論　文

ロバスト高速オンライン多変量ノンパラメトリック密度推定法

中村　圭宏[†a)]　　長谷川　修[††b)]

Robust Fast Online Multivariate Non-Parametric Density Estimation

Yoshihiro NAKAMURA[†a)] and Osamu HASEGAWA[††b)]

あらまし　近年，センサーやネットワーク技術の発展に伴って，実環境から大量のデータが継続的にリアルタイムに生成されるようになってきている．どのようなデータが得られるかあらかじめ仮定するのが困難な場合が多く，また実環境のデータは少なからずノイズを含んでいる．確率密度推定は機械学習において重要なタスクであるが，このようなビッグデータに対して行うのは困難である．そこで，本研究では実環境の大量のデータに対応するために，高いロバスト性を有する高速オンライン多変量ノンパラメトリック密度推定法を提案する．提案手法はカーネル密度推定法と自己増殖型ニューラルネットワーク（SOINN）を拡張した手法である．SOINN はデータのプロトタイプをノードとするネットワークとしてデータを学習するクラスタリング手法であるが，そのネットワークに対して新たに統計的な意味付けをし拡張することで確率密度推定を可能にした．提案手法は評価実験において，学習時間，ロバスト性に関して，既存手法より高い性能を示し，推定精度に関しても，同等またはそれ以上の性能を示すことができた．

キーワード　カーネル密度関数，オンライン学習，SOINN，ビッグデータ

1. まえがき

近年，センサーやネットワーク技術の発展に伴って，大量のデータが生成されるようになってきている．このようなデータは，「ビッグデータ」と呼ばれ，これらを解析し活用することに対して関心，期待が高まっている [1]〜[4]．

ビックデータの特徴として，以下の 3 点を Laney [5] が挙げている：

- Volume：データ量．ペタバイト級にもおよぶ膨大なデータが生成されるようになってきており，また，その量は増加し続けている．
- Velocity：データが生成される速度．膨大な数のセンサーから継続的にリアルタイムにデータが生成されるため，単位時間あたりに生成されるデータ量（速度）は非常に大きい．
- Variety：データの種類の多様性．センサーデータなどの環境から取り入れたものや，ライフログや購買データ，株価などの人間の活動において生成されるデータ，Web 上に生成されたテキストデータや画像データなど多岐にわたるデータが生成されている．

これらは頭文字を取って「3Vs」として知られており，ビッグデータを処理する上で考慮すべき特徴である．

データ解析や機械学習において確率密度推定は重要なタスクである．しかし，このようなビッグデータに対して，確率密度推定を行うことは既存手法には困難である．なぜなら，次の三つの条件を満たす必要があると考えられるからである．

まず第 1 に，オンライン学習手法である点である．ビックデータの本質はデータの生成される速度 Velocity にある．データが高速に大量に生成されているため，結果的にデータの総量 Volume が莫大になるのである．そのため，ビックデータに対応するには継続的にリアルタイムに大量に生成されるデータを処理しなくてはならず，バッチ学習では対応できない．なぜなら，学習が終わる前に新たなデータが生成されてしまい，またデータ量が増えるに従って計算量が大幅に増

† 東京工業大学大学院総合理工学研究科知能システム科学専攻，横浜市
Department of Computational Intelligence and System Science, Tokyo Institute of Technology, Yokohama-shi, 226–8503 Japan
†† 東京工業大学像情報工学研究所，横浜市
Imaging Science and Engineering Laboratory, Tokyo Institute of Technology, Yokohama-shi, 226–8503 Japan
a) E-mail: nakamura.y.ba@m.titech.ac.jp
b) E-mail: hasegawa.o.aa@m.titech.ac.jp

加してしまうため，学習が間に合わないからである．したがって，生成されるデータを逐次的に学習できる高速なオンライン学習手法である必要がある．

第 2 に，ノンパラメトリックな手法である点である．ビッグデータの解析の主な目的はデータマイニング・知識発見であり，多様な種類のデータを用いるとともに，いかようなデータが取得できるかをあらかじめ予測することが難しい場合が多い．したがって，観測データが既知のパラメトリックな分布から発生したと仮定し，そのパラメータを推定することで密度関数を推定するパラメトリック密度推定法は適用できない．実際のデータの分布が仮定した分布と異なっていた場合，大幅な推定精度の低下を招いてしまうためである．パラメトリック密度推定法を用いることができるのは，よく知られた問題や専門家によって解析されモデルが立てられる種類のデータに対してだけである．未知の分布や既知のパラメトリックな分布としてモデル化できない分布に対して，パラメトリック密度推定法を用いて密度関数を推定することは困難である．これに対して，ノンパラメトリック密度推定法は，分布を仮定せず，データのみから密度関数を推定するため，このような問題は生じない．

第 3 に，高いロバスト性を有する点である．データは実環境から取り入れられたものであり，ノイズを多く含むと考えられる．ノイズを多く含むデータを用いて学習した場合，ロバスト性のない手法では，ノイズに対しても適合してしまい過学習を起こし汎化性能の低下につながる．また．ノイズの分布を仮定することでノイズに対処することも考えられるが，仮定したノイズの分布が実際のノイズの分布と異なっていた場合，著しい性能の低下を招いてしまう．人手でノイズを除去することもできるが，大規模に高速に生成されるデータにおいては非常に困難である．したがって，ノイズを含むデータからでも学習できる高いロバスト性を有する手法が必要である．

ロバスト性に関しては様々な分野で研究されており，各分野でロバスト性の定義が微妙に異なっている [6]〜[8]．本論文においては，ロバスト性を，外れ値や欠損値などのノイズを含まない理想的な状態ではないデータを学習した場合でも，理想的なデータの場合と同様の学習結果が得られる性質と定義する．ノイズは 2 種類に大別されると考えられる：

(1) 外れ値や異常値，拡散または出現頻度の低い分布から発生したサンプルなどの，目的の分布とは無関係であり，環境から発生したと考えられるノイズ，
(2) クラス内変動，分散，または揺らぎなどの，現象に内在し確率分布として捉えるべきノイズ．

密度推定におけるロバスト性とは (1) のノイズを除去できる性質のことである [9]．

本研究ではビックデータに対して確率密度推定を行うために，上記の三つの条件を満たした確率密度推定法を提案する．

2. 関連研究

代表的なノンパラメトリック密度推定法としてカーネル密度推定法 (kernel density estimation：KDE) [10] が挙げられる．入力データ $\{\boldsymbol{x}_i|\boldsymbol{x}_i \in \mathbb{R}^d,\ i = 1, 2, \cdots, N\}$ に対する KDE による推定密度関数 $\hat{p}(\boldsymbol{x})$ は，

$$\hat{p}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} K_{\boldsymbol{H}}(\boldsymbol{x} - \boldsymbol{x}_i) \tag{1}$$

と表せる．ここで K はカーネル関数と呼ばれ，次のガウスカーネルがよく用いられる：

$$K_{\boldsymbol{H}}(\boldsymbol{x} - \boldsymbol{\mu}) = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{H}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{H}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right). \tag{2}$$

ここで $\boldsymbol{H}$ はバンド幅行列と呼ばれるパラメータである．

このバンド幅行列は性能に大きく影響を与えるため，この最適化に関して多くの研究が行われているが [11]〜[16]，これらの手法はバッチ学習手法であり，大規模データに対応できない．KDE では入力サンプル全てに対してカーネルを配置するため，サンプル数の増加に伴い，時間計算量，空間計算量ともに増加してしまう．しかし，精度を出すためにはサンプル数が必要 [11] であり，計算量と精度がトレードオフの関係にある．

計算量を抑えるために，サンプルの分布の複雑さに応じて適切にカーネルを配置したり，オンライン学習手法を取り入れるなどの工夫が必要である．カーネルの数を設定する手法が幾つか提案されている [17]〜[19]．オンライン化の方法としては，損失やゆう度の最適化問題を考え勾配法を適用する [20] のが一般的だが，KDE に対して適用した手法は探した限り見つからなかった．KDE はサンプルが増えるたびにモデルが変化するため，オンラインに最適化を行うのが困難なためではないかと思われる．クラスタリングなどを

用いることで KDE をオンライン化した手法は考案されている [21]〜[23]．しかし，これらはロバスト性と学習速度が十分とは言えない．

式 (1) が示すように，KDE では入力サンプル全てに対してカーネルを配置し，その線形和で密度関数を表現する．入力サンプルにノイズを含む場合，ノイズにもカーネルを配置するため，ノイズに対しても適合してしまい過学習を起こしてしまう．ノイズに対処する手法としては，ノイズに対する事前知識を導入する手法が考案されている [24]．KDE をカーネル法として捉え，ロバストな推定を行うことでロバストな KDE を実現している手法 [9] もあるが，この手法はバッチ学習手法であり，大規模データには対応できない．

3. 提案手法

提案手法は KDE と Self-Organizing Incremental Neural Network (SOINN) [25] を拡張した手法である．SOINN は様々な応用が考えられてきたが [26]〜[31]，基本的にはクラスタリング器として用いられてきた．しかし本論文では，SOINN のネットワーク構造がサンプルの分布の情報を保持していると新たに意味付けることで確率密度推定を行う．

提案手法の概略図を図 1 に示す．SOINN を拡張したアルゴリズムを用い，サンプルのプロトタイプをノードとするネットワークとしてサンプル集合を学習し，そのネットワークの局所構造を利用して各ノードに配置するカーネルの形状と大きさを決定し，カーネルの線形和として密度関数を推定する．

3.1 Self-Organizing Incremental Neural Network

3.1.1 SOINN の概要

SOINN は Growing Neural Gas (GNG) [32] を拡張したプロトタイプベースの教師なし学習手法である．SOINN には複数のバージョンのアルゴリズムが存在するが [25], [33], [34]，本論文では Adjusted SOINN [34] を SOINN と呼称することとする．オリジナルの SOINN [25] は 2 層構造であるのに対し，Adjusted SOINN は 1 層構造であり，またハイパーパラメータの数も少なく扱いやすいことから，近年の SOINN の応用研究では Adjusted SOINN が用いられることが多い [26]〜[30]．本論文でもこの理由から Adjusted SOINN を用いる．

SOINN による学習の例を図 2 に示す．SOINN はオンライン学習及び追加学習が可能であり，複雑で非定常な分布からのサンプルも，ノイズを除去しつつ，入力サンプル数より大幅に少ないノード数のネットワーク構造として学習できる．ノード数やノードの初期位置を事前に設定する必要がないため追加学習に適している．

SOINN ではネットワーク構造を用いて入力サンプル集合を学習する．オンラインに入力されるサンプルに対して，ネットワーク上のノードが競合学習を行う．それに伴って，ノードの挿入・削除・位置の更新及びエッジの挿入・削除が必要に応じて行われ，サンプル集合の分布を近似するようにネットワーク構造が自己

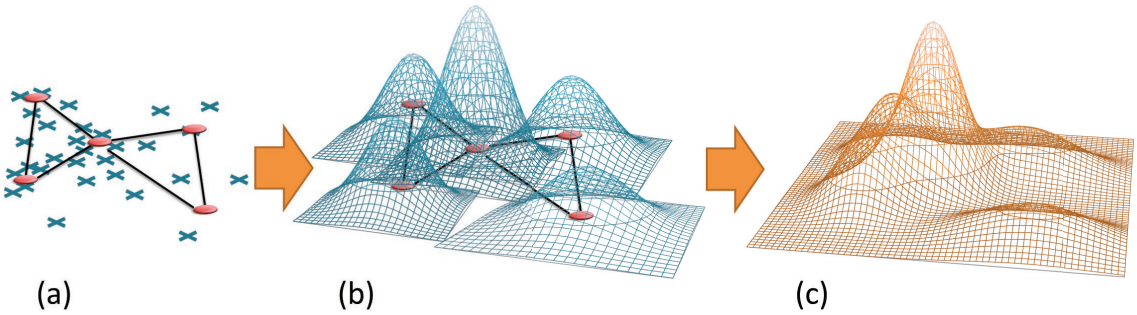


図 1 提案手法の概略図：(a) サンプルのプロトタイプがノードであるネットワークとして，サンプルをオンラインに学習．(b) ネットワークの局所構造を利用して各ノードに配置するカーネルの形状と大きさを決定．(c) 各カーネルの線形和として密度関数を推定

Fig. 1 The overview of poposed method: (a) Online learn samples as networks whose nodes are prototypes of samples. (b) Determine the shape and size of each kernel on a node by the local structure of a network around the node. (c) Estimate the probability density function of samples as linear summation of the kernels.

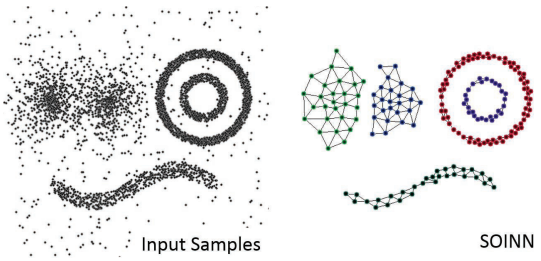


図 2　SOINN による学習．左図がノイズを含んだ入力サンプル集合，右図が SOINN による学習結果
Fig. 2　Learning by SOINN. Left figure is the set of input samples including noise. Right figure is the result of learning by SOINN.

表 1　変数，パラメータ及び記号の定義
Table 1　The definition of variables, parameters and symbols.

$\boldsymbol{\xi}$	入力サンプル．$\boldsymbol{\xi} \in \mathbb{R}^d$.
$\mathcal{N}$	全ノードの集合.
$\mathcal{E}$	全エッジの集合．$\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$.
$\mathcal{P}_i$	ノード i の隣接ノード集合.
$\boldsymbol{w}_i$	ノード i の位置ベクトル．$\boldsymbol{w}_i \in \mathbb{R}^d$.
t_i	ノード i が競合学習において勝者になった回数.
a_e	エッジ e の年齢.
Θ_i	エッジ，ノード作成におけるノード i のしきい値.
$\psi_k(t)$	競合学習の勝者ノードの更新における係数決定のための関数．$\psi_1(t) = t^{-1}$，$\psi_2(t) = (100t)^{-1}$
λ	ノード削除ハイパーパラメータ.
age_{max}	エッジ削除ハイパーパラメータ.
ρ	しきい値ハイパーパラメータ.
$\boldsymbol{I}$	単位行列.
$\lvert\mathcal{S}\rvert$	集合 $\mathcal{S}$ の要素数.

組織的に更新されていく．

3.1.2　SOINN のネットワークの統計的解釈

SOINN のアルゴリズムを Algorithm 1 に示す．また，アルゴリズム内に現れる記号を表 1 に定義した．SOINN では，入力サンプルの最近傍 2 ノード（第 1 勝者ノード，第 2 勝者ノード）のしきい値領域に入力サンプルが入らなかった場合は新規ノードが作成され，入った場合は第 1，第 2 勝者ノード間にエッジが作成され新規ノードは作成されず，入力サンプルは第 1 勝者ノードの勝者回数としてカウントされ，第 1 勝者ノード及びその隣接ノードの位置ベクトルを入力サンプルの方向に更新する．

この第 1 勝者ノードの更新，Algorithm 1 の 18 行目は，平均のオンライン更新と一致する．$\{x_1, x_2, \cdots, x_n\}$ の平均を

$$\boldsymbol{w}^{(n)} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$$

Algorithm 1 SOINN

1: **if** 初回の学習である **then**
2:　$\mathcal{N} \leftarrow \{c_1, c_2\}$：学習データからランダムに選択したベクトルを位置ベクトルとしてもつ二つのノード c_1，c_2 で全ノードの集合を初期化
3:　$\mathcal{E} \leftarrow \phi$
4: **end if**
5: **while** 入力パターン $\boldsymbol{\xi} \in \mathbb{R}^d$ が存在する **do**
6:　$s_1 \leftarrow \arg\min_{c \in \mathcal{N}} ||\boldsymbol{\xi} - \boldsymbol{w}_c||$：$\boldsymbol{\xi}$ に対する第 1 勝者ノード s_1 を探索
7:　$s_2 \leftarrow \arg\min_{c \in \mathcal{N} \setminus \{s_1\}} ||\boldsymbol{\xi} - \boldsymbol{w}_c||$：$\boldsymbol{\xi}$ に対する第 2 勝者ノード s_2 を探索
8:　式 (6) によって類似度しきい値 Θ_{s_1}，Θ_{s_2} を計算
9:　**if** $||\boldsymbol{\xi} - \boldsymbol{w}_{s_1}|| > \Theta_{s_1}$ **or** $||\boldsymbol{\xi} - \boldsymbol{w}_{s_2}|| > \Theta_{s_2}$ **then**
10:　$\mathcal{N} \leftarrow \mathcal{N} \cap \{\boldsymbol{\xi}\}$：$\boldsymbol{\xi}$ を新規ノードとして追加
11:　**else**
12:　**if** $(s_1, s_2) \notin \mathcal{E}$：$s_1$ と s_2 の間にエッジがない **then**
13:　$\mathcal{E} \leftarrow \mathcal{E} \cap \{(s_1, s_2)\}$：エッジ (s_1, s_2) を追加
14:　**end if**
15:　$a_{(s_1,s_2)} \leftarrow 0$：エッジ (s_1, s_2) の年齢を 0 にリセット
16:　$a_{(s_1,i)} \leftarrow a_{(s_1,i)} + 1$ $(\forall i \in \mathcal{P}_{s_1})$
：s_1 につながる全エッジの年齢をインクリメント
17:　$t_{s_1} \leftarrow t_{s_1} + 1$：$s_1$ の勝者回数をインクリメント
18:　$\boldsymbol{w}_{s_1} \leftarrow \boldsymbol{w}_{s_1} + \psi_1(t_{s_1})(\boldsymbol{\xi} - \boldsymbol{w}_{s_1})$
：s_1 の位置ベクトルを更新
19:　$\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i + \psi_2(t_i)(\boldsymbol{\xi} - \boldsymbol{w}_i)$ $(\forall i \in \mathcal{P}_{s_1})$
：s_1 の全隣接ノードの位置ベクトルを更新
20:　エッジ $\mathcal{E}_{old} = \{e \mid e \in \mathcal{E},\ a_e > age_{max}\}$ を削除
21:　ノード $\{i \mid \exists j\ (i,j) \in \mathcal{E}_{old}, |\mathcal{P}_i| = 0\}$ を削除
：20 行目において削除されたエッジに接続していたノードのうち，接続エッジがなくなったノードを削除.
22:　**end if**
23:　**if** 入力パターン数が λ の倍数 **then**
24:　$|\mathcal{P}_i| \leq 1$ であるノード i を全て削除
25:　**end if**
26: **end while**

とすると，$\{x_1, x_2, \cdots, x_{n+1}\}$ の平均 $\boldsymbol{w}^{(n+1)}$ は

$$\begin{aligned}\boldsymbol{w}^{(n+1)} &= \frac{1}{n+1}(n\boldsymbol{w}^{(n)} + \boldsymbol{x}_{n+1}) \\ &= \boldsymbol{w}^{(n)} + \frac{1}{n+1}(\boldsymbol{x}_{n+1} - \boldsymbol{w}^{(n)}) \qquad (3)\end{aligned}$$

と，$\boldsymbol{w}^{(n)}$ と $\boldsymbol{x}_{n+1}$ から計算でき，この式は Algorithm 1 の 18 行目と一致する．つまり，各ノードは勝者回数としてカウントされたサンプルを代表しており，それらの平均に位置している．Algorithm 1 の 19 行目はスムージングのためであり，$\psi_2(t)$ の係数は実験的に決められたものである．

また，Algorithm 1 の 16，20 行目が示すとおり，エッジは，端点のノード間へのサンプルの出現頻度が周辺のエッジと比較して低かった場合，切断されるようになっている．つまり，ノード間にエッジがあるということは，そのノード間のサンプルの出現頻度が相対的に大きいということになり，ノードが代表するサ

ンプルがエッジの方向に広がっているということを意味すると考えられる．

以上をまとめると，ノードは周辺のサンプルを代表するプロトタイプであり，そのノードの周辺ネットワークはノードの周辺サンプルの広がりの大きさとその方向を表していると考えられる．

3.2 推定確率密度関数

提案手法では **3.1.2** で述べた SOINN のネットワーク構造が表現するサンプルの分布の情報を用いて，密度推定を行う．

提案手法の推定密度関数 $\hat{p}(\boldsymbol{x})$ を以下のように定義する：

$$\hat{p}(\boldsymbol{x}) = \frac{1}{T_{\mathcal{N}}} \sum_{n \in \mathcal{N}} t_n K_{\boldsymbol{C}_n}(\boldsymbol{x} - \boldsymbol{w}_n). \tag{4}$$

ここで $T_{\mathcal{N}} = \sum_{n \in \mathcal{N}} t_n$，$K$ は式 (2) のガウスカーネルである．各ノードの位置にガウスカーネルを配置して，その線形和としてサンプル集合の確率密度を推定する．ノード n に配置したカーネルは，ノード n が代表しているサンプルの数を表している勝者回数 t_n によって重み付けされている．

ガウスカーネルの形状を決定する共分散行列 $\boldsymbol{C}_n$ は各ノードの周辺のネットワーク構造から各ノードごとに適応的に決定される．この共分散行列を局所ネットワーク共分散行列と定義する：

$$\boldsymbol{C}_n = \frac{1}{T_{\mathcal{P}_n}} \sum_{p \in \mathcal{P}_n} t_p (\boldsymbol{w}_p - \boldsymbol{w}_n)(\boldsymbol{w}_p - \boldsymbol{w}_n)^T. \tag{5}$$

ここで，$T_{\mathcal{P}_n} = \sum_{i \in \mathcal{P}_n} t_i$ である．局所ネットワーク共分散行列は，共分散行列を拡張したものである．通常の共分散行列と異なる点は，中心が平均ではなく対象ノード n の位置ベクトルである点と，対象ノード n の隣接ノードのみで計算している点，勝者回数 t で重み付けしている点である．n の隣接ノードが代表しているサンプルが，n から隣接ノードまでの距離と方向に広がっているとして，n の周辺のサンプルの広がりをこの局所ネットワーク共分散行列で近似しており，これをノードを中心とするガウスカーネルに組み込むことで，周辺のサンプルの密度を近似する．なお，ネットワークの形状によっては $\boldsymbol{C}_n$ が正則にならない場合があるが，その場合は対角成分に微小な値を加えることで補正している．

3.3 学習アルゴリズム

Algorithm 2 に学習アルゴリズムの全体を示す．

Algorithm 2 学習アルゴリズム

if 初回の学習である **then**
  $\mathcal{N} \leftarrow \{c_1, c_2\}$ (c_1, c_2：学習データからランダムに選択したベクトルを位置ベクトルとしてもつ二つのノード)
  $\mathcal{E} \leftarrow \phi$
end if
while 入力パターン $\boldsymbol{\xi} \in \mathbb{R}^d$ が存在する **do**
  $s_1 \leftarrow \arg\min_{c \in \mathcal{N}} ||\boldsymbol{\xi} - \boldsymbol{w}_c||$
  $s_2 \leftarrow \arg\min_{c \in \mathcal{N} \setminus \{s_1\}} ||\boldsymbol{\xi} - \boldsymbol{w}_c||$
  if $(\boldsymbol{\xi} - \boldsymbol{w}_{s_1})^T \boldsymbol{M}_{s_1}^{-1} (\boldsymbol{\xi} - \boldsymbol{w}_{s_1}) > 1$
    or $(\boldsymbol{\xi} - \boldsymbol{w}_{s_2})^T \boldsymbol{M}_{s_2}^{-1} (\boldsymbol{\xi} - \boldsymbol{w}_{s_2}) > 1$ **then**
    $\mathcal{N} \leftarrow \mathcal{N} \cap \{\boldsymbol{\xi}\}$
  else
    if $(s_1, s_2) \notin \mathcal{E}$ **then**
      $\mathcal{E} \leftarrow \mathcal{E} \cap \{(s_1, s_2)\}$
    end if
    $a_{(s_1, s_2)} \leftarrow 0$
    $a_{(s_1, i)} \leftarrow a_{(s_1, i)} + 1$ $(\forall i \in \mathcal{P}_{s_1})$
    $t_{s_1} \leftarrow t_{s_1} + 1$
    $\boldsymbol{w}_{s_1} \leftarrow \boldsymbol{w}_{s_1} + \psi_1(t_{s_1})(\boldsymbol{\xi} - \boldsymbol{w}_{s_1})$
    エッジ $\mathcal{E}_{old} = \{e \mid e \in \mathcal{E},\ a_e > age_{max}\}$ を削除
    ノード $\{i \mid \exists j\ (i, j) \in \mathcal{E}_{old}, |\mathcal{P}_i| = 0\}$ を削除
  end if
  if 入力パターン数が λ の倍数 **then**
    $|\mathcal{P}_i| = 0$ であるノード i を全て削除
    $\mathcal{N}$ に対して k 近傍グラフ $G = (\mathcal{N}, \mathcal{E}_k)$ を作成
    $\mathcal{E} \leftarrow \mathcal{E} \cap \{(i, j) \mid (i, j) \in \mathcal{E}_k, (j, i) \in \mathcal{E}_k\}$
  end if
end while
$\mathcal{N}$ に対して k 近傍グラフ $G = (\mathcal{N}, \mathcal{E}_k)$ を作成
$\mathcal{E} \leftarrow \mathcal{E} \cap \{(i, j) \mid (i, j) \in \mathcal{E}_k, (j, i) \in \mathcal{E}_k\}$

3.3.1 しきい値領域の決定

学習アルゴリズムにおいて SOINN から拡張した重要な点に，しきい値領域の決定が挙げられる．新規入力サンプルが勝者ノードのしきい値領域内に入るかどうかでエッジ及び新規ノードの作成の判断を行うため，SOINN のアルゴリズムにおいて最も重要な部分である．

SOINN において，ノード i のしきい値領域は次の Θ_i を半径とする超球である：

$$\Theta_i = \begin{cases} \max\limits_{p \in \mathcal{P}_i} ||\boldsymbol{w}_p - \boldsymbol{w}_i|| & (\mathcal{P}_i \neq \phi) \\ \min\limits_{p \in \mathcal{N} \setminus \{i\}} ||\boldsymbol{w}_p - \boldsymbol{w}_i|| & (otherwise). \end{cases} \tag{6}$$

図 3 の (a) に SOINN のしきい値領域を図示する．あるノードの周りのネットワークがある方向に大きく広がっていた場合，そのノード周辺のサンプルはネットワークに沿った方向に広がっており，エッジのない方向にはあまりサンプルが存在しないことを意味する．したがって，エッジのない方向にしきい値領域が広がると，サンプルの密度と関係のない必要以上に長い

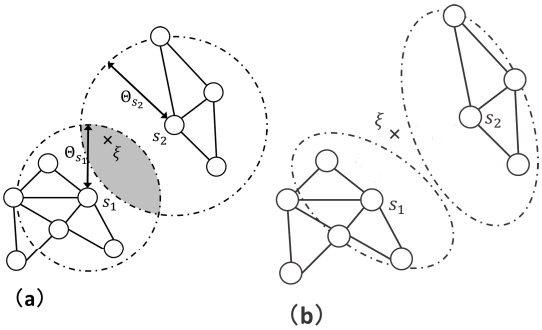


図 3　(a) SOINN のしきい値領域，(b) 提案手法のしきい値領域
Fig. 3　(a) The threshold of SOINN. (b) The threshold of proposed method.

エッジができてしまい，その後のネットワーク構成に悪影響を及ぼし，ネットワークが適切に密度を表さなくなってしまう可能性が高い．

しきい値領域をサンプルの密度によって適切に決めることで，ネットワークのひずみをなくし，より密度を表現するネットワーク構造を作ることができると考えられる．図 3 の (b) に提案手法のしきい値領域を図示している．提案手法では，ネットワーク構造がサンプルの密度を表しているとし，しきい値領域をネットワークに沿う形にし，エッジのない方向にはしきい値領域が広がらないようにしている．こうすることで，SOINN ではサンプルの密度と関係のないエッジになっていた入力サンプルは，しきい値領域に入らずにノードとなり，その後の入力サンプルによってネットワークに組み込まれるか，ノイズとして削除されるか決定される．

提案手法ではしきい値領域をネットワークに沿う形にするために，局所ネットワーク共分散行列式 (5) を用いたマハラノビス距離によってしきい値領域を定義する．ノード i のしきい値領域は，サンプル $\boldsymbol{\xi}$ に対して

$$(\boldsymbol{\xi}-\boldsymbol{w}_i)^T\boldsymbol{M}_i^{-1}(\boldsymbol{\xi}-\boldsymbol{w}_i)\leq 1,$$

を満たす領域とする．ここで

$$\boldsymbol{M}_i=\boldsymbol{C}_i+\rho\gamma_i\boldsymbol{I},$$
$$\gamma_i=\begin{cases}\min\limits_{p\in\mathcal{P}_i}||\boldsymbol{w}_p-\boldsymbol{w}_i|| & (\mathcal{P}_i\neq\phi)\\ \min\limits_{p\in\mathcal{N}\setminus\{i\}}||\boldsymbol{w}_p-\boldsymbol{w}_i|| & (otherwise)\end{cases}$$

である．単位行列を加えているのは，スムージングのためである．

3.3.2　その他の拡張

しきい値領域決定以外にも適切なネットワークを形成するように変更を加えている．SOINN のアルゴリズム Algorithm 1 における 19 行目は廃止している．これはスムージングのために実験的に決められたものであり，**3.1.2** で示したことを考慮すると，密度推定の観点からは必要ないと考えられるためである．22 行目では $|\mathcal{P}_i|\leq 1$ ではなく $|\mathcal{P}_i|=0$ としている．23〜24，27〜28 行目ではネットワークの調整を行っている．SOINN ではノード間のしきい値領域内にサンプルが入力されないとエッジが作成されないため，高次元空間やサンプル数が少ない場合，十分な数のエッジが作成されない．これを補うために，ここでは k 近傍グラフに基いてエッジを追加している．ノード集合に対して k 近傍グラフを形成し，そのグラフにおいて双方向にエッジが存在するノードのペアに対して，エッジを追加する．ノードが密に分布する場所では k 近傍グラフのエッジは双方向に形成されるので，密度を表すエッジが張れると考えられる．

また，6〜7 行目の競合学習の勝者選択においては，マハラノビス距離ではなく，SOINN と同様にユークリッド距離を用いている．これは，各ノードを同一の基準で評価することで，競合学習を適切に行うためである．それに加えて，ここでは全ノードと距離計算を行う必要があるため，マハラノビス距離を用いると計算コストが大きく掛かってしまうためである．

4.　評 価 実 験

ロバスト性，学習時間，オンライン学習，精度に関して提案手法の優位性を示すための評価実験を行った．全ての実験は 3.20 GHz CPU × 8，8.00 GB RAM，の計算機上の MATLAB 環境下で行った．

4.1　比 較 手 法

既存の比較手法として，クラスタリングによりカーネルの形状と数を適応的に決定するオンライン KDE (oKDE) [23]，ゆう度を評価尺度とし交差検証法によりバンド幅行列を最適化するバッチ学習による手法 (CVML) [16]，ロバスト性を考慮したバッチ手法の KDE (RKDE) [9] を用いた．RKDE は他の手法でバンド幅行列を求める必要があるため，CVML が最適化したバンド幅行列を RKDE のバンド幅行列として用いた．

4.2　ロバスト性

ロバスト性を評価するために，学習データにノイズ

が含まれる場合の推定精度を既存手法と比較した．学習データは

$$\boldsymbol{X} \sim f = (1-\alpha)f_0 + \alpha f_1,$$

とする．f_0 が真の分布，f_1 がノイズの分布であり，α はノイズの割合を表す．各手法は $\boldsymbol{X}$ を学習し，f_0 を推定する．α を変化させ，それに伴う真の分布の推定精度の変化を評価した．真の分布には

$$f_0(\boldsymbol{x}) = N(\boldsymbol{x}; \boldsymbol{t}, 0.2 \cdot \boldsymbol{I}), \quad (7)$$
$$\boldsymbol{t} = [a, \sin(3a)]^T, a \in [-2, 2],$$
$$f_0(\boldsymbol{x}) = N(\boldsymbol{x}; \boldsymbol{t}, 0.25 \cdot \boldsymbol{I}), \quad (8)$$
$$\boldsymbol{t} = [r\cos(\theta), r\cos(\theta), \theta]^T,$$
$$r = 10 - 0.5\theta, \theta \in [-7, 7],$$

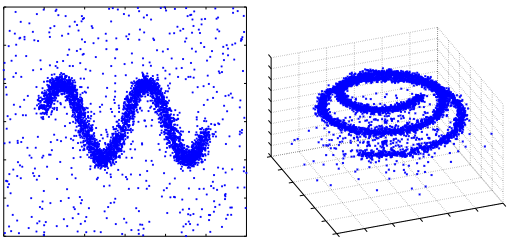
の二つの分布を用い，ノイズの分布 f_1 にはそれぞれ一様分布（各次元の範囲を $[-4, 4]$ とした）とガウス分布 $N(\boldsymbol{x}; [0, 0, -15]^T, 30 \cdot \boldsymbol{I})$ を用いた．学習サンプル数は 5,000 とした．図 4 にそれぞれ図示する．

評価尺度には JS divergence を用いた．JS divergence は確率分布間の距離尺度であり，推定分布と真の分布を比較した場合，値が低いほど推定精度が高いといえる．KL divergence と違い，対称であり無限大に発散することがないので今回はこれを用いた．推定確率分布 $\hat{f}$ と f_0 間の JS divergence $D_{JS}(\hat{f}||f_0)$ は

$$D_{JS}(\hat{f}||f_0) = \frac{1}{2}D_{KL}(\hat{f}||m) + \frac{1}{2}D_{KL}(f_0||m)$$

である．ここで $m = \frac{1}{2}(\hat{f} + f_0)$，$D_{KL}$ は KL divergence であり，

$$D_{KL}(p||q) \approx \frac{1}{n}\sum_{i=1}^{n} \log \frac{p(x_i)}{q(x_i)}$$



(a) (b)

図 4 (a)，(b) はそれぞれ式 (7)，式 (8) の分布に 20%のノイズを含めた学習データ

Fig. 4 Training samples generated from Eqs. (7) and (8) distributions with 20% noise samples.

である．ここで $\{x_i\}_{i=1}^n$ は p からのサンプルである．よって，

$$D_{JS}(\hat{f}||f_0) \approx \frac{1}{2n}\sum_{i=1}^{n} \log \frac{\hat{f}(x_{\hat{f}_i})}{m(x_{\hat{f}_i})} + \frac{1}{2n}\sum_{i=1}^{n} \log \frac{f_0(x_{f_0 i})}{m(x_{f_0 i})}$$

となる．ここで，$\{x_{\hat{f}_i}\}_{i=1}^n$ は $\hat{f}$ からのサンプルであり，$\{x_{f_0 i}\}_{i=1}^n$ は f_0 からのサンプルである．今回の実験においては，$n = 20{,}000$ とした．各ノイズ割合に対して 20 回ずつ実験を行い，その平均を評価した．

oKDE のハイパーパラメータを $D_{th} = 0.01$ とした．提案手法のハイパーパラメータは式 (7) の場合，$\lambda = 500$，$age_{max} = 100$，$\rho = 0.1$，$k = 2d$，式 (8) の場合，$\lambda = 1000$，$age_{max} = 100$，$\rho = 1.0$，$k = 2d$ とした．

図 5 が実験結果である．図 5 (a) において，学習データにおけるノイズ割合が 0%の場合では，提案手法は他手法とほぼ同等の精度を出している．既存手法の CVML と oKDE は学習データにおけるノイズの割合が増加するに従って大きく精度が低下している．CVML は 0%と 5%の差が大きいことからノイズに敏感に反応してしまうことが分かる．oKDE はカーネルの形状，大きさ，及び数をデータに合わせて適応的に設定するため CVML ほど敏感には反応していないと考えられる．しかし，それだけではロバスト性は十分ではなくノイズの割合が増加するに従って精度が低下している．これに対して，ロバスト性のある提案手法と RKDE はノイズが増加しても精度が低下しない．特に提案手法は 0%のときと同程度の水準を保っている．図 5 (b) においても同様のことが言える．このことから提案手法が高いロバスト性を有していると示された．

4.3 学習時間

学習データ数に対する学習時間の変化を評価した．ノイズを含まない式 (7) の分布からのサンプルを用いて，データ数を 1,000 から 100,000 まで増加させたときの計算時間の変化を既存手法と比較した．各手法のパラメータは **4.2** と同じとした．

実験結果を図 6 に示す．CVML 及び RKDE は非常に学習に時間が掛かるため，学習サンプル数が 10,000 の時点で実験を中止した．このとき，提案手法は CVML より 1,390 倍高速であった．CVML はバッチ手法であり，学習サンプル数の増加に伴って学習時

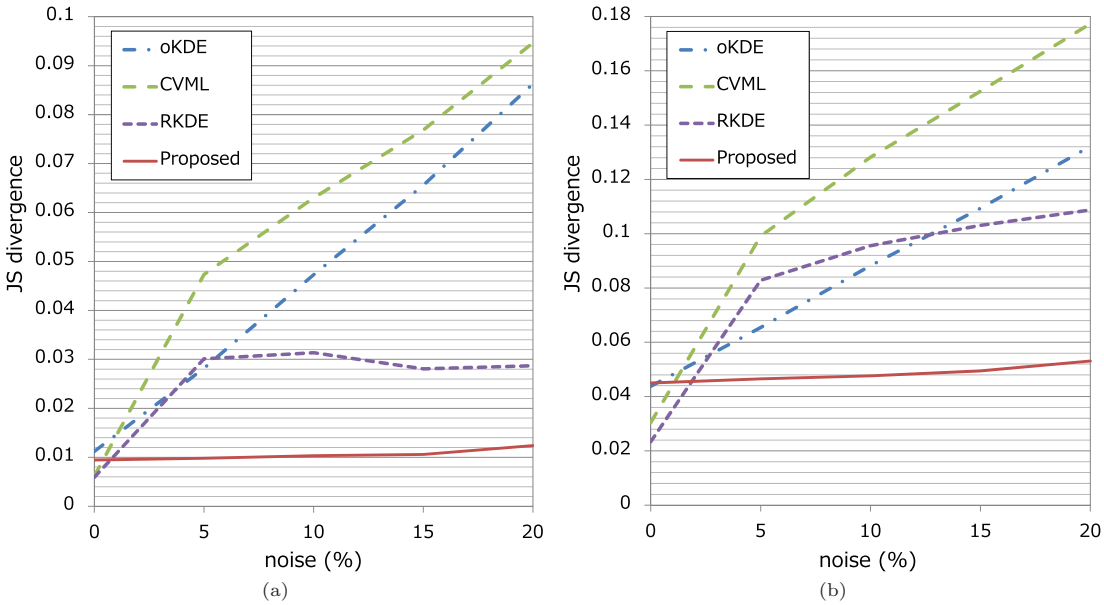


図 5　ロバスト性に関する既存手法との比較実験の結果．(a)，(b) はそれぞれ式 (7)，式 (8) の分布に対する実験結果

Fig. 5　The experimental results about robustness comparing state-of-arts and proposed method. (a) and (b) correspond to the results with Eqs. (7) and Eqs. (8) distributions, respectively.

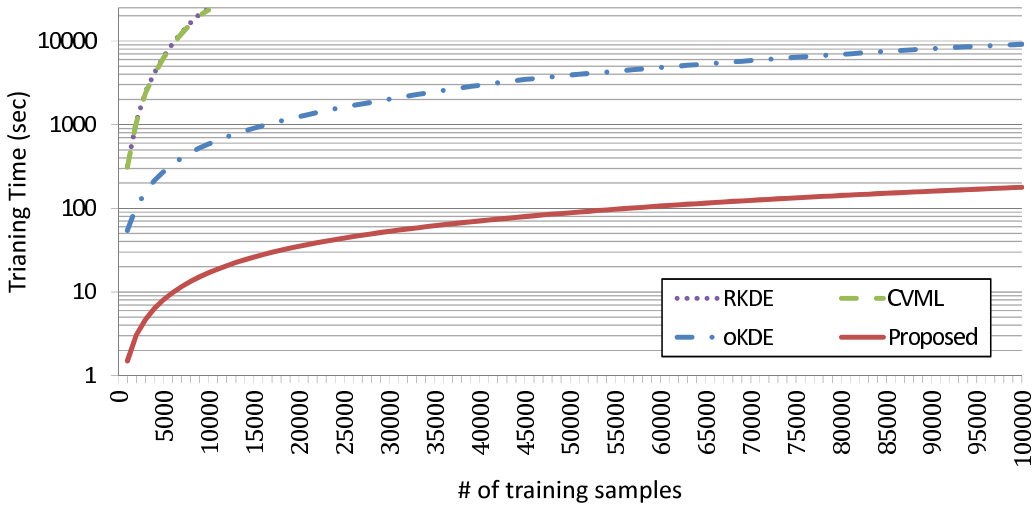


図 6　学習時間に関する評価実験の結果．提案手法は CVML [16] より 1,390 倍，oKDE [23] より 51.4 倍高速である．縦軸が対数表示であることに注意．

Fig. 6　The experimental result about training time of online methods. Our proposed method is 1,390 times faster than CVML [16], and 51.4 times faster than oKDE [23]. Note that vertical axis is plotted on a logarithmic scale.

間が指数関数的に増加している．oKDE と提案手法はオンライン手法であるため CVML ほど学習時間は掛かっていないが，oKDE は学習サンプル数が増加するとともに学習時間が大きく増加しているのに対し，提案手法は増加が小さい．提案手法は，学習サンプル数が 100,000 のとき oKDE より 51.4 倍高速であった．oKDE はクラスタリングを用いて混合数を減らし，バンド幅行列を入力サンプルごとに計算し直しているの

に対し，提案手法はプロトタイプのネットワーク構造の中にカーネルの配置位置とバンド幅行列の情報を含めているため，より高速に実行できるのではないかと考えられる．

4.4 オンライン学習

オンライン学習手法はデータを追加的に学習できる．そこで，オンライン学習手法である提案手法と oKDE に対して，学習サンプルを追加したときの性能の変化を評価した．学習サンプル数を 100,000 まで追加的に増やし，1,000 サンプル学習するごとに各手法の学習時間と推定精度を評価した．サンプルを発生させる分布及び各手法のパラメータは **4.3** と同じ設定にした．推定精度の評価指標としては JS divergence を用いた．

実験結果を図 7 と図 8 に示す．

図 7 は学習したサンプル数に伴う学習時間の増加量，つまり 1,000 サンプルを追加的に学習するのに要した時間の変化を示す．学習したサンプル数の増加に伴い，oKDE の学習時間の増加量は増加する傾向にある．これに対して，提案手法の増加量は初期は増加するが，その後ほぼ一定である．つまり，提案手法の増加量はこれまで学習したサンプル数に直接的には依存しないことを示している．提案手法の学習における計算はノードとの距離計算が大きな部分を占めているが，ネットワークがデータの分布を十分近似するとノードの数はほとんど変化しないため，学習時間の増加量もほとんど変化しない．このことから，提案手法はデータ量が増大して大規模になっても高速に学習を行えることが分かる．

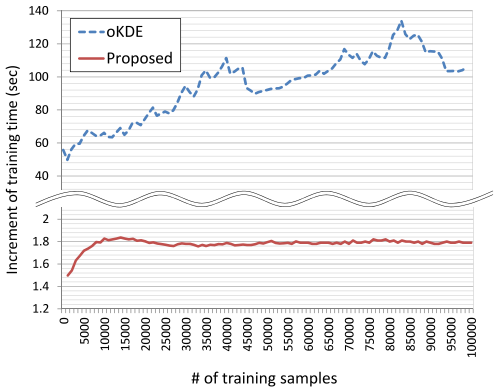


図 7 追加的学習における学習時間の増加量に関する評価実験結果

Fig. 7 The experimental result about increment of training time of incremental learning.

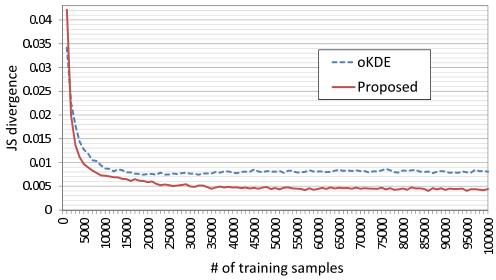


図 8 学習サンプル数の増加に伴う推定精度の変化の評価実験結果

Fig. 8 The experimental result about change in estimate accuracy with increasing training samples.

図 8 は学習したサンプル数に伴う推定精度の変化を示す．oKDE は学習サンプル数が増えるに従って学習時間の増加量が大きくなるにもかかわらず，JS divergence の値は学習サンプル数が 10,000 を超えたあたりからほとんど変わらず，学習の効果がない．これに対して，提案手法はサンプル数が増えても学習時間の増加量はほとんど増えないが JS divergence の値は徐々に下がっている．

4.5 実データに対する密度推定

UCI Machine Learning Repository のデータセットを使って実データにおける確率密度推定を評価した．実験に用いたデータセットを表 2 に示す．データセットのクラスごとに実験を行った．ただし，Skin は正例，負例の 2 クラスあるが，正例 (50,859 サンプル) のみ用いた．各実験のサンプル集合を白色化により正規化し，そこからランダムに抽出した 75%のサンプルを学習データ，残りをテストデータとし，密度関数の推定精度を評価した．これを各実験ごとに 20 回繰り返し，データセットごとに平均し評価した．評価尺度には Negative Log Likelihood (NLL) を用いた．NLL は無限大に発散する可能性があるため，順序平均をとることで対応した．提案手法のハイパーパラメータを $\lambda = 1000$，$age_{max} = 100$，$\rho = 1.0$，$k = 2d$，oKDE

表 2 実験に用いた実データセット一覧

Table 2 The list of real data set used in experiments.

	サンプル数	次元数	クラス数
Iris	150	4	3
Wine	178	13	3
Pima	768	8	2
Wine_Red	1,599	11	6
Wine_White	4,898	11	7
MAGIC	19,020	10	2
Skin	245,057	3	2

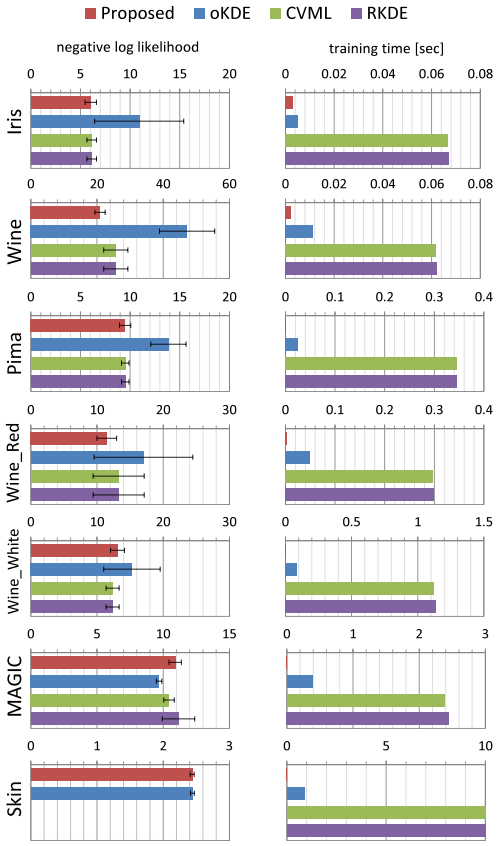


図 9　実データに対する評価実験の結果．密度推定の精度と 1 サンプルあたりの学習時間を評価した．提案手法の学習時間は値が小さすぎ見難いため，表 3 に記した

Fig. 9　The experimental results of real data. We evaluated methods about estimation accuracy and training time per sample. Our proposed method's training times are too small to see in this chart, so they showed in Table 3.

のハイパーパラメータを $D_{th} = 0.1$ とした．

実験結果を図 9 に示す．左列が NLL，右列が 1 サンプルあたりの学習時間である．棒グラフは平均値，それに付随するバーは標準偏差を表している．Skin データセットにおいては，CVML と RKDE は 5 日間計算し続けたが 1 度も実験が終わらなかったため値は記していない．提案手法は既存のバッチ学習手法と同等またはそれ以上の推定精度を出している．また，学習時間も他手法と比べて非常に短い．提案手法の学習時間は値が小さすぎ，図 9 では見難いため，表 3 に記した．他手法は学習サンプル数が増えるに従って学習時間が増加しているのに対し，提案手法はほとんど変化がない．以上のことから，提案手法は実データに対する密度推定において実用的に使用することができる．

表 3　提案手法の実データに対する実験結果

Table 3　Our proposed method's experimental results of real data.

	NLL	training time [sec]
Iris	6.03 ± 0.827	3.03×10^{-3}
Wine	20.9 ± 1.45	1.91×10^{-3}
Pima	9.49 ± 0.591	2.14×10^{-3}
Wine_Red	11.5 ± 2.11	2.31×10^{-3}
Wine_White	13.1 ± 1.28	2.42×10^{-3}
MAGIC	10.9 ± 0.440	2.02×10^{-3}
Skin	2.43 ± 0.0361	2.77×10^{-3}

4.6 まとめ

4. の実験結果を表 4 にまとめた．提案手法はロバスト性及び高速なオンライン学習に関して既存手法より優れており，推定精度に関しても他手法と同等またはそれ以上の性能を示した．

5. 考　　察

ここでは，提案手法における KDE 及び SOINN からの拡張の効果と，高次元への対応について考察する．

5.1 ナイーブな手法との比較

提案手法では，SOINN を改良した学習アルゴリズムでサンプル集合をネットワーク構造として学習し，そのネットワークの局所的な構造から各ノードに配置するガウスカーネルの形状及び大きさを決定し，それらのカーネルの線形和として確率密度関数を推定する．このようにすることで単純に SOINN のネットワークの各ノードに対してガウスカーネルを配置する場合よりも性能が向上していることをここで検証したい．

4.2 の実験を SOINN と KDE をナイーブに組み合わせた手法（naive）と，学習は SOINN で行い，密度関数を提案手法の方法を用いて推定している手法（adaptive）と提案手法（Proposed）を比較した．naive は，学習を SOINN で行い，バンド幅はゆう度交差検証法により決定している．adaptive では，SOINN で学習したネットワーク構造に対して式 (4) を用いて推定密度関数を算出する．各手法において SOINN のハイパーパラメータは $\lambda = 500$，$age_{max} = 100$，$\rho = 0.1$，$k = 2d$ とした．

図 10 が実験の結果である．naive と adaptive を比較すると，adaptive の方が性能が向上しており，特にノイズの割合が増加すると顕著になる．SOINN はノードの密度だけではなく，エッジやノードの勝者回数などネットワークとしてサンプルの分布を保持し

表 4　既存手法との比較
Table 4　The comparison between our proposed method and others.

	CVML [16]	RKDE [9]	oKDE [23]	**Proposed**
Non-parametric	○	○	○	○
Fast online learning	×	×	△	○
Robustness	×	△	×	○
Accuracy	○	○	△	○

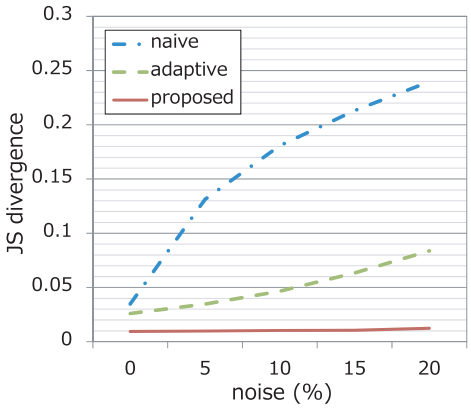


図 10　ロバスト性に関して，SOINN と KDE をナイーブに組み合わせた場合との比較実験の結果
Fig. 10　The experimental result about robustness comparing proposed method and a naive method integrating SOINN into KDE.

ており，提案手法の推定密度関数式 (4) がこのネットワークが保持している密度の情報を引き出せていることが分かる．また，naive は通常の KDE と同様に各ノードに配置しているカーネルは全て同じであるのに対し，adaptive では式 (4)，式 (5) を用い，各ノードに配置するカーネルの形状及び大きさをネットワークから適応的に決定しているためロバスト性が向上し，ノイズの割合が増加すると性能の差が顕著になったと考えられる．naive，adaptive と提案手法を比較すると，提案手法の方が大きく性能が向上している．従来の SOINN には，ネットワークにひずみがあり，そのため分布の推定精度が低下してしまっている．これは分布を適切に表現しないエッジができてしまうことが原因であると考えられ，エッジ作成時のしきい値決定を改良している提案手法の学習アルゴリズムがサンプル集合の密度関数を表すネットワーク構造を SOINN よりも学習できていることが分かる．

5.2　高次元への対応

一般にノンパラメトリック密度推定法は高次元データに対しては推定精度が良くない [35], [36]．提案手法もノンパラメトリック密度推定法であるためデータの次元数が大きくなるに連れて推定精度が低下してしまう．

これに対処するために，次元削減や特徴抽出の手法と統合することが考えられる．オンラインに主成分分析及びカーネル主成分分析を行う手法 [37]～[39] が考案されているため，これらと組み合わせることで，提案手法のオンライン性を失うことなく高次元データを低次元に写像し密度推定することができる．また，提案手法はグラフ構造を基盤としており，グラフ構造を用いた多様体学習手法 [40]～[42] と統合できるのではないかと考えている．多様体上の密度関数を考える [43], [44] ことで，高次元空間を直接扱うことを避けることができる．現在，提案手法と多様体学習を統合した手法を考案中である．

6.　む　す　び

本研究ではビッグデータに対処するために，ロバスト性があり，高速にオンライン学習可能なノンパラメトリック確率密度推定法を考案した．提案手法の基盤となっている SOINN のネットワーク構造に対して新たに統計的な解釈を与えることで密度推定を可能にした．ノンパラメトリック密度推定の伝統的な手法である KDE は精度を高めるために大量のサンプル数が必要であり，サンプル数の増加に伴って計算量が大きくなってしまうが，提案手法はサンプルのプロトタイプを密度の分布に従ってネットワーク構造にすることで計算量を大きく抑えている．評価実験において，学習時間，ロバスト性に関して，既存のノンパラメトリック密度推定法より高い性能を示し，推定精度に関しても，同等またはそれ以上の性能を示すことができた．

今後は **5.2** で述べた高次元への対応のための提案手法の拡張と，提案手法の実環境のビックデータ解析への応用に取り組んでいきたいと考えている．ノイズを含む実環境からデータの分布を推定できるようになれば，確率的な予測や推論がより実際的なアプリケーションとして構築でき，その応用範囲は広いと考えられる．

謝辞　本研究は，科学技術振興機構の戦略的創造研究推進事業からの支援を頂きました．記して感謝致します．また，査読者の方から非常に有益なご意見を頂きました．併せて感謝致します．

文　　献

[1] V. Mayer-Schönberger and K. Cukier, Big Data: A Revolution that Will Transform how We Live, Work, and Think, Houghton Mifflin Harcourt, 2013.

[2] W. Fan and A. Bifet, "Mining big data: Current status, and forecast to the future," SIGKDD Explor. Newsl., vol.14, pp.1–5, 2013.

[3] J.K. Laurila, D. Gatica-Perez, I. Aad, J. Blom, O. Bornet, T. Do, O. Dousse, J. Eberle, and M. Miettinen, "The mobile data challenge: Big data for mobile computing research," Mobile Data Challenge by Nokia Workshop, 2012.

[4] D. Howe, M. Costanzo, P. Fey, T. Gojobori, L. Hannick, W. Hide, D.P. Hill, R. Kania, M. Schaeffer, S. St Pierre, S. Twigger, O. White, and S.Y. Rhee, "Big data: The future of biocuration," Nature, vol.455, pp.47–50, 2008.

[5] D. Laney, "3D data management: Controlling data volume, velocity, and variety," Technical report, META Group, 2001.

[6] P.J. Huber, Robust statistics, Wiley, 1981.

[7] M. Hubert, P.J. Rousseeuw, and K. Vanden Branden, "ROBPCA: A new approach to robust principal component analysis," Technometrics, vol.47, no.1, pp.64–79, 2005.

[8] P. Xanthopoulos, P.M. Pardalos, and T.B. Trafalis, Robust data mining, SpringerBriefs in Optimization, 2013.

[9] J. Kim and C.D. Scott, "Robust kernel density estimation," J. Machine Learning Research, vol.13, pp.2529–2565, 2012.

[10] E. Parzen, "On estimation of a probability density function and mode," The Annals of Mathematical Statistics, vol.33, pp.1065–1076, 1962.

[11] B.W. Silverman, Density Estimation, Chapman and Hall, 1986.

[12] P. Hall, S.J. Sheather, M.C. Jones, and J.S. Marron, "On optimal data-based bandwidth selection in kernel density estimation," Biometrika, vol.78, no.2, pp.263–269, 1991.

[13] M.C. Jones, J.S. Marron, and S.J. Sheather, "A brief survey of bandwidth selection for density estimation," J. American Statistical Association, vol.91, no.433, pp.401–407, 1996.

[14] A. Bowman, "An alternative method of cross-validation for the smoothing of density estimates," Biometrika, vol.71, pp.353–360, 1984.

[15] D. Chaudhuri, B. Chaudhuri, and C. Murthy, "A data driven procedure for density estimation with some applications," Pattern Recognit., vol.29, no.10, pp.1719–1736, 1996.

[16] J.M. Leiva and A. Artés, "Algorithms for Gaussian bandwidth selection in kernel density estimators," Neural Information Processing Systems, 2008.

[17] E. López-Rubio and J.M. Ortiz-de-Lazcano-Lobato, "Soft clustering for nonparametric probability density function estimation," Pattern Recognit. Lett., vol.29, no.16, pp.2085–2091, 2008.

[18] M. Girolami and C. He, "Probability density estimation from optimally condensed data samples," IEEE Trans. Pattern Anal. Mach. Intell., vol.25, no.10, pp.1253–1264, 2003.

[19] Z. Deng, F. Chung, and S. Wang, "FRSDE: Fast reduced set density estimator using minimal enclosing ball approximation," Pattern Recognit., vol.41, no.4, pp.1363–1372, 2008.

[20] L. Bottou, "Large-scale machine learning with stochastic gradient descent," Proc. 19th International Conference on Computational Statistics (COMPSTAT '2010), eds. Y. Lechevallier and G. Saporta, pp.177–187, Springer, Paris, France, Aug. 2010. http://leon.bottou.org/papers/bottou-2010

[21] A. Declercq and J.H. Piater, "Online learning of Gaussian mixture models — A two level approach," VISAPP, pp.605–611, 2008.

[22] K. Tabata, M. Sato, and M. Kudo, "Data compression by volume prototypes for streaming data," Pattern Recognit., vol.43, no.9, pp.3162–3176, 2010.

[23] M. Kristan, A. Leonardis, and D. Skočaj, "Multivariate online kernel density estimation with Gaussian kernels," Pattern Recognit., vol.44, no.10-11, pp.2630–2642, 2011.

[24] Y. Li, D. de Ridder, R. Duin, and M. Reinders, "Integration of prior knowledge of measurement noise in kernel density classification," Pattern Recognit., vol.41, no.1, pp.320–330, 2008.

[25] F. Shen and O. Hasegawa, "An incremental network for on-line unsupervised classification and topology learning," Neural Netw., vol.19, no.1, pp.90–106, 2006.

[26] A. Sudo, A. Sato, and O. Hasegawa, "Associative memory for online learning in noisy environments using self-organizing incremental neural network," IEEE Trans. Neural Netw., vol.20, no.6, pp.964–972, June 2009.

[27] N. Makibuchi, F. Shen, and O. Hasegawa, "Online knowledge acquisition and general problem solving in a real world by humanoid robots," ICANN, vol.3, pp.551–556, 2010.

[28] S. Okada and T. Nishida, "Online incremental clustering with distance metric learning for high dimensional data," 2011 International Joint Conference on Neural Networks (IJCNN), pp.2047–2054, July 2011.

[29] P. Kankuekul, A. Kawewong, S. Tangruamsub, and O. Hasegawa, "Online incremental attribute-based

zero-shot learning," 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.3657–3664, June 2012.
[30] A. Kawewong, R. Pimup, and O. Hasegawa, "Incremental learning framework for indoor scene recognition," AAAI, pp.496–502, 2013.
[31] F. Shen, Q. Ouyang, W. Kasai, and O. Hasegawa, "A general associative memory based on self-organizing incremental neural network," Neurocomputing, vol.104, pp.57–71, 2013.
[32] B. Fritzke, "A growing neural gas network learns topologies," Advances in Neural Information Processing Systems 7, pp.625–632, MIT Press, 1995.
[33] S. Furao, T. Ogura, and O. Hasegawa, "An enhanced self-organizing incremental neural network for online unsupervised learning," Neural Netw., vol.20, no.8, pp.893–903, 2007.
[34] F. Shen and O. Hasegawa, "A fast nearest neighbor classifier based on self-organizing incremental neural network," Neural Netw., vol.21, no.10, pp.1537–1547, 2008.
[35] V.N. Vapnik, Statistical learning theory, Wiley, 1998.
[36] W. Härdle, Nonparametric and Semiparametric Models, Springer Series in Statistics, Springer-Verlag GmbH, 2004.
[37] T. Oyama, K.S. Githinji, S. Tsuge, Y. Mitsukura, and M. Fukumi, "Fast incremental algorithm of simple principal component analysis," IEEJ Trans. Electronics, Information and Systems, vol.129, no.1, pp.112–117, Jan. 2009.
[38] T.-J. Chin and D. Suter, "Incremental kernel principal component analysis," IEEE Trans. Image Process., vol.16, no.6, pp.1662–1674, 2007.
[39] Y. Takeuchi, S. Ozawa, and S. Abe, "An efficient incremental kernel principal component analysis for online feature selection," IJCNN, pp.2346–2351, 2007.
[40] J.B. Tenenbaum, V. de Silva, and J.C. Langford, "A global geometric framework for nonlinear dimensionality reduction," Science, vol.290, no.5500, p.2319, 2000.
[41] Z. Zhang and H. Zha, "Principal manifolds and nonlinear dimension reduction via local tangent space alignment," SIAM J. Scientific Computing, vol.26, pp.313–338, 2002.
[42] K. Yu and T. Zhang, "Improved local coordinate coding using local tangents," ICML, eds. J. Fürnkranz and T. Joachims, pp.1215–1222, Omnipress, 2010.
[43] B. Pelletier, "Kernel density estimation on Riemannian manifolds," Statistics & Probability Letters, vol.73, no.3, pp.297–304, 2005.
[44] G. Henry and D. Rodriguez, "Kernel density estimation on Riemannian manifolds: Asymptotic results," J. Mathematical Imaging and Vision, vol.34, no.3, pp.235–239, 2009.

（平成 25 年 11 月 8 日受付，26 年 3 月 7 日再受付）

中村　圭宏　（学生員）

2011 名工大・工・情報工卒．2013 東京工業大学大学院総合理工学研究科知能システム科学専攻修士課程了．現在，同大学院博士課程在学中．主に機械学習の研究に従事．

長谷川　修　（正員）

1993 東京大学大学院博士課程了．博士(工学)．同年電子技術総合研究所入所．1999 年 6 月より 1 年間カーネギーメロン大学ロボティクス研究所滞在研究員．2001 産業技術総合研究所主任研究員．2002 年 5 月東京工業大学大学院理工学研究科付属像情報工学研究施設准教授．2002 から 3 年間科技団さきがけ研究 21 研究員．パターン認識，ニューラルネット，実世界知能システムなどの研究に従事．人工知能学会，日本認知科学会，日本顔学会，IEEE CS 等各会員．