# パーシステントホモロジーと機械学習

ディープラーニングと物理学/学習物理領域セミナー
2024-02-15

九州大学 マス・フォア・インダストリ研究所
池 祐一

- 富士通研究所およびInriaとの共同研究
- 学術変革領域 (A)データ記述科学の創出と諸分野への横断的展開

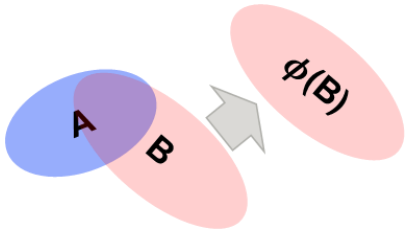
# 自己紹介

## 池 祐一（いけ ゆういち）1990年新潟生まれ

■東大数理で博士取得
（層理論とシンプレクティック幾何）

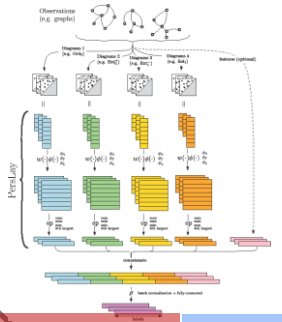
■富士通研究所入社

2018




■東京大学に異動

2020



■九州大学に異動

■教科書を出版

2022

富士通

東京大学

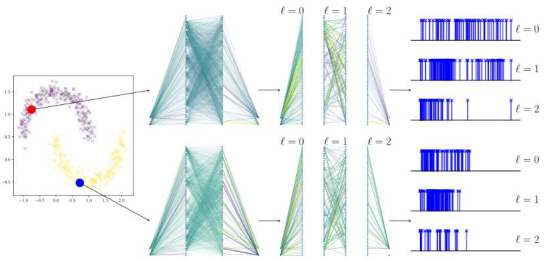
九大IMI

2019

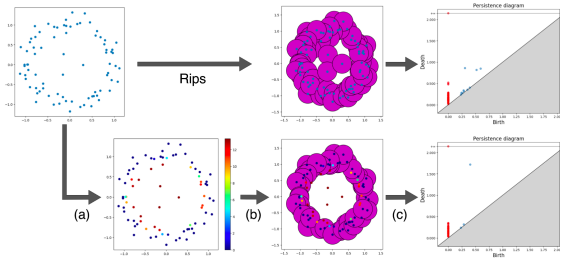
■フランスInriaとの共同研究のため2ヶ月フランス滞在

■ACT-X採択

2021



2023

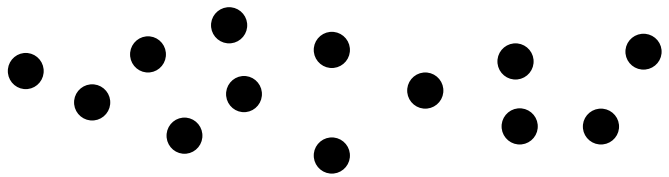


# 位相的データ解析

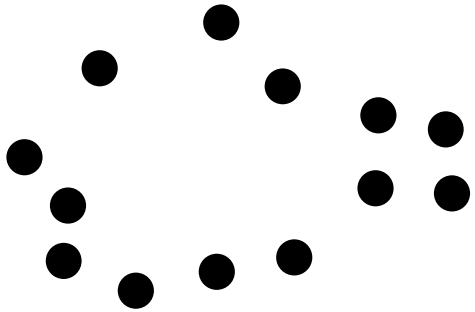
- パーシステントホモロジー (PH)
- フィルトレーションとパーシステンス図
- PHと機械学習のつながりいろいろ

# 位相的データ解析（TDA）のアイデア

■データの**大まかな形**（**トポロジー**）を用いて解析する手法



**穴がない**

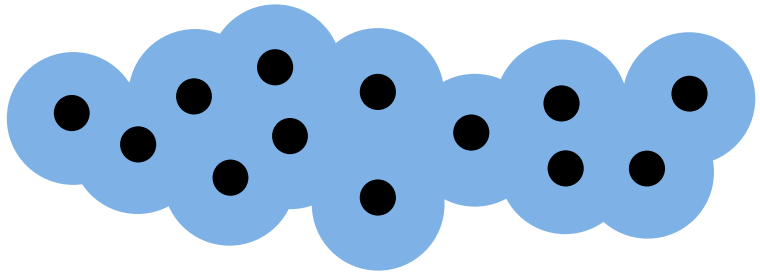


**穴が1つ**

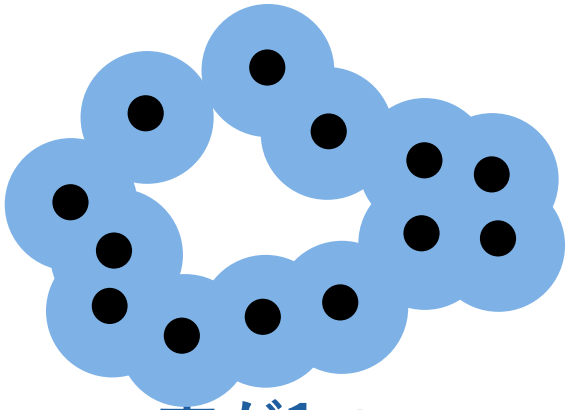
■Q. どうやって離散的なデータから「トポロジー」を取り出すか？

# 位相的データ解析（TDA）のアイデア

■データの**大まかな形**（**トポロジー**）を用いて解析する手法



**穴がない**



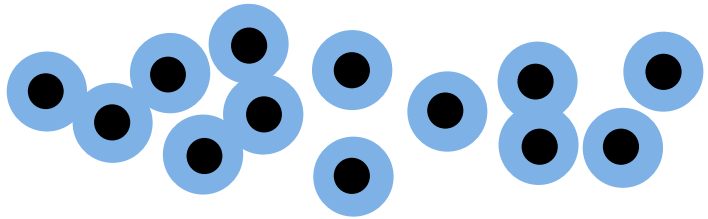
**穴が1つ**

■Q. どうやって離散的なデータから「トポロジー」を取り出すか？

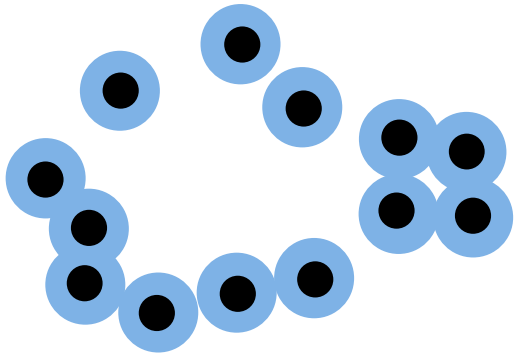
■アイデア1：**データ点中心の閉球の和集合**を考える

# 位相的データ解析（TDA）のアイデア

■データの**大まかな形**（**トポロジー**）を用いて解析する手法



**穴がない**



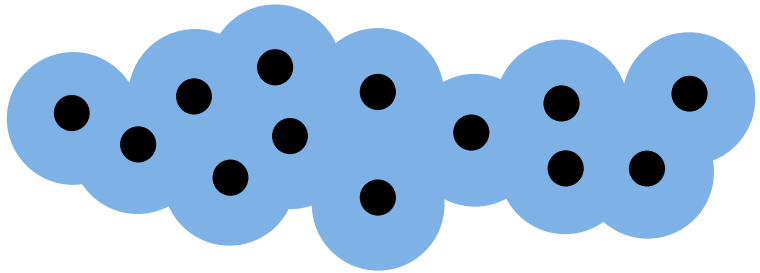
**穴が1つ**

■Q. どうやって離散的なデータから「トポロジー」を取り出すか？

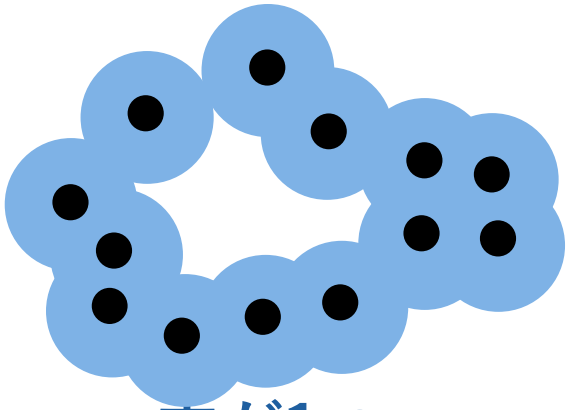
■アイデア1：**データ点中心の閉球の和集合**を考える
→ **どのように半径を設定すればよいか分からない**

# 位相的データ解析（TDA）のアイデア

■データの**大まかな形**（**トポロジー**）を用いて解析する手法



**穴がない**



**穴が1つ**

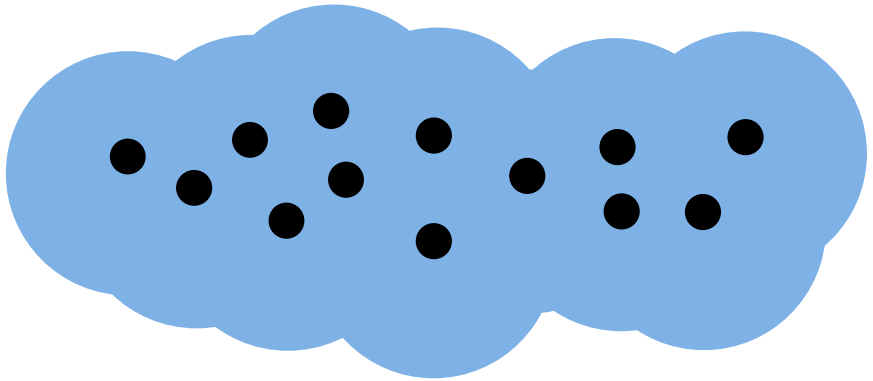
■Q. どうやって離散的なデータから「トポロジー」を取り出すか？

■アイデア1：**データ点中心の閉球の和集合**を考える
→ **どのように半径を設定すればよいか分からない**

■アイデア2：**すべての半径**を考えて**トポロジーの発展を追跡**する
**パーシステントホモロジー**と呼ばれる道具

# 位相的データ解析（TDA）のアイデア

■データの**大まかな形**（**トポロジー**）を用いて解析する手法



**穴がない**

■Q. どうやって離散的なデータから「トポロジー」を取り出すか？

■アイデア1：**データ点中心の閉球の和集合**を考える
→ **どのように半径を設定すればよいか分からない**

■アイデア2：**すべての半径**を考えて**トポロジーの発展を追跡**する
**パーシステントホモロジー**と呼ばれる道具
→ **ノイズと本質的な特徴量を区別できる・マルチスケールの解析**

# フィルトレーションとパーシステンス図

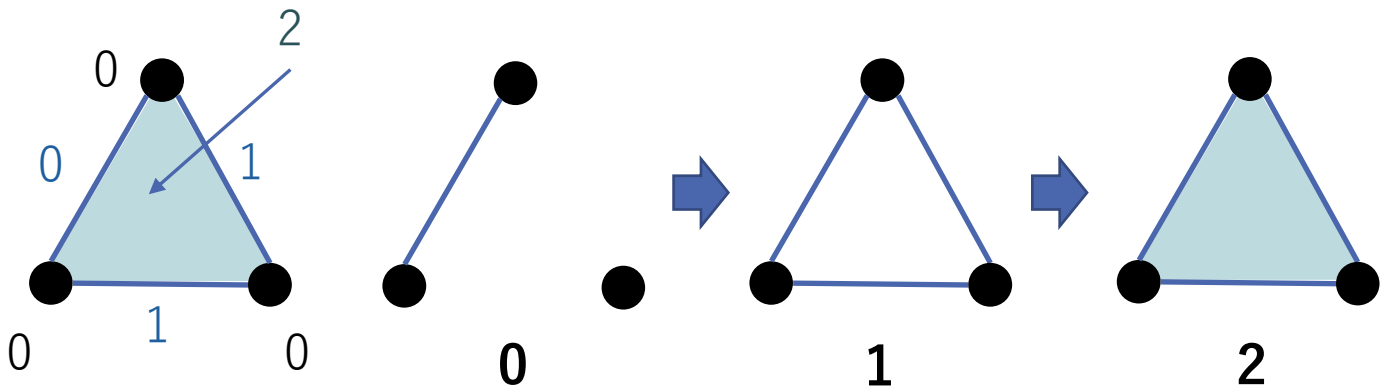
**フィルトレーション**: 単体的複体の増大族

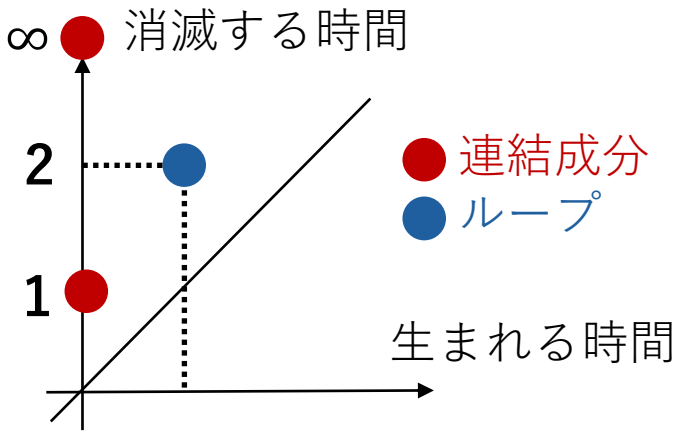
■**単体的複体**: 有限集合$V$の部分集合の集合$K$であって次を満たす：

$$\sigma \in K, \tau \subset \sigma \Rightarrow \tau \in K$$

■$\mathcal{K} = (K_r)_r, K_r \subset K$が$K$のフィルトレーション: $\Leftrightarrow K_r \subset K_s\ (r \leq s)$, $\bigcup_r K_r = K$

$\leftrightarrow$ 関数$F: K \to \mathbb{R}$ s.t. $\sigma \subset \tau \Rightarrow F(\sigma) \leq F(\tau)$, $K_r = \{\sigma \in K \mid F(\sigma) \leq r\}$

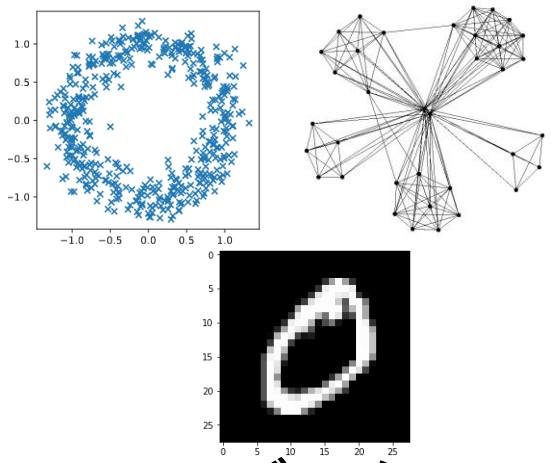


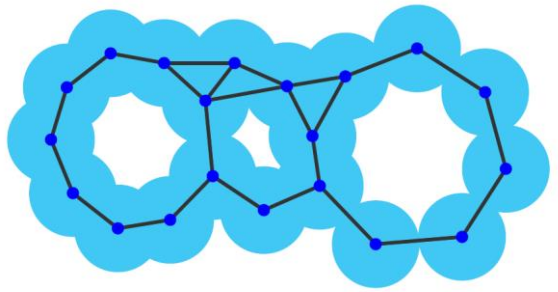

⇝ **パーシステンス図 (PD)**: 各連結成分やループといった「穴」の**生成・消滅時刻**を2Dにプロット

# データからパーシステンス図まで

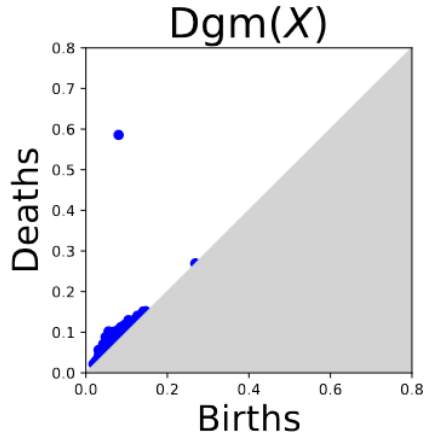
## データからフィルトレーションを定めることでPDを計算する



データ



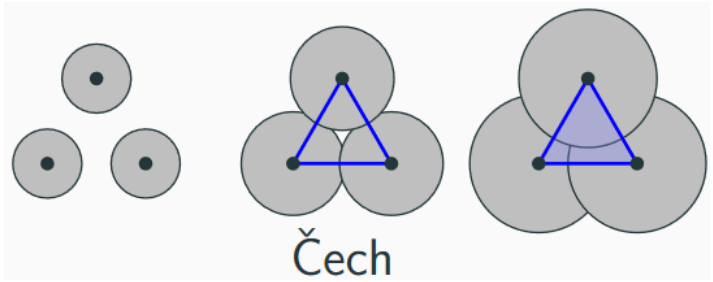
フィルトレーション




パーシステンス図

### Čechフィルトレーション

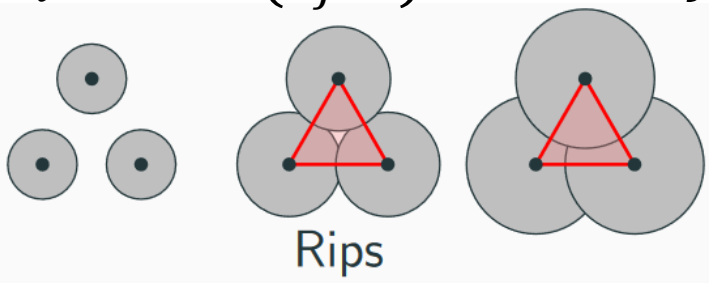
$\{x_0, \dots x_k\}$: 単体:$\Leftrightarrow \bigcap_i B(x_i; r) \neq \emptyset$




### Ripsフィルトレーション

$\{x_0, \dots x_k\}$: 単体 :$\Leftrightarrow$

$B(x_i; r) \cap B(x_j; r) \neq \emptyset \ (\forall i, j)$




### 劣位集合フィルトレーション

$f: V(K) \to \mathbb{R}$ 頂点上の関数

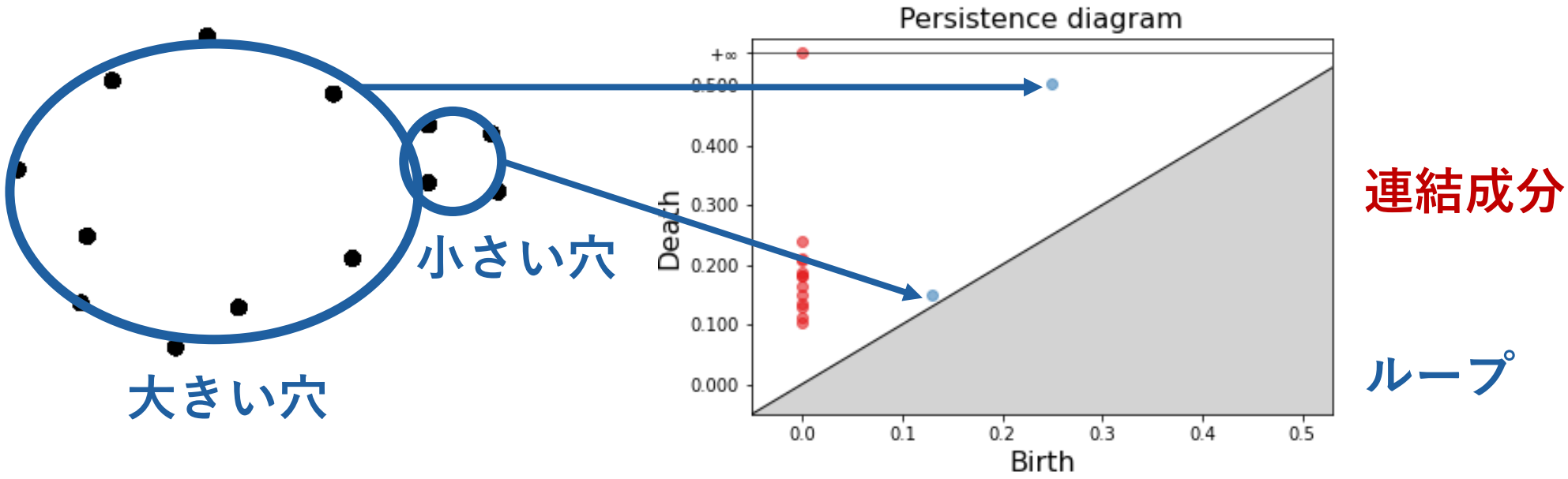
$F(\sigma) := \max_{v \in \sigma} f(v)$

グラフ・画像データで使われる

# PDは何がうれしいのか (1/2)

■対角線の遠くが本質的な形をあらわし，対角線の近くはノイズであると区別ができる

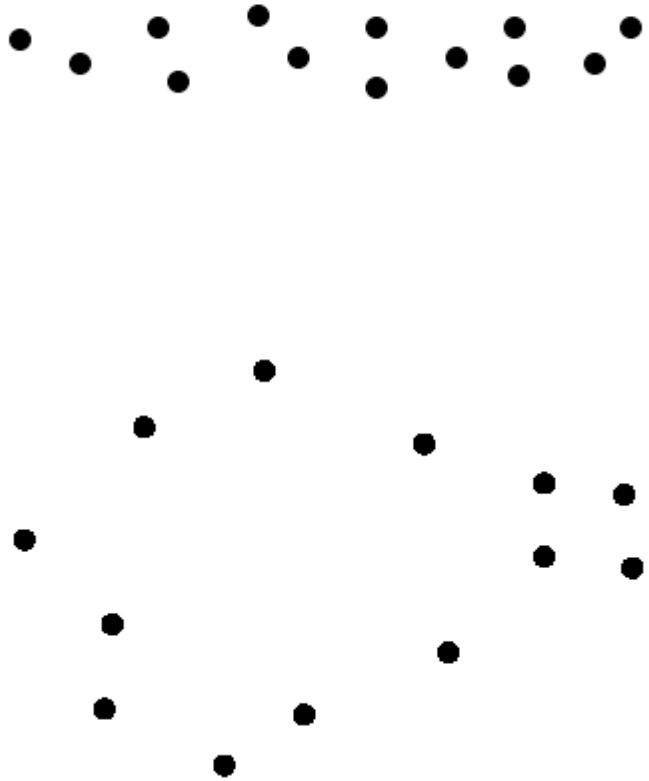
■パーシステンス図の各点が，**データのどのような「形」を意味するか**も分かる

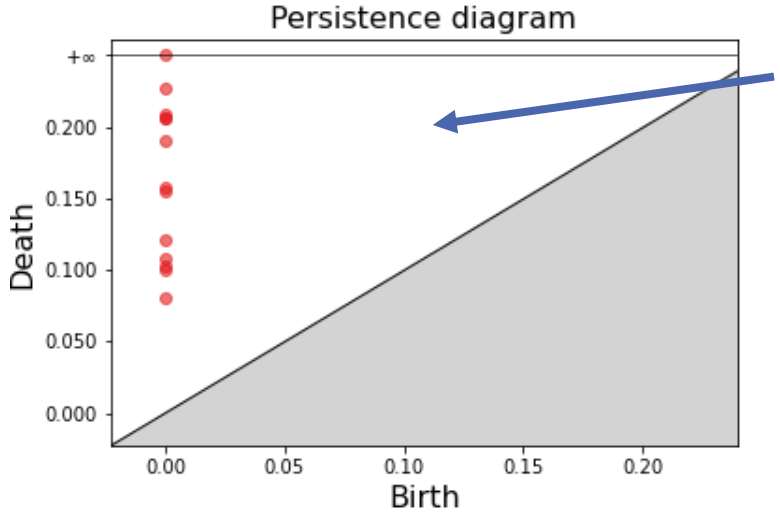


# PDは何がうれしいのか (2/2)

もっと直接的には．．．

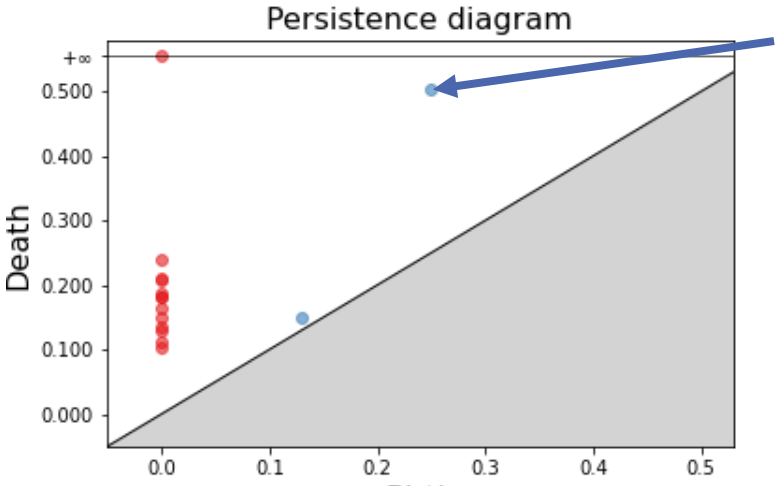
**「形」が異なるデータをパーシステンス図で区別**できる






穴がない

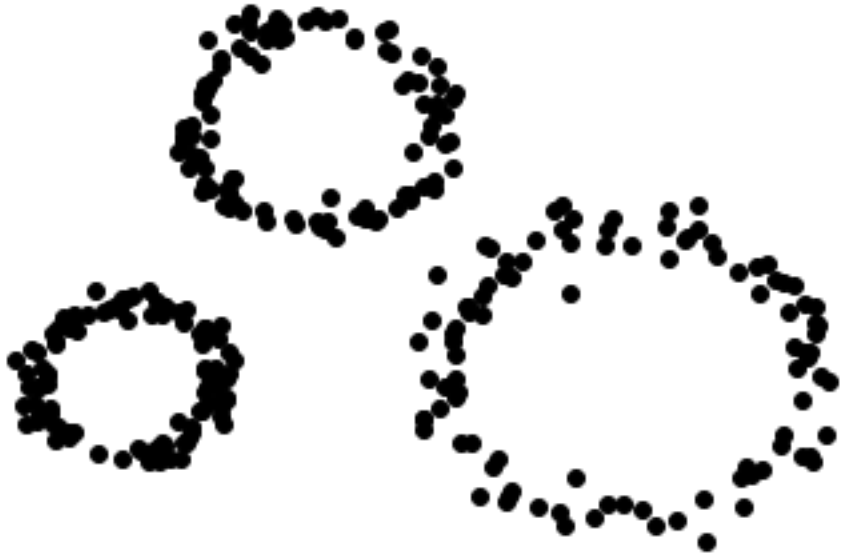
連結成分



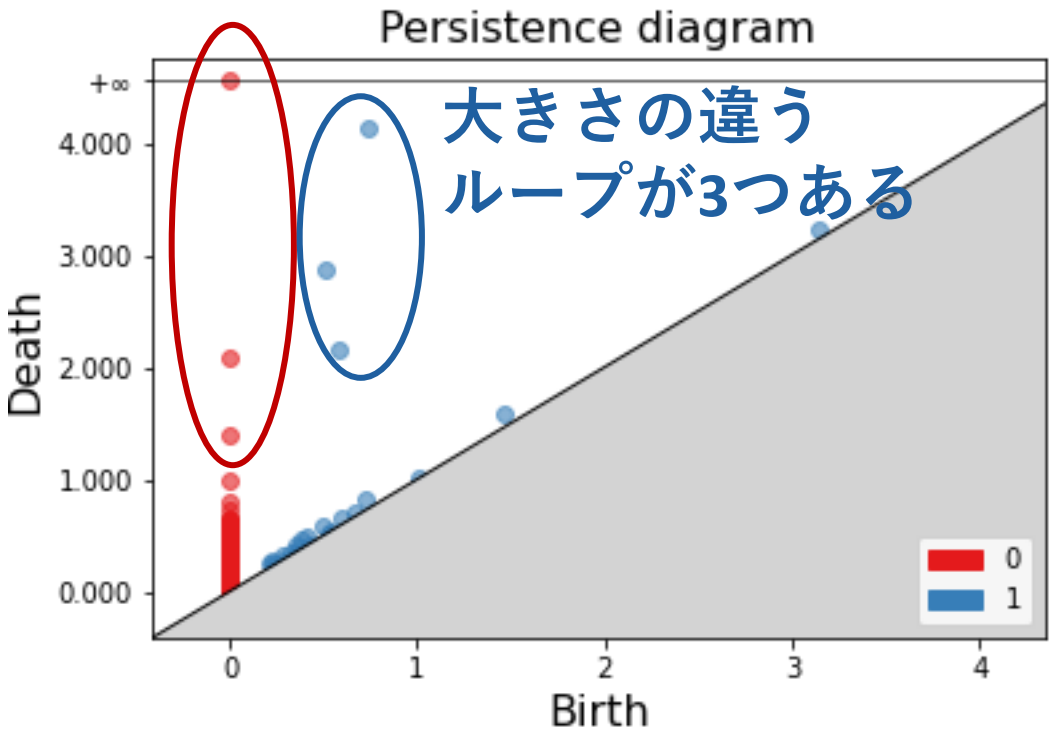

穴がある

ループ

# PHの簡単な使用例



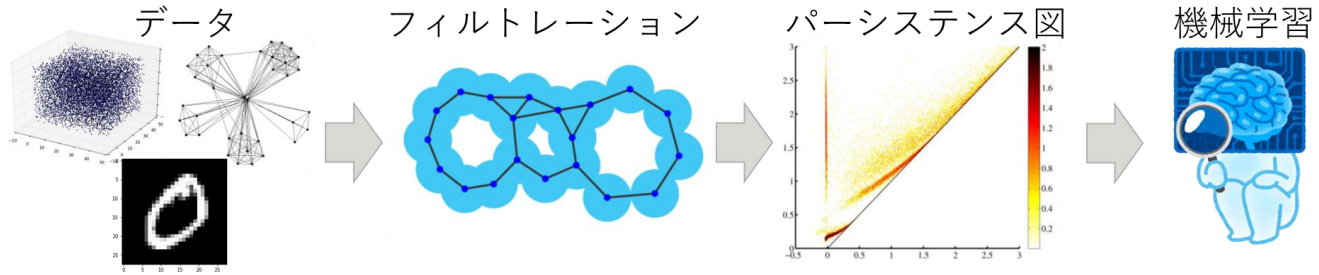
連結成分が3つある



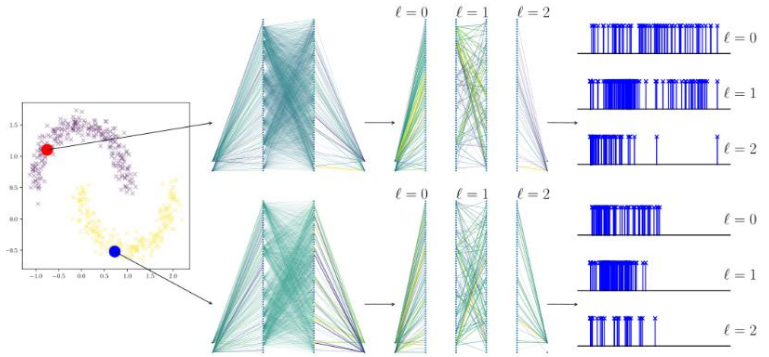
# PHと機械学習のつながり（1/2）

## 1. PHを機械学習へ応用する

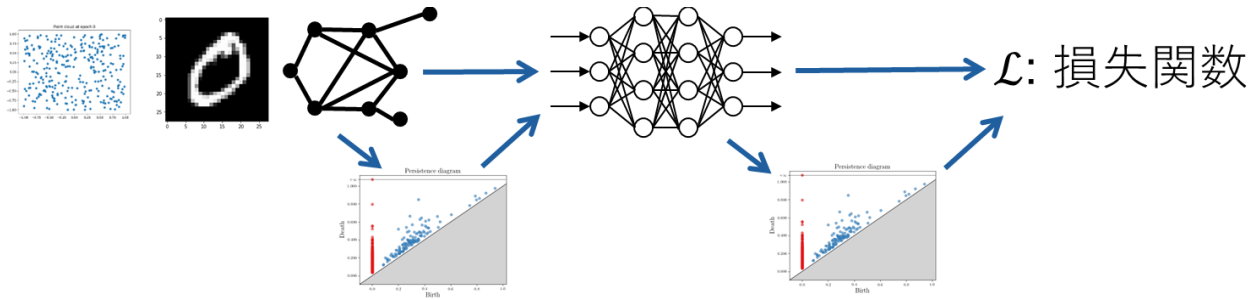
■PHを機械学習の入力に使う




■ネットワークからPH的特徴量を取り出す



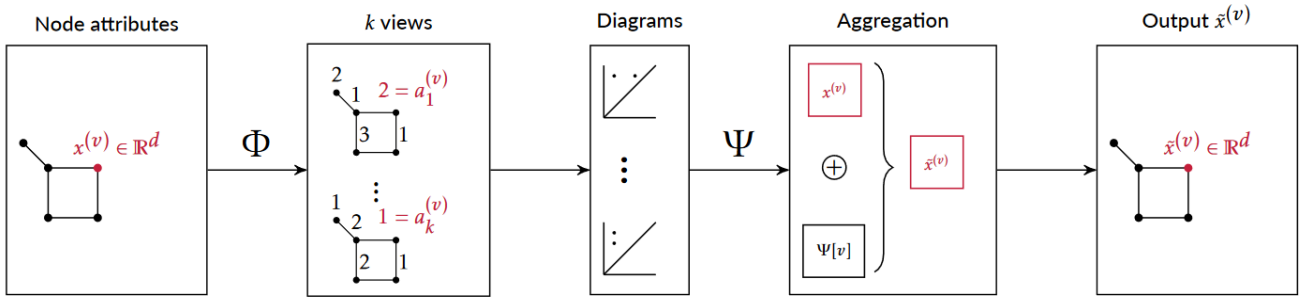
■PHを損失関数に組み込む

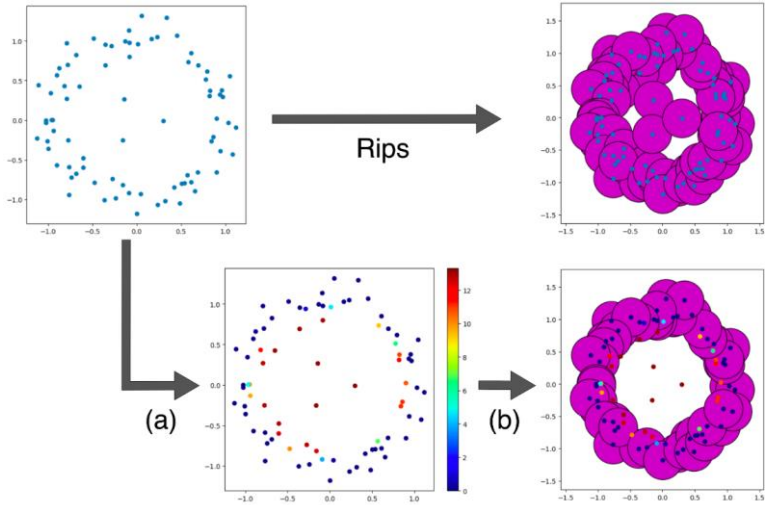


# PHと機械学習のつながり（2/2）

## 2. 機械学習からPHへフィードバックする

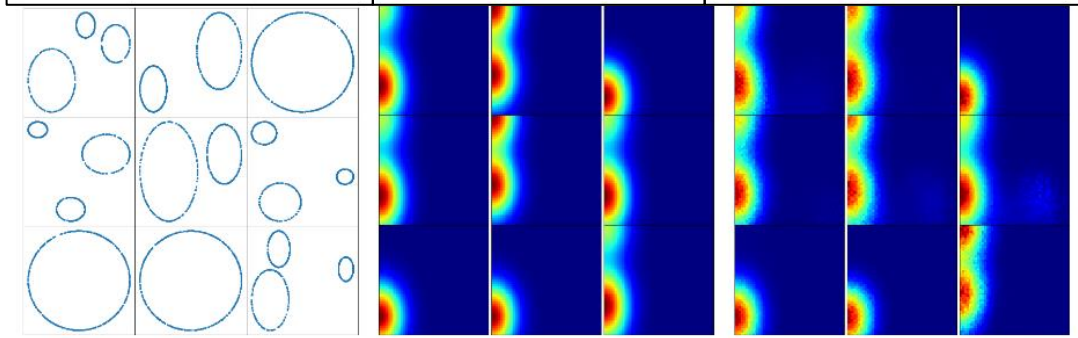
### ■フィルトレーションを学習する




### ■PDを計算する関数を機械学習で近似する

| データ | 本物の出力 | 機械学習で近似 |
|---|---|---|

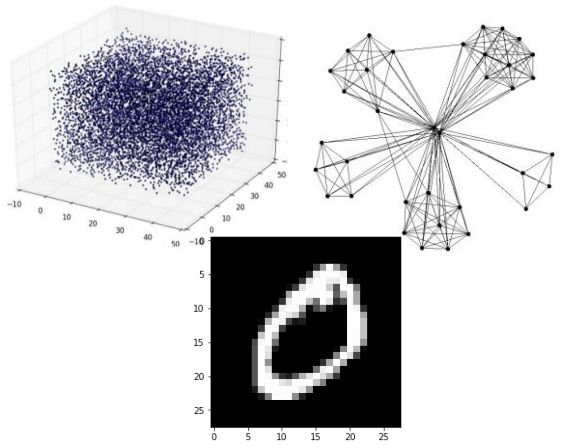


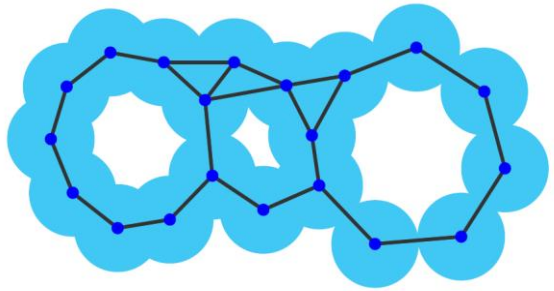
# PHを機械学習の入力に使う

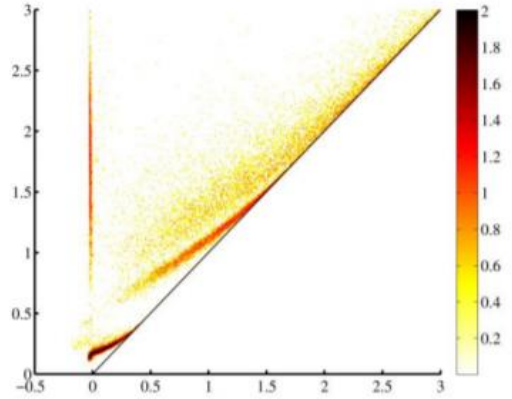
- トポロジー情報を使って機械学習を行う
- パーシステンス図のベクトル化
- いろいろな応用

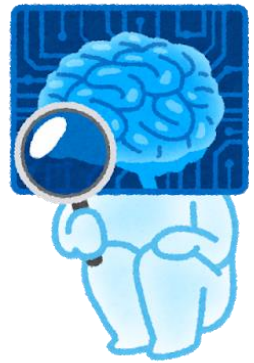
# PHを機械学習の入力とするパイプライン

データ　　フィルトレーション　　パーシステンス図　　機械学習









**例**
- シミュレーションで生成された点群
- センサーによる振動
- グラフ・画像

**Software**
Ripser, GUDHI, HomCloud, …

**機械学習に入力するためにベクトル化**

Figures from K. Fukumizu, Persistence Weighted Gaussian Kernel for Topological Data Analysis

# パーシステンス図のベクトル化 1

パーシステンス図の**ベクトル化手法**が多く提案されている

■PHと機械学習を組み合わせる際に有用

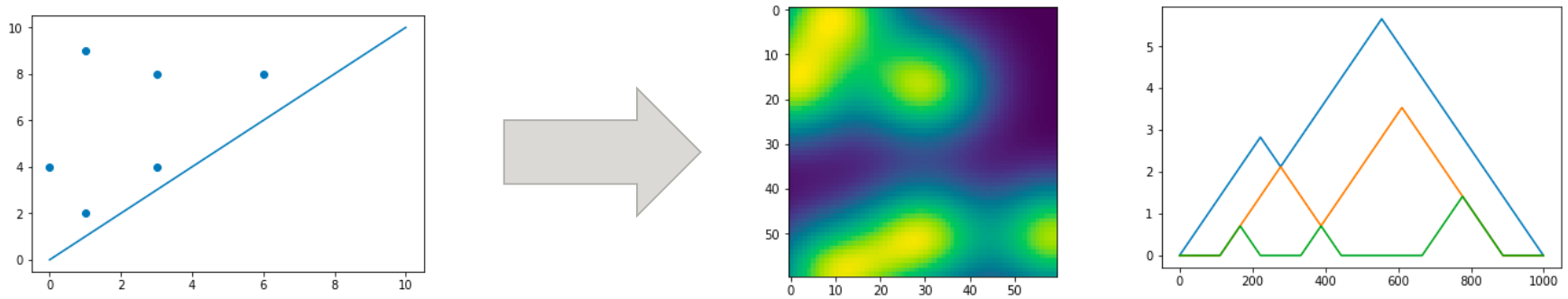
■**パーシステンスイメージ** (PI)：ガウス関数によるカーネル密度推定：

$$\rho(D)(z) \coloneqq \sum_{u \in D} w(u) \cdot \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|z-u\|^2}{2\sigma^2}\right), \qquad w\colon \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$$

領域を分割してそこで$\rho(D)$を積分することで有限次元ベクトルを得る

■**パーシステンスランドスケープ** (PL)：三角状の関数を用いて定義

$$\lambda_k(D)(t) \coloneqq k - \max \min\{t - b_i, d_i - t\}_+, \qquad D = \{(b_i, d_i)\}_i$$



■**Weighted Gaussian kernel**, …

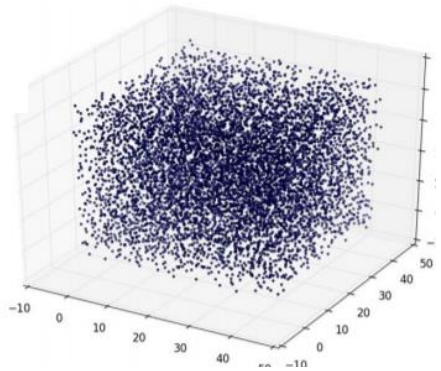
# シリカガラスの相転移温度解析

**G. Kusano, K. Fukumizu, Y. Hiraoka: Persistence weighted Gaussian kernel for topological data analysis, 2016**

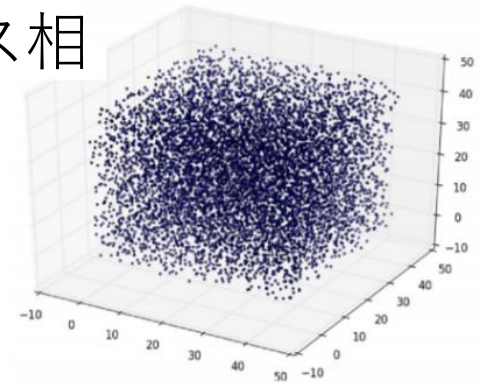
■$SiO_2$が液相からガラス相に変化する温度を原子配置の点群から推定

■**点群をPDに変換**して，それが変化する温度を調べる

液相

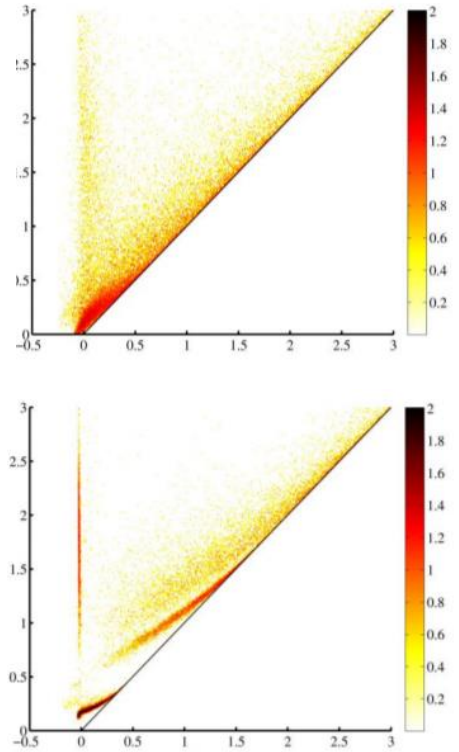


ガラス相



PD

1次の穴



※点が多いのでPDを密度で表示

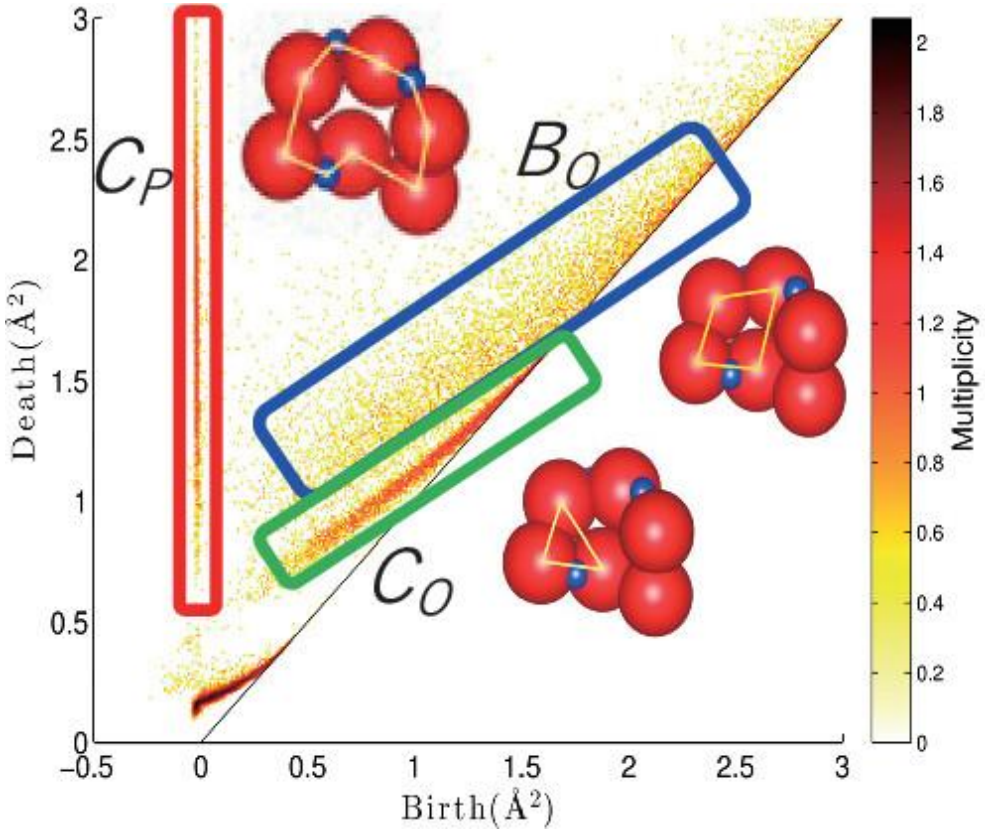
**変化点は従来手法で説明される相転移温度に合致**
（カーネル法を利用）

Figures from K. Fukumizu, Persistence Weighted Gaussian Kernel for Topological Data Analysis

# シリカガラスの解析：PDの意味

$SiO_2$のガラス相のPDに特徴的な曲線の意味を調べる

**PDの点が**原子配置点群の**どの形に対応するかを出力可能（逆解析）**




- $C_P$: ⋯-O-Si-O-Si-O-⋯という共有結合によるリング
- $C_O$: Si原子をまたぐ3つのO原子が作る三角形
- $B_O$: いろいろな構造が混在
  たとえば，Si原子をまたいだ4個以上のO原子のなす多角形
- $C_O, B_O, C_P$が**中距離秩序**に対応すると信じられている

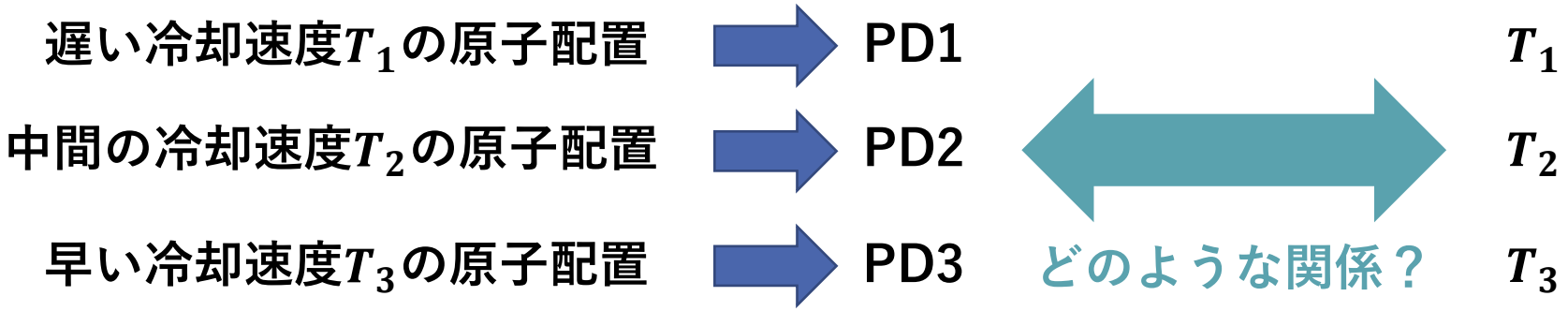
Figure from Obayashi et al., Persistent homology analysis for materials research and persistent homology software: Homcloud

# 金属ガラスの解析

**Hirata et al., Structural changes during glass formation extracted by computational homology with machine learning, 2020**

Pd 80%とSi 20%からなる金属ガラスの解析

- ■高速冷却のシミュレーションで原子配置を用意
- ■複数の冷却速度でデータを生成
- ■原子配置のPDと冷却速度の関係を調べた（ベクトル化+線形回帰）

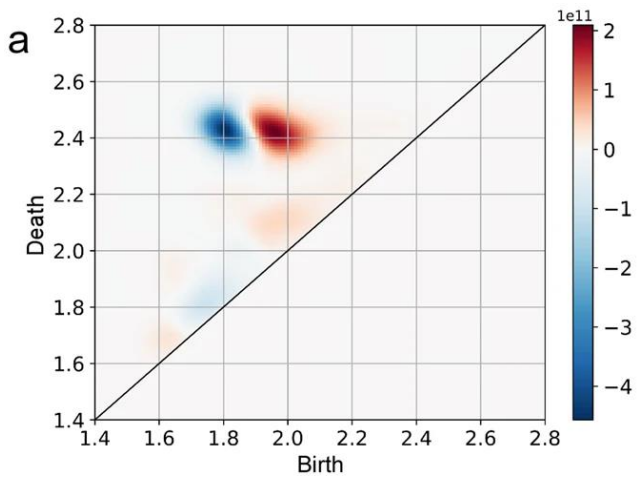



1. PDをパーシステンスイメージに変換する
2. 冷却温度とパーシステンスイメージに対して線形回帰を行う

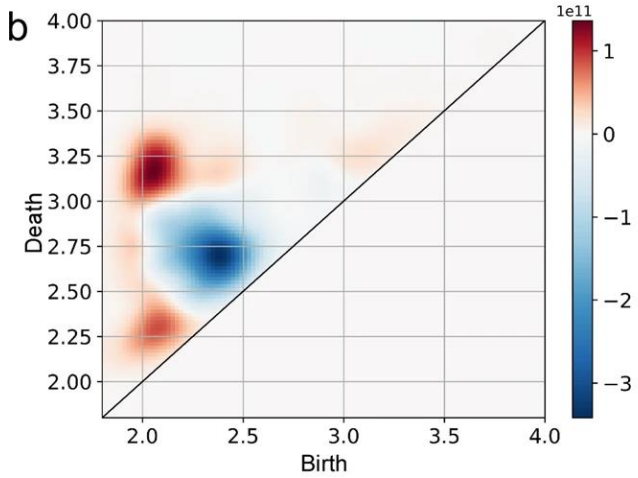
# 金属ガラスの解析

1. PDをパーシステンスイメージに変換する
2. 冷却温度とパーシステンスイメージに対して線形回帰を行う

**Pd原子配置の2次元の穴に関するPD**




**Si原子配置の1次元の穴に関するPD**



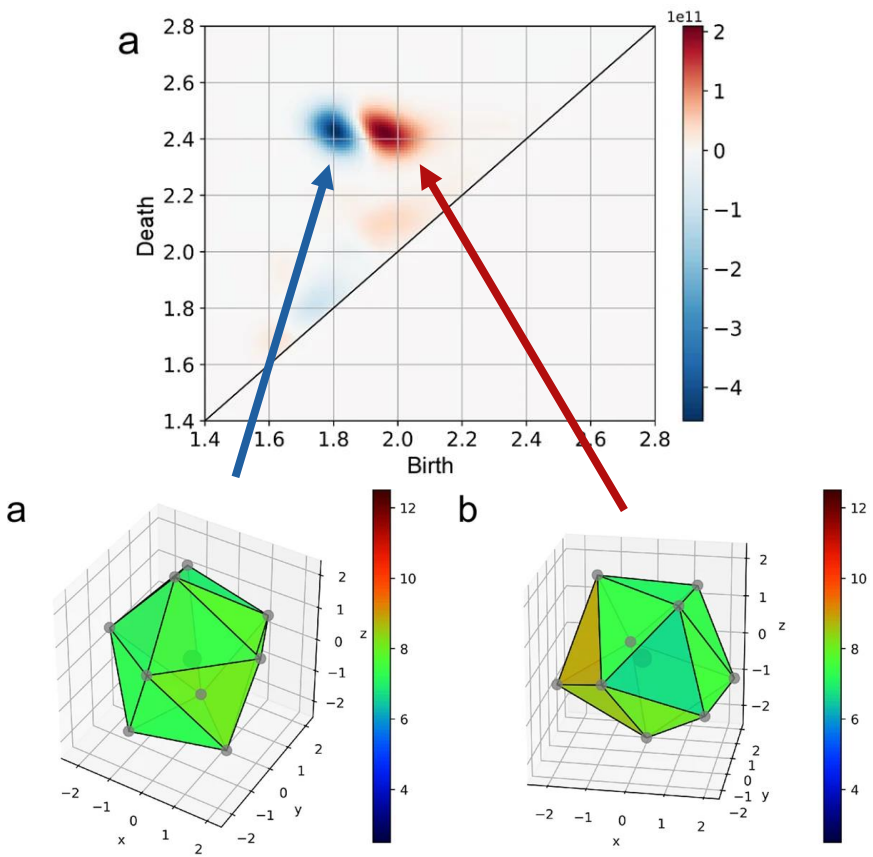

回帰係数が正か負かで，その範囲のPDの点が冷却温度に
正に効くか負に効くか分かる

■冷却温度が早いほど，赤い領域の点が多く，青い領域の点が少ない

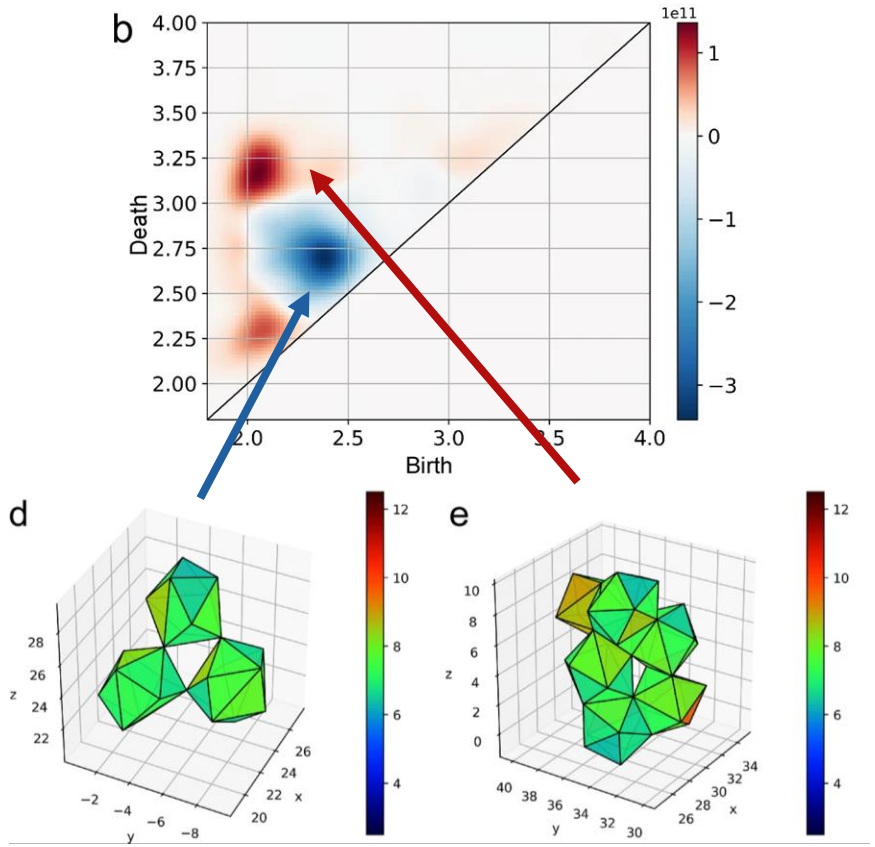
# 金属ガラスの解析

逆解析により具体的にどの原子配置の構造が冷却速度に
関係するかもわかる

**Pd原子配置の2次元の穴に関するPD**



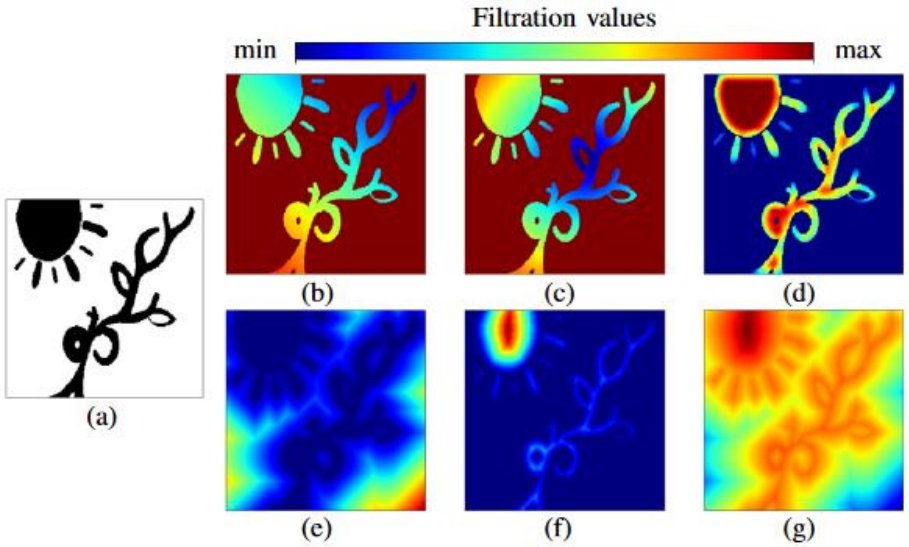

**Si原子配置の1次元の穴に関するPD**

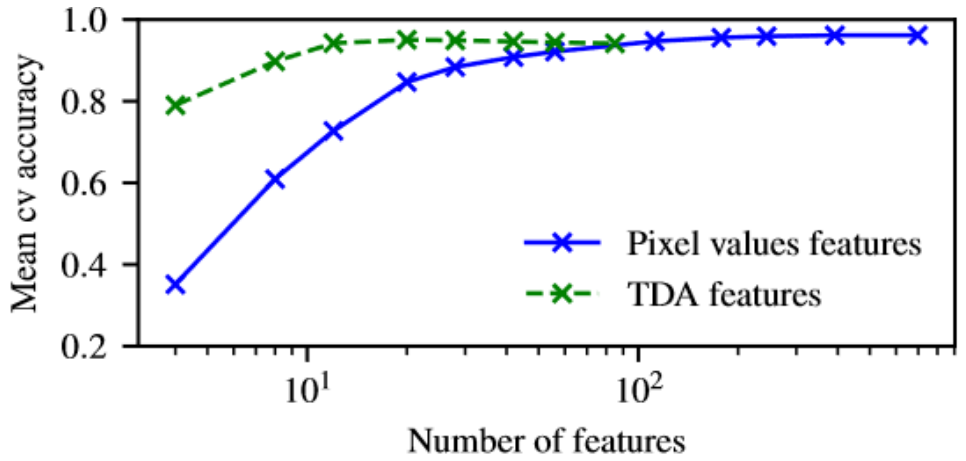


# 2次元画像分類への応用 1

## Garin and Tauzin, A Topological "Reading" Lesson: Classification of MNIST using TDA, 2019

- **■手書き文字画像**データセット（MNIST）の**分類にPHを用いる**
- ■様々なフィルトレーションとベクトル化手法で特徴量を生成
  - ■画素を頂点として，縦横の辺，囲まれる四角形で複体を作る
  - ■ベクトル化はPLに加えてベッチ曲線・生存時間の長さの和・エントロピーを使用
- ■ランダムフォレストを用いて分類
  - ■重要度が高いPD特徴量を使うと，**画素値よりも少ない特徴量で精度よく分類可能**



# パーシステンス図のベクトル化 2

PDのベクトル化は**機械学習で学習する**ことも可能

## ■Hofer et al., Deep Learning with Topological Signatures, 2017

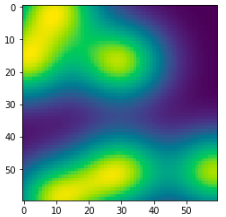
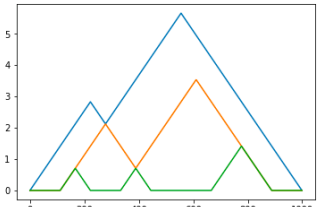
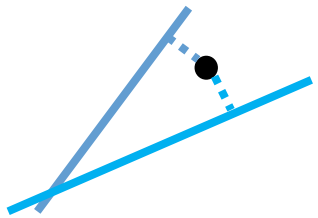
- ■ 中心$\mu_i$・共分散行列$\sigma_i$のガウス関数の値をPDの各点にわたって和を取り，ベクトル化
- ■ これら**$\mu_i$および$\sigma_i$たちを**タスクに応じて**学習させる**

## ■Carrière et al., PersLay, 2020: DeepSetに基づく構造

$$\mathrm{PersLay}(D) \coloneqq \mathrm{op}\left(\{w(p)\cdot\phi(p)\}_{p\in D}\right)$$

- ■ $\phi\colon \mathbb{R}^2 \to \mathbb{R}^q$ 点変換写像，たとえば

1. $\phi_\Gamma\colon p \mapsto \left[\exp\left(-\frac{|p-\boldsymbol{c_k}|^2}{2\sigma^2}\right)\right]_{k=1}^{q}$ ガウス関数
2. $\phi_\Lambda\colon p \mapsto [\max\{0, y-|\boldsymbol{t_k}-x|\}]_{k=1}^{q}$ 三角形の関数
3. $\phi_L\colon p \mapsto [\langle p, \boldsymbol{e_k}\rangle + \boldsymbol{b_k}]_{k=1}^{q}$ 直線からの符号付き距離

- ■ $w\colon \mathbb{R}^2 \to \mathbb{R}$ 重み関数，
- ■ op: 任意の置換不変な操作（和・最大・最小・$k$番目に大きい値など）
- ■ タスクに応じて**$\phi$が持つパラメータを学習させる**

最近はTransformerを用いたベクトル化学習も提案されている

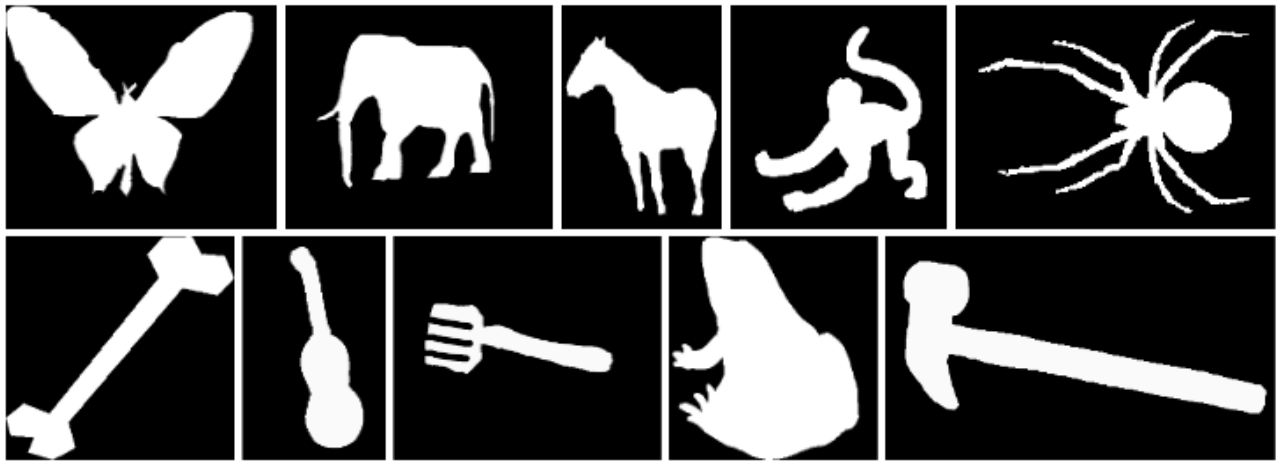
# 2次元画像分類への応用 2

## Hofer et al., Deep Learning with Topological Signatures, 2017

■白黒画像の分類に**PHとベクトル化学習**を用いる

Animal
20クラス

MPEG-7
70クラス

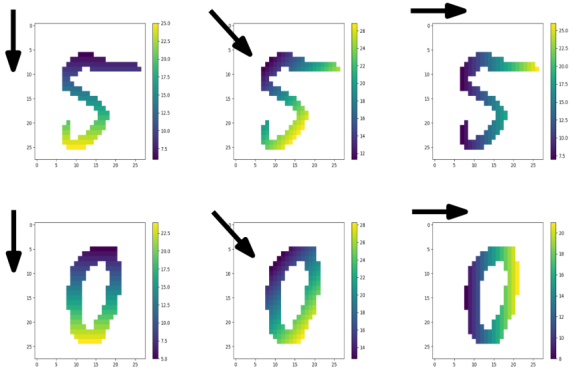


| | MPEG-7 | Animal |
|---|---|---|
| ‡Skeleton paths | 86.7 | 67.9 |
| ‡Class segment sets | 90.9 | 69.7 |
| †ICS | 96.6 | 78.4 |
| †BCF | 97.2 | 83.4 |
| **Ours** | 91.8 | 69.5 |

■**白い画素の格子点が頂点**と見て，それらを**縦横でつなぎ単体的複体**を作る

■円周上に32個等角度に点を取り，それらの方向に増加する**1次関数の劣位集合フィルトレーション**のPHを計算

■出来た32個の0次PDをベクトル化してCNNに入力



**0次PDのみでも表現を学習することで良い精度**

# グラフ分類への応用

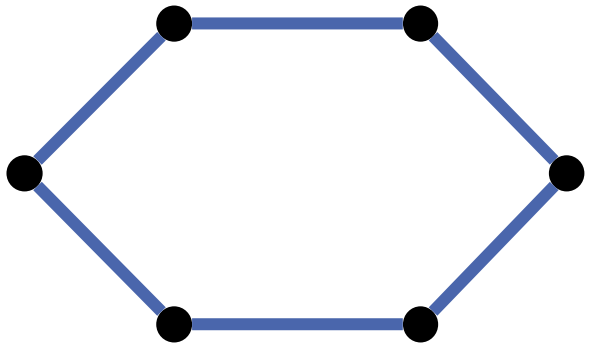
グラフがたくさんあるときに，それらを分類したい

■たとえばタンパク質グラフから種類を分類するなど

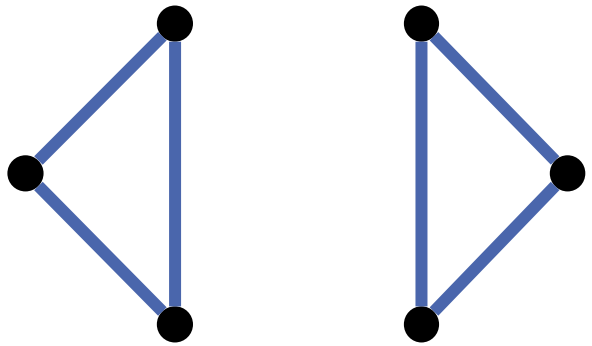
■各点に何本辺があるかなどの**近傍情報だけを使うことが多い**

**PHを使うとグラフの全体的な構造を分類に使える**

**すべての点につながっている辺は2本**



**連結成分は1つ**
**ループは1つ**



**連結成分は2つ**
**ループは2つ**

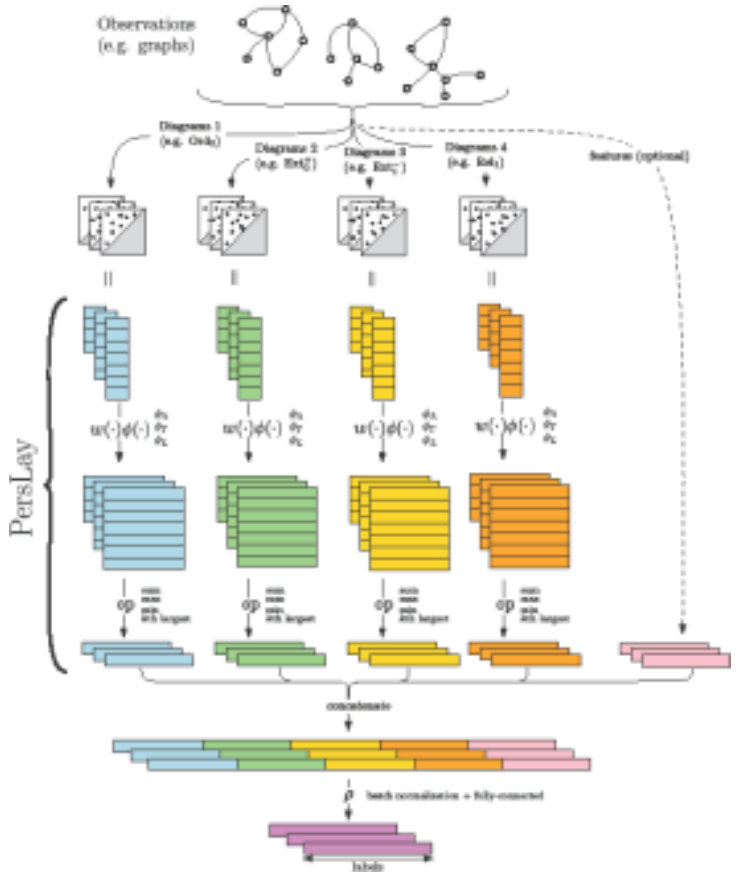
■たとえば，頂点の**次数関数**や**熱の伝導に関係する関数**に対する劣位集合フィルトレーションのパーシステントホモロジーを使う

# グラフ分類への応用

PDを使うと**分類精度が実際に向上する場合がある**

**PersLay**：単純なネットワークにベクトル化学習したPDを入力

↝PDしか使わずに（当時）最先端の分類器と同等の精度を達成




| Dataset | SV[1] | RetGK* [2] | FGSD [3] | GCNN [4] | GIN [5] | PersLay Mean | PersLay Max |
|---|---|---|---|---|---|---|---|
| REDDIT5K | — | 56.1 | 47.8 | 52.9 | 57.0 | 55.6 | 56.5 |
| REDDIT12K | — | 48.7 | — | 46.6 | — | 47.7 | 49.1 |
| COLLAB | — | 81.0 | 80.0 | 79.6 | 80.1 | 76.4 | 78.0 |
| IMDB-B | 72.9 | 71.9 | 73.6 | 73.1 | 74.3 | 71.2 | 72.6 |
| IMDB-M | 50.3 | 47.7 | 52.4 | 50.3 | 52.1 | 48.8 | 52.2 |
| COX2* | 78.4 | 80.1 | — | — | — | 80.9 | 81.6 |
| DHFR* | 78.4 | 81.5 | — | — | — | 80.3 | 80.9 |
| MUTAG* | 88.3 | 90.3 | 92.1 | 86.7 | 89.0 | 89.8 | 91.5 |
| PROTEINS* | 72.6 | 75.8 | 73.4 | 76.3 | 75.9 | 74.8 | 75.9 |
| NCI1* | 71.6 | 84.5 | 79.8 | 78.4 | 82.7 | 73.5 | 74.0 |
| NCI109* | 70.5 | — | 78.8 | — | — | 69.5 | 70.1 |

※10-foldの交差検証

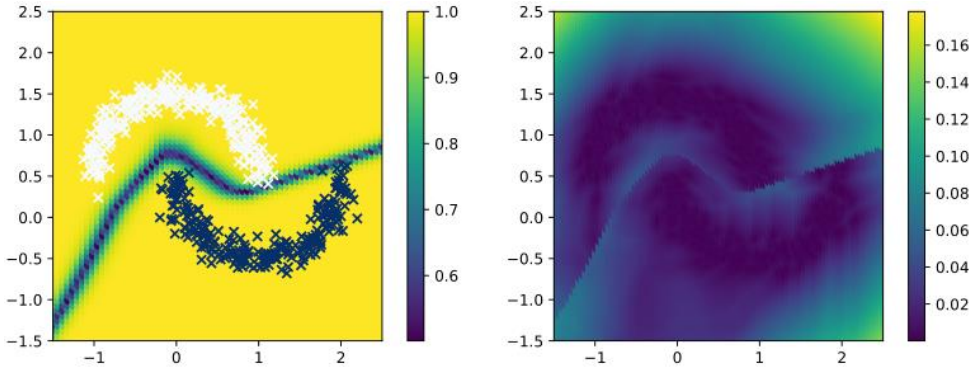
**トポロジー情報を上手く使うことは有効**

# ネットワークのPH的特徴量

# ネットワークのPH的特徴量：TU

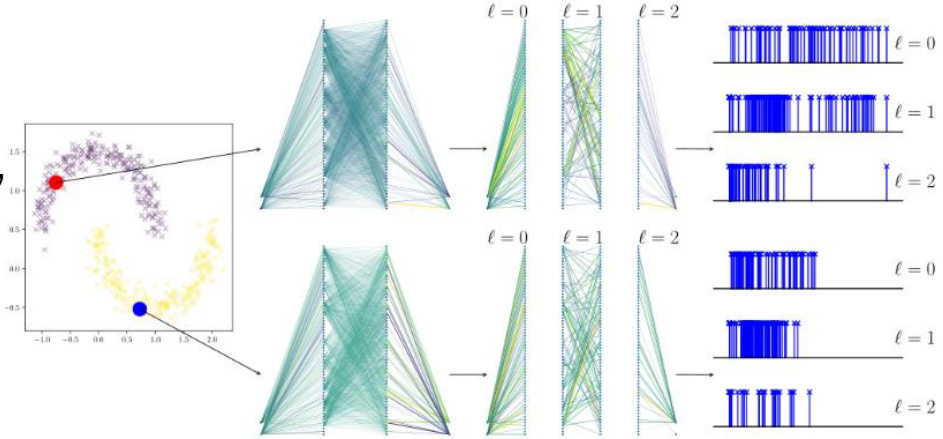
学習済み分類器 (NN) は最終層が予想確率をあらわす

■予想確率の最大はどのデータに対しても高く，**全く別のデータが来ても自信をもって間違えてしまう**（左図）



**ネットワークの活性度を表す重み付きグラフのPDを使う**（右図）

[Gebhart and Schrater, 2017],
[Gebhart et al., 2019],
[Rieck et al., Neural persistence, 2019],
**[Lacombe et al., Topological Uncertainty, 2021]**

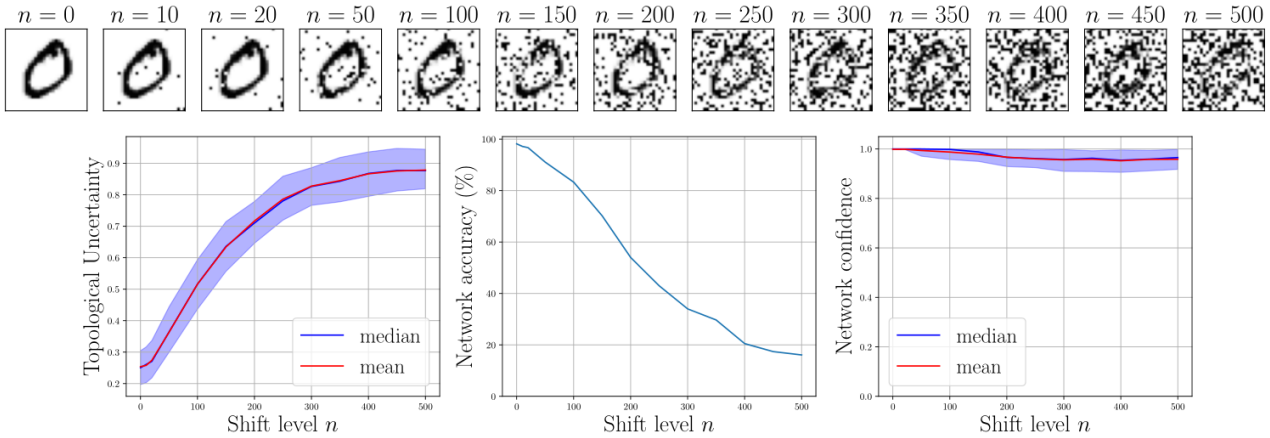


学習データでの活性度と似ているかをPD同士を比較してはかる

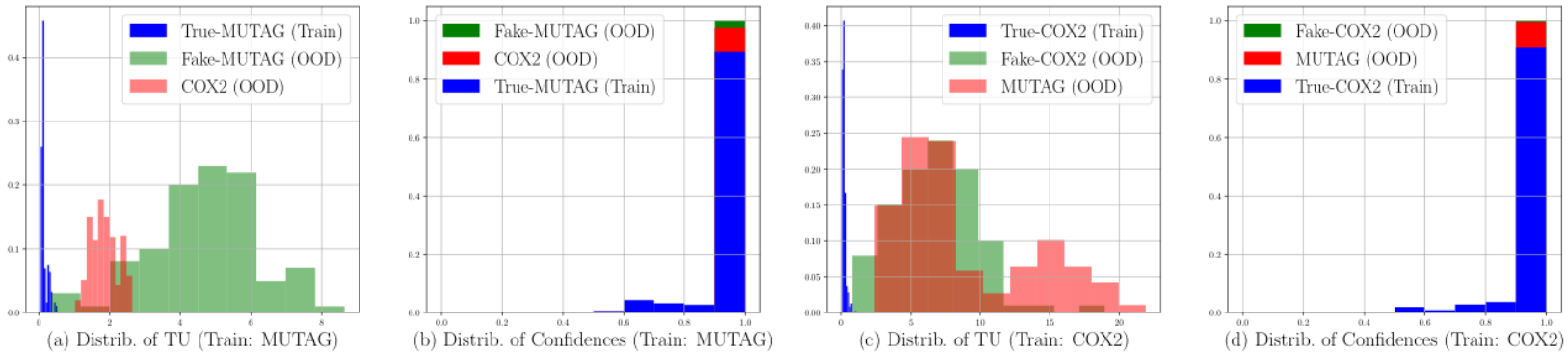
Figures from [Lacombe et al., Topological Uncertainty, 2021]

# ネットワークのPH的特徴量：TU

■**入力の変化に対して敏感**：NNの性能保持に役立つと期待




■ **分布外データ検出**：全然別のデータが来た時にアラートを出せる



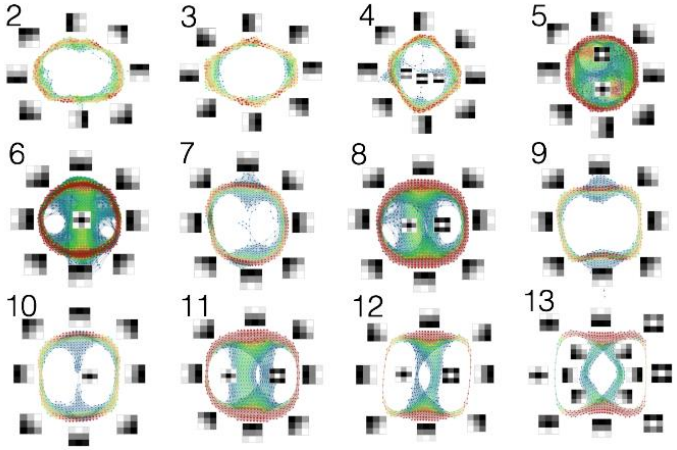

Figures from [Lacombe et al., Topological Uncertainty, 2021]

# ネットワークのPH的特徴量：他の試み

その他にも**NNそのものをPH的な特徴量で調べる**アプローチがある

■**Carlsson and Bruel-Gabrielsson, Topological approaches to deep learning, 2020**

- ■CNNのフィルターをPHとMapperで調べている
- ■浅い層と深い層のフィルターのをトポロジー的に比較




■**Barannikov et al., Representation topology divergence: A method for comparing neural network representations, 2022**

- ■NNによる潜在表現同士の差をはかる指標RTDをPHで構成
- ■右図：90個の単語埋め込み表現の比較.　log perplexityと整合.

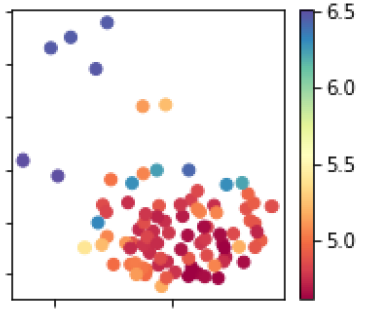


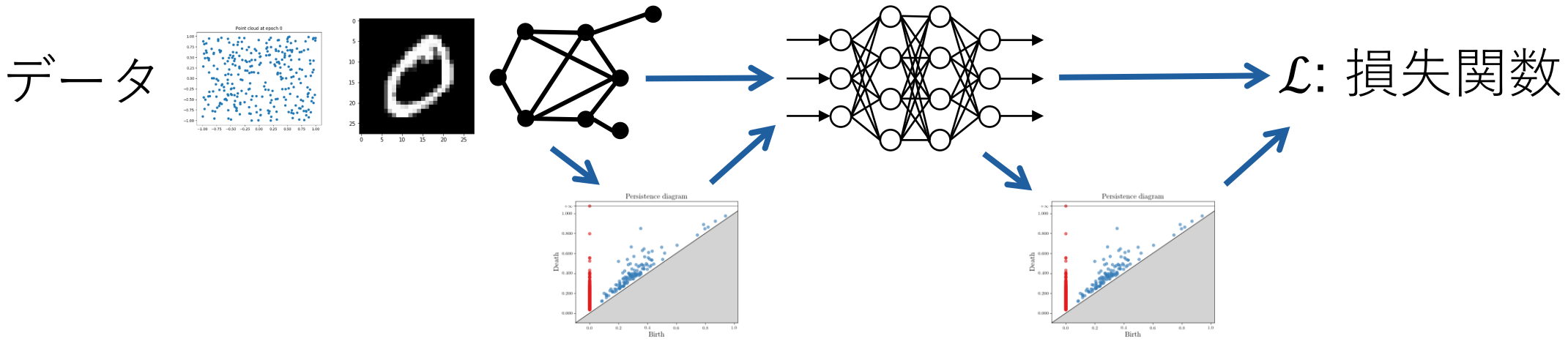

Figure 5: Multi-dimensional scaling of 90 architectures selected randomly from NAS-Bench-NLP. Color depicts log. perplexity.

# PHを損失関数に組み込む

- トポロジー的損失関数の利用
- 微分可能性と勾配法による最適化

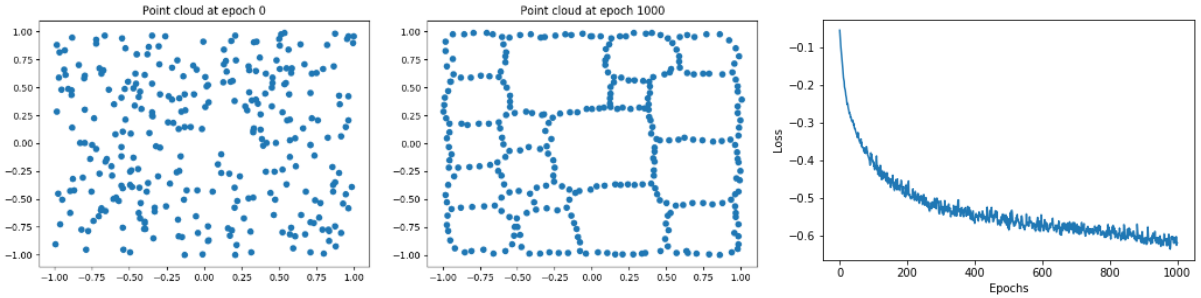
# トポロジー的損失関数の利用

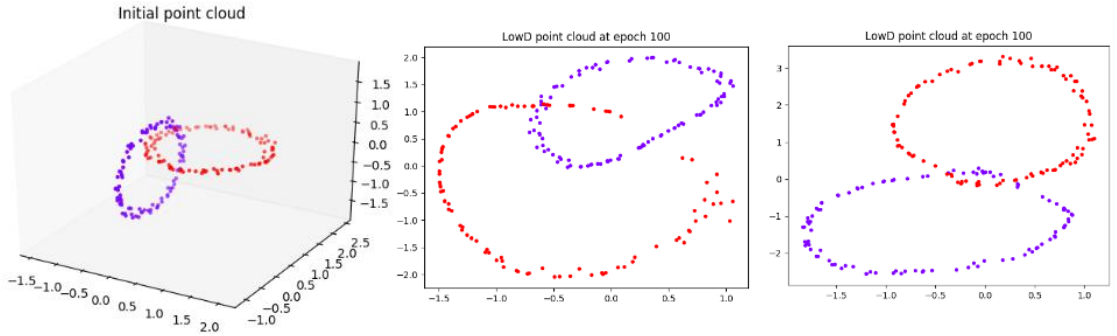
## PHを損失関数に組み込んで学習器をトポロジー的にコントロール




■**A Topology Layer for Machine Learning, AISTATS2020**: 点群の変形

■**Topological Autoencoders, ICML2020**: 位相的性質も保つ次元圧縮



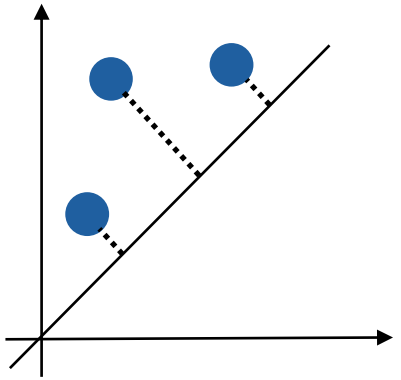



# おもちゃの例 (1/2)

点群を**ループの個数が多くなる**ように変形する

■1次（ループに関する）のPD $D_1(X)$を考えて,

$$\mathcal{L}(X) = -\sum_{p \in D_1(X)} \|p - \pi_\Delta(p)\|_\infty^2 + d(X, C)$$



とする．ここで$\pi_\Delta$は対角線への射影で$C$は正方形.

■**勾配降下法** $x_{k+1} = x_k - \alpha_k \nabla \mathcal{L}(x_k)$ により最適化


実はパーシステンス図を与える写像は，ほとんど至るところで微分可能で，劣微分を使って最適化を行うことができる

# パラメータ付けられたフィルトレーション

**復習**：単体的複体$K$のフィルトレーション

$\leftrightarrow$ 関数$F\colon K \to \mathbb{R}$ s.t. $\sigma \subset \tau \Rightarrow F(\sigma) \leq F(\tau)$

$K$に順序を与えると

$\leftrightarrow$ ベクトル$F \in \mathbb{R}^{|K|}$ s.t. $\sigma \subset \tau \Rightarrow F_\sigma \leq F_\tau$

$$\mathrm{Filt}_K \coloneqq \left\{ F \in \mathbb{R}^{|K|} \,\middle|\, \sigma \subset \tau \Rightarrow F_\sigma \leq F_\tau \right\}$$

**定義** $A \subset \mathbb{R}^d$で**パラメータ付けられたフィルトレーション**とは
関数$F\colon A \to \mathrm{Filt}_K$のこと

**例**

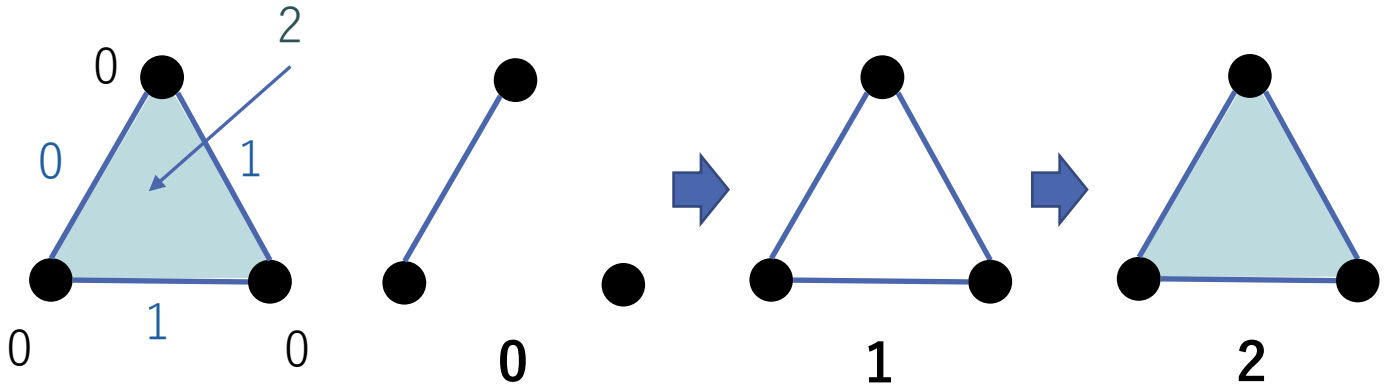
- **劣位集合フィルトレーション** $F\colon \mathbb{R}^{|V(K)|} \to \mathbb{R}^{|K|},\ F_\sigma(f) \coloneqq \max_{i\in\sigma} f_i$
- **Ripsフィルトレーション** $F\colon (\mathbb{R}^d)^N \to \mathbb{R}^{|\Delta_N|},\ F_\sigma(x) \coloneqq {}^1\!/_2 \max_{i,j\in\sigma} \|x_i - x_j\|$
- **MLのパラメータ** $f_\theta\colon \mathbb{R}^d \to \mathbb{R}^D\ (\theta \in \Theta), P \subset \mathbb{R}^d$: 有限集合

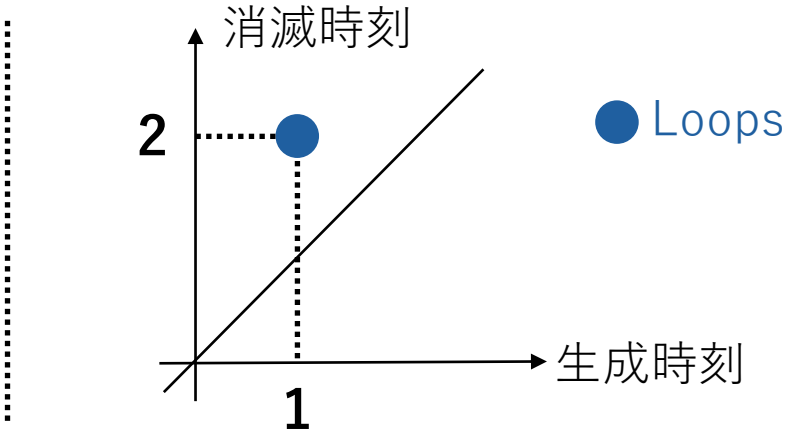
$$F\colon \Theta \to \mathbb{R}^{|\Delta_N|},\ F_\sigma(\theta) \coloneqq {}^1\!/_2 \max_{i,j\in\sigma} \|f_\theta(x_i) - f_\theta(x_j)\|$$

# パーシステンス図の対応の微分可能性

**Q. パーシステンス図を与える写像を微分することはできるか？**

**A. 多くの点では（ほとんど至るところで）微分できる**




1. **生成・消滅単体の組**$\{(\sigma_{b_i}, \sigma_{d_i})\}_i$を見つける（組合せ的）

   e.g.　　：生成単体　　：消滅単体

2. 各組に**フィルトレーションの値を対応**させる$\{(F(\sigma_{b_i}), F(\sigma_{d_i}))\}_i$ e.g. (1,2)

■$F$が滑らかにパラメータ付けられている ⇒ 微分$x \mapsto \{(\nabla_x F(\sigma_{b_i}), \nabla_x F(\sigma_{d_i}))\}_i$
を単体の順序が変わらない範囲で考えることが可能

# 応用例 (1/2)：画像の劣位集合

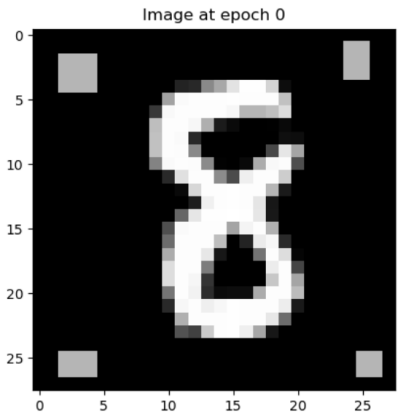
**画像を変形**して染みをのせたMNIST画像から染みを削除したい

- 損失関数として$\mathcal{L}_1(I) = \sum_{p \in D_0(I)} \|p\|_2^2$を用いて勾配法を適用すると，連結成分を減らすように働き中央の画像のようになる
- ピクセル値が0か1になるように

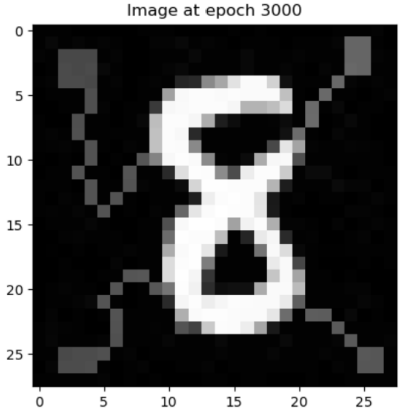
$$\mathcal{L}_2(I) = \sum_{p \in D_0(I)} \|p\|_2^2 + \sum_i \max\{|x_i|, |1 - x_i|\}$$

を用いると結果は右の画像のようになる

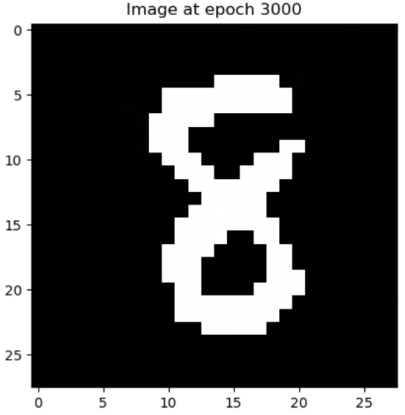
**元画像**




**$\mathcal{L}_1$の結果**




**$\mathcal{L}_2$の結果**



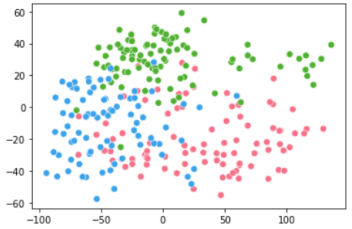
# 応用例 (2/2)：埋め込みのTDA的統制

**Vandaele et al., Topologically Regularized Data Embeddings, 2022: データ埋め込み $E: \mathbb{X} \to \mathbb{R}^d$ に応用**

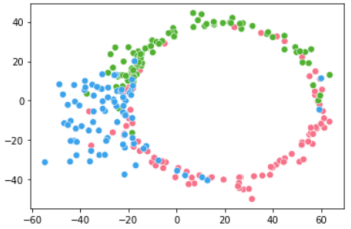
■$E$のPDの点を$d_k - b_k$が降順になるように並べて

$$\mathcal{L}_{\text{top}}(E) := \mu \sum_{k=i, d_k<\infty}^{k=j} (d_k - b_k),\ \mu \in \{\pm 1\} \text{ を考える}$$

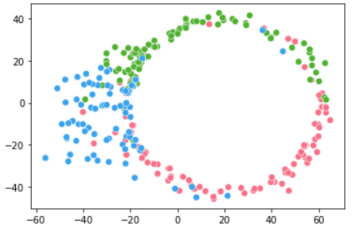
■$\mathcal{L}_{\text{emb}}(E, \mathbb{X}) + \lambda_{\text{top}} \mathcal{L}_{\text{top}}(E)$を最適化することで$E$のトポロジーを統制



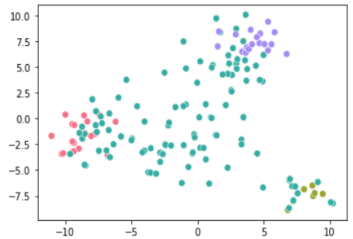
(a) Ordinary PCA embedding.



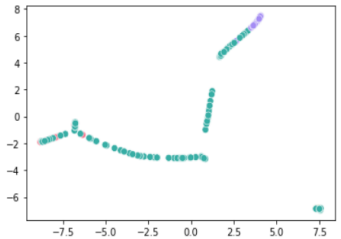
(b) Top. optimized embedding.



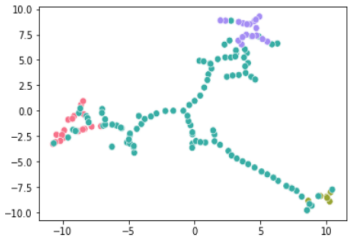
(c) Top. regularized embedding.



(a) Ordinary UMAP embedding.



(b) Top. optimized embedding.



(c) Top. regularized embedding.

$\left\| x - \frac{1}{|E|} \sum_{y \in E} y \right\| \geq \tau$
を満たす点のPDを用いる

# トポロジー情報の機械学習

- フィルトレーションを学習する
- PDを計算する関数を機械学習で近似する

# PH損失関数の別の応用：フィルター学習

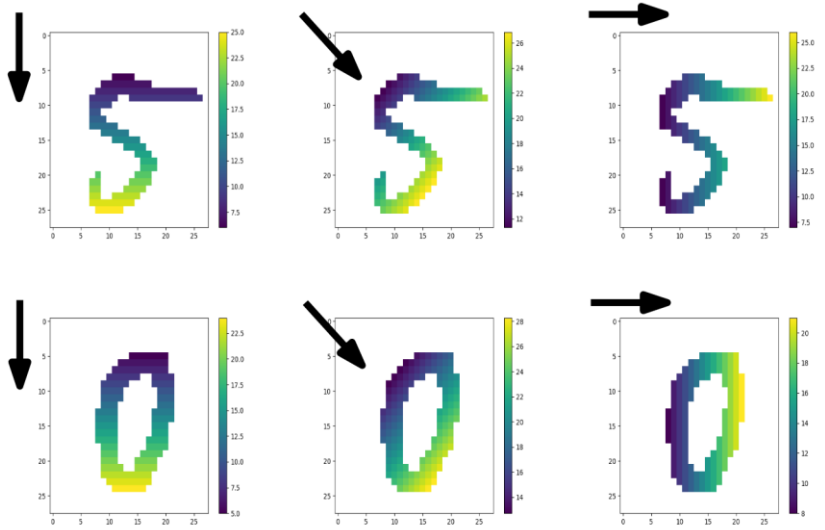
分類に用いる**TDAのフィルターで最適なものを学習**する

■MNIST画像を劣位フィルトレーションに関する0次PDで分類する問題

■ある方向への1次関数の族をとり

$$\mathcal{L}(f) = \sum_{l} \frac{\sum_{y_i=y_j=l} d(D_0(I_i, f), D_0(I_j, f))}{\sum_{y_i=l} d(D_0(I_i; f), D_0(I_j; f))}$$

を最適化する．最適化前と後でのランダムフォレストでの分類精度は下の表

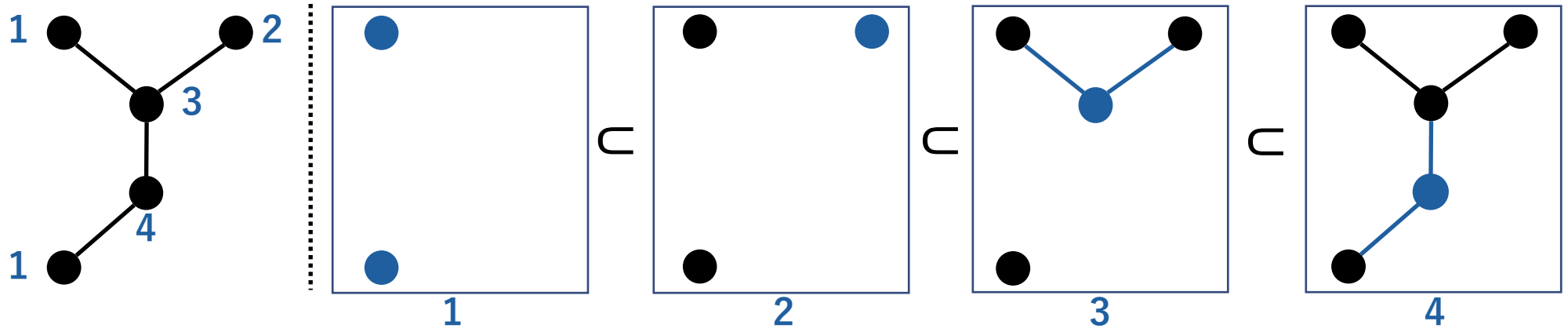


| Dataset | Baseline | Before | After | Difference | Dataset | Baseline | Before | After | Difference |
|---|---|---|---|---|---|---|---|---|---|
| vs01 | 100.0 | 61.3 | 99.0 | **+37.6** | vs26 | 99.7 | 98.8 | 98.2 | -0.6 |
| vs02 | 99.4 | 98.8 | 97.2 | -1.6 | vs28 | 99.1 | 96.8 | 96.8 | 0.0 |
| vs06 | 99.4 | 87.3 | 98.2 | **+10.9** | vs29 | 99.1 | 91.6 | 98.6 | **+7.0** |
| vs09 | 99.4 | 86.8 | 98.3 | **+11.5** | vs34 | 99.8 | 99.4 | 99.1 | -0.3 |
| vs16 | 99.7 | 89.0 | 97.3 | **+8.3** | vs36 | 99.7 | 99.3 | 99.3 | -0.1 |
| vs19 | 99.6 | 84.8 | 98.0 | **+13.2** | vs37 | 98.9 | 94.9 | 97.5 | **+2.6** |
| vs24 | 99.4 | 98.7 | 98.7 | 0.0 | vs57 | 99.7 | 90.5 | 97.2 | **+6.7** |
| vs25 | 99.4 | 80.6 | 97.2 | **+16.6** | vs79 | 99.1 | 85.3 | 96.9 | **+11.5** |

# グラフのフィルトレーション学習 (1/2)

## Hofer et al., Graph Filtration Learning, 2020

- **フィルター関数をエンドツーエンドに学習**
- **頂点上の関数**はグラフの**劣位集合フィルトレーション**を与える




- 頂点上の関数をグラフニューラルネットワーク (GNN) として実現してそれをタスクに応じて学習させる
  - $V \ni v \mapsto \mathrm{GNN}(G, l(v)), l: V \to \mathbb{R}^n$はノードの表現
  - 実験では隠れ層64次元の1-GINを用いて，後ろに2層のMLPを使っている

# グラフのフィルトレーション学習 (2/2)

## Hofer et al., Graph Filtration Learning, 2020

■フィルター関数をエンドツーエンドに学習

■出来上がったPDのベクトル化は分類に用いられる

■微分可能性により，PDの情報を含んだ損失関数を勾配法で最適化できる

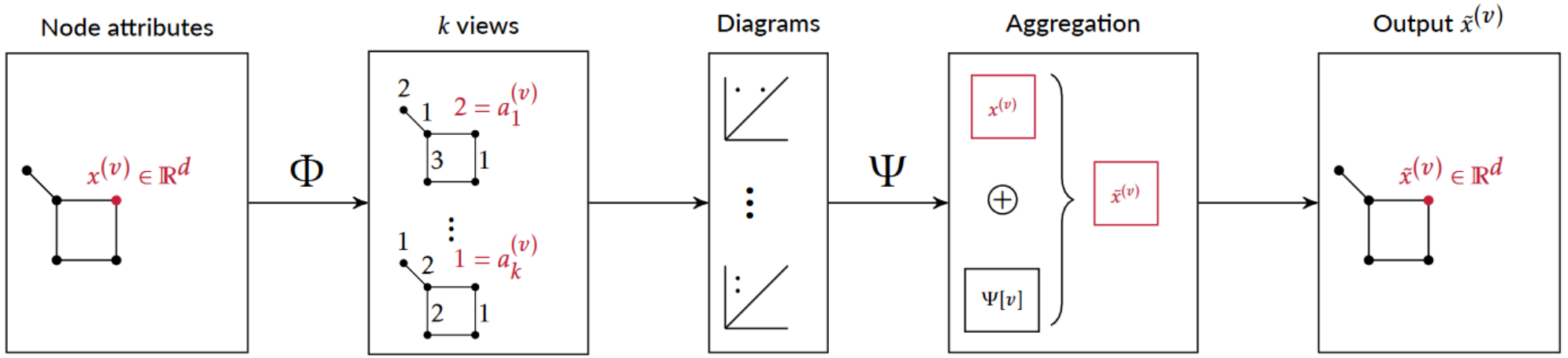
| Method | REDDIT-BINARY | REDDIT-MULTI-5K | IMDB-BINARY | IMDB-MULTI |
|---|---|---|---|---|
| | *Initial node features: uninformative* | | *Initial node features:* $l(v) = \deg(v)$ | |
| PH-only | $90.3_{\pm 2.6}$ | $55.7_{\pm 2.1}$ | $68.9_{\pm 3.5}$ | $46.1_{\pm 4.2}$ |
| 1-GIN (GFL) | $90.2_{\pm 2.8}$ | $55.7_{\pm 2.9}$ | $74.5_{\pm 4.6}$ | $49.7_{\pm 2.9}$ |
| 1-GIN (SUM) (Xu et al., 2019) | $81.2_{\pm 5.4}$ | $51.0_{\pm 2.2}$ | $73.5_{\pm 3.8}$ | $50.3_{\pm 2.6}$ |
| 1-GIN (SP) (Zhang et al., 2018a) | $76.8_{\pm 3.6}$ | $48.5_{\pm 1.8}$ | $73.0_{\pm 4.0}$ | $50.5_{\pm 2.1}$ |
| Baseline (Zaheer et al., 2017) | $77.5_{\pm 4.2}$ | $45.7_{\pm 1.4}$ | $72.7_{\pm 4.6}$ | $49.9_{\pm 4.0}$ |
| *State-of-the-Art (NN)* | | | | |
| DCNN (Wang et al., 2018) | n/a | n/a | 49.1 | 33.5 |
| PatchySAN (Niepert et al., 2016) | 86.3 | 49.1 | 71.0 | 45.2 |
| DGCNN (Zhang et al., 2018a) | n/a | n/a | 70.0 | 47.8 |
| 1-2-3-GNN (Morris et al., 2019) | n/a | n/a | 74.2 | 49.5 |
| 5-GIN (SUM) (Xu et al., 2019) | 88.9 | 54.0 | 74.0 | 48.8 |
| | Sparse | | Dense | |

フィルトレーションを学習した方が精度が高まる場合がある

# Topological Graph Layer

## Horn et al., Topological Graph Neural Networks, 2022

- **トポロジー的情報を含めて頂点上のベクトルを変換する**層を提案
- フィルトレーション学習とベクトル化学習を両方行うパイプライン




- $\Phi$: $\mathbb{R}^d \to \mathbb{R}^k$ フィルトレーション学習を行う（たとえば2層のNN）
- $\Psi$: PDのベクトル化（たとえばDeepSets, rational hat function, PersLayなど）

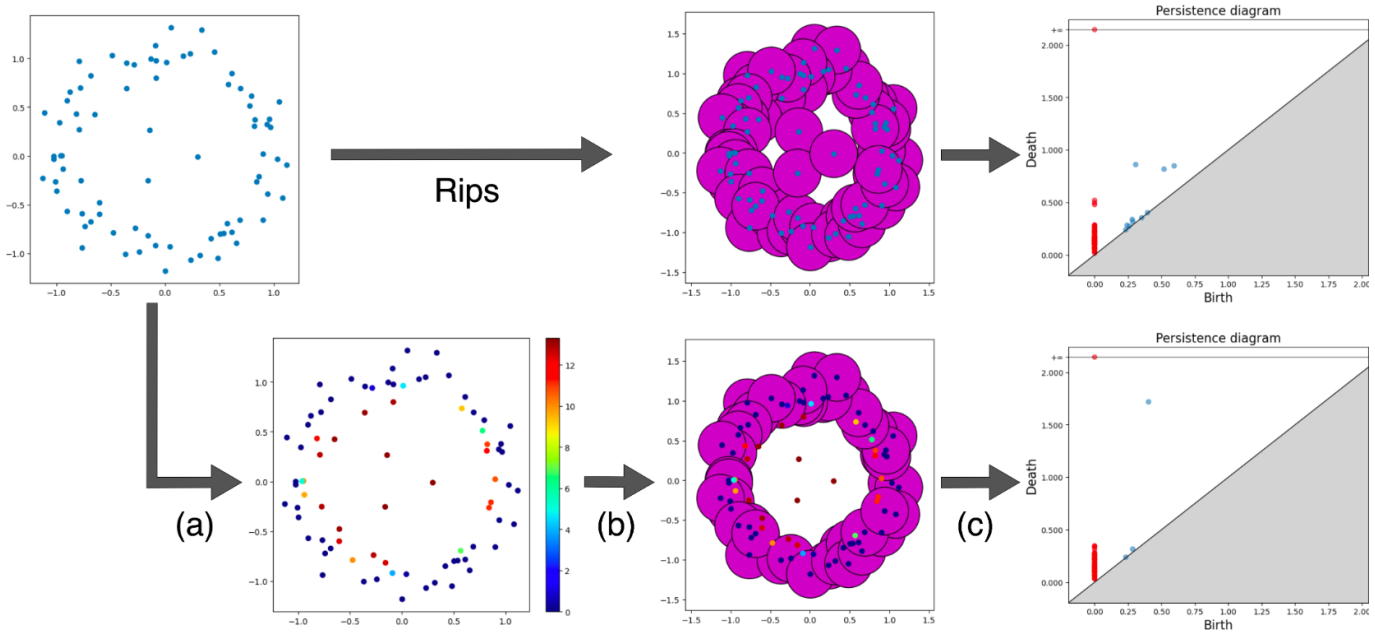
- GNNに簡単に組み込むことが可能
- 単純なGNNでは区別できないグラフを区別可能にできる

Figures from B. Rieck, Topology-Based Graph Representation Learning

# 点群のフィルトレーション学習 (1/2)

## Nishikawa et al., Adaptive Topological Feature via Persistent Homology: Filtration Learning for Point Clouds, 2023

- ■分類タスクに応じて**点群のフィルトレーションを学習**
  - ■球の膨張の**開始時刻を点ごとに学習**
  - ■**等長変換不変な構造**を提案



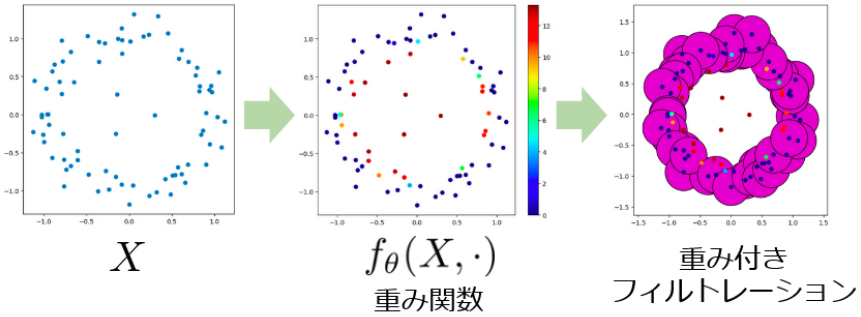

ユーザがフィルトレーションを指定しなくても，**データから自動的に学習**

タンパク質分類の精度

| DistMatrixNet | Rips | DTM | Ours |
|---|---|---|---|
| 65.0 ± 12.0 | 79.9 ± 3.0 | 78.0 ± 1.6 | **81.9 ± 2.1** |

# 点群のフィルトレーション学習 (2/2)

■各点$x$の球の**膨張の開始時刻**を点群$X$と点$x$を入力とするニューラルネットワーク$f_\theta(X,x)$で定める

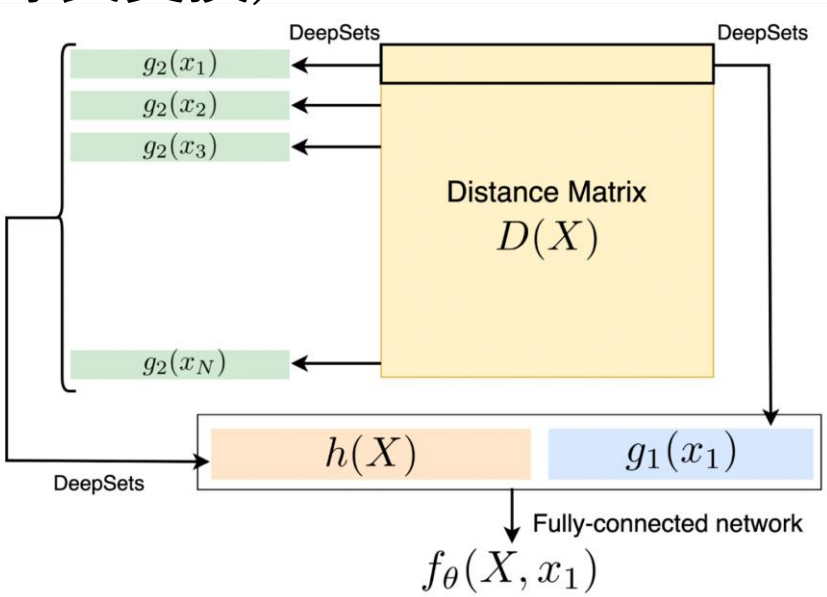



■$f_\theta(X,x)$が満たす条件

1. **点の順序について不変**
2. **点群全体$X$の性質と各点$x$の性質の両方**を反映
3. **等長不変変換**：$f_\theta(TX,Tx)=f_\theta(X,x)$（$T$: 等長変換）

■**距離行列**と**DeepSets**を用いて
これらの条件を満たすように構成

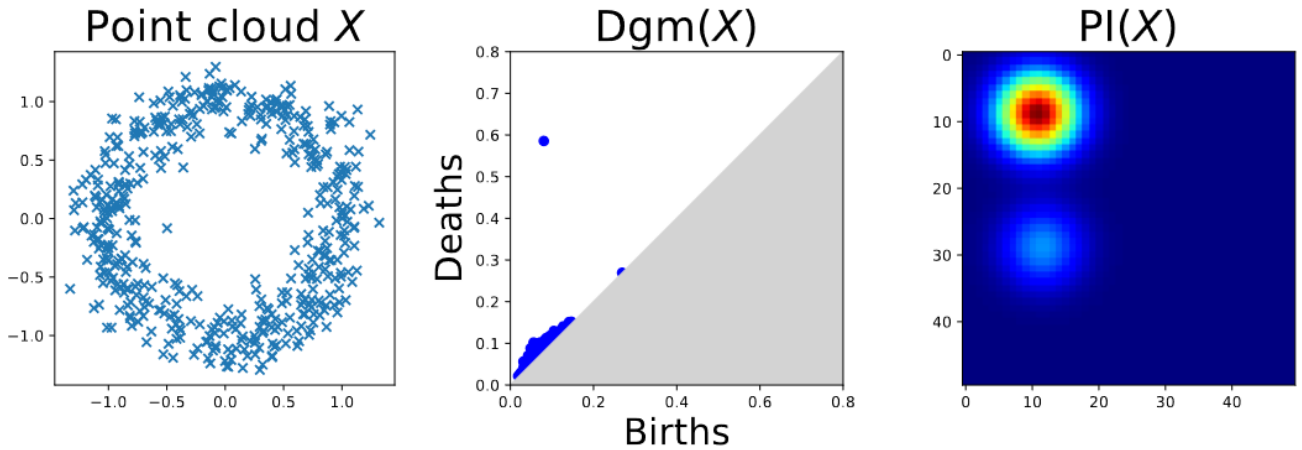
■**PersLayでベクトル化**して分類に使用
全体を**タスクに合わせて学習**



# PDの厳密計算の問題点

TDAの実応用においては以下のパイプラインを使うことが多い：

**点群 ↦ PD ↦ ベクトル化**



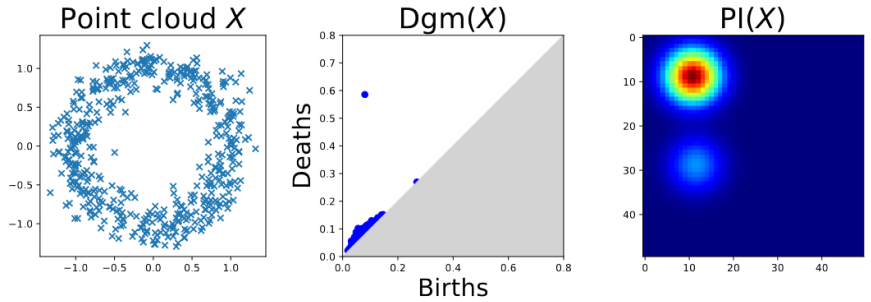

このパイプラインにおいては**２つの問題点**がある

1. 点群に対するPDを計算するための**計算量**
   - ■（ナイーブな）アルゴリズムの計算量は$\mathcal{O}(n^3), n$ =#単体
   - ■**Čech・Ripsフィルトレーションの場合**: $n = 2^{\#P} - 1!$
2. 低レベルの**外れ値に対しても敏感**（1点の変更で大きく変化）

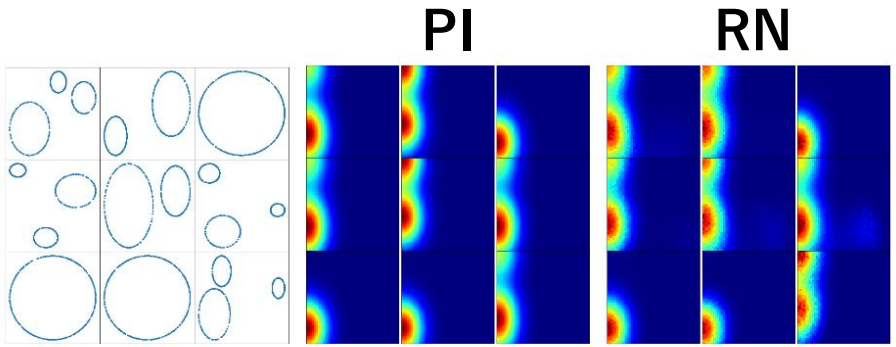
# RipsNet: NNによるPDの推定

**de Surrel et al., RipsNet, ICLR Workshop on Geometrical and Topological Representation Learning 2022**
(cf. Zhou et al., 2022; 3次元点群データ， Kaji, 2019; Som et al., 2020; 画像データ， Yan et al., 2022; グラフデータ)

■点群 ↦ PD ↦ ベクトル化のパイプラインを学習する**データドリブンな手法**




■ニューラルネットワークで**PDを推定する**



# RipsNet: 構造と学習

■PDベクトル化手法 $\mathbf{PV}: \boldsymbol{X} \mapsto \boldsymbol{D}(\boldsymbol{X}) \mapsto \mathbf{PV}(\boldsymbol{X})$ を固定（主にPIとPL）

■点群$X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$ から**PVを学習するRN**：

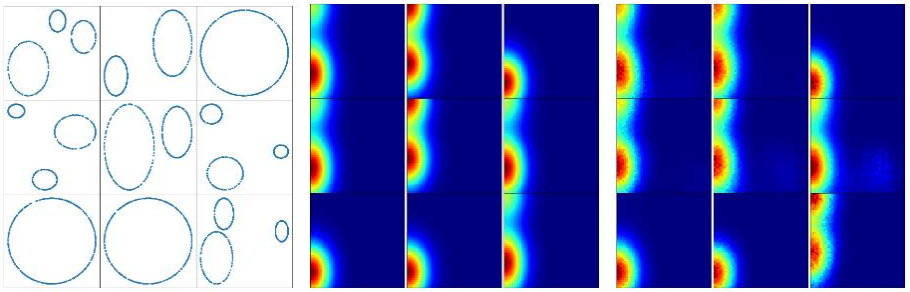
$$\mathrm{RN}: X \mapsto \phi_2\left(\mathrm{op}(\{\phi_1(x)\}_{x \in X})\right)$$

$\phi_1: \mathbb{R}^d \to \mathbb{R}^{d_1}, \phi_2: \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ MLP，opは置換不変な作用素

■**PDのベクトル化**と**L2損失**で学習：$\mathrm{minimize} \sum_i \|\mathrm{RN}(X_i) - \mathrm{PV}(X_i)\|^2$
（PHは学習データ生成のみに使用）

■学習後のRNは効率的にPVを推定

**PI** **RN**



**計算時間の比較**

| Data | Gudhi (s) | Gudhi$^{\mathrm{DTM}}$ (s) | **RN (s)** |
|---|---|---|---|
| LS | $56.3 \pm 1.5$ | $155.9 \pm 8.1$ | $\mathbf{0.3 \pm 0.0}$ |
| PI | $69.5 \pm 3.1$ | $173.7 \pm 13.3$ | $\mathbf{0.4 \pm 0.0}$ |
| P | $5.3 \pm 1.4$ | $44.7 \pm 6.6$ | $\mathbf{0.2 \pm 0.0}$ |
| UMD | $8.0 \pm 1.4$ | $55.7 \pm 3.6$ | $\mathbf{0.2 \pm 0.0}$ |
| $\lambda = 2\%$ | $118.4 \pm 4.7$ | $178.5 \pm 8.1$ | $\mathbf{0.2 \pm 0.0}$ |
| $\lambda = 5\%$ | $117.8 \pm 4.5$ | $180.0 \pm 9.2$ | $\mathbf{0.2 \pm 0.0}$ |

実際のPD計算

# 実験：分類タスク

## ■分類タスクにおいて比較

■$\widetilde{\mathrm{PI}}, \widetilde{\mathrm{PL}}$: ノイズありの結果

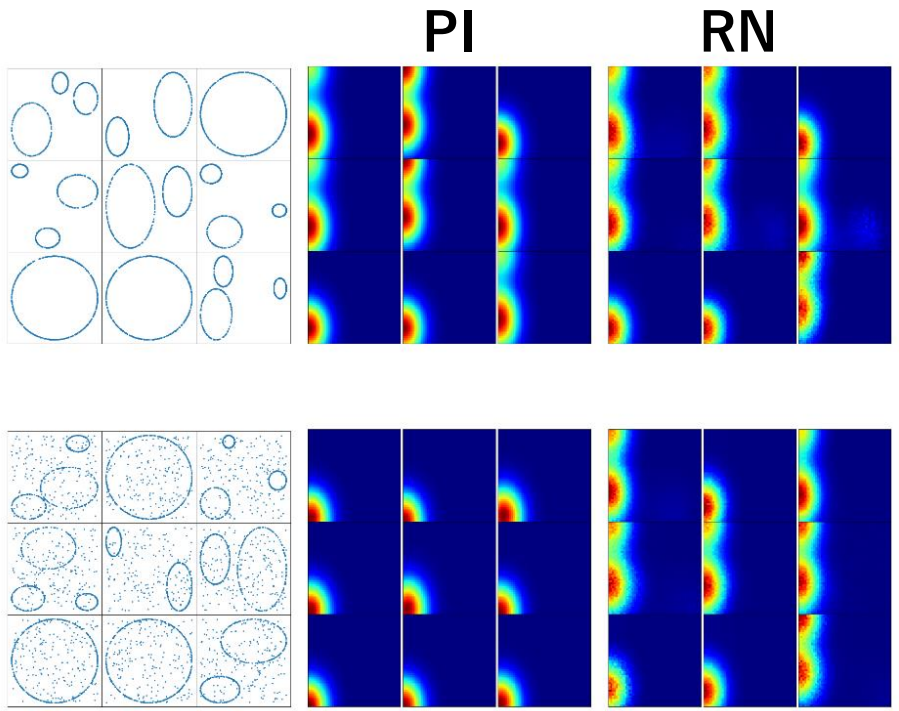
## ■人工データ

■$\mathbb{R}^2$内の円状データ（ノイズなし・あり）

■$N = 600, \lambda = 1/3$

■PIまたはRipsNet + XGBoost

■直接DeepSetsを使う手法とも比較



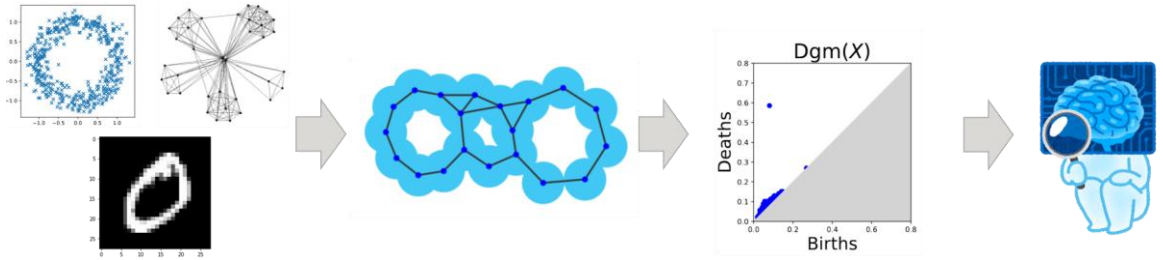

■RipsNetが**ノイズありの状況では他手法に勝った**

| Synth. Data | $\mathrm{Cl}^{\mathrm{XGB}}_{\mathrm{Gudhi}}$ | $\mathrm{Cl}^{\mathrm{XGB}}_{\mathrm{Gudhi^{DTM}}}$ | $\mathrm{Cl}^{\mathrm{XGB}}_{\mathrm{RN}}$ | $\mathbf{DS}_1$ | $\mathbf{DS}_2$ |
|---|---|---|---|---|---|
| PL | $99.9 \pm 0.1$ | $99.9 \pm 0.1$ | $80.7 \pm 3.0$ | $66.4 \pm 2.3$ | $66.0 \pm 2.4$ |
| PI | $100.0 \pm 0.0$ | $100.0 \pm 0.1$ | $81.6 \pm 5.3$ | - | - |
| $\widetilde{\mathrm{PL}}$ | $66.7 \pm 0.0$ | $66.7 \pm 0.0$ | $76.3 \pm 2.3$ | $66.8 \pm 1.0$ | $66.6 \pm 2.3$ |
| $\widetilde{\mathrm{PI}}$ | $33.3 \pm 0.0$ | $65.0 \pm 1.3$ | $77.4 \pm 4.4$ | - | - |

# まとめ

## 1. PHを機械学習の入力に使う

- PDをベクトル化して機械学習に入力することでトポロジー情報を使える
- ベクトル化を学習する手法もある

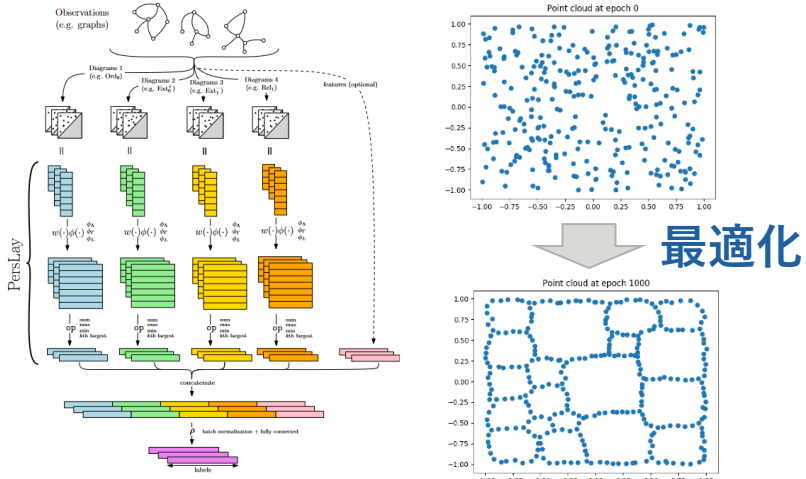



## 2. NNからPH的特徴量を取り出す

- NNの状況を調べたり監視したりできる

## 3. PHを損失関数に組み込む

- PDの関数を損失関数に組み込むことによって学習機をトポロジー的にコントロールできる
- TDA損失関数の設計が今後の課題




## 4. トポロジー的情報の機械学習

- フィルトレーションをデータから学習
- データをPDに変換する関数をNNで学習する試み