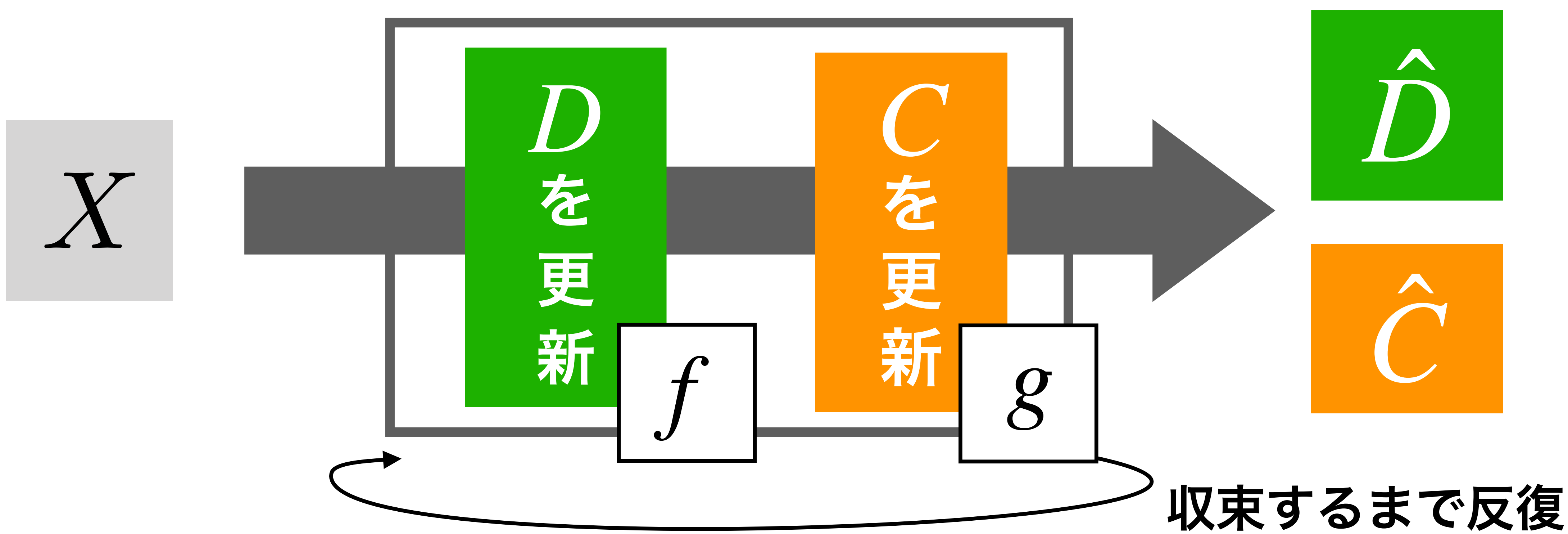


※背景色：再構成誤差 $||X - D^{*}\hat{C}||_F^2$ の等高線プロット

高速化の検証実験

- LNMF (提案モデル) を収束するまで更新して以下を比較
  - そのときの再構成誤差をMU (比較手法) が達成する更新回数
  - LNMF, MUそれぞれ1回の更新あたりの計算時間

検証データ数：500



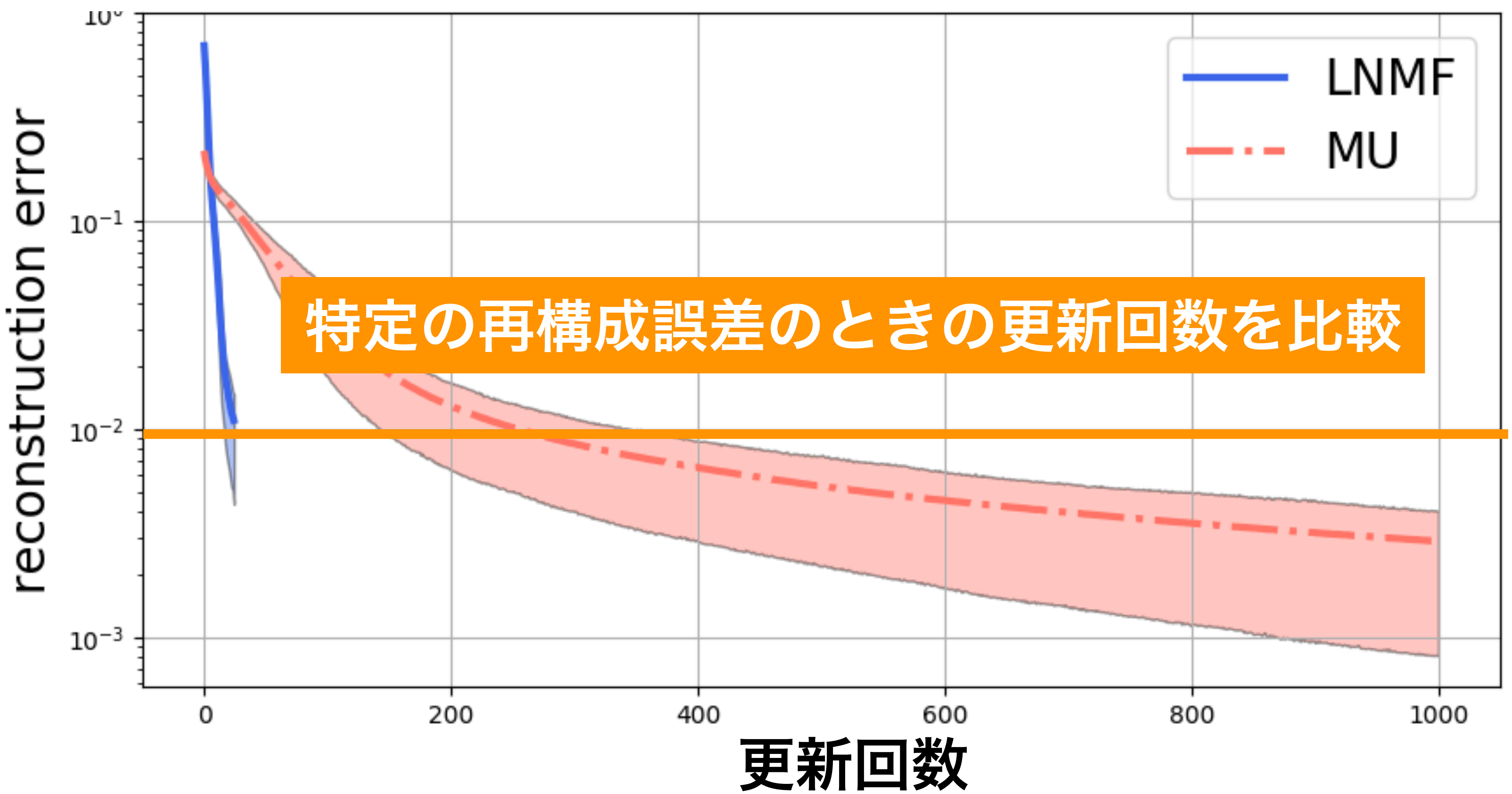
提案手法 vs. MU | 高速化の検証

LNMF (提案手法) と **MU (乗法更新則)** の比較

更新毎の再構成誤差

再構成誤差

$$\frac{||X - \hat{D}\hat{C}||_F^2}{||X||_F^2}$$



提案手法 vs. MU | 高速化の検証

MU (乗法更新則) と比べて

手法	更新回数	計算時間
LNMF	27.43 ± 11.56	1483.70 ± 243.23
MU	384.80 ± 317.89	771.03 ± 69.68

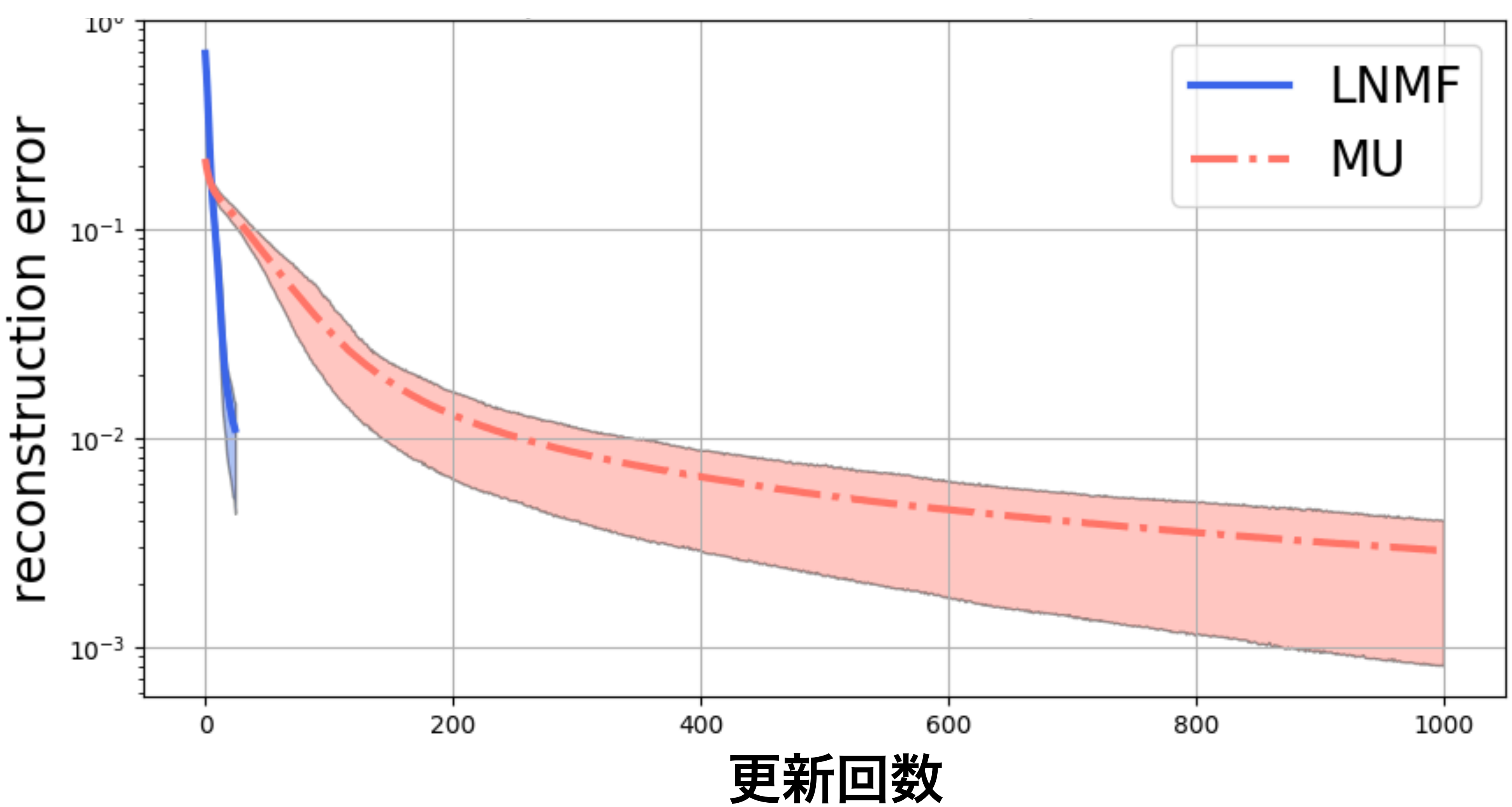
更新回数 **平均 1/14倍**

実測値 **平均7倍**

の高速化

更新毎の再構成誤差

$$\frac{||X - \hat{D}\hat{C}||_F^2}{||X||_F^2}$$



総括

まとめ

- NMFの学習による高速化手法の提案
- 既存のアルゴリズムと比較して約7倍の高速化を達成
- 収束状況の可視化
  - 解近傍での収束が早く，全体的な高速化につながった

今後の課題

- 提案手法の解析・理由付け
- 実データ実験
  - 教師となる正解行列：既存のアルゴリズムを十分に回したもの

発表まとめ

背景

- NMFの更新回数を減らしたい
- 大規模な行列でも効率的に計算したい

手法

- NMFの更新分を学習器によって推定
- 正解行列を用意しその近傍で学習
- 行列サイズが大きいときはサンプリング

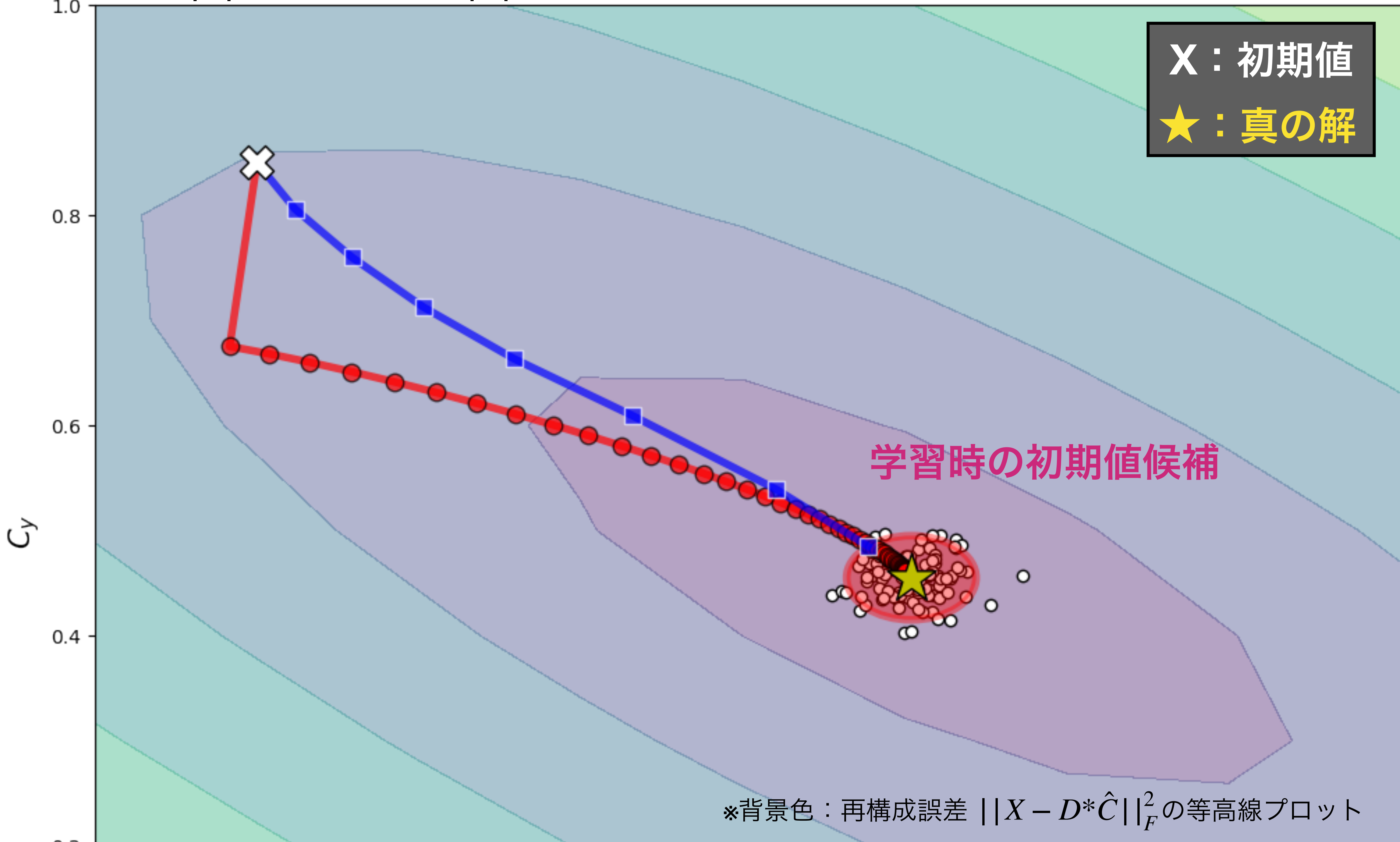
実験

- D^*, C^* を一様分布から生成しXを作成
- MUと比較して約7倍の高速化
- 収束状況の可視化を行った．MUよりも直線的なフローを獲得している

APPENDICES

LNMF vs. MU | 収束状況の可視化

- 初期値を同じにして更新過程を比較
- LNMF：7回 / MU：50回

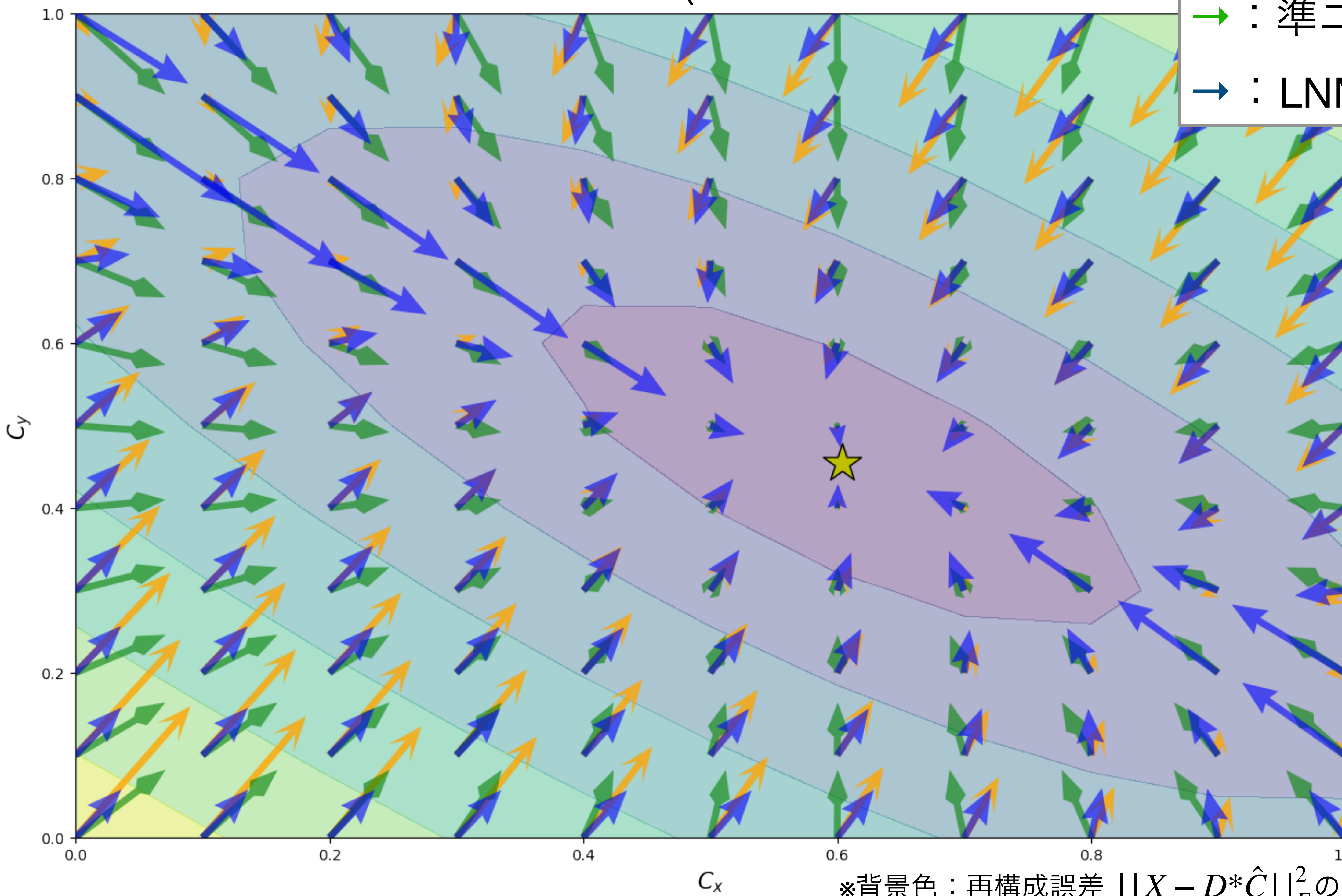


※背景色：再構成誤差 $||X - D^{*}\hat{C}||_F^2$ の等高線プロット

LNMF vs. MU | 収束状況の可視化

- 勾配とニュートン法，及びLNMFの更新フローの比較
- ベクトルの長さは更新幅を表す (勾配法の学習率は1とする

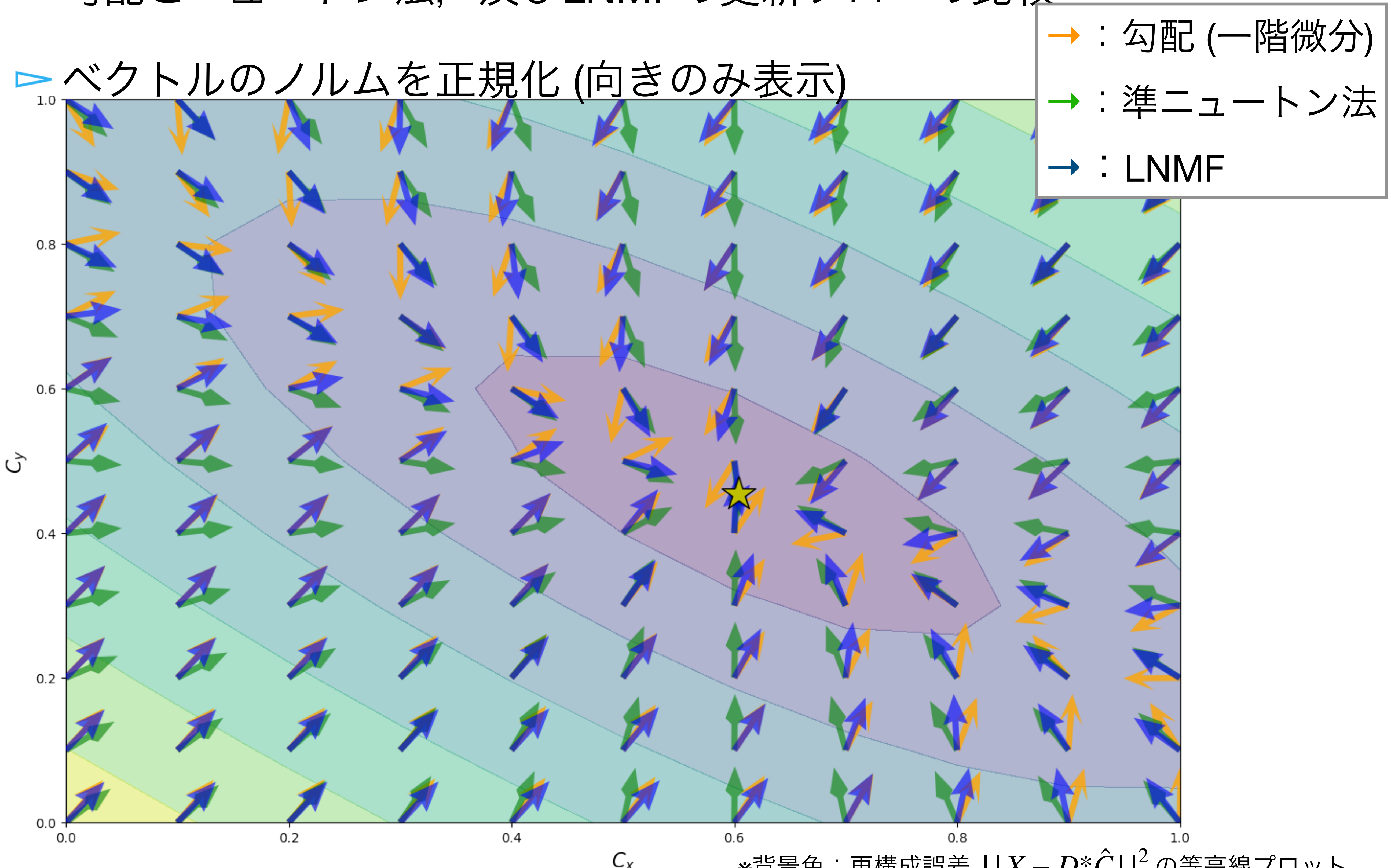
→ ：勾配 (一階微分)
→ ：準ニュートン法
→ ：LNMF



※背景色：再構成誤差 $||X - D^*\hat{C}||_F^2$ の等高線プロット

LNMF vs. MU | 収束状況の可視化

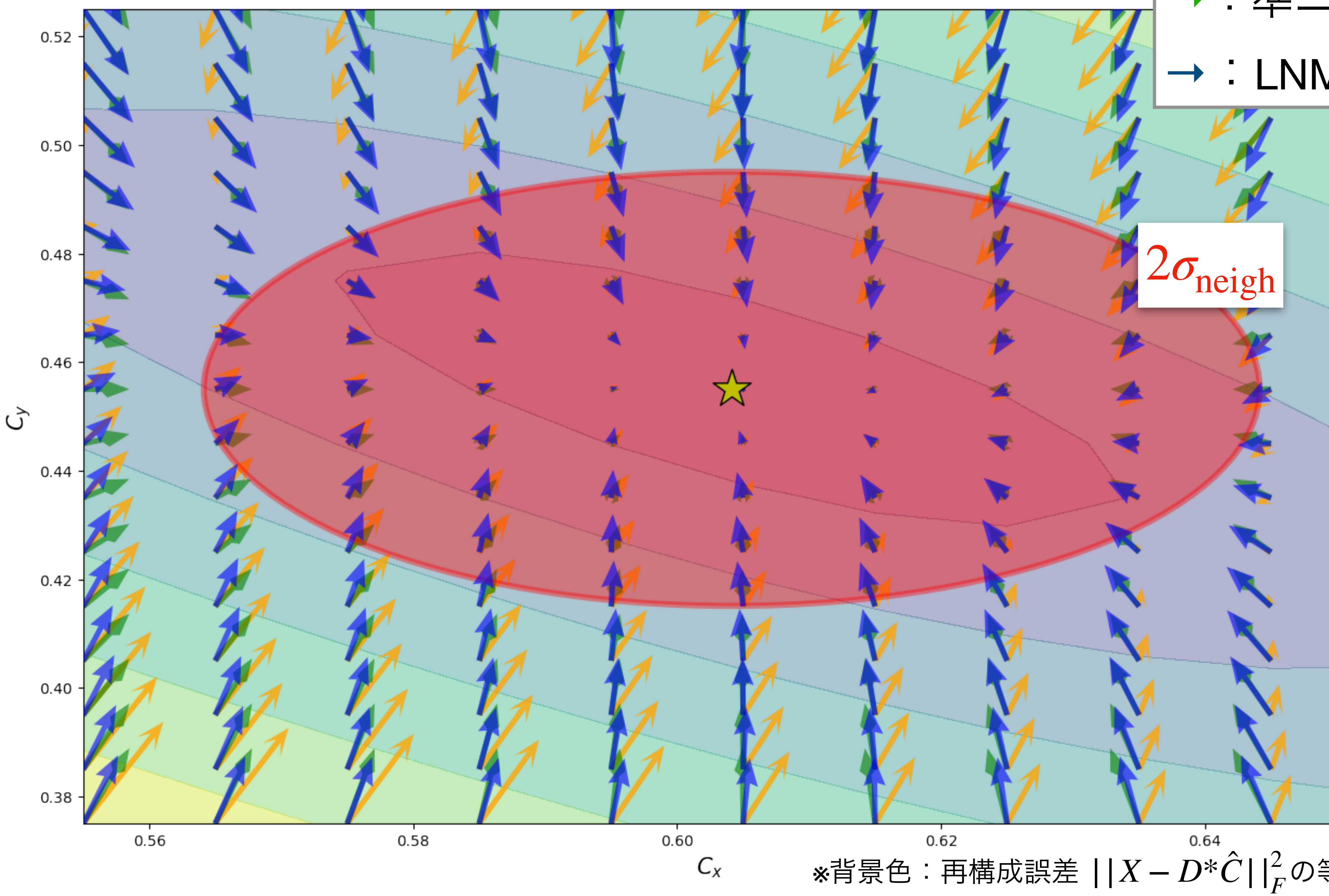
- 勾配とニュートン法，及びLNMFの更新フローの比較
- ベクトルのノルムを正規化 (向きのみ表示)



※背景色：再構成誤差 $||X - D^*\hat{C}||_F^2$ の等高線プロット

LNMF vs. MU | 収束状況の可視化 (解近傍)

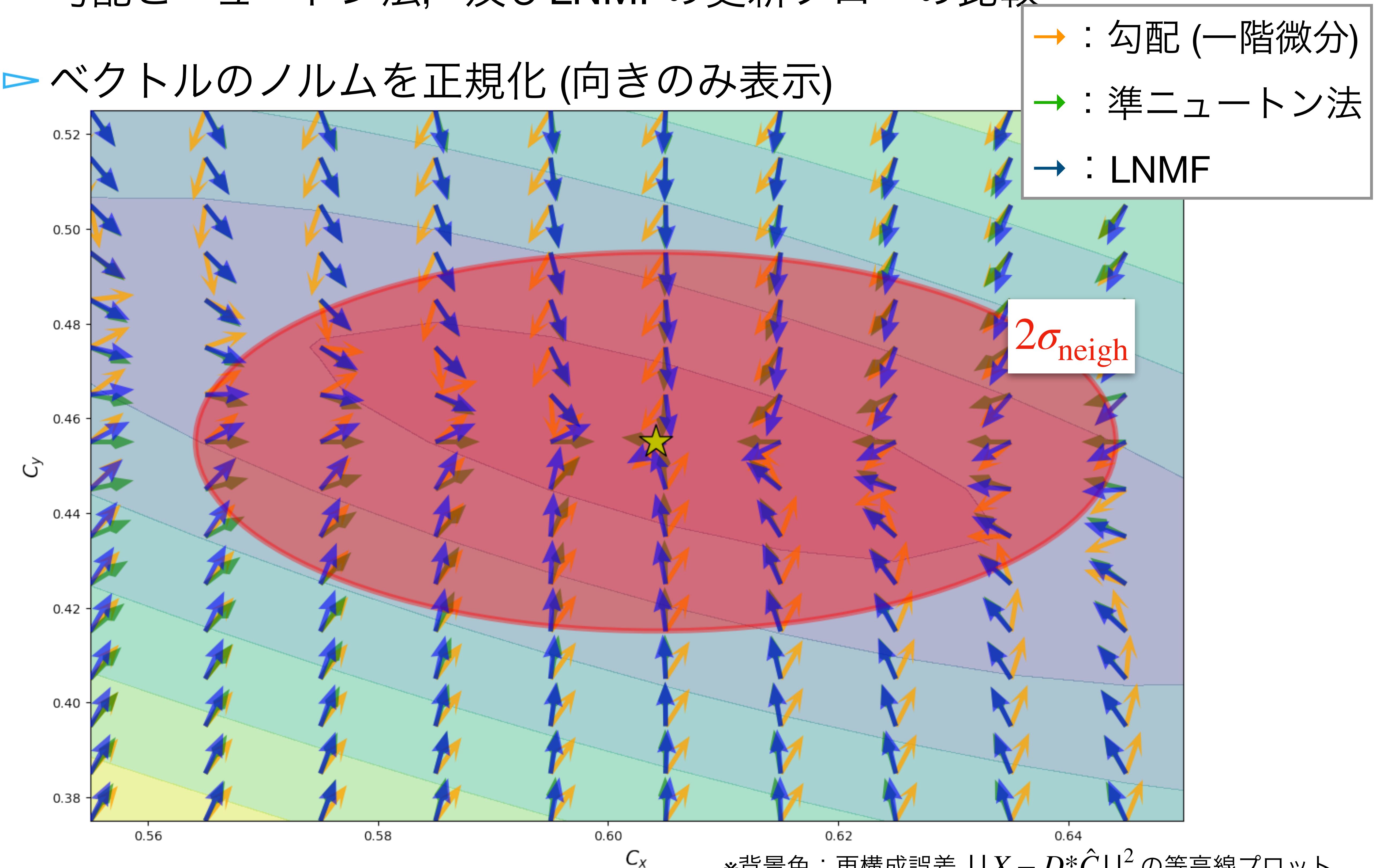
- 勾配とニュートン法，及びLNMFの更新フローの比較
- ベクトルの長さは更新幅を表す (勾配法の学習率は1とする



※背景色：再構成誤差 $||X - D^*\hat{C}||_F^2$ の等高線プロット

LNMF vs. MU | 収束状況の可視化 (解近傍)

- 勾配とニュートン法，及びLNMFの更新フローの比較
- ベクトルのノルムを正規化 (向きのみ表示)

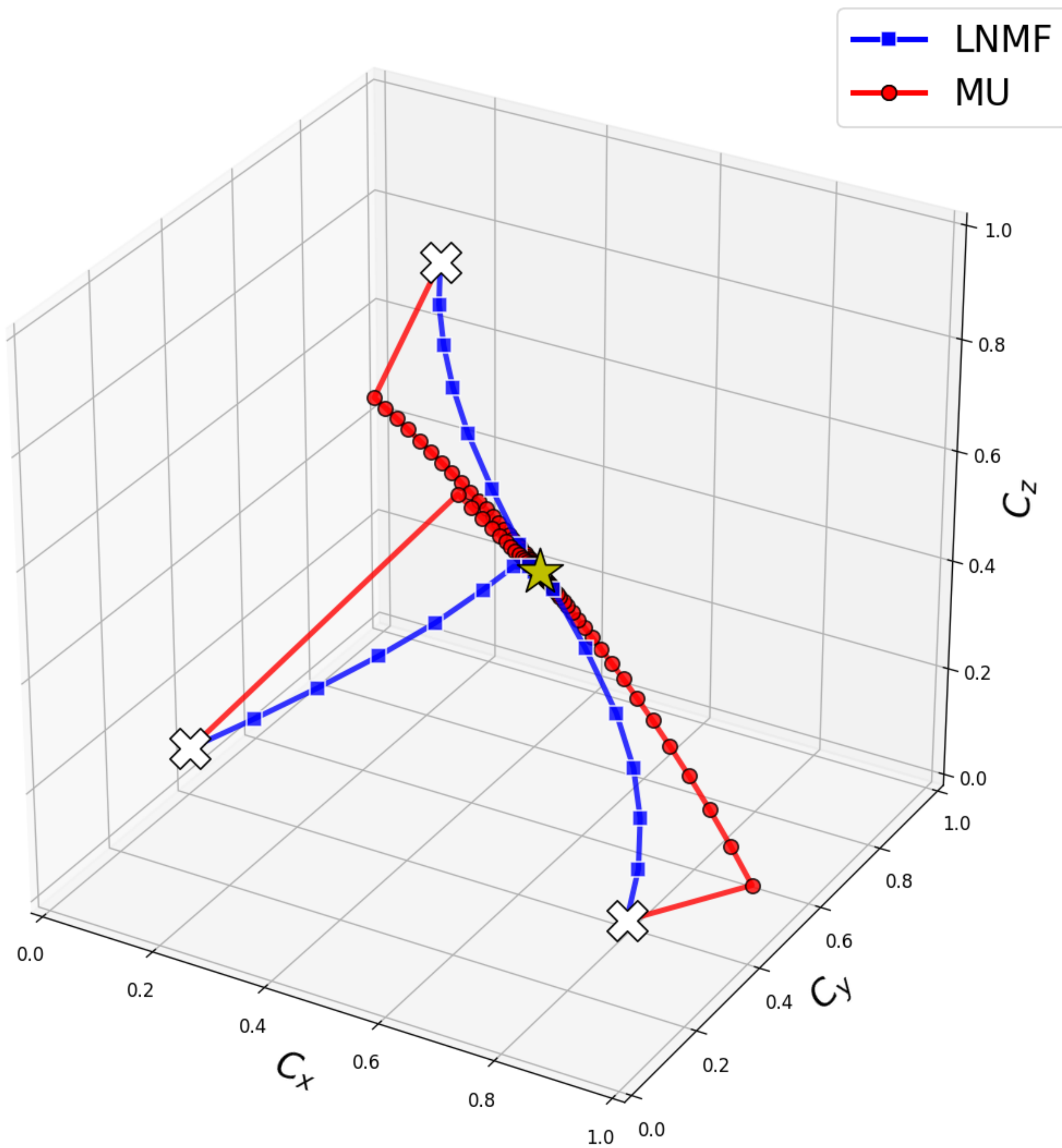


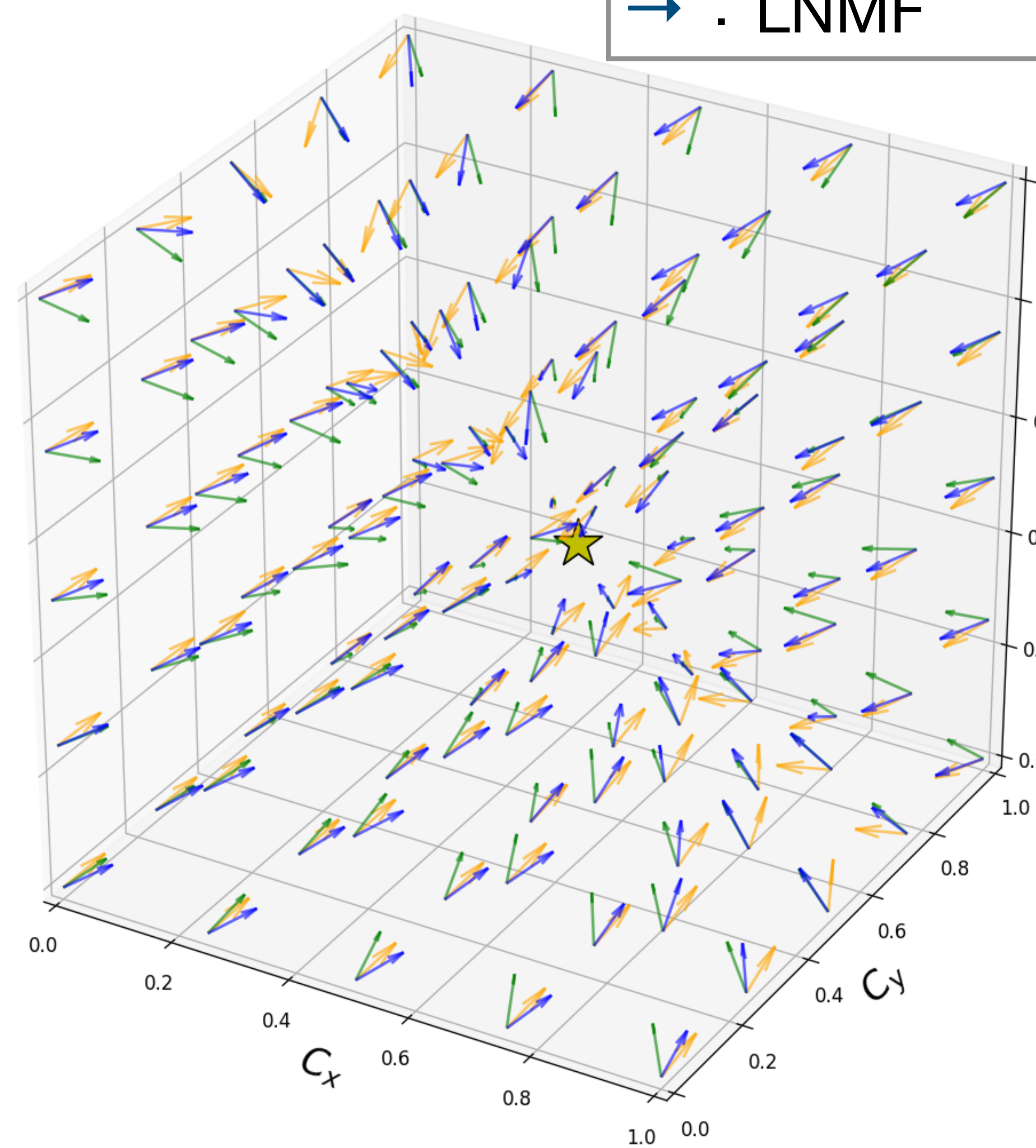
※背景色：再構成誤差 $||X - D^{*}\hat{C}||_F^2$ の等高線プロット

LNMF vs. MU | 収束状況の可視化 (3D)

- ▻ 勾配とニュートン法，及びLNMFの更新フローの比較
- ▻ ベクトルのノルムを正規化 (向きのみ表示)

→ : 勾配 (一階微分)
→ : 準ニュートン法
→ : LNMF



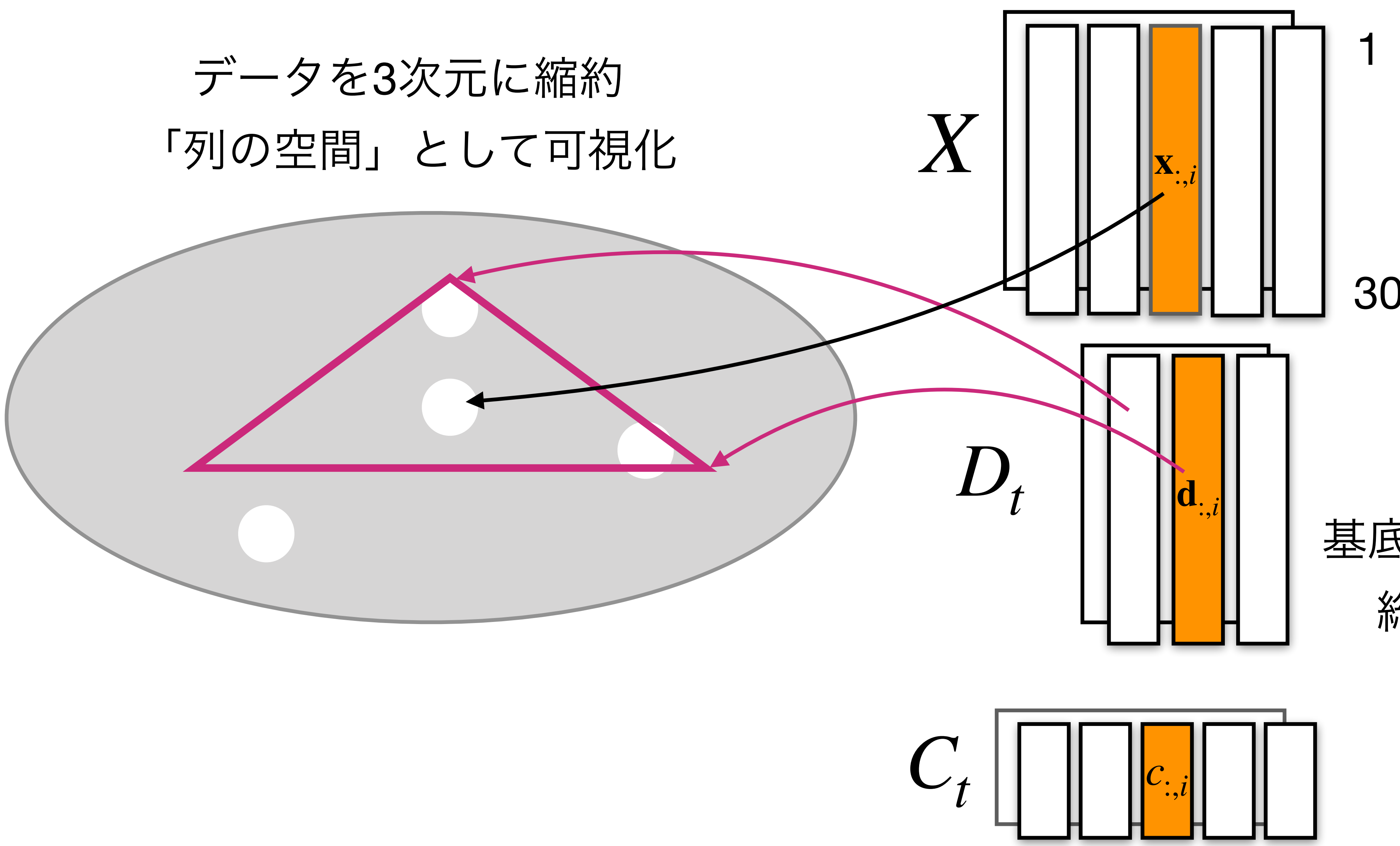


LNMF vs. MU | NMFの幾何的可視化

- XおよびDCが表現できる範囲を3Dで可視化して，収束状況を定性的に見たい

データを3次元に縮約

「列の空間」として可視化



Xの列の総和を1

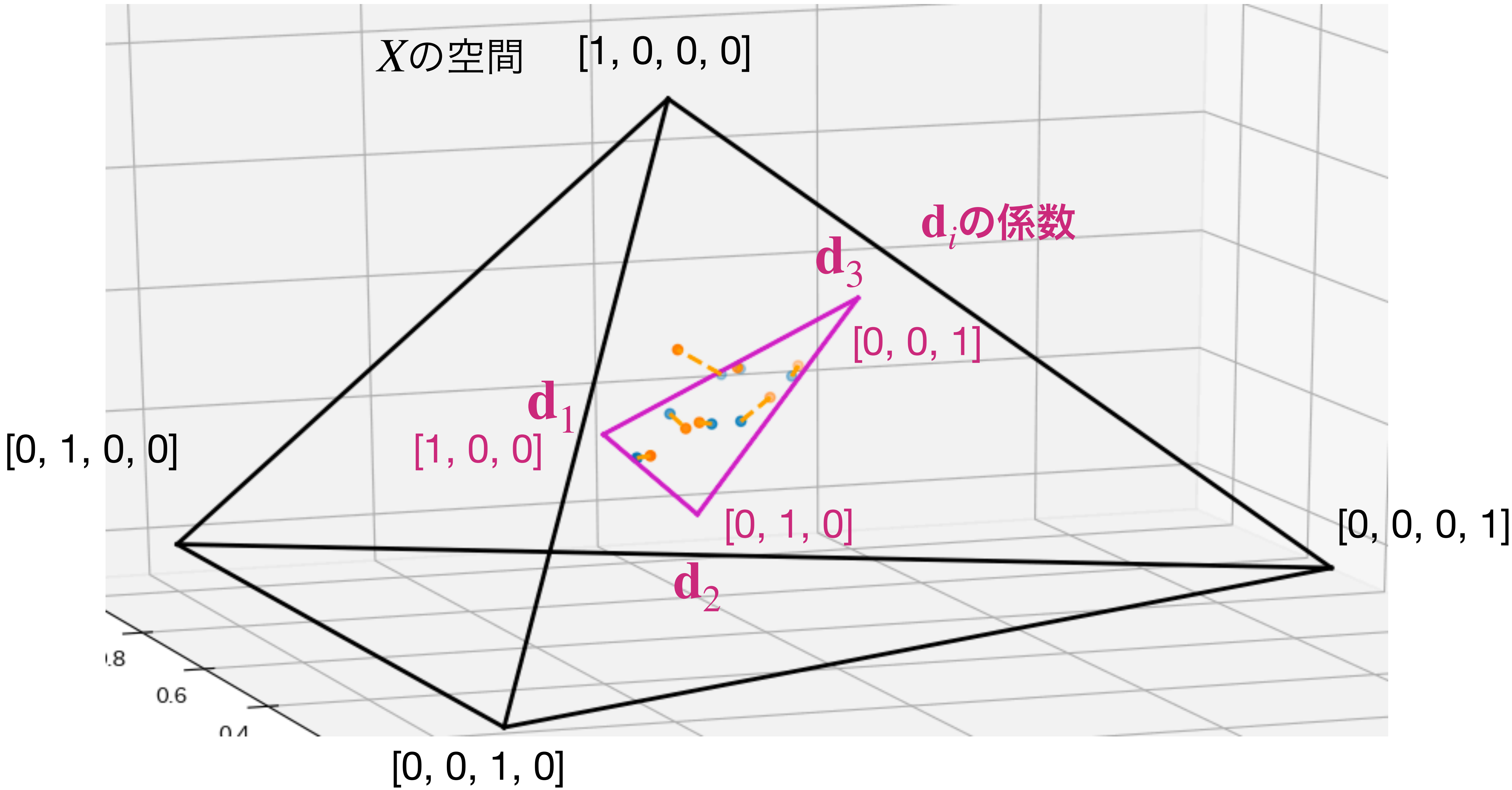
基底・係数ベクトルの

総和を1に正規化

$$\sum_i \mathbf{x}_{i,j} = 1$$

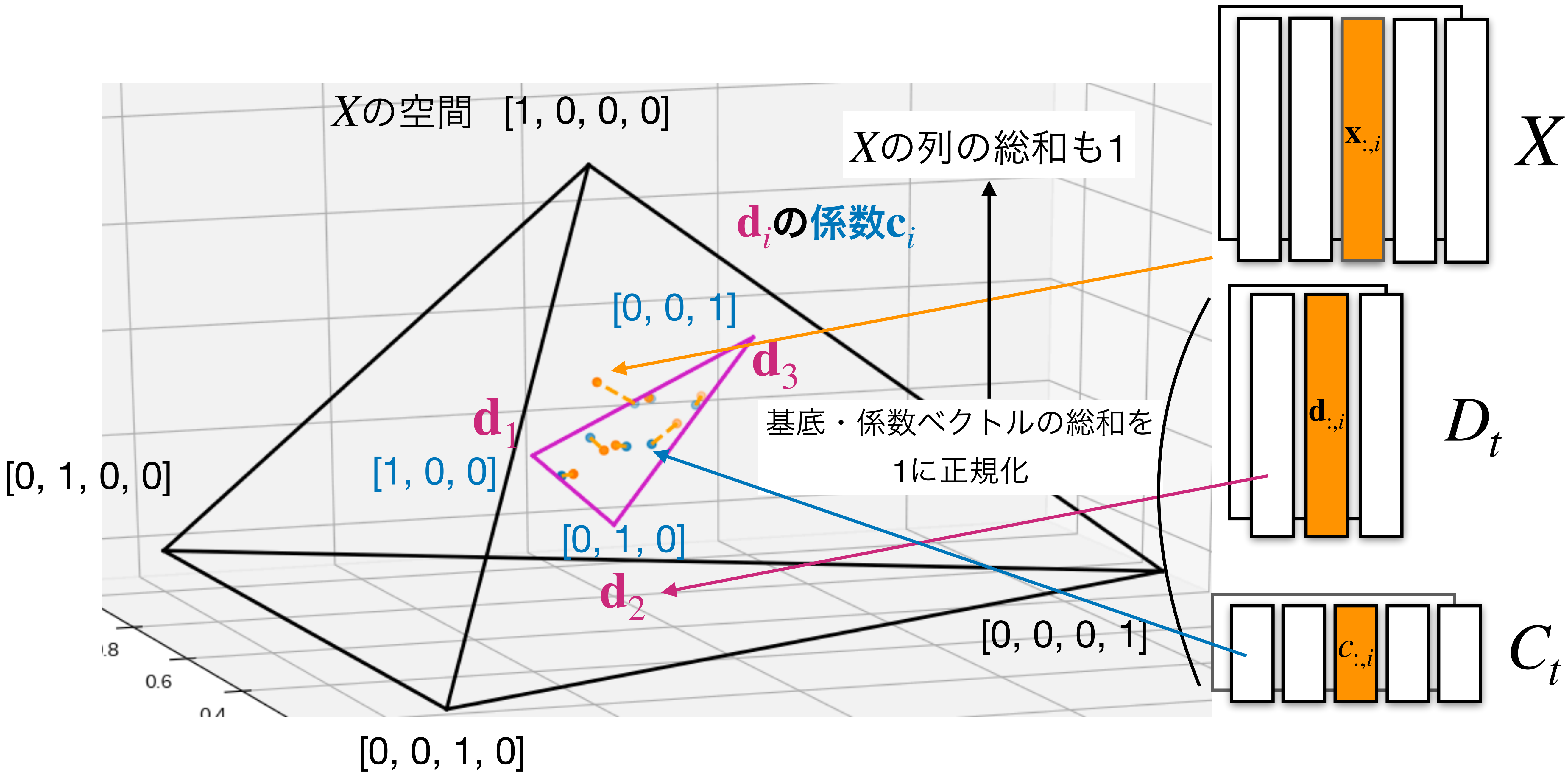
NMFの幾何的可視化 | 説明

- データX: 4次元・足して1の制約→正四面体の中に収まる
- 基底数: 3つ・DCの列も足して1の制約→基底が表現できる範囲は三角形 (2次元) で表せる
- 再構成誤差: 基底部分を表す平面にデータ点を射影したときの距離



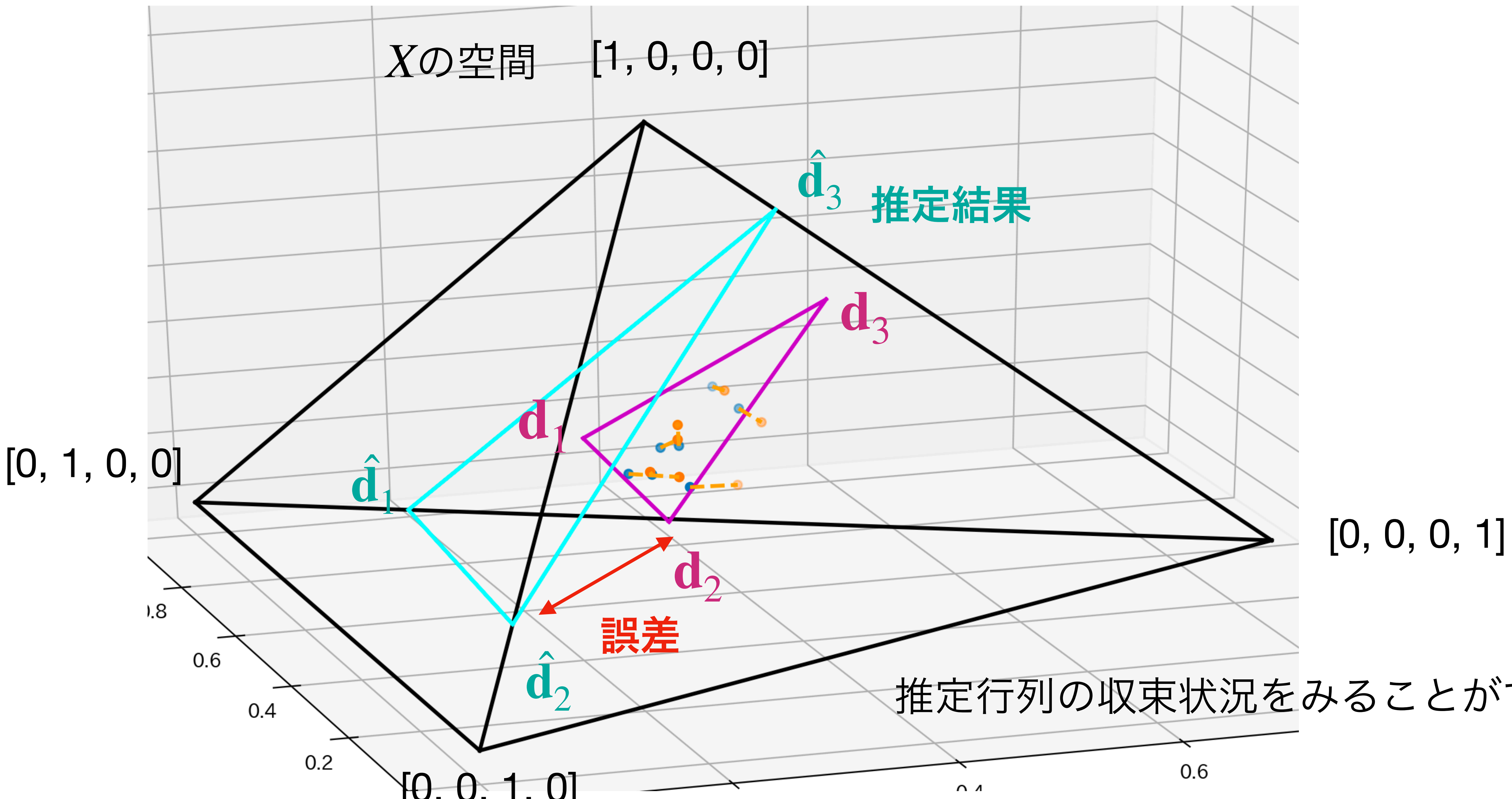
NMFの幾何的可視化 | 説明

- データX: 4次元・足して1の制約→正四面体の中に収まる
- 基底数: 3つ・DCの列も足して1の制約→基底が表現できる範囲は三角形 (2次元) で表せる
- 再構成誤差: 基底部分を表す平面にデータ点を射影したときの距離



NMFの幾何的可視化 | 説明

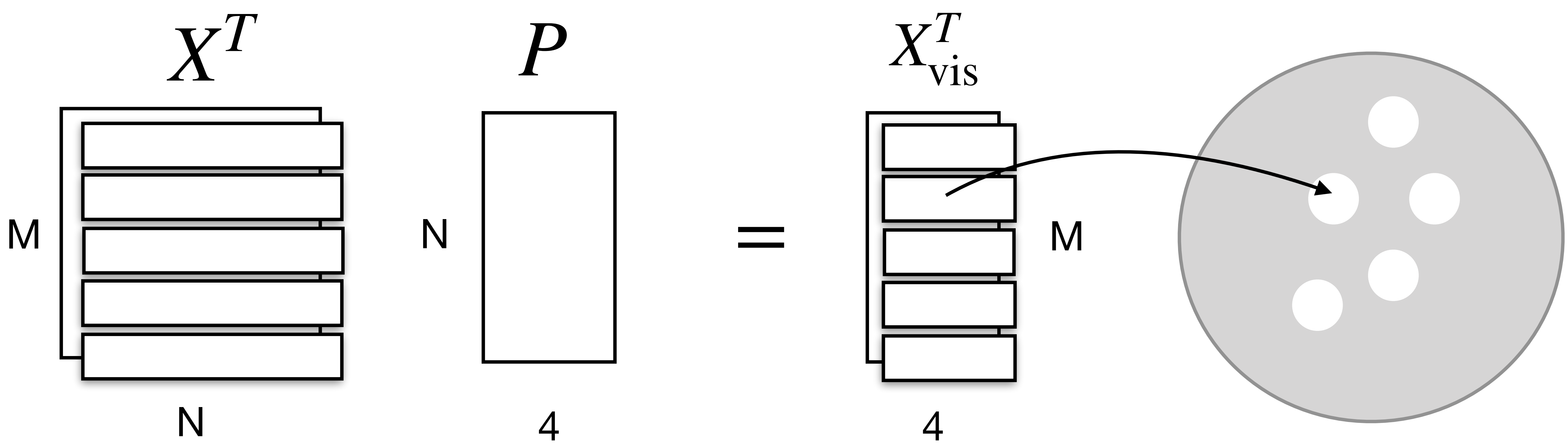
- データX: 4次元・足して1の制約→正四面体の中に収まる
- 基底数: 3つ・DCの列も足して1の制約→基底が表現できる範囲は三角形 (2次元) で表せる
- 再構成誤差: 基底部分を表す平面にデータ点を射影したときの距離



推定行列の収束状況をみることができる

データの次元が4以上のときの可視化 (K=3)

- 適当な射影行列Pをかけて4次元に次元縮約

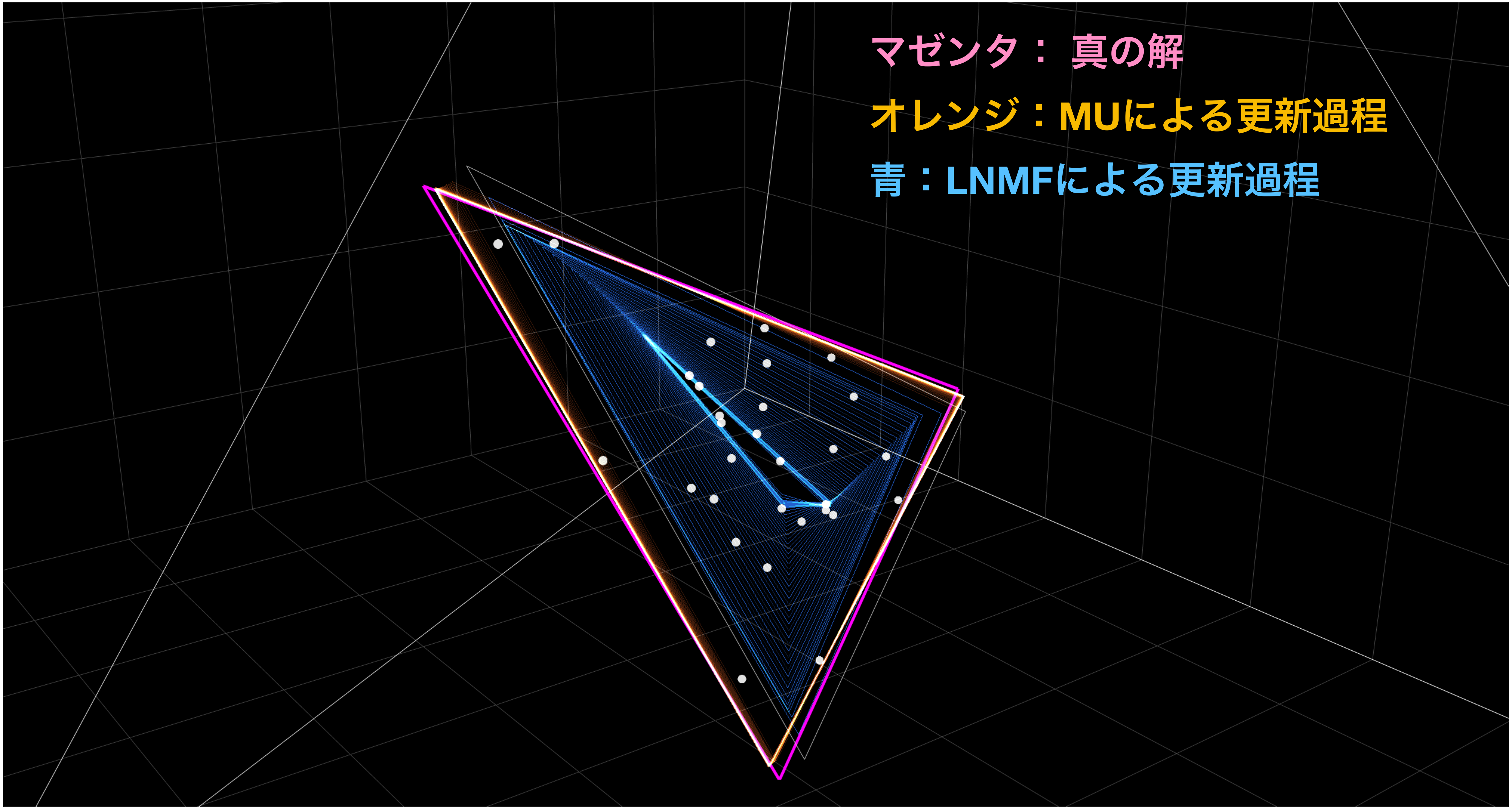


データを実質3次元に縮約 (3Dで可視化)　$\sum_i \mathbf{x}_{i,j} = 1$

今回は　$[\mathbf{x}_{1,j}, \mathbf{x}_{2,j}, \mathbf{x}_{3,j}, \mathrm{mean}(\mathbf{x}_{4,j}, \cdots, \mathbf{x}_{30,j})], \sum_i \mathbf{x}_{i,j} = 1$

行列の更新過程の可視化デモ

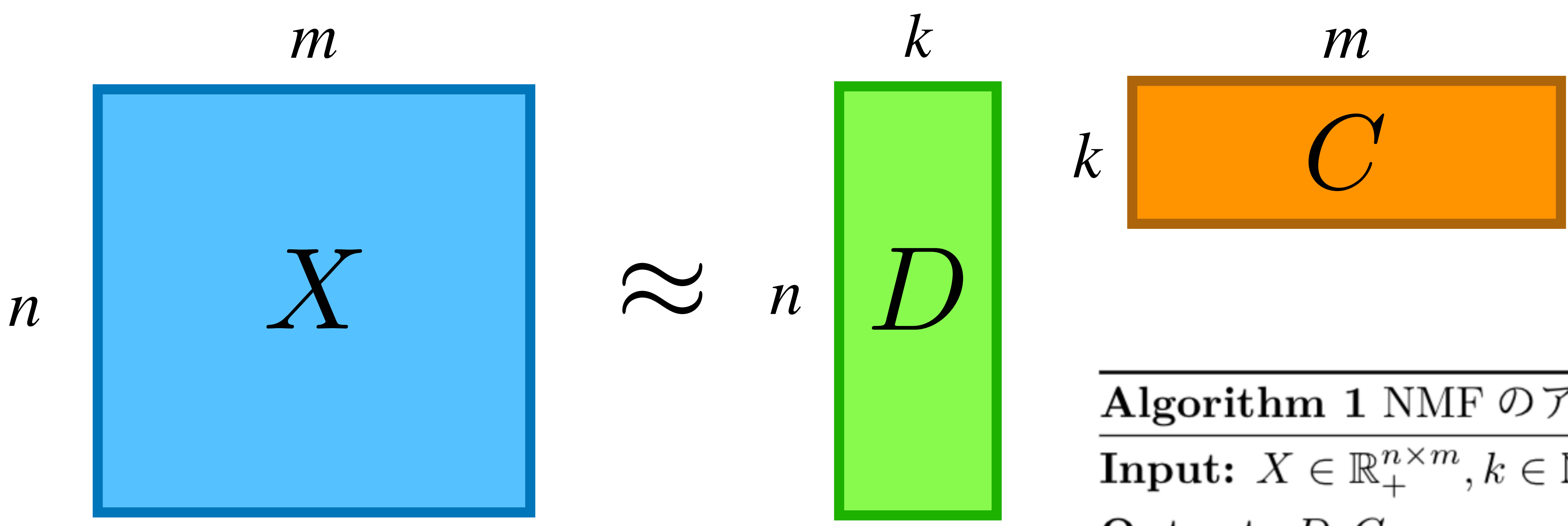
- 凡例は以下



NMF: Nonnegative Matrix Factorization

- 因子行列に非負制約を課した行列分解 (NMF) の高速化を考える

$X \in \mathbb{R}_+^{n \times m}, D \in \mathbb{R}_+^{n \times k}, C \in \mathbb{R}_+^{k \times m}$ $\quad X \approx DC, 1 \leq k \leq \min\{n, m\}$



目的関数

$$\min_{D,C} D(X||DC)$$ (今回はユークリッド距離)

$$\text{s.t.} \ \ D \geq 0, C \geq 0$$

※$A \geq 0$:行列Aの各要素が非負

Algorithm 1 NMF のアルゴリズム

Input: $X \in \mathbb{R}_+^{n \times m}, k \in \mathbb{N}$

Output: D, C

Initialize D, C

repeat

$\quad D \leftarrow \arg\min_{D \geq 0} ||X - DC||_F$

$\quad C \leftarrow \arg\min_{C \geq 0} ||X - DC||_F$

until convergence

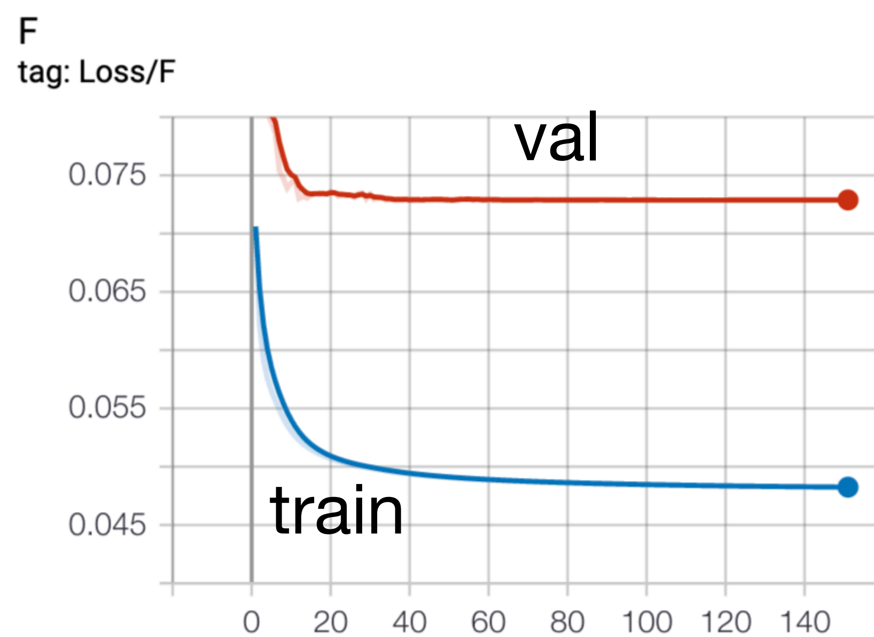
D, Cに関して交互最適化を行う

LNMFの学習過程

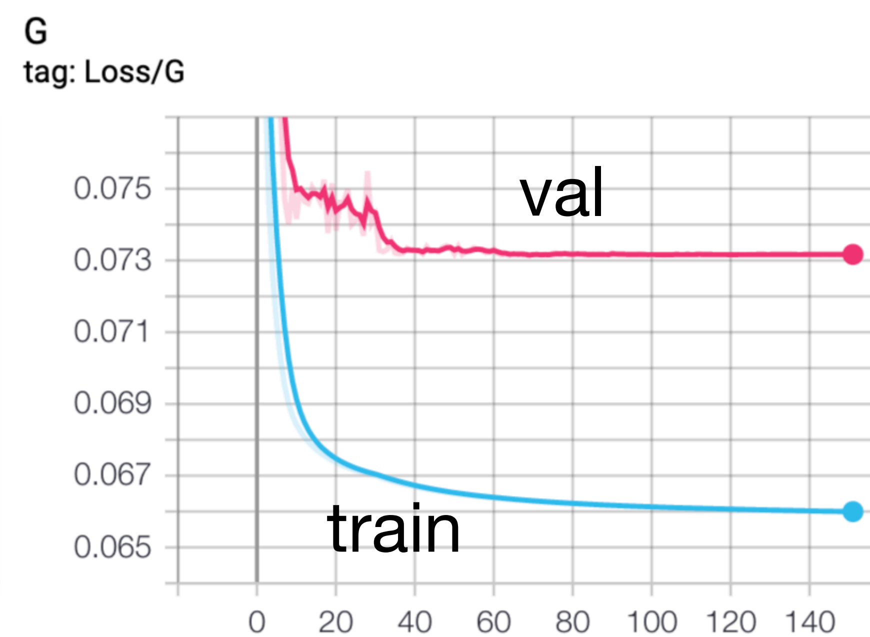
- 訓練データを与える毎の行列の推定誤差及び再構成誤差

行列の推定誤差

$$F : ||D - \hat{D}||_F^2$$
$$G : ||C - \hat{C}||_F^2$$



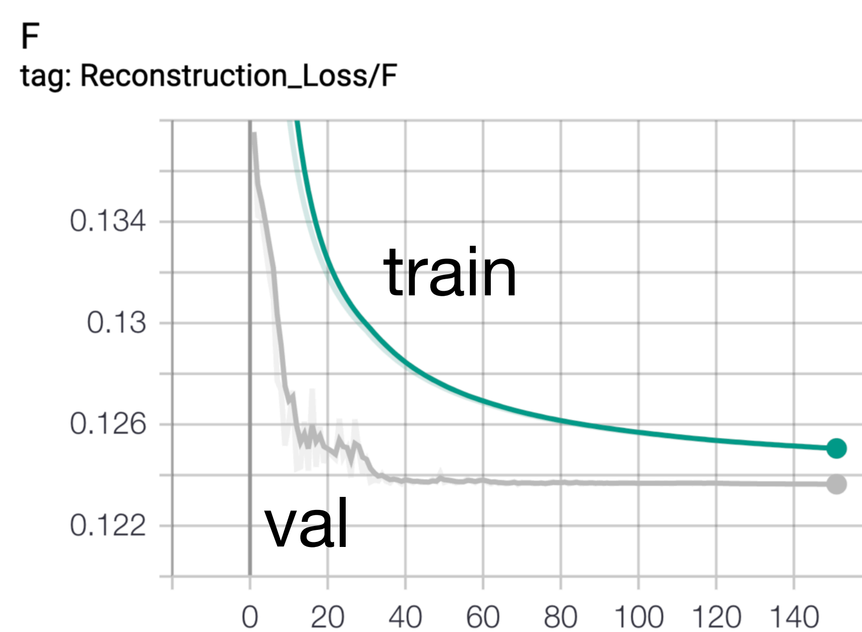
訓練データ数 (x10万)



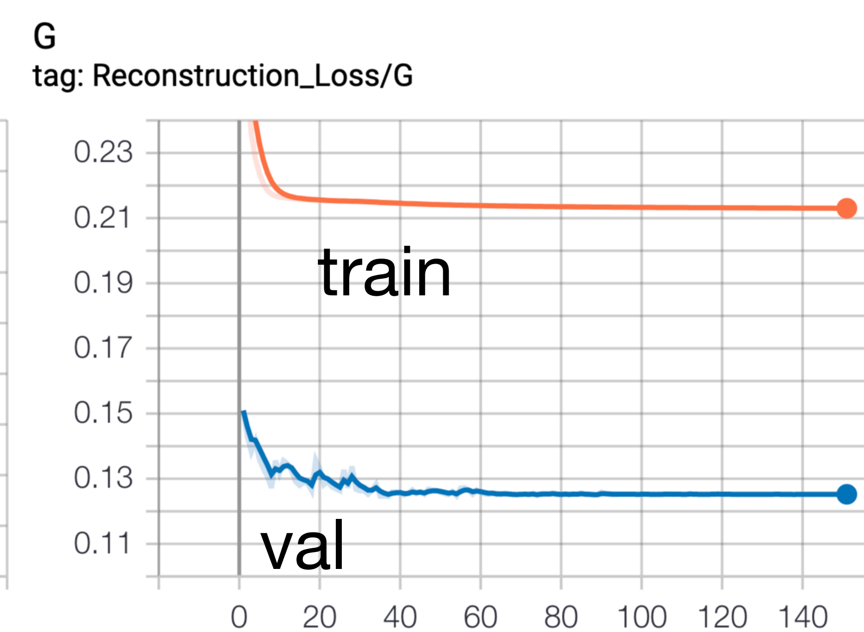
訓練データ数 (x10万)

再構成誤差

$$F : ||X - \hat{D}C||_F^2$$
$$G : ||X - D\hat{C}||_F^2$$



訓練データ数 (x10万)



訓練データ数 (x10万)

LNMFの学習 | 実験設定 (ノイズあり)

学習データ (train)	評価データ (val)	最適化	(n, m, k) (行, 列, 基底数)	活性化関数
10M	5K	Adam	(30,30,3)	tanh

- 人工データ (以下のセットを1つのサンプルとする)
  $$D^*_{ij}, C^*_{jk} \sim \mathcal{U}(0,1)$$
  - 真の解 D^*, C^*
  $$e \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$$
  - $X = \left[D^* C^* + e\right]_+$

Xのノイズの分散とS/N比 ($\sigma_* \fallingdotseq 0.29$)

σ_{noise}	0.1	0.02	0.001
S/N[dB]	18.4	46.4	98.4

- D, Cの初期化: 解の近傍 ($\sigma_{\text{neigh}} = 0.02$)

$$\epsilon_{\text{neigh}} \sim \mathcal{N}(0, \sigma^2_{\text{neigh}})$$
$$D_0 = \left[D^* + \epsilon_{\text{neigh}}\right]_+$$

目的関数

$$\min_{\hat{D}_1 \geq 0} ||D^* - \hat{D}_1||_F^2 + ||X - \hat{D}_1 C_*||_F^2$$
$$\min_{\hat{C}_1 \geq 0} ||C^* - \hat{C}_1||_F^2 + ||X - D_* \hat{C}_1||_F^2$$

実験結果 | 学習時のノイズの分散による変化

- 学習時と評価時でノイズの分散を変えたときの更新回数の比率を比較
- 表の数値はLNMFの更新回数がMUに比べて何分の一かを示す

学習時ノイズ \ 評価時ノイズ	$\sigma_{\text{noise}} = 0$ $\text{S/N} = \infty$	$\sigma_{\text{noise}} = 0.001$ $\text{S/N} = 98.4$	$\sigma_{\text{noise}} = 0.01$ $\text{S/N} = 46.4$	$\sigma_{\text{noise}} = 0.1$ $\text{S/N} = 18.4$
$\sigma_{\text{noise}} = 0,\ \text{S/N} = 0$	16.47 ± 26.31	16.37 ± 25.96	16.27 ± 24.19	13.47 ± 17.95
$\sigma_{\text{noise}} = 0.001,\ \text{S/N} = 98.4$	13.99 ± 17.04	14.04 ± 17.24	14.35 ± 18.70	12.78 ± 17.09
$\sigma_{\text{noise}} = 0.01,\ \text{S/N} = 46.4$	15.76 ± 21.17	15.80 ± 21.22	15.70 ± 21.18	13.57 ± 17.61
$\sigma_{\text{noise}} = 0.1,\ \text{S/N} = 18.4$	10.60 ± 16.12	10.60 ± 16.24	10.51 ± 16.22	8.59 ± 13.00

実験結果 | 更新毎の再構成誤差 (ノイズ毎)

学習

テスト

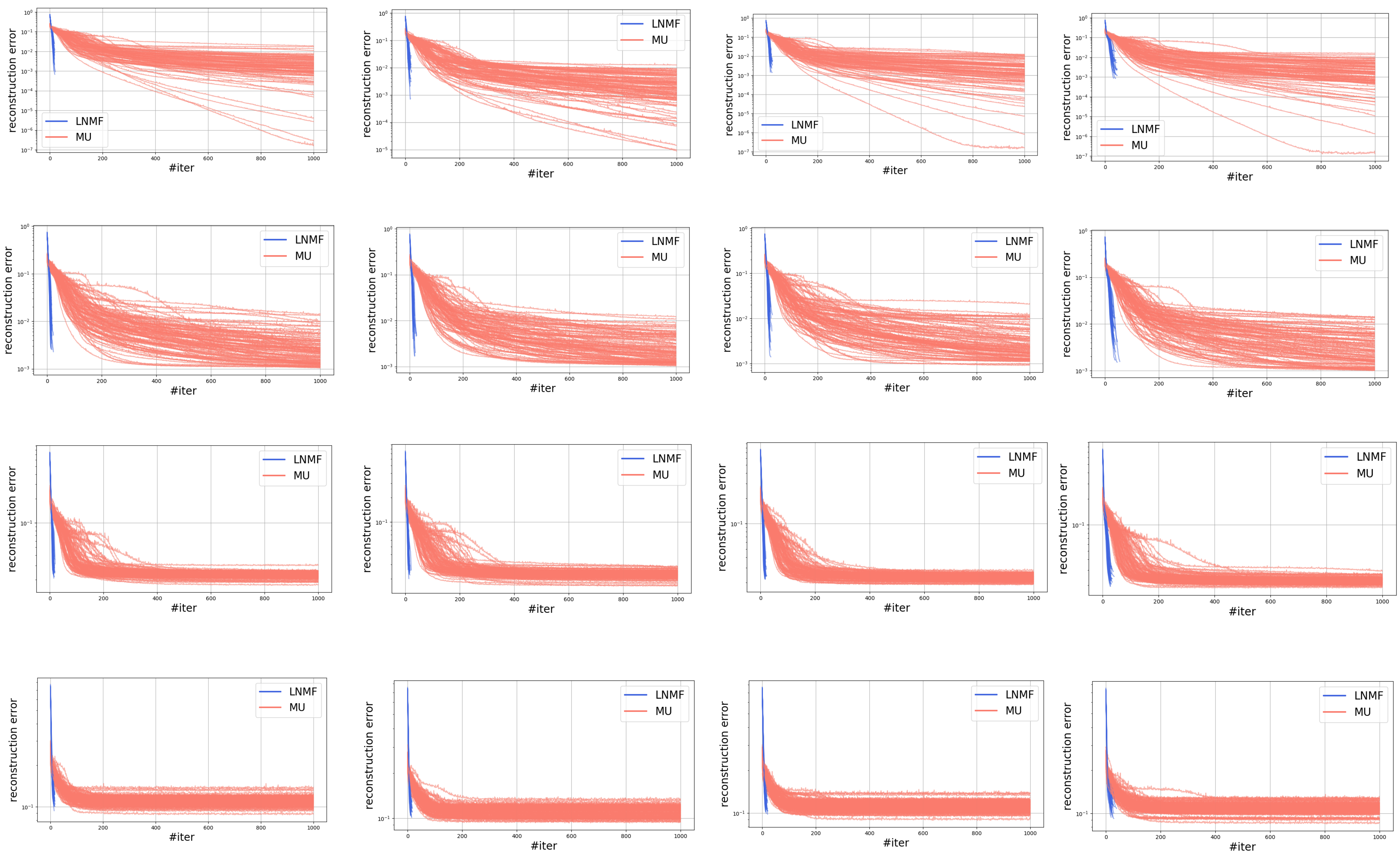
$\sigma_{\text{noise}} = 0$ | $\sigma_{\text{noise}} = 0.001$ | $\sigma_{\text{noise}} = 0.02$ | $\sigma_{\text{noise}} = 0.1$

$\sigma_{\text{noise}} = 0$

$\sigma_{\text{noise}} = 0.001$

$\sigma_{\text{noise}} = 0.02$

$\sigma_{\text{noise}} = 0.1$



実験結果 | 更新毎の再構成誤差 (ノイズ毎)

学習

テスト

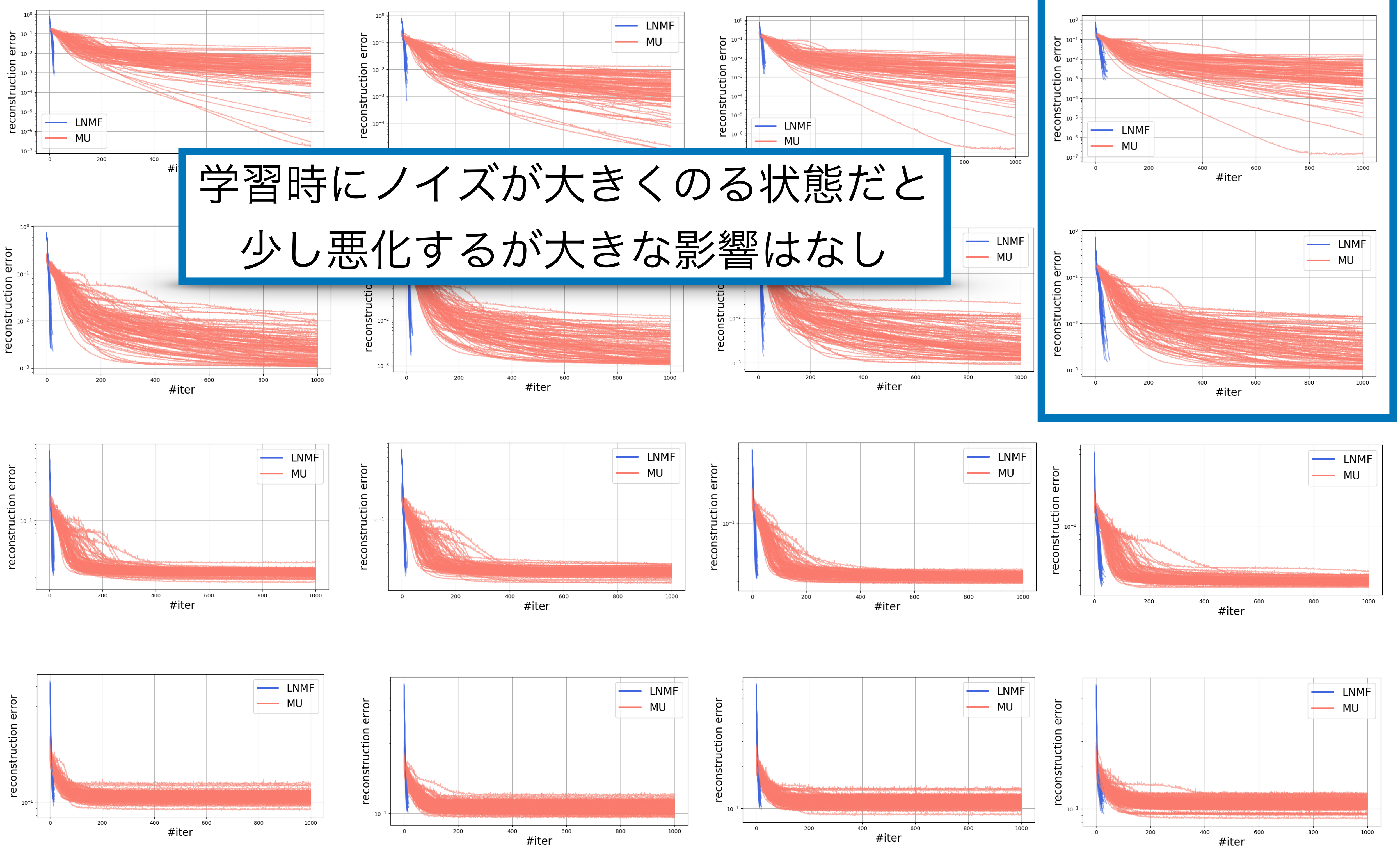
$\sigma_{\text{noise}} = 0$ | $\sigma_{\text{noise}} = 0.001$ | $\sigma_{\text{noise}} = 0.02$ | $\sigma_{\text{noise}} = 0.1$

$\sigma_{\text{noise}} = 0$

$\sigma_{\text{noise}} = 0.001$

$\sigma_{\text{noise}} = 0.02$

$\sigma_{\text{noise}} = 0.1$



学習時にノイズが大きくのる状態だと
少し悪化するが大きな影響はなし

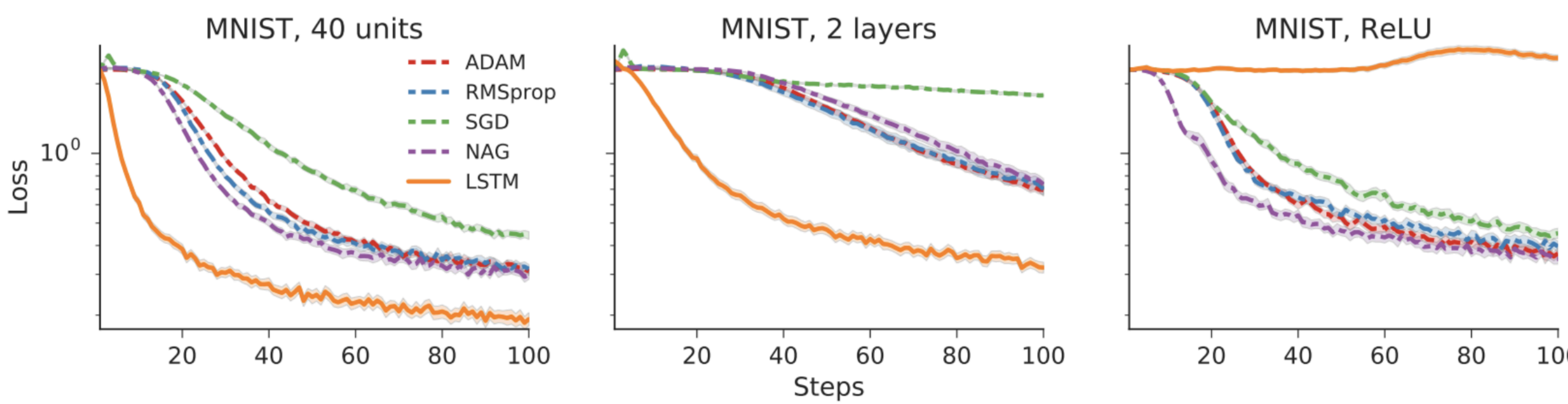
関連研究 | 学習による高速化手法

- スパースコーディングに対する高速化は応用/発展が進んでいる
  - “Learning fast approximations of sparse coding”, [Gregor+, 2010]
  - “Learning step sizes for unfolded sparse coding” [Ablin+, 2019]
- 勾配降下法に対する学習による高速化
  - “Learning to learn using gradient descent”, [Hochreiter+, 2001]
  - “Learning to learn by gradient descent by gradient descent”, [Andrychowicz+, 2016]

- 行列分解に対して学習の高速化を適用した例はない
  - 教師ありNMF (片方の行列を固定) のものはある
    - "Deep NMF for speech separation” [Roux+, 2014]
    - 片方の行列を学習するパラメータとしている

勾配降下法の学習による高速化

- Learning to learn using gradient descent [Hochreiter+, 2001]
- Learning to learn by gradient descent by gradient descent [Andrychowicz+, 2016]
  - 勾配法の最適化をLSTMによって行った．Adamなどよりも早い収束を達成
  - LNMFと同様に，ニューラルネットに目的関数の勾配を入力してパラメータの更新分を推定させる



目的関数$f(\theta)$をパラメータθに対して最適化する問題

$$\theta^* = \arg\min_{\theta\in\Theta} f(\theta)$$

に対して，fが微分可能なら勾配降下法でパラメータを求めることができる

$$\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t).$$

この代わりに，学習器の出力からパラメータの更新分を推定する

$$\theta_{t+1} = \theta_t + g_t,$$

$$\begin{pmatrix} g_t \\ h_{t+1} \end{pmatrix} = u\left(\nabla_t, h_t; \phi\right).$$

← LSTM

この学習器uはLSTMで，元の目的関数を最小化するように学習しておく

$$\mathcal{L}_{\mathrm{traj}}(\phi) = \mathbb{E}_f\left[\sum_{t=1}^{T} w_t f(\theta_t)\right]$$

最終的に得られるパラメータだけでなく，更新過程でも損失をとる

- LSTMの出力でパラメータ更新
- 更新フローを図にすると以下

$$\mathcal{L}_{\mathrm{traj}}(\phi) = \mathbb{E}_f \left[\sum_{t=1}^{T} w_t f(\theta_t) \right],$$

$$\text{where} \quad \theta_{t+1} = \theta_t + g_t,$$

$$\begin{pmatrix} g_t \\ h_{t+1} \end{pmatrix} = u(\nabla_t, h_t; \phi).$$

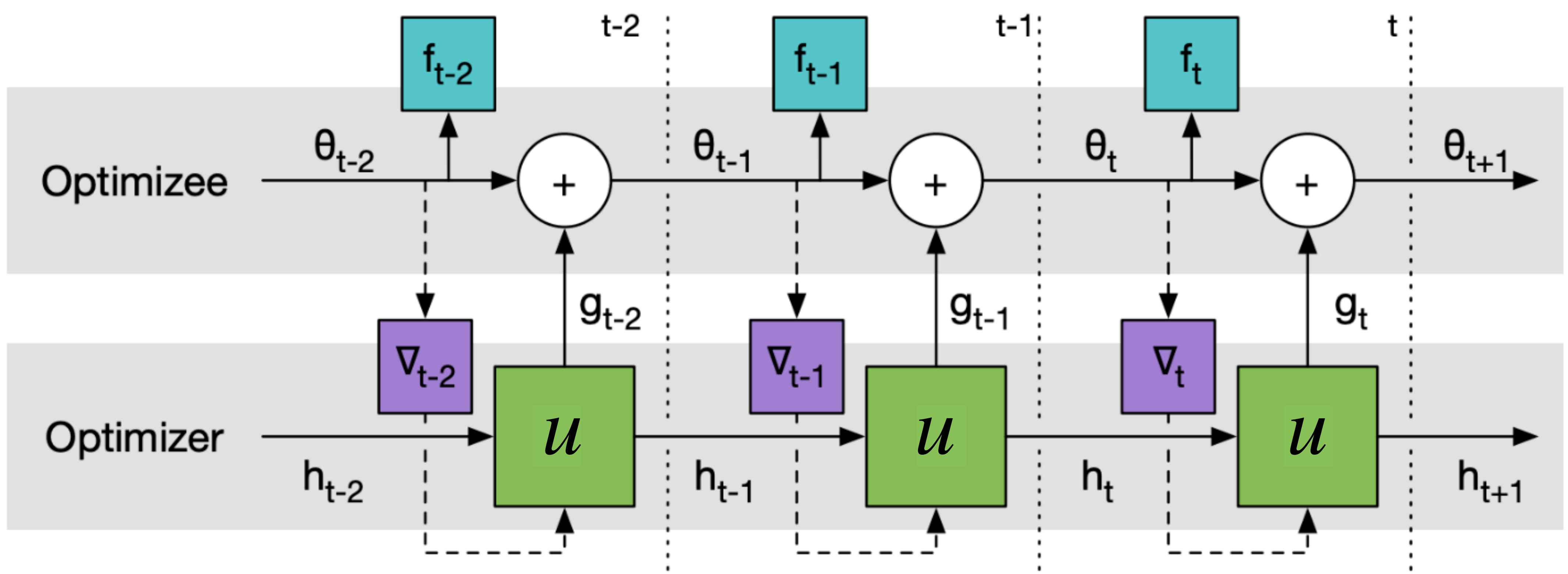


Figure 2: Computational graph used for computing the gradient of the optimizer.

学習による高速化手法 | Learned ISTA

- Gregor and LeCunがスパースコーディングの高速化を行った (Learned ISTA)
  - “Learning Fast Approximations of Sparse Coding”, ICML2010

目的関数

$$\min_{Z} ||X - W_d Z||_2^2 + \alpha ||Z||_1$$

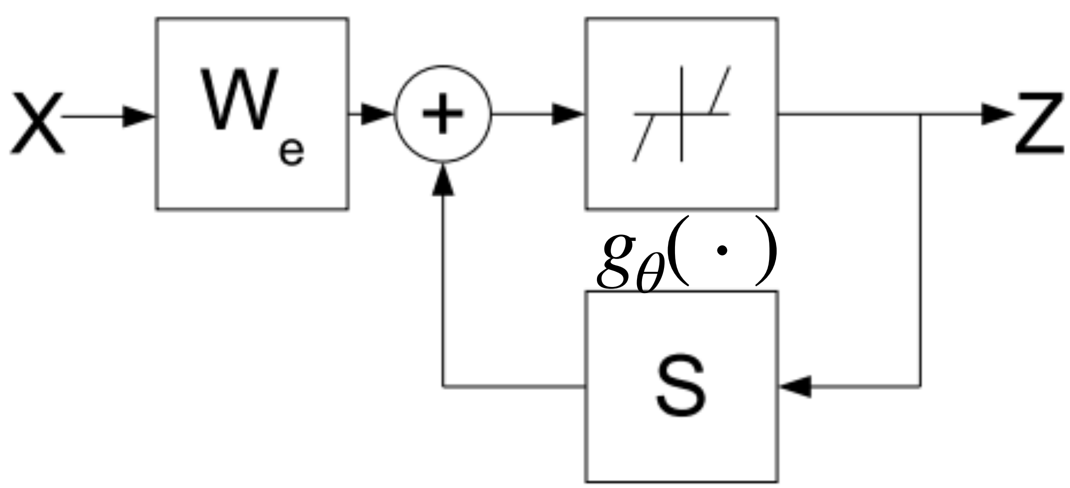
Xの例: 画像データなど

ISTAアルゴリズム

$$\theta = \frac{\alpha}{L}, S = I - \frac{1}{L} W_d^T W_d, W_e = \frac{1}{L} W_d$$

$$Z^{k+1} = g_\theta(S Z^k + W_e X)$$

※$L > W_d^T W_d$の最大固有値



学習による高速化手法 | Learned ISTA

- Gregor and LeCunがスパースコーディングの高速化を行った (Learned ISTA)
  - “Learning Fast Approximations of Sparse Coding”, ICML2010

目的関数

$$\min_{Z} ||X - W_d Z||_2^2 + \alpha ||Z||_1$$

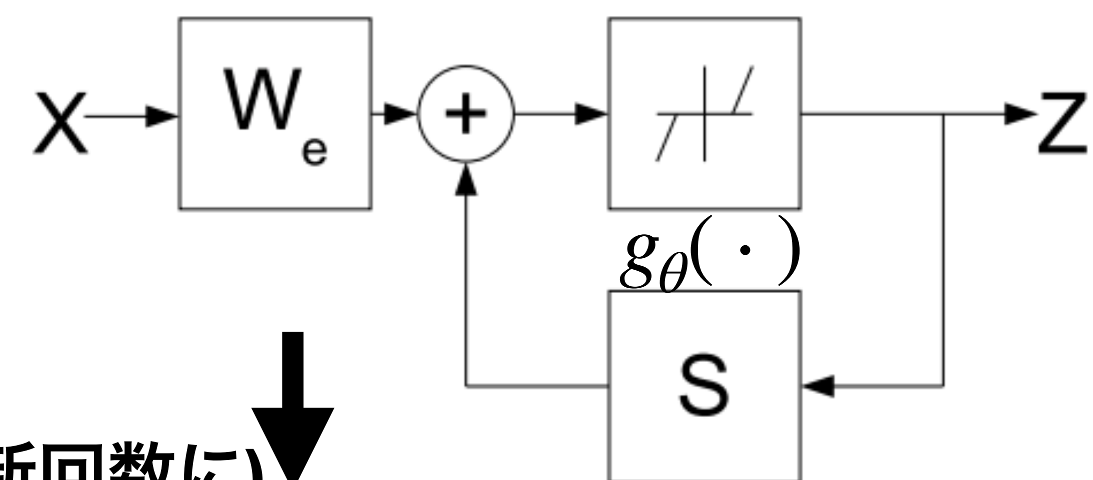
Xの例: 画像データなど

ISTAアルゴリズム

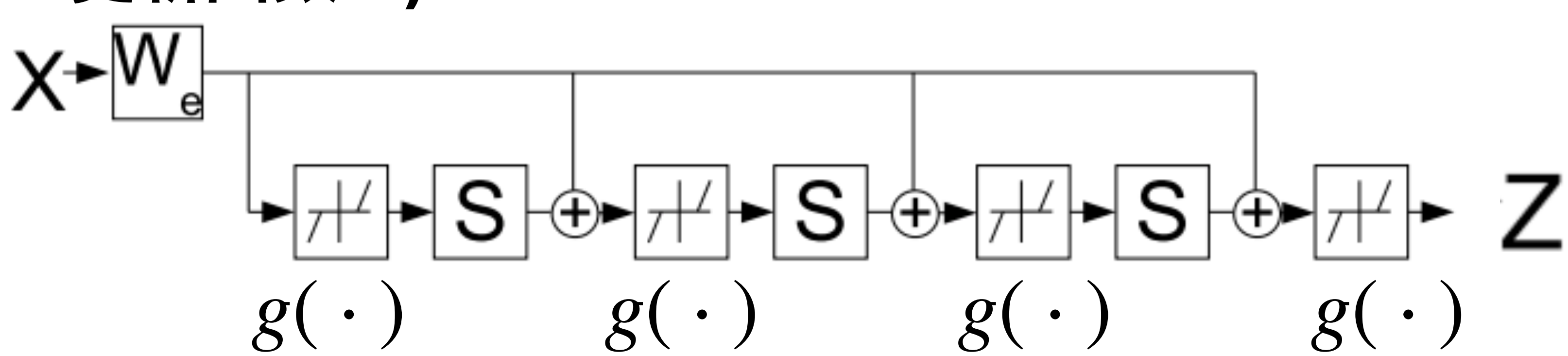
$$\theta = \frac{\alpha}{L}, S = I - \frac{1}{L}W_d^T W_d, W_e = \frac{1}{L}W_d$$

$$Z^{k+1} = g_\theta(SZ^k + W_e X)$$

※$L > W_d^T W_d$の最大固有値



Unrolling ISTA
(ループ展開して少ない更新回数に)



学習による高速化手法 | Learned ISTA

- Gregor and LeCunがスパースコーディングの高速化を行った (Learned ISTA)
  - “Learning Fast Approximations of Sparse Coding”, ICML2010

目的関数

$$\min_{Z} ||X - W_d Z||_2^2 + \alpha ||Z||_1$$

ISTAアルゴリズム

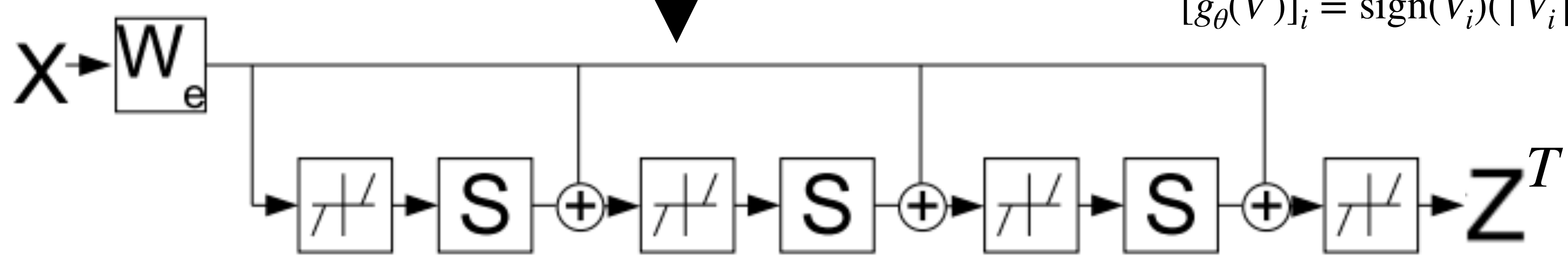
$$\theta = \frac{\alpha}{L}, S = I - \frac{1}{L} W_d^T W_d, W_e = \frac{1}{L} W_d$$

$$Z^{k+1} = g_\theta(SZ^k + W_e X)$$

$L > W_d^T W_d$の最大固有値

$[g_\theta(V)]_i = \mathrm{sign}(V_i)(|V_i| - \theta_i)_+$

Unrolling ISTA
(ループ展開して少ない更新回数に)



$g_\theta(\cdot)$ $g_\theta(\cdot)$ $g_\theta(\cdot)$

バイアス$W_e X$, 重み行列Sと非線形関数$g_\theta(\cdot)$

学習による高速化手法 | Learned ISTA

- Gregor and LeCunがスパースコーディングの高速化を行った (Learned ISTA)
  - “Learning Fast Approximations of Sparse Coding”, ICML2010

目的関数

$$\min_{Z} ||X - W_d Z||_2^2 + \alpha ||Z||_1$$

ISTAアルゴリズム

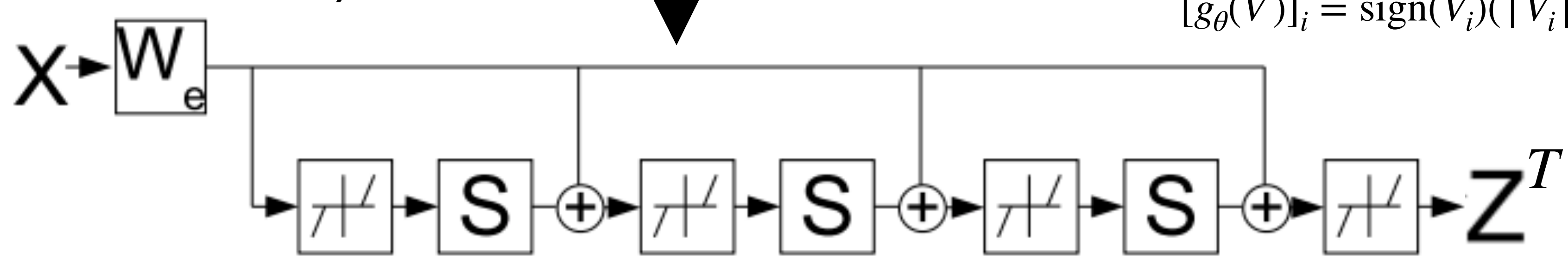
$$\theta = \frac{\alpha}{L}, S = I - \frac{1}{L} W_d^T W_d, W_e = \frac{1}{L} W_d$$

$$Z^{k+1} = g_\theta(SZ^k + W_e X)$$

$L > W_d^T W_d$の最大固有値

$[g_\theta(V)]_i = \mathrm{sign}(V_i)(|V_i| - \theta_i)_+$

Unrolling ISTA
ループ展開して少ない更新回数に)



$g_\theta(\cdot)$ $g_\theta(\cdot)$ $g_\theta(\cdot)$

バイアス$W_e X$, 重み行列Sと非線形関数$g_\theta(\cdot)$

パラメータW_e, S, θ のT 層のニューラルネット

- Gregor and LeCunがスパースコーディングの高速化を行った (Learned ISTA)
  - “Learning Fast Approximations of Sparse Coding”, ICML2010

目的関数

$$\min_{Z} ||X - W_d Z||_2^2 + \alpha ||Z||_1$$

ISTAアルゴリズム

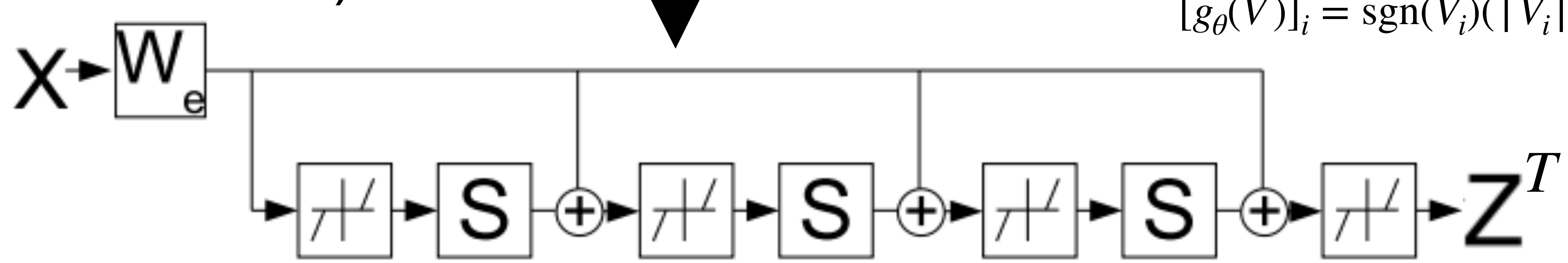
$$\theta = \frac{\alpha}{L}, S = I - \frac{1}{L} W_d^T W_d, W_e = \frac{1}{L} W_d$$

$$Z^{k+1} = g_\theta(SZ^k + W_e X)$$

$L > W_d^T W_d$の最大固有値

$[g_\theta(V)]_i = \mathrm{sgn}(V_i)(|V_i| - \theta_i)_+$

Unrolling ISTA
ループ展開して少ない更新回数に)



$g_\theta(\cdot)$ $g_\theta(\cdot)$ $g_\theta(\cdot)$

バイアス$W_e X$, 重み行列Sと非線形関数$g_\theta(\cdot)$

パラメータを$||Z^* - Z^T||_2^2$として学習！

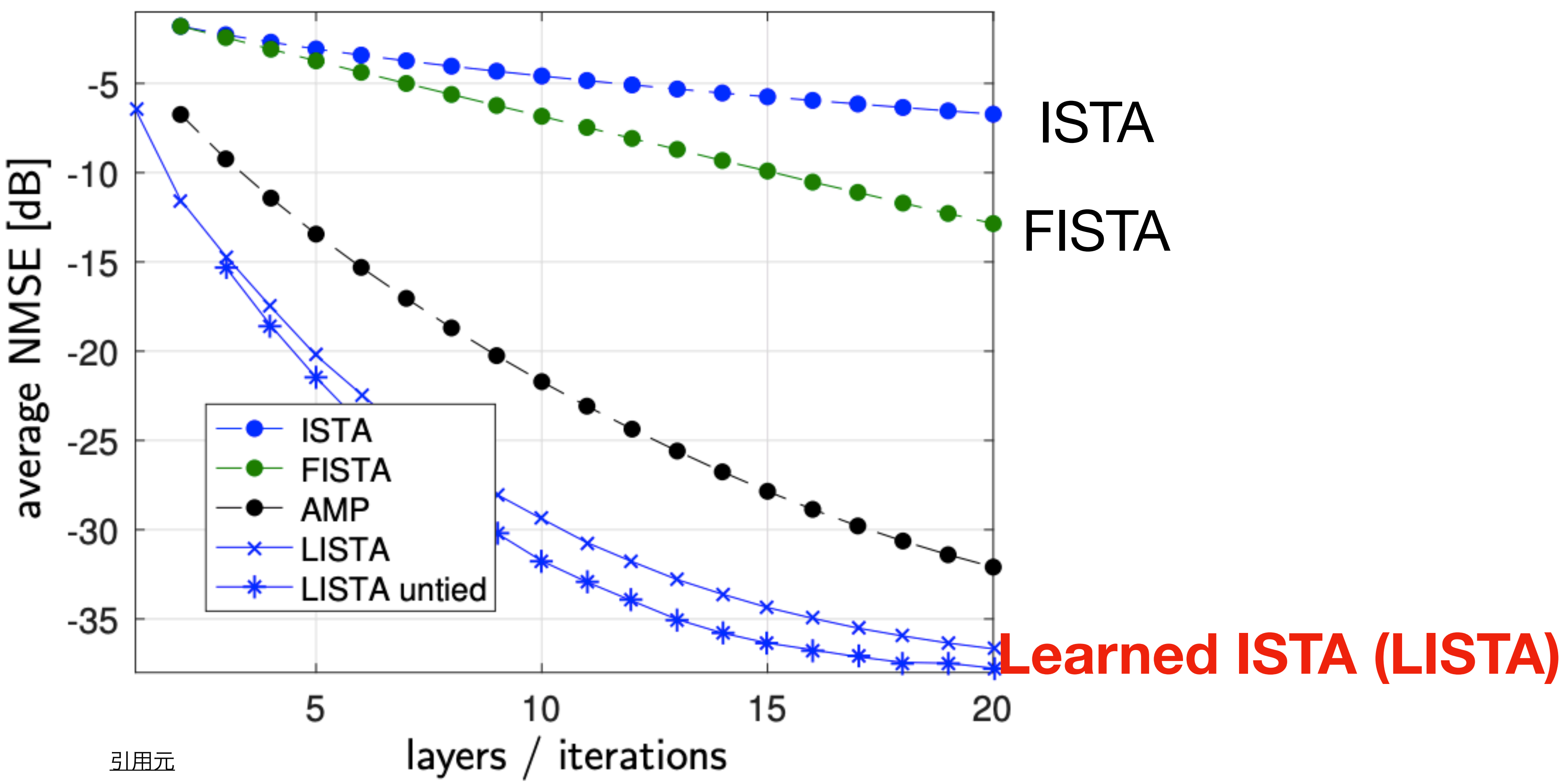
Learned ISTA (LISTA)

パラメータW_e, S, θ のT 層のニューラルネット

学習による高速化手法 | Learned ISTAの成果

▻ 更新毎の誤差の比較

○ ISTA, FISTAと比べて同イテレーション数でより早い収束

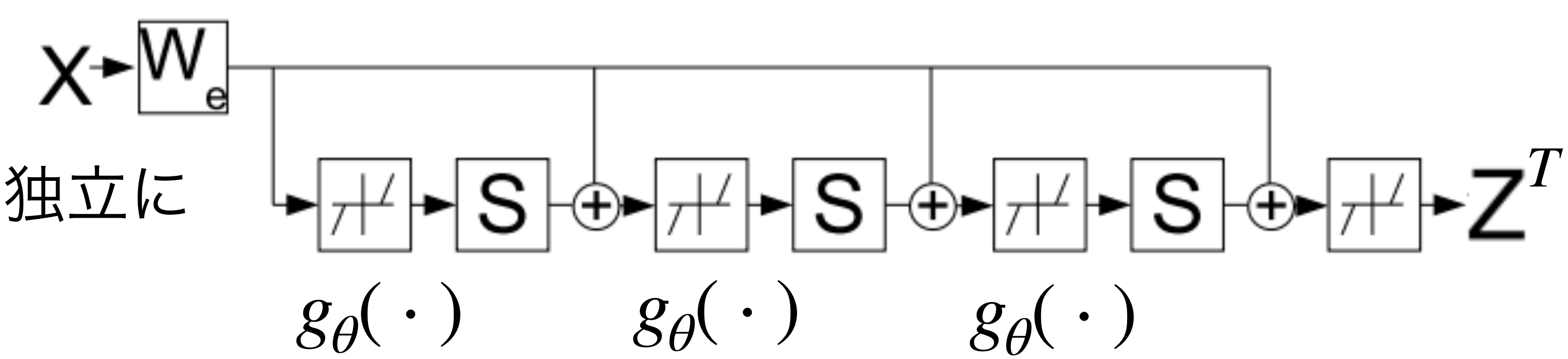


引用元

LISTA untied：

各層で用いるパラメータを独立にすることでさらなる改善

(Deep neural networks)



勾配の前処理

- ニューラルネットにNMFの目的関数に対する勾配を入力している
- 勾配の値は変動が大きい．大きすぎる・小さすぎる勾配に対処したい
- 入力する勾配に以下の前処理を適用 (左下の図中緑線)

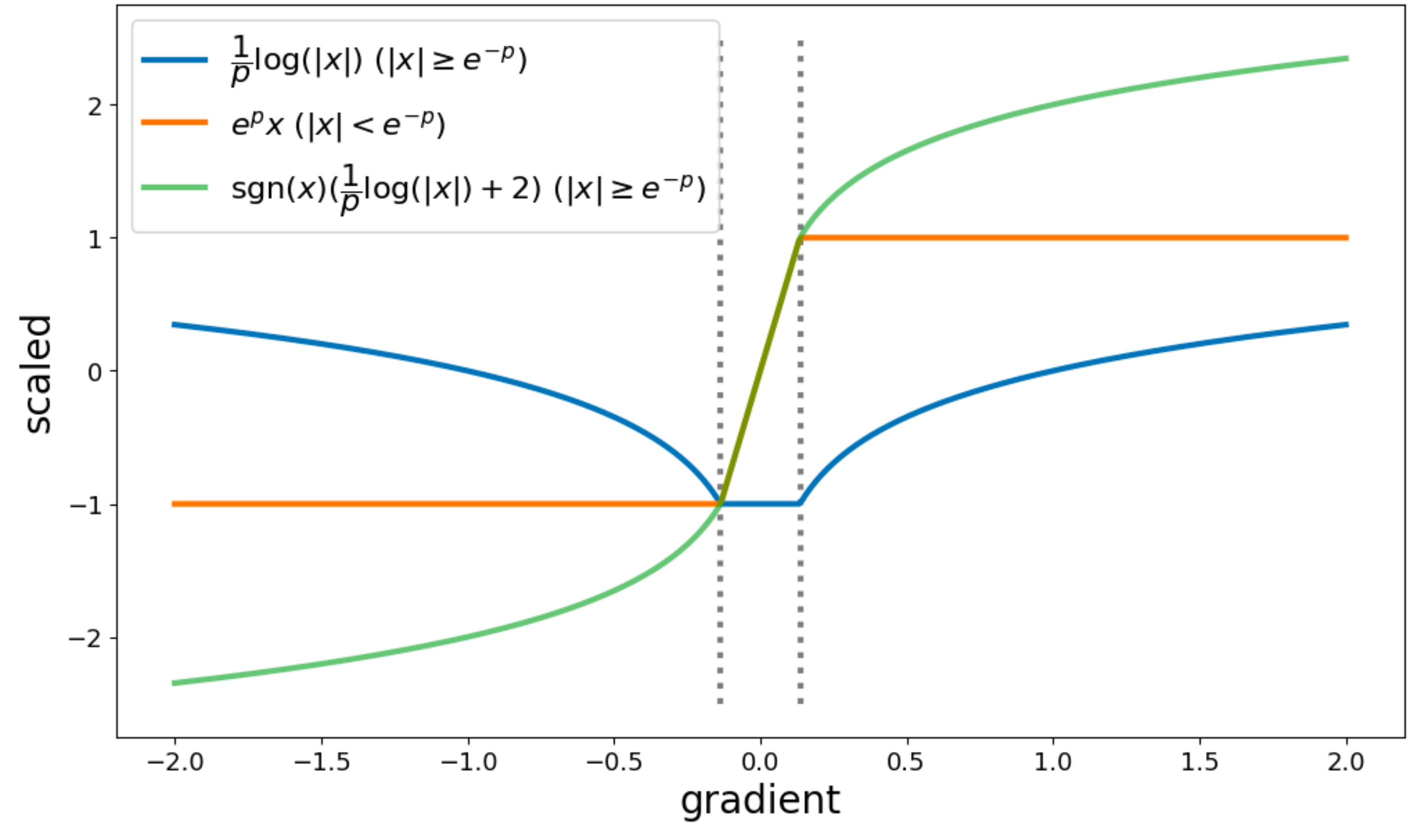
勾配の前処理

$$\nabla\mathcal{L}_{\mathrm{scaled}} = \begin{cases} \mathrm{sgn}(\nabla\mathcal{L})\left(\frac{1}{p}\log|\nabla\mathcal{L}| + 2\right) & (|\nabla\mathcal{L}| \geq e^{-p}) \\ e^{p}\nabla\mathcal{L} & \text{otherwise} \end{cases}$$

$p = 1$のとき

実験では$p = 10$

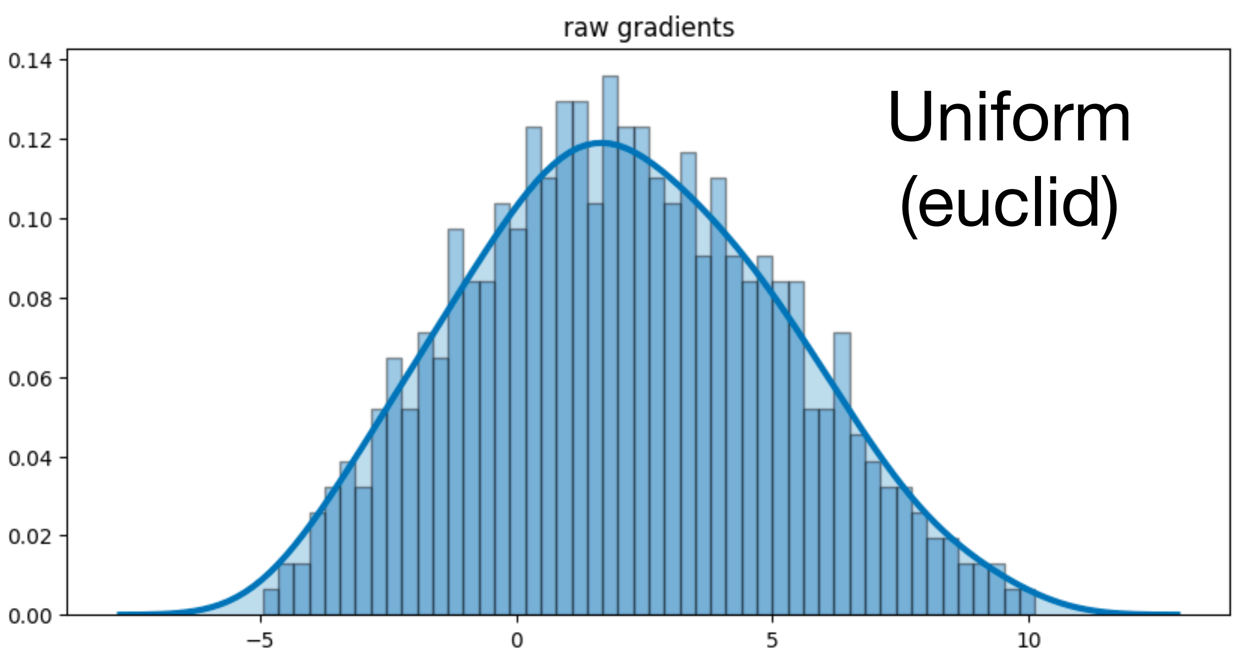
- 絶対値を取って対数変換
- 元の符号を保存
- 小さい値にはe^pをかける

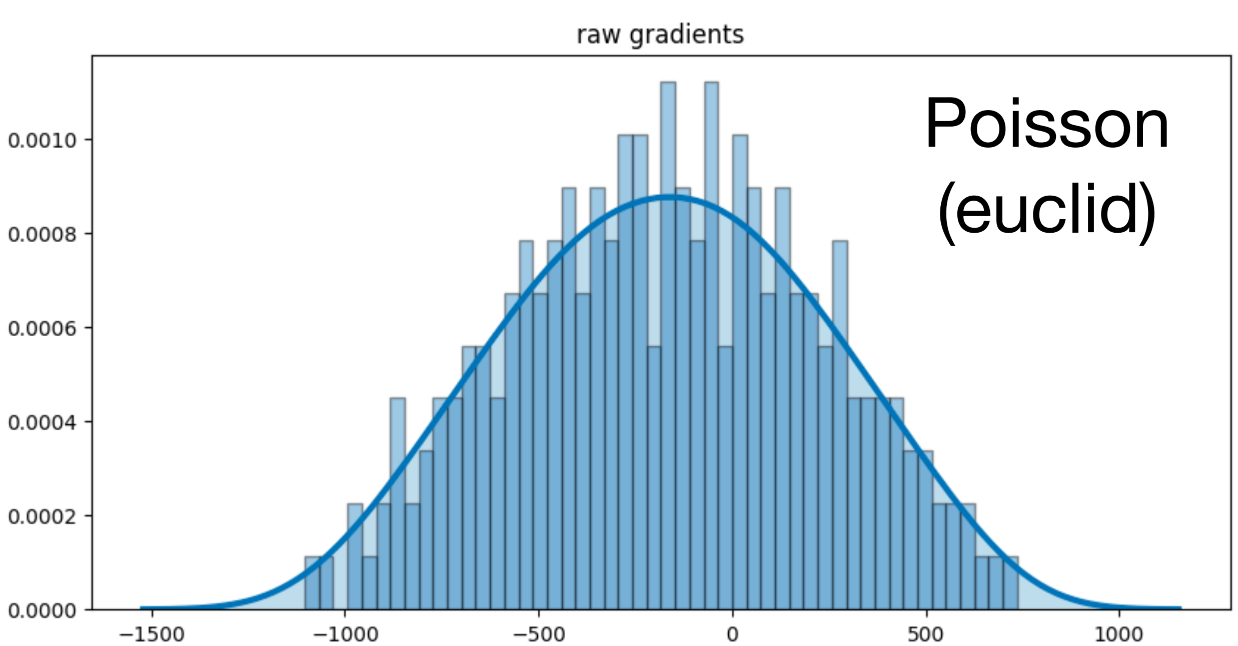


前処理結果

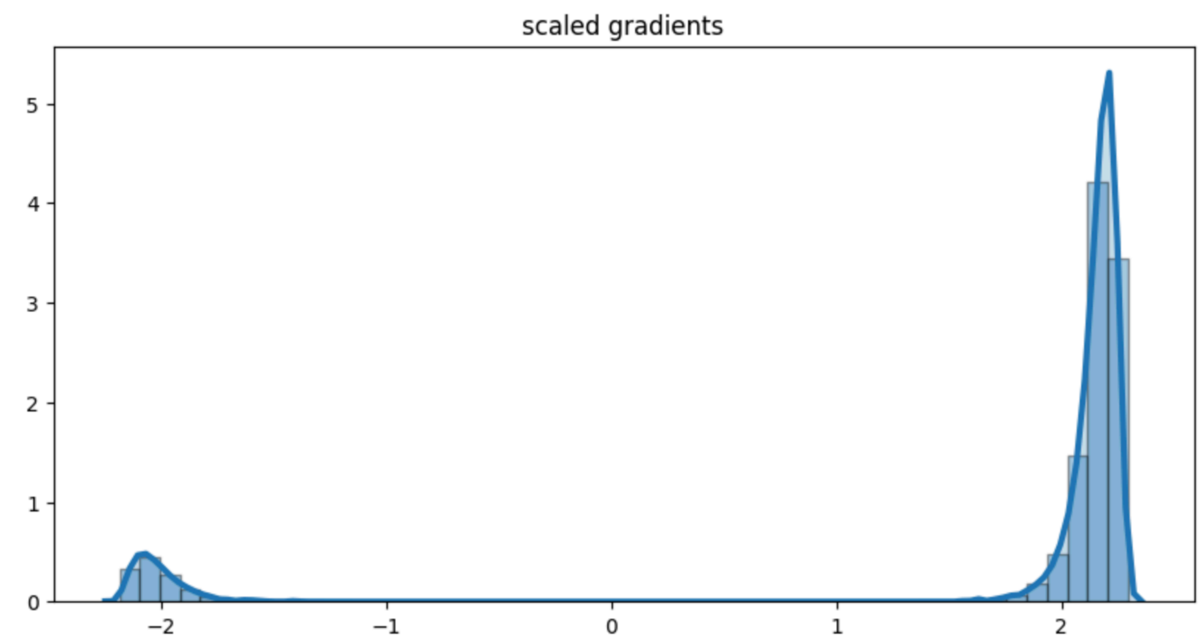
- 前処理前後の勾配の分布の変化

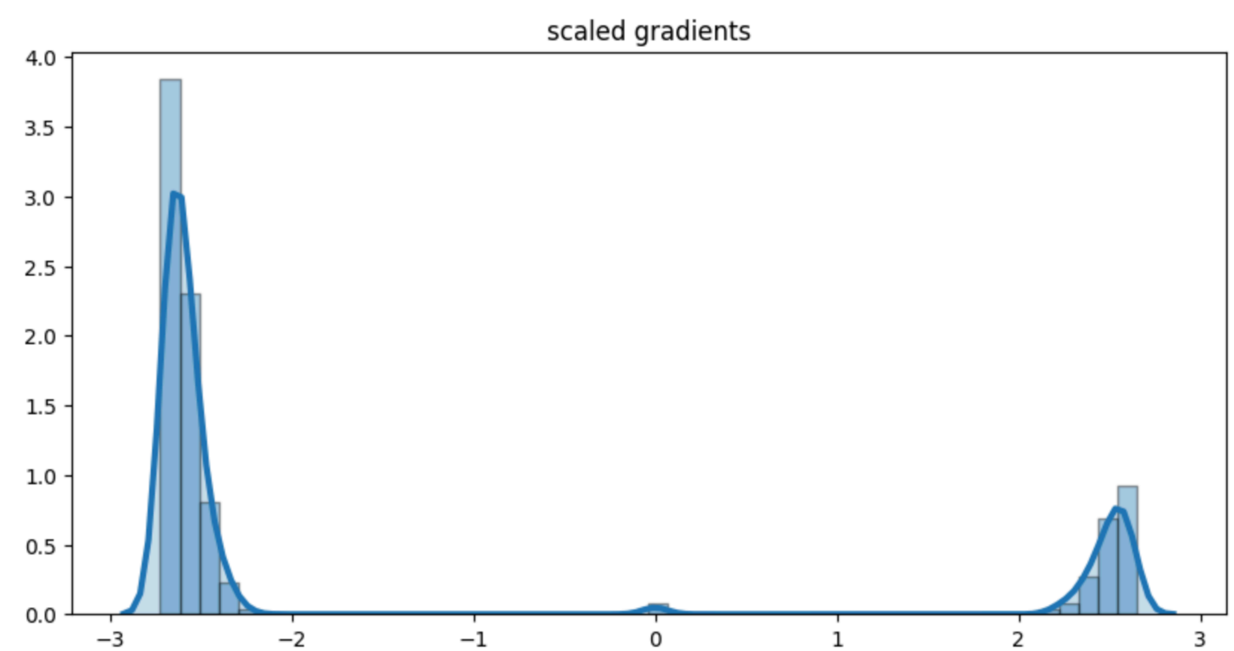
元の勾配





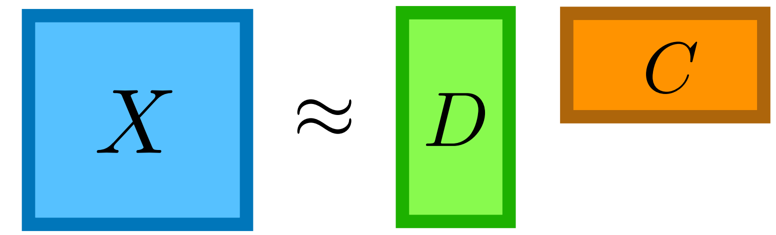
前処理後

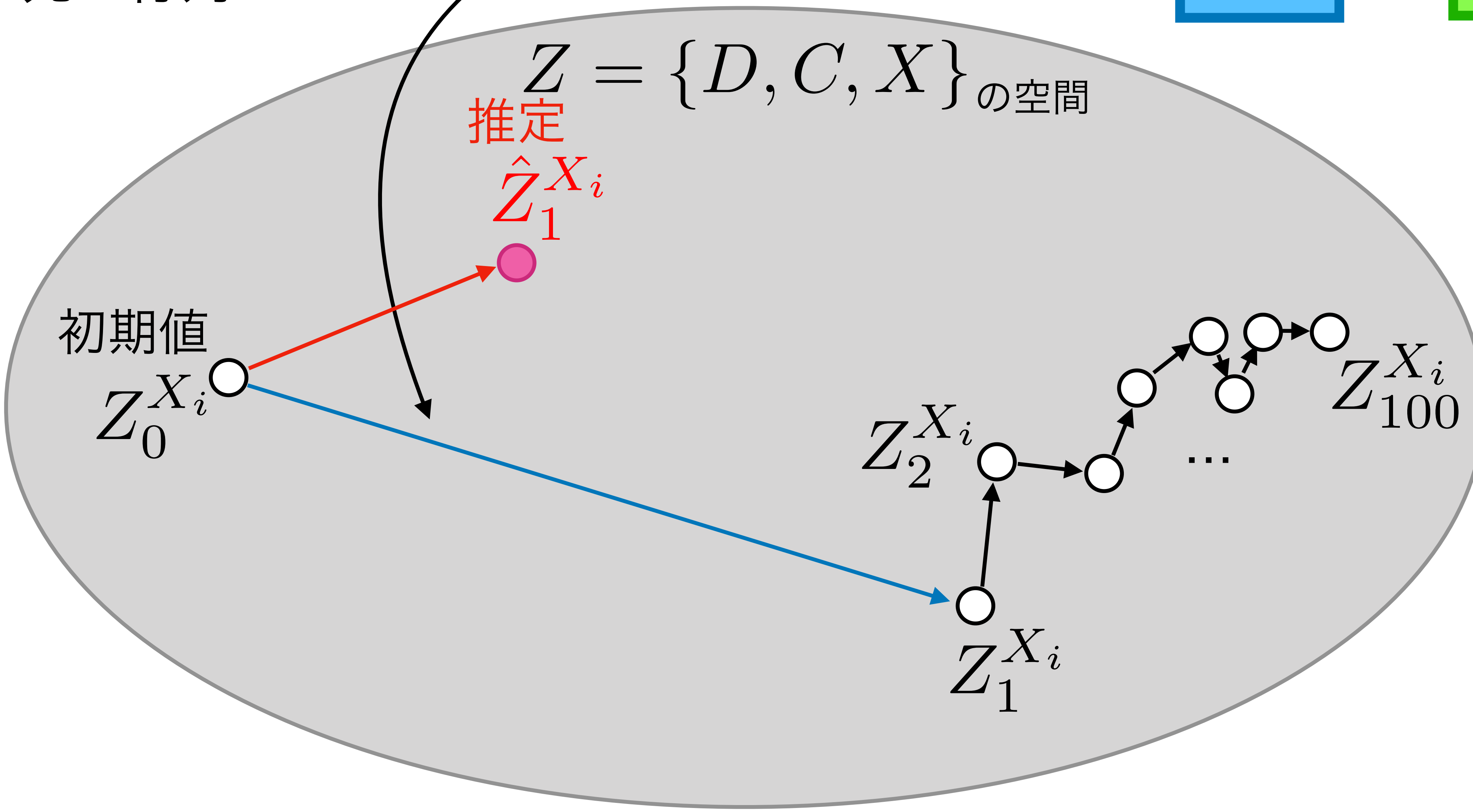




LNMFのイメージ

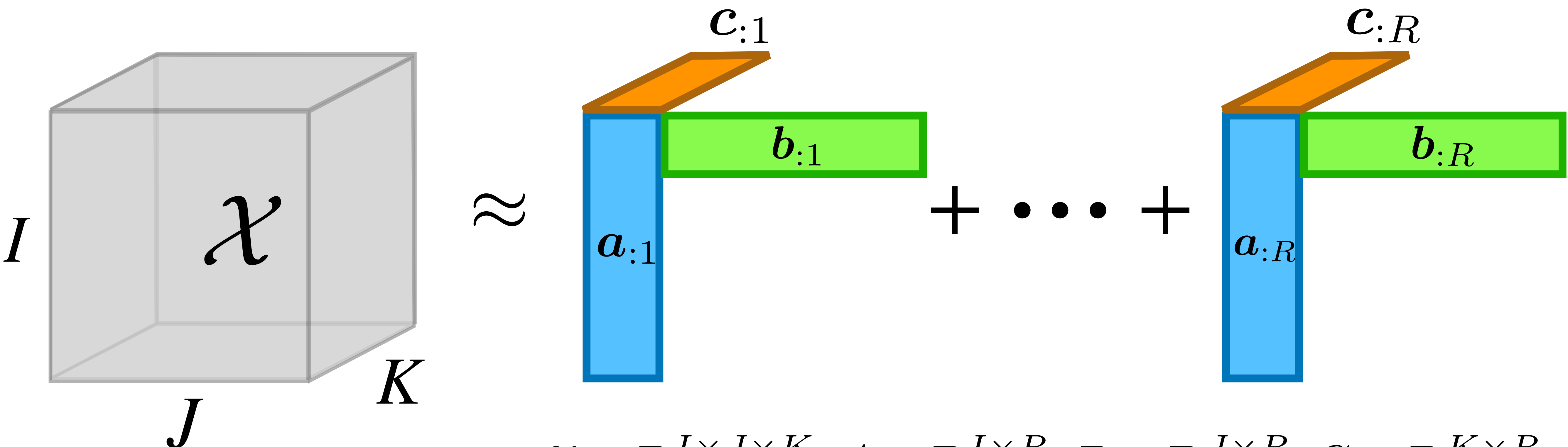
- 以下の2つを学習器への入力として，D, Cの更新分を推定する
  - 目的関数 (再構成誤差) のD, Cに対する**勾配**
  - 元の行列 X





NCP: Nonnegative CP decomposition

- 対象のテンソル分解モデル: 非負制約付きCP分解



$$\mathcal{X} \in \mathbb{R}_+^{I \times J \times K}, A \in \mathbb{R}_+^{I \times R}, B \in \mathbb{R}_+^{J \times R}, C \in \mathbb{R}_+^{K \times R}$$

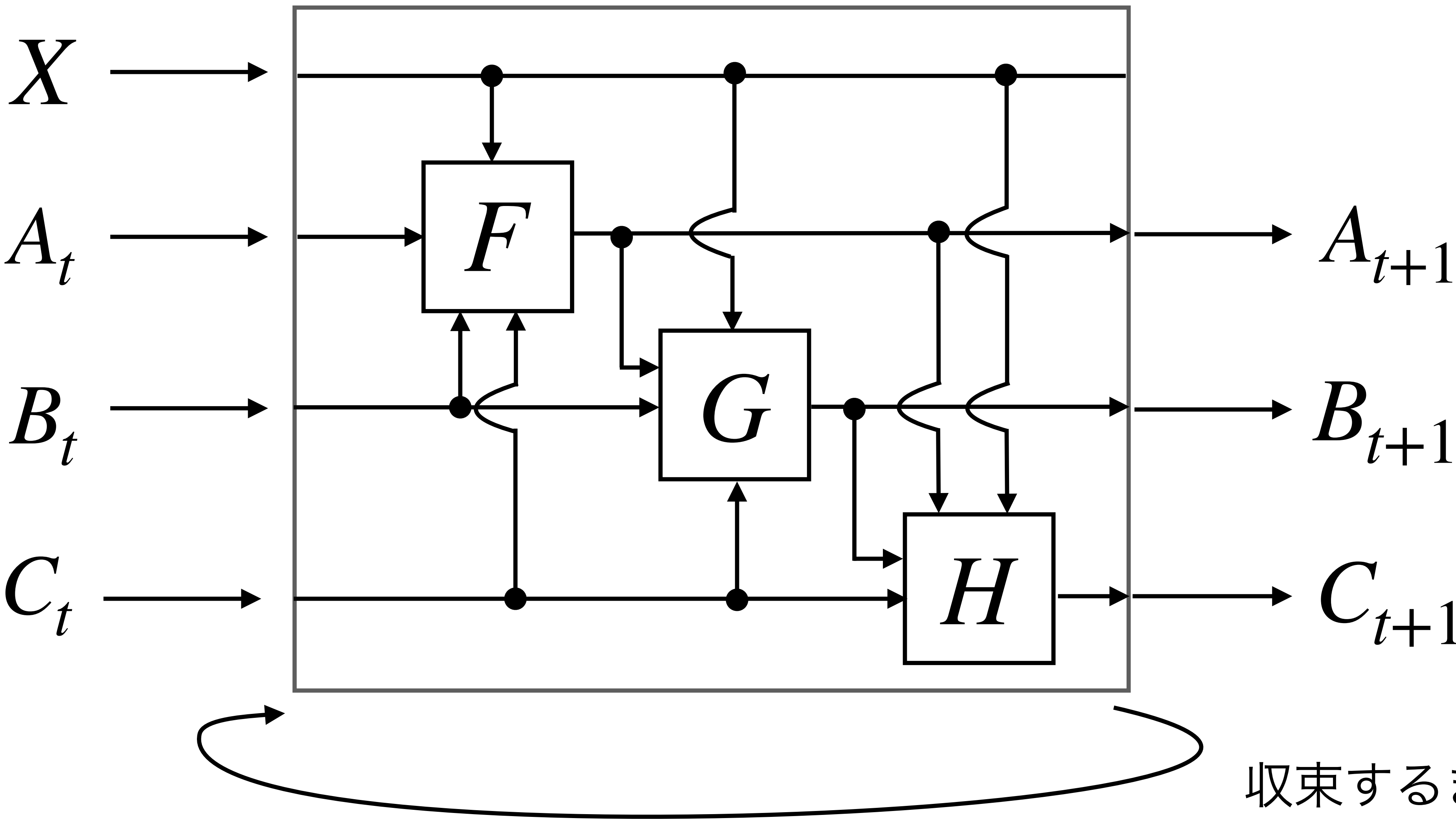
$$\mathcal{X} \approx \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \qquad x_{ijk} \approx \sum_{r=1}^{R} \lambda_r a_{ir} b_{jr} c_{kr}$$

$\circ$: outer product

$$\min_{A,B,C} \mathcal{L}_{euc}^{CP}(A,B,C) \qquad \mathcal{L}_{euc}^{CP} = \frac{1}{2} ||\mathcal{X} - \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r||_F^2$$

$$\text{s.t. } A_{ir} \geq 0, B_{jr} \geq 0, C_{kr} \geq 0$$

NCPの更新フロー



収束するまで反復

Input : X, R
$\hat{A}, \hat{B}, \hat{C}$ = Initialize(X, R)
repeat
  $\hat{A} \leftarrow F(X, \hat{A}, \hat{B}, \hat{C})$
  $\hat{B} \leftarrow G(X, \hat{A}, \hat{B}, \hat{C})$
  $\hat{C} \leftarrow H(X, \hat{A}, \hat{B}, \hat{C})$

  normalize $\hat{A}, \hat{B}, \hat{C}$
  calculate $\hat{\Lambda}$
until convergence
Output : $\hat{A}, \hat{B}, \hat{C}, \hat{\Lambda}$

Fの例： 乗法更新式 (ユークリッド距離)

$$F(X, A, B, C) = A \circledast \frac{X_{(1)}(B \odot C)}{A(B^T B \circledast C^T C)}$$

$X_{(n)}$: mode-n 行列化

$\odot$: khatri-rao積

$\circledast$: hadamard積

NCP 射影勾配法 (行ごと)

$$\min_{A,B,C} \mathcal{L}_{euc}^{CP}(A,B,C) \qquad \mathcal{L}_{euc}^{CP} = \frac{1}{2}||\mathcal{X} - \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r||_F^2$$

$$\text{s.t.} \;\; A_{ir} \geq 0, B_{jr} \geq 0, C_{kr} \geq 0$$

Input : X, R

$\hat{A}, \hat{B}, \hat{C}$ = Initialize(X, R)

repeat

  for $i \in [I]$: $\hat{\mathbf{a}}_{i,:} \leftarrow \left[\hat{\mathbf{a}}_{i,:} - \eta \nabla_{\hat{a_{i,:}}} \mathcal{L}_{euc}^{CP}\right]_+$

  for $j \in [J]$: $\hat{\mathbf{b}}_{i,:} \leftarrow \left[\hat{\mathbf{b}}_{i,:} - \eta \nabla_{\hat{b_{i,:}}} \mathcal{L}_{euc}^{CP}\right]_+$

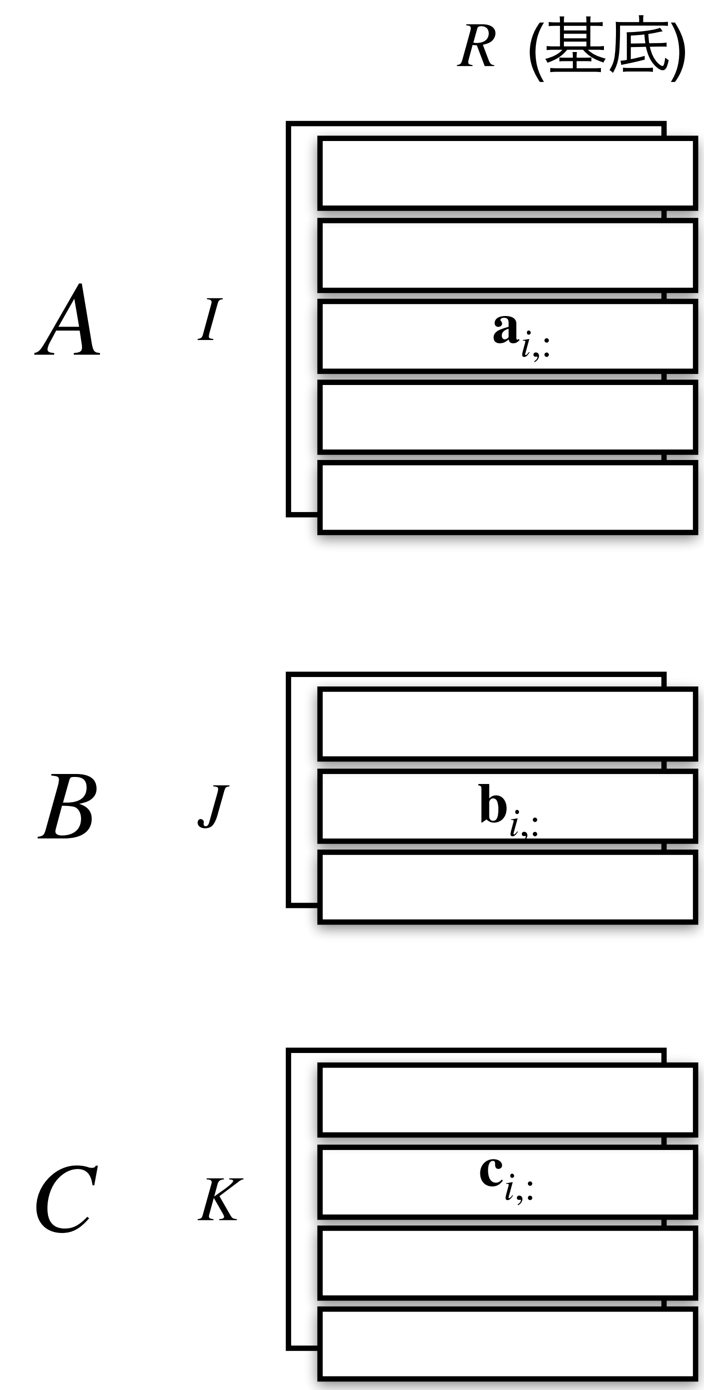
  for $k \in [K]$: $\hat{\mathbf{c}}_{i,:} \leftarrow \left[\hat{\mathbf{c}}_{i,:} - \eta \nabla_{\hat{c_{i,:}}} \mathcal{L}_{euc}^{CP}\right]_+$

  normalize $\hat{A}, \hat{B}, \hat{C}$

  calculate $\hat{\Lambda}$

until convergence

Output : $\hat{A}, \hat{B}, \hat{C}, \hat{\Lambda}$



NCP 射影勾配法 (行ごと)

$$\min_{A,B,C} \mathcal{L}^{CP}_{euc}(A,B,C) \qquad \mathcal{L}^{CP}_{euc} = \frac{1}{2}||\mathcal{X} - \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r||_F^2$$

$$\text{s.t.} \;\; A_{ir} \geq 0, B_{jr} \geq 0, C_{kr} \geq 0$$

Input : X, R

$\hat{A}, \hat{B}, \hat{C} = \text{Initialize}(X, R)$

repeat

for $i \in [I]$: $\hat{\mathbf{a}}_{i,:} \leftarrow \left[\hat{\mathbf{a}}_{i,:} - \eta \nabla_{a_{\hat{i},:}} \mathcal{L}^{CP}_{euc}\right]_+$

for $j \in [J]$: $\hat{\mathbf{b}}_{i,:} \leftarrow \left[\hat{\mathbf{b}}_{i,:} - \eta \nabla_{b_{\hat{i},:}} \mathcal{L}^{CP}_{euc}\right]_+$

for $k \in [K]$: $\hat{\mathbf{c}}_{i,:} \leftarrow \left[\hat{\mathbf{c}}_{i,:} - \eta \nabla_{c_{\hat{i},:}} \mathcal{L}^{CP}_{euc}\right]_+$

nor

calculate $\hat{\Lambda}$

until convergence

Output : $\hat{A}, \hat{B}, \hat{C}, \hat{\Lambda}$

LNMFと同様勾配とX (モード行列化) をNNに入れる

(Learned NCP)