Nonparametric Density Estimation Based on Self-Organizing Incremental Neural Network for Large Noisy Data

Yoshihiro Nakamura and Osamu Hasegawa, *Member, IEEE*

***Abstract*— With the ongoing development and expansion of communication networks and sensors, massive amounts of data are continuously generated in real time from real environments. Beforehand, prediction of a distribution underlying such data is difficult; furthermore, the data include substantial amounts of noise. These factors make it difficult to estimate probability densities. To handle these issues and massive amounts of data, we propose a nonparametric density estimator that rapidly learns data online and has high robustness. Our approach is an extension of both kernel density estimation (KDE) and a self-organizing incremental neural network (SOINN); therefore, we call our approach KDESOINN. An SOINN provides a clustering method that learns about the given data as networks of prototype of data; more specifically, an SOINN can learn the distribution underlying the given data. Using this information, KDESOINN estimates the probability density function. The results of our experiments show that KDESOINN outperforms or achieves performance comparable to the current state-of-the-art approaches in terms of robustness, learning time, and accuracy.**

***Index Terms*— Big data, kernel density estimation (KDE), online learning, robustness, self-organizing incremental neural network (SOINN).**

I. Introduction

WITH THE ongoing expansion and development of communication networks and sensors, massive amounts of data are continuously generated. These data sets have collectively been called big data, and there are many expectations to use such data sets on the basis of their analysis [1]–[4].

Laney [5] considered the following three concepts as the characteristics of big data.

1) *Volume (The Volume of Data):* Huge amounts of data have been generated, and the data sets continue to grow.
2) *Velocity (Data Generation Speed):* Increasing numbers of sensors are continuously collecting and generating data in real time, meaning that the amount of generated data per unit time (i.e., the velocity) is very high.
3) *Variety (The Variety of Data):* Many different types of data are generated, such as sensor data from real environments, life-log data, point-of-sale data, and textual and image data on the Web.

The above characteristics are known as the 3Vs, and should be considered when dealing with big data.

Probability density estimation is a critical task in the fields of machine learning and data analysis; however, it is difficult to do such estimation through the conventional methods for big data, because density estimators for big data must have the characteristics described below.

First, density estimators for big data must be online learning methods. The most importance of big data is the velocity of data generation. Massive amounts of data are generated very quickly, and the total volume of the data becomes gigantic. Online learning methods do not need the entire data for learning. They can learn samples one by one and update their status sequentially. Therefore, they can handle the situation where data are continuously generated.

Second, density estimators for big data must be nonparametric. The purpose of analyzing big data is primarily for data mining and knowledge-discovery-based studies and applications. In many cases, beforehand, determination of the distribution underlying the observed data is extremely difficult, for example, when analyzing a new type of data that no one has analyzed and when analyzing data that include complex dependences between variables. As a result, parametric density estimators that assume known parametric distributions generating observed samples are not applicable. If the assumptions are incorrect, then the performance of the estimators decreases dramatically. In general, parametric density estimators are usable only for data sets that experts have studied and modeled. It is difficult for parametric density estimators to estimate unknown distributions or distributions that cannot be modeled by known parametric distributions. Conversely, nonparametric density estimators that do not require initial assumptions regarding distributions and estimate distributions only from the given data are not problematic.

Third, density estimators for big data must be very robust. Data are generated from real environments and often include substantial amounts of noise. Such noise, when included in the training data, becomes one of the causes of overfitting and decreases the performance. Methods have been developed to eliminate noise by assuming extent of the distribution of noise, but if this assumed distribution does not correspond

Manuscript received October 17, 2014; revised September 27, 2015; accepted September 28, 2015. Date of publication January 21, 2016; date of current version December 22, 2016. This work was supported by Japan Science and Technology Agency, Core Research for Evolutional Science and Technology (CREST).

Y. Nakamura is with the Department of Computational Intelligence and System Science, Tokyo Institute of Technology, Yokohama 152-8550, Japan (e-mail: nakamura.y.ba@m.titech.ca.jp).

O. Hasegawa is with the Imaging Science and Engineering Laboratory, Tokyo Institute of Technology, Yokohama 152-8550, Japan (e-mail: hasegawa.o.aa@m.titech.ac.jp).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2015.2489225

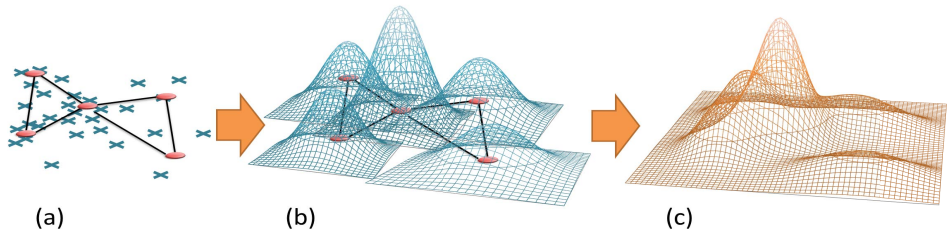

Fig. 1. Overview of the KDESOINN process. (a) Learn samples as networks whose nodes are the prototypes of the samples in an online manner. (b) Determine the shape and size of each kernel at each node on the basis of the local structure of the network around the node. (c) Estimate the probability density function of the samples as a summation of the kernels.

to the true distribution of noise, performance and accuracy decrease severely. Noise may be eliminated manually, but it is difficult to do so with massive data sets generated in real time. Thus, highly robust methods that can successfully learn about data that include noise are required.

Robustness has been studied in many fields, and its definitions in those fields differ [6]–[9]. In this paper, according to [9], we define robustness as a function that provides almost the same results as learning data without noise when learning with noisy data. There are two types of noise.

1) Noises generated from the environment, which are not relevant to the objective distribution, such as outliers, abnormal samples, and samples generated from diffuse or not abundant distributions.
2) Noises, such as variance and fluctuation, that we must recognize as probability distributions.

In density estimation, robustness is essentially a function to eliminate the noise defined in (1).

To estimate probability density functions for big data, we propose a new method that fulfills the three features described above.

Compared with our previous work [10], we improved our algorithm by achieving better performance and added more detailed explanation and empirical studies.

II. Related Works

A typical approach to nonparametric density estimation is kernel density estimation (KDE) [11]. Given training samples $\{\boldsymbol{x}_i | \boldsymbol{x}_i \in \mathbb{R}^d, i = 1, 2, \ldots, N\}$, KDE is

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

where K is a kernel function and

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

is the Gaussian kernel that is often used as K, where $\boldsymbol{H}$ is a parameter called the bandwidth matrix and $|\cdot|$ is a determinant.

This parameter materially affects the performance of the estimator; therefore, there are numerous works on optimizing it [12]–[17]. Because these methods are batch methods, they cannot handle big data. With an increase in the training data, the time and space complexities of KDE increase, but to achieve high performance, KDE needs numerous samples [12]. There is a tradeoff between complexity and performance. Hence, some approaches, such as placing kernels at adequate locations or introducing an online approach, are necessary. Methods that set the number of kernels have been proposed in [18]–[20]. Typically, to introduce an online approach, the problem is considered to be an optimization problem of a loss or likelihood function, and a gradient descent method is used to solve it [21]; however, there is no such a method of KDE. The KDE model depends on training samples; hence, the model changes with an increase in the samples. Thus, it is difficult to optimize it online. There are some online methods of KDE using other approaches, including online clustering [22]–[24]. However, their training speed is not fast enough and does not present high robustness to noise.

From (1), we note that KDE places kernels on all the training samples and estimates the density function as the summation of these kernels. If the training samples include noise, the KDE locates the kernels on the noise, which becomes one of the causes of overfitting and leads to a decrease in performance. Kim and Scott [9] considered KDE as a kernel method and optimized it using robust estimation, thus realizing robust KDE; however, their method is also a batch approach, and, therefore, cannot handle huge amounts of data.

III. Proposed Method

Our proposed method, KDESOINN, is an extension of KDE and self-organizing incremental neural networks (SOINNs) [25]. There are many applications of SOINNs [26]–[31]. In all of them, an SOINN is used as a clustering method; however, we determined that the network structure of SOINNs has information regarding the distribution of underlying observed samples. Using this information, KDESOINN estimates the probability density function.

Fig. 1 shows an overview of the KDESOINN process. Our proposed method first learns from observed samples by interpreting them as networks using a modified SOINN algorithm. In these networks, the nodes are the prototypes of the samples. Next, our proposed method determines the shape and size of each kernel at each node on the basis of the local structure of the network around the node and estimates the probability density function of samples as a summation of these kernels.

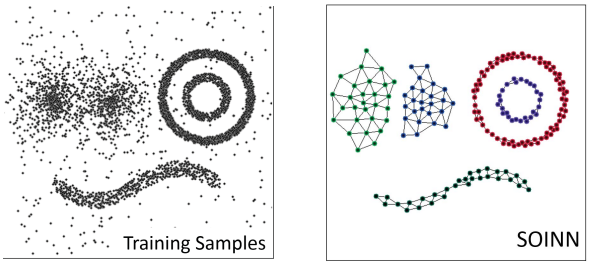

Fig. 2. Example of how a SOINN learns. (a) Set of training samples, including noise. (b) SOINN's learned results.

A. SOINN

1) Overview of SOINN: SOINN is a prototype-based unsupervised learning method that is an extension of growing neural gas [32]. There are several versions of the SOINN algorithms, including those in [25], [33], and [34]. In this paper, we adopt the adjusted SOINN [34] as an SOINN; many recent applied studies of SOINNs used the adjusted SOINN [26]–[30], because it is easier to use than the original SOINN. The original SOINN [25] is a two-layer framework with many parameters. Conversely, the adjusted SOINN is a one-layer framework with fewer parameters, which is why we opted to use the adjusted SOINN in this paper.

Fig. 2 shows an example of how the SOINN learns. Note that the SOINN can perform online learning. In general, it learns samples from noisy complex nonstationary distributions as network structures, in which the number of nodes is much less than the number of samples. Furthermore, it does not request the initialization of the number of nodes or the location of nodes; therefore, it can be applied to incremental learning.

More specifically, the SOINN learns the training samples as network structures through competitive learning. The nodes on the network sequentially handle the samples that are sequentially inputted. Accordingly, the nodes or edges are inserted or deleted, or the locations of nodes are updated. Thus, the network structure is updated to approximate the distribution that underlies the samples.

2) Statistical Analysis of SOINN Networks: Algorithm 1 shows the SOINN algorithm, and Table I summarizes the definitions of variables, parameters, and symbols used in the algorithm. The SOINN creates a new node if a new sample is not in both the threshold regions of the two nearest neighbor nodes of the new sample, i.e., the first and second winners. Otherwise, the SOINN does not create a new node, instead creates a new edge between the first and second winner nodes; in such cases, the new sample causes the number of wins of the first winner to be incremented by one. Next, the positional vectors of the first winner and its adjacency nodes are updated to move toward the new sample.

The update of the first winner, line 19 of Algorithm 1, corresponds to an online update of the mean. Given that the mean of $\{x_1, x_2, \ldots, x_n\}$ is

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

Algorithm 1 SOINN

1: **if** initialization **then**
2: randomly select c_1, c_2 from learning data set
3: $\mathcal{N} \leftarrow \{c_1, c_2\}$
4: $\mathcal{E} \leftarrow \phi$
5: **end if**
6: **while** There is $\boldsymbol{\xi} \in \mathbb{R}^n$. **do**
7: $s_1 \leftarrow \arg\min_{c\in\mathcal{N}} ||\boldsymbol{\xi} - \boldsymbol{w}_c||$
8: $s_2 \leftarrow \arg\min_{c\in\mathcal{N}\setminus\{s_1\}} ||\boldsymbol{\xi} - \boldsymbol{w}_c||$
9: Calculate thresholds $\Theta_{s_1}, \Theta_{s_2}$ with Eq. (6)
10: **if** $||\boldsymbol{\xi} - \boldsymbol{w}_{s_1}|| > \Theta_{s_1}$ **or** $||\boldsymbol{\xi} - \boldsymbol{w}_{s_2}|| > \Theta_{s_2}$ **then**
11: $\mathcal{N} \leftarrow \mathcal{N} \cup \{\boldsymbol{\xi}\}$
12: **else**
13: **if** $(s_1, s_2) \notin \mathcal{E}$ **then**
14: $\mathcal{E} \leftarrow \mathcal{E} \cup \{(s_1, s_2)\}$
15: **end if**
16: $a_{(s_1,s_2)} \leftarrow 0$
17: $a_{(s_1,i)} \leftarrow a_{(s_1,i)} + 1\ (\forall i \in \mathcal{P}_{s_1})$
18: $t_{s_1} \leftarrow t_{s_1} + 1$
19: $\boldsymbol{w}_{s_1} \leftarrow \boldsymbol{w}_{s_1} + \psi_1(t_{s_1})(\boldsymbol{\xi} - \boldsymbol{w}_{s_1})$
20: $\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i + \psi_2(t_i)(\boldsymbol{\xi} - \boldsymbol{w}_i)\ (\forall i \in \mathcal{P}_{s_1})$
21: delete edges $\mathcal{E}_{old} = \{e \mid e \in \mathcal{E},\ a_e > age_{max}\}$
22: delete nodes $\{i \mid \exists j\ (i, j) \in \mathcal{E}_{old}, |\mathcal{P}_i| = 0\}$
23: **end if**
24: **if** the number of input samples is the multiple of λ **then**
25: delete nodes $\{i \mid |\mathcal{P}_i| \leq 1\}$
26: **end if**
27: **end while**

Then, $\boldsymbol{w}^{(n+1)}$, the mean of $\{x_1, x_2, \ldots, x_{n+1}\}$, is

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

which corresponds to line 19 of Algorithm 1. Each node acts on behalf of the samples that are counted for the node in competitive learning. More specifically, the node is located on the mean of these samples. Furthermore, line 20 of Algorithm 1 can be considered as smoothing. Note that the coefficient of $\psi_2(t)$ was determined experimentally.

According to lines 17 and 21 of Algorithm 1, an edge is cut off if the frequency of samples between the endpoints of the edge is relatively lower than that of edges connected to the endpoints. For an edge to exist between nodes means that the frequency of samples between the nodes is relatively high and the samples of which the nodes are representatives scatter in the direction of the edge.

We note here that each node is a prototype representative of samples around the node and that the network around the node expresses the breadth and direction of the spread of samples.

B. Proposed Density Estimator

KDESOINN estimates the density function using information expressed by SOINN networks that we presented in Section III-A2.

TABLE I
DEFINITIONS OF VARIABLES, PARAMETERS, AND SYMBOLS

$\boldsymbol{\xi}$	Input samples. $\boldsymbol{\xi} \in \mathbb{R}^d$.
$\mathcal{N}$	The node set in SOINN and in the proposed method.
$\mathcal{E}$	The edge set in SOINN and in the proposed method. $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$.
$\mathcal{P}_i$	The set of nodes connected to node i.
$\boldsymbol{w}_i$	The positional vector of node i. $\boldsymbol{w}_i \in \mathbb{R}^d$.
t_i	The number of wins of node i in competitive learning.
a_e	The age of edge e
Θ_i	The threshold of node i to create a node and edge.
$\psi_k(t)$	Functions to decide a coefficient of local optimization. $\psi_1(t) = t^{-1}\psi_2(t) = (100t)^{-1}$.
λ	The parameter for deletion of nodes.
age_{max}	The parameter for deletion of edges.
ρ	The parameter for thresholds.
$\boldsymbol{I}$	Identity matrix.
$\lvert\mathcal{S}\rvert$	The number of elements of the set $\mathcal{S}$.
$E(\mathcal{G})$	The set of edges in graph $\mathcal{G}$

The estimated density function obtained through KDESOINN $\hat{p}(\boldsymbol{x})$ is defined as

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

where $T_{\mathcal{N}} = \sum_{n \in \mathcal{N}} t_n$ and K is the Gaussian kernel function from (2). KDESOINN locates the kernels on nodes and estimates the density function as the summation of such kernels. The kernel located on node n is weighted by winning times t_n, the number of samples for which node n is the representative.

Covariance matrix $\boldsymbol{C}_n$ determines the shape of the Gaussian kernel, and is adaptively determined by the network around node n. We call this matrix the local network covariance matrix and define it as

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

where $T_{\mathcal{P}_n} = \sum_{i \in \mathcal{P}_n} t_i$. This covariance matrix is an extension of the original covariance matrix. The differences from the original covariance matrix include the following: 1) the center of this covariance matrix is not the mean of samples, but rather the positional vector of node n; 2) it is calculated only from the adjacent nodes of n; and 3) that the positional vectors of adjacent nodes are weighted by their winning times t_p. The local network covariance matrix extracts distribution information expressed by the networks; i.e., how broad samples around node n scatter toward edges connecting n. The Gaussian kernel $K_{\boldsymbol{C}_n}$ approximates the density of samples around n. Note that depending on the shape of the network around n, $\boldsymbol{C}_n$ may not be regular; in such cases, it is necessary to modify it by adding an identity matrix multiplied by some small value.

C. Learning Algorithm

Algorithm 2 shows the learning algorithm of KDESOINN; the algorithm is a modified SOINN that determines the distribution of underlying training samples.

Algorithm 2 Learning Algorithm

if initialization **then**
  randomly select c_1, c_2 from learning data set
  $\mathcal{N} \leftarrow \{c_1, c_2\}$
  $\mathcal{E} \leftarrow \phi$
end if
while There is $\boldsymbol{\xi} \in \mathbb{R}^n$. **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**
    $\mathcal{N} \leftarrow \mathcal{N} \cup \{\boldsymbol{\xi}\}$
  else
    if $(s_1, s_2) \notin \mathcal{E}$ **then**
      $\mathcal{E} \leftarrow \mathcal{E} \cup \{(s_1, s_2)\}$
    end if
    $a_{(s_1,s_2)} \leftarrow 0$
    $a_{(s_1,i)} \leftarrow a_{(s_1,i)} + 1 \forall i \in \mathcal{P}_{s_1}$
    $t_{s_1} \leftarrow t_{s_1} + 1$
    $\boldsymbol{w}_{s_1} \leftarrow \boldsymbol{w}_{s_1} + \psi_1(t_{s_1})(\boldsymbol{\xi} - \boldsymbol{w}_{s_1})$
    delete edges $\mathcal{E}_{old} = \{e \mid e \in \mathcal{E},\ a_e > age_{max}\}$
    delete nodes $\{i \mid \exists j\ (i, j) \in \mathcal{E}_{old}, |\mathcal{P}_i| = 0\}$
  end if
  if the number of input samples is the multiple of λ **then**
    delete nodes $\{i \mid |\mathcal{P}_i| = 0\}$
    create a k-NN graph $\mathcal{G}$ whose set of nodes is $\mathcal{N}$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(i, j) \mid (i, j) \in E(\mathcal{G}), (j, i) \in E(\mathcal{G})\}$
  end if
end while
create a k-NN graph $\mathcal{G}$ whose set of nodes is $\mathcal{N}$
$\mathcal{E} \leftarrow \mathcal{E} \cup \{(i, j) \mid (i, j) \in E(\mathcal{G}), (j, i) \in E(\mathcal{G})\}$

1) Determining Threshold Regions: One of the extensions we applied to our learning algorithm is the determination of threshold regions, which is the most important part of the SOINN algorithm, because this part determines to add new edges and nodes.

In SOINN's algorithm, the threshold region of node n is a hypersphere with radius Θ_i defined as

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

The threshold Θ_i is defined as the Euclidean distance from the boundary to the center of the Voronoi region V_i of node i [34]. In competitive learning, node i represents the sample prototype inside V_i. The threshold region includes V_i; however, V_i does not necessarily form a hypersphere.

Fig. 3(a) shows the threshold regions of SOINN; they spread not only in the direction in which networks spread, but also in the direction in which networks do not spread. The network around a node spreads widely in certain directions, meaning that the samples around the node scatter along the network and that few samples are in the direction without edges. Thus, the threshold regions cause some long edges that do not express the breadth of samples, which has harmful effects for creating

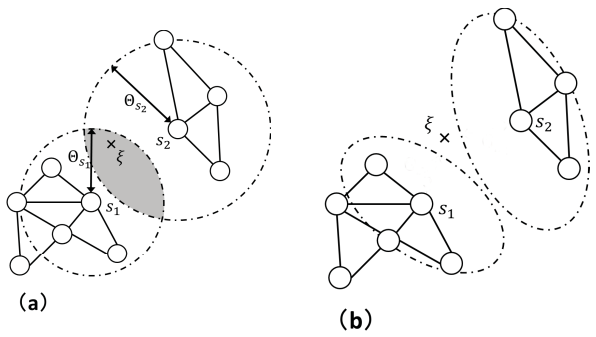

Fig. 3. (a) SOINN's threshold regions. (b) KDESOINN's threshold regions.

the networks i.e., the networks become unable to express distributions.

To eliminate network distortion and create more adequate networks that more accurately express the distribution of the observed samples, KDESOINN decides the threshold regions adaptively from the samples. Fig. 3(b) shows the threshold regions of KDESOINN. In KDESOINN, the threshold region of a node is a hypersphere, converted into a shape that conforms to the local structure of the network around the node. Thus, the threshold regions do not spread in the direction without edges. Samples that became edges unrelated to the density of samples in the SOINN are outside of the threshold regions and, therefore, become nodes. As to whether they belong to specific networks, this is decided by the samples inputted later.

To convert threshold regions into shapes that conform to networks, KDESOINN defines threshold regions using the Mahalanobis distance with local network covariance matrices, as expressed by (5). Given the new sample $\boldsymbol{\xi}$, the threshold region of node i is

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

where

$$\boldsymbol{M}_i = \boldsymbol{C}_i + \rho \gamma_i \boldsymbol{I}$$
$$\gamma_i = \begin{cases} \min\limits_{p \in \mathcal{P}_i} ||\boldsymbol{w}_p - \boldsymbol{w}_i|| & (\mathcal{P}_i \neq \phi) \\ \min\limits_{p \in \mathcal{N} \setminus \{i\}} ||\boldsymbol{w}_p - \boldsymbol{w}_i|| & (\text{otherwise}). \end{cases} \tag{7}$$

In [10], the threshold region was defined as heuristics; a node's threshold region becomes strangely distorted in shape when the node's kernel is thin, decreasing the algorithm's accuracy and robustness. This tends to happen when the networks are less developed or when the data are placed in a high-dimensional space. In this paper, this conflict was resolved by fitting the node's threshold region to the kernel using the Mahalanobis distance.

In addition, an identity matrix is added in (7) to achieve smoothing. If (7) does not have the smoothing, the threshold region of a node that has few edges is too thin to create a new edge. γ_i was determined experimentally by analogy with (6). When $\mathcal{P}_i = \phi$, $\boldsymbol{C}_i = \boldsymbol{0}$ and the threshold region is a hypersphere. When $\mathcal{P}_i \neq \phi$, $\boldsymbol{C}_i \neq \boldsymbol{0}$ and if Θ_i is used as γ_i, the threshold region becomes too large and hinders the proper formation of networks. Therefore, (6) was modified to (7).

2) Adjusting Networks: At lines 24 and 25, and 28 and 29, KDESOINN adjusts its networks. Because the SOINN does not create an edge unless a new sample is in a threshold region, the SOINN cannot create networks when the dimensions of data are very high or when only a few samples are available. To resolve this problem, this part of the algorithm adds edges on the basis of a k-nearest neighbor (k-NN) graph. More specifically, it adds edges between the pairs of nodes that have duplex edges in a k-NN graph on nodes. At the location where there are many nodes, the duplex edges of a k-NN graph are likely created, meaning that edges created this way express the density of samples.

The previously applied method [10] did not include this adjustment. It has fewer edges and nodes than KDESOINN placed in a higher dimensional space. Consequently, its accuracy was smaller than that of KDESOINN.

3) Other Extensions: There are additional extensions. We deleted line 20 of Algorithm 1, which was introduced experimentally for smoothing. Considering what we presented in Section III-A2, we feel that it is unnecessary for estimating density. Furthermore, on line 23, we use $|\mathcal{P}_i| = 0$ instead of $|\mathcal{P}_i| \leq 1$.

In lines 7 and 8 of Algorithm 2, for selecting the winners of the competitive learning process, KDESOINN uses the Euclidean distance similar to the SOINNN instead of the Mahalanobis distance. Evaluating nodes with a uniform measure, such as the Euclidean distance, result in an adequate execution of competitive learning. In addition, it is necessary to calculate the distances between the new sample and all the nodes here, and each node has each covariance matrix $\boldsymbol{M}_i$ that is used to calculate the Mahalanobis distance between the new sample and the node. Using the Mahalanobis distance here would require too large of a computational cost.

IV. Experimental Study

To compare our proposed method with the current state-of-the-art approaches, we conducted experiments focused on robustness, learning time, and accuracy. All the experiments were performed on a PC with a quad-core 3.4-GHz CPU and 16-GB RAM; the implementation language and environment was MATLAB.

A. Comparative Methods

We compared our proposed method with the following other methods: 1) an online KDE (oKDE) approach that used clustering to adaptively decide the number of kernels and their shapes [24]; 2) a batch KDE approach that optimized a bandwidth matrix with cross-validation of maximum likelihood (CVML) [17]; 3) a batch KDE approach focused on robustness (RKDE) [9]; and 4) our previous method [10]. Note that RKDE required a bandwidth matrix optimized by another method; thus, we used a bandwidth matrix optimized by CVML.

B. Robustness

To evaluate the robustness of KDESOINN, we compared it with the other methods while learning from noisy data.

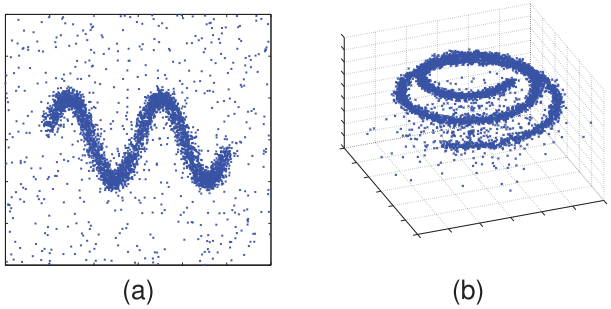

Fig. 4. Training samples generated from (a) (8) and (b) (9) with distributions including 20% of noise.

The training data were

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

where f_0 is a true distribution, f_1 is a contaminating distribution, and α is the proportion of contamination. Each method learned $\boldsymbol{X}$ and estimated f_0. We evaluated how estimate accuracies changed as α were increased. We used the following two distributions as a true distribution f_0:

$$\begin{aligned} f_0(\boldsymbol{x}) &= N(\boldsymbol{x}; \boldsymbol{t}, 0.2 \cdot \boldsymbol{I}) \\ \boldsymbol{t} &= [a, \sin(3a)]^T, \quad a \in [-2, 2] \end{aligned} \tag{8}$$

$$\begin{aligned} f_0(\boldsymbol{x}) &= N(\boldsymbol{x}; \boldsymbol{t}, 0.25 \cdot \boldsymbol{I}) \\ \boldsymbol{t} &= [r\cos(\theta), r\cos(\theta), \theta]^T \\ r &= 10 - 0.5\theta, \quad \theta \in [-7, 7]. \end{aligned} \tag{9}$$

We used a uniform distribution with the range of dimension as $[-4, 4]$ and the Gaussian distribution $N(\boldsymbol{x}; [0, 0, -15]^T, 30 \cdot \boldsymbol{I})$ as a contaminating distribution f_1. The number of training samples was 5000. Fig. 4 shows the noisy training data sets obtained from (8) and (9).

We used the Jensen–Shannon (JS) divergence as our evaluation measure. The JS divergence is considered as a distance measure between distributions, with a lower value between an estimated distribution and a true distribution, indicating that the estimator has high accuracy. The JS divergence is more suitable to measure robustness than the most popular Kullback–Leibler (KL) divergence, because the JS divergence is symmetric and enables evaluation outside the area of true distribution without diverging infinitely.

The JS divergence between an estimated distribution $\hat{f}$ and f_0 is

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

where $m = (1/2)(\hat{f} + f_0)$ and D_{KL} is the KL divergence

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

where $\{x_i\}_{i=1}^n$ is the set of samples from p. Then

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

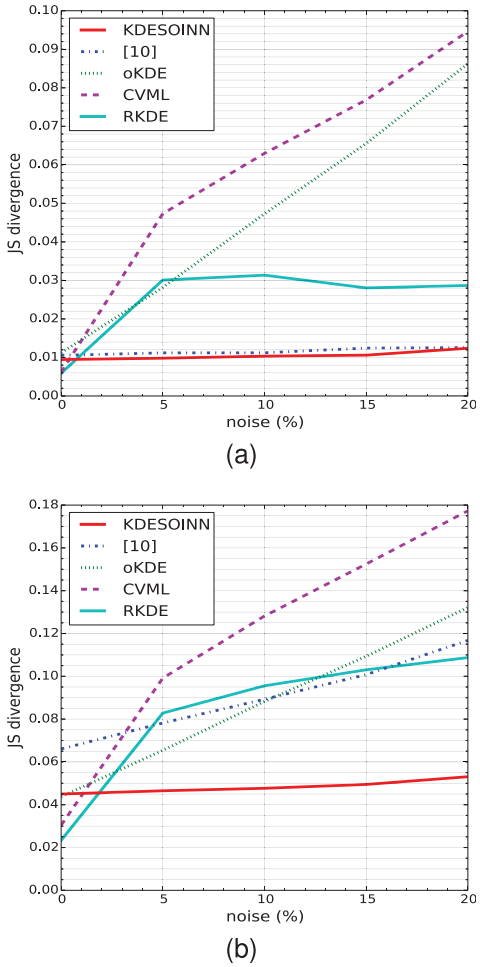

Fig. 5. Experimental results regarding robustness, comparing KDESOINN with oKDE [24], CVML [17], RKDE [9], and our previous method [10]. (a) and (b) Results obtained through (8) and (9) distributions, respectively. Horizontal axis: ratio of noise included in the training data, corresponding with α. Vertical axis: JS divergence, which is our measure of the estimation accuracy.

where $\{x_{\hat{f}i}\}_{i=1}^n$ is the set of samples from $\hat{f}$ and $\{x_{f_0 i}\}_{i=1}^n$ is the set of samples from f_0. In our experiments, $n = 20\,000$. We evaluated each method by calculating the mean of results across 20 experiments for each noise ratio.

The parameter of oKDE, D_{th}, was set to 0.01. The parameters of KDESOINN were set as follows: 1) when f_0 was (8), $\lambda = 500$, $\mathrm{age}_{\mathrm{max}} = 100$, $\rho = 0.1$, and $k = 2d$ and 2) when f_0 was (9), $\lambda = 1000$, $\mathrm{age}_{\mathrm{max}} = 100$, $\rho = 1.0$, and $k = 2d$. The parameters of [10] were set as follows: 1) when f_0 was (8), $\lambda = 300$, $\mathrm{age}_{\mathrm{max}} = 50$, and $\rho = 0.1$ and 2) when f_0 was (9), $\lambda = 3000$, $\mathrm{age}_{\mathrm{max}} = 100$, and $\rho = 1.2$.

Fig. 5 shows our experimental results. In Fig. 5(a), when the noise ratio α is 0, KDESOINN achieved performance comparable to the other comparative methods. As α was increased, the accuracies of oKDE and CVML decreased. The CVML accuracy decreased sharply when α transitioned from 0 to 0.05, indicating that CVML is sensitive to noise. We assumed that oKDE is less sensitive than CVML, because oKDE adaptively sets the shapes, sizes, and number of its kernels, but the oKDE robustness was not enough. Its accuracy

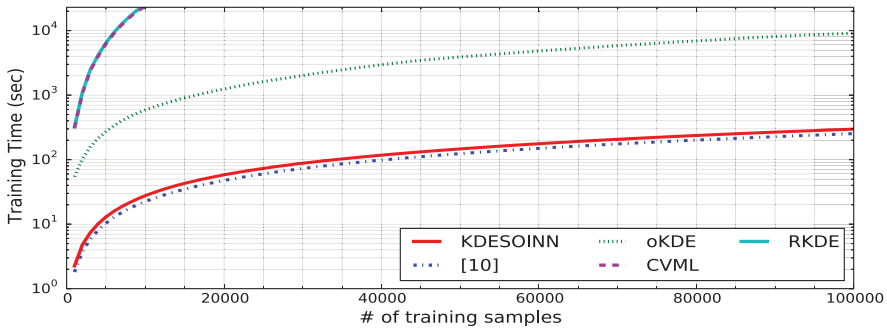

Fig. 6. Experimental result regarding the training times of methods. This figure shows that the increase in the training times of methods with increasing the number of training samples from 1000 to 100 000. KDESOINN is 1390 times faster than CVML [17] and 51.4 times faster than oKDE [24]. Note that the vertical axis is plotted on a logarithmic scale.

TABLE II
LIST OF REAL DATA SETS USED IN EXPERIMENTS IN SECTION IV-E

	#samples	#dimension	#classes
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	50,859	3	1
SUSY	2,287,827	18	1
HIGGS	5,829,123	28	1

decreased with increase in the noise. Conversely, KDESOINN and RKDE, which have strong robustness, kept their accuracies even as the noise increased. In particular, KDESOINN kept its accuracy at the same level as that of $\alpha = 0$. Fig. 5(b) presents these results, showing that KDESOINN, indeed, has high robustness. However, the robustness of the previous method [10] is not high; as α increased, the method's accuracy decreased. This is possibly the result of the strangely distorted shape of its threshold regions.

C. Training Time

We compared KDESOINN with the other methods in terms of increased training time with increase in the number of training samples. We used the samples from (8) without noise as training data, and the number of samples was increased from 1000 to 100 000. Each method's parameters were set as presented in Section IV-B.

Fig. 6 shows our experimental results. CVML and RKDE took so much time to learn that we canceled their experiments when the number of training samples reached 10 000. From our results, KDESOINN was 1390 times faster than CVML. The training times of CVML and RKDE exponentially increased as the number of training data points increased, primarily, because they are batch methods. Because oKDE, KDESOINN, and our previous method [10] are online methods, their training times were less than those of the batch methods. The training times of oKDE increased more than those of KDESOINN. More specifically, KDESOINN was 51.4 times faster than oKDE when the number of training samples was 100 000. We observe here that oKDE clusters and recalculates its bandwidth matrix when it finds a new sample, whereas KDESOINN takes the locations and bandwidth matrices of its kernels into its networks of prototypes. We claim that this is why KDESOINN was much faster than oKDE. KDESOINN is as fast as the previous method [10], though the performance of KDESOINN gets better than that of [10].

D. High Dimensionality

We compared KDESOINN with the other methods with respect to learning high-dimensional data. The aim was to evaluate how estimate accuracies changed with increasing data dimensions. The following distribution is accepted:

$$f(\boldsymbol{x}) = N(\boldsymbol{x}; \boldsymbol{t}, 0.25 \cdot \boldsymbol{I})$$
$$\boldsymbol{t} = [a, \boldsymbol{0}]^T, \quad a \in [-10, 10] \tag{10}$$

as the true distribution. Each method learns 5000 samples from (10) and estimates the distribution. We evaluated each method by calculating the mean KL divergence across 20 experiments for each dimension separately.

Fig. 7 presents the experimental results. The previous method [10] does not perform well when the number of dimensions becomes high. KDESOINN performs better, and its accuracy is higher than that of CVML. The previous method cannot generate networks in high-dimensional space properly. However, KDESOINN can generate, because the threshold regions are set properly and the networks are adjusted. Hypothetically, KDESOINN networks make up for the limited number of samples placed in high-dimensional space. As a result, KDESOINN performs better than CVML. Conversely, oKDE exhibits very good performance. It maintains high accuracy when the dimensionality becomes high, presumably because its compression scheme works very well with a simple true distribution.

E. Density Estimation on Real Data Sets

We evaluated the performance of each method with regard to density estimation using real data sets from the UCI Machine Learning Repository. Table II shows the list of data sets used. We conducted experiments with each class of each data

TABLE III
COMPARISON BETWEEN KDESOINN AND EXISTING METHODS

	CVML [17]	RKDE [9]	oKDE [24]	[10]	**KDESOINN**
Non-parametric	◯	◯	◯	◯	◯
Fast online learning	×	×	△	◯	◯
Robustness	×	△	×	△	◯
High Dimensionality	△	△	◯	×	△
Accuracy	◯	◯	△	△	◯

◯ means a method has the functional capability. △ means a method has the functional capability with lower performance than ◯. × means a method unable to realize the functional capability.

set. Although the Skin data set has 245 057 samples and two classes, positive and negative, only the positive class was used, which includes 50 859 samples. In addition, the SUSY and HIGGS data sets have two classes, background and signal, but only the signal class was used. In each experiment, samples were whitened as standardization. As such, 75% of samples were randomly selected and used as training data, while the rest were used as test data. Each experiment was conducted 20 times, and we evaluated the average of the results of each data set. We used negative log-likelihood (NLL) as our evaluation measure; NLL has a chance to diverge infinitely; thus, we used the medians of NLL to avoid such divergence. The parameter of oKDE, D_{th}, was set to 0.1. The parameters of KDESOINN were set as follows: 1) $\lambda = 1000$; 2) $\text{age}_{\text{max}} = 100$; 3) $\rho = 0.6$; and 4) $k = 2d$. The parameters of our previous method were set as follows: 1) $\lambda = 300$; 2) $\text{age}_{\text{max}} = 50$; and 3) $\rho = 0.5$.

Fig. 8 shows our experimental results. In Fig. 8, the left column shows NLLs, and the right column shows training times per sample. Note that the bar charts are averages, and the error bars are standard deviations. With several data sets, Skin, SUSY, and HIGGS, several methods did not finish even one experiment over a span of 5 d; hence, we cannot show their NLL. KDESOINN outperformed or achieved performance comparable to the batch methods, with the required training times much shorter than those of the batch methods. oKDE's accuracy is not good in spite of high accuracy as mentioned in Section IV-D, presumably because the distributions of real data are complex. The training times of all the methods except for KDESOINN and the previous method [10] increased with increase in the number of training samples. The previous method is slightly faster than KDESOINN, but its accuracy is lower, especially when processing high-dimensional data. The reason is that KDESOINN creates more proper networks by proper threshold regions and adjusting networks. Consequently, KDESOINN is considered faster and more accurate, and it can be applied in practice for density estimation on real data sets.

F. Summary

Table III summarizes our experimental results. From our results, we observe that KDESOINN was superior to the other methods with regard to robustness and fast online learning. Regarding estimation accuracy, KDESOINN outperformed or achieved performance comparable to the other methods.

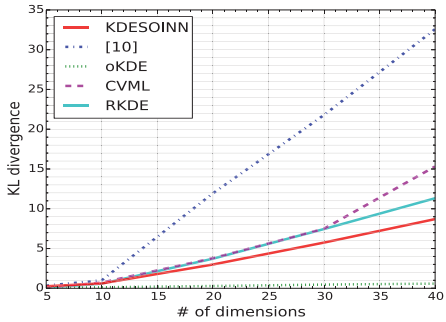

Fig. 7. Experimental results regarding high dimensionality. Horizontal axis: number of dimensions. Vertical axis: KL divergence, which is a measure of the estimation accuracy.

V. DISCUSSION

In this section, we discuss the effects of the extensions on our proposed method and strategies for handling high-dimensional data sets.

A. Comparison With Naive Approaches

Our proposed method, KDESOINN, learns samples as networks using a learning algorithm extended from SOINN, then determines the shape and size of each kernel located on a node of networks from the local network around the node, finally estimating the probability function by calculating the sum of kernels. In this section, we verify that this approach results in improving the performance of KDESOINN over other naive approaches that simply place kernels on the nodes of the SOINNs.

We compared KDESOINN with two naive approaches labeled *Naive* and *Adaptive,* and performed the same experiments as mentioned in Section IV-B. The *Naive* method is an approach that naively integrates SOINNN into KDE. More specifically, it learns samples using SOINN and places kernels on the nodes of SOINNs. The bandwidth matrix of its kernels is optimized through CVML. The *Adaptive* method is an approach that learns from samples using an SOINN, but estimates the density function using our proposed density estimator, defined in (4). All method parameters were set as follows: 1) $\lambda = 500$; 2) $\text{age}_{\text{max}} = 100$; 3) $\rho = 0.1$; and 4) $k = 2d$.

Fig. 9 shows our experimental results. The *Adaptive* method is superior to the *Naive* method, especially when the noise ratio becomes high. SOINN identifies information regarding

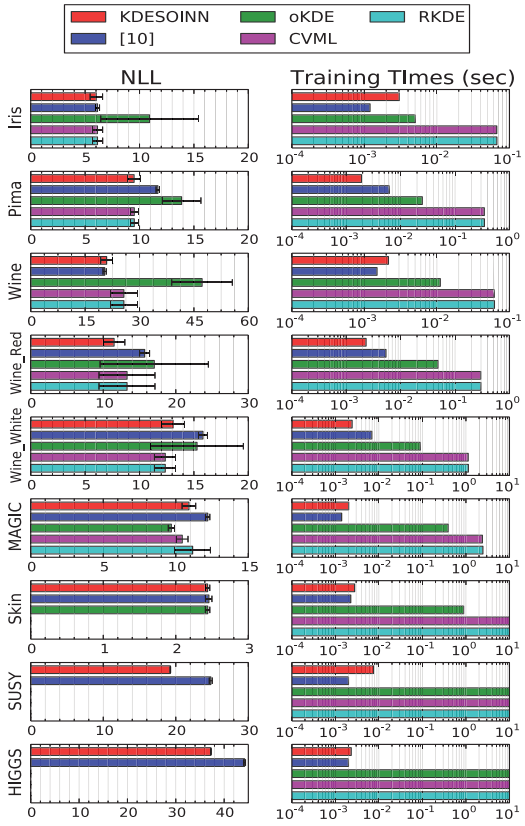

Fig. 8. Experimental results of density estimation on real data sets. Left column: estimation accuracies (NLL). Bar charts: averages. Error bars: standard deviations. Right column: training times per sample. Note that its axes are plotted on a logarithmic scale. From these results, KDESOINN outperformed or achieved performance comparable to batch methods with regard to accuracy and training times.

distribution using not only nodes but also networks, including edges and winning times. Our experimental results show that the estimated density function through KDESOINN [i.e., (4)] extracts the information retained by the networks. In addition, the *Naive* method places the same kernel as that of a typical KDE, but the *Adaptive* method adaptively changes the shape and size of each kernel to each node using (4) and (5), and the difference between these two approaches in terms of performance was large when the noise ratio increased.

Compared with the *Naive* and *Adaptive* methods, KDESOINN had a much better performance. The networks of SOINN had deformations, causing declines in the precision of the estimates. This deformation resulted from creating edges that do not appropriately express the distribution. The learning algorithm of our proposed method, by improving the decision of threshold regions, learned more appropriate densities of samples on the basis of network structure.

B. Handling High-Dimensional Data Sets

In general, the performance of nonparametric density estimators decreases with increase in the number of dimensions [35], [36]. KDESOINN is a nonparametric density estimator; thus, its performance decreases with increase in the number of dimensions. In Section IV-D, we showed the results

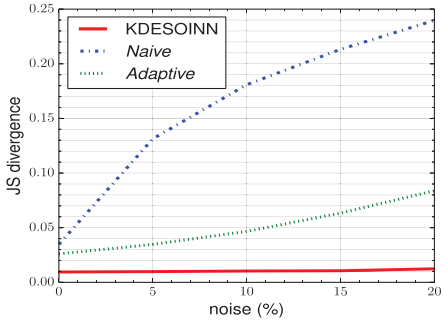

Fig. 9. Experimental results regarding robustness comparing KDESOINN with two methods that naively integrate SOINN into KDE.

with only dozens of dimensions of data. It is not enough for some applications, such as image recognition.

To address this problem, dimensionality reduction and feature extraction can be used. There are methods that execute principal component analysis (PCA) or Kernel PCA in an online fashion [37]–[39]. We can combine such methods with KDESOINN to estimate density functions of higher dimensional data by mapping to lower-dimensional space in an online manner. Furthermore, we feel that KDESOINN can be integrated with manifold learning methods that use graph structures [40]–[42] because KDESOINN is based on a graph structure. Considering density functions on manifolds [43], [44] can avoid the need to handle high-dimensional space directly.

VI. Conclusion

In this paper, we proposed KDESOINN, a robust fast online nonparametric density estimator capable of handling the challenges inherent in big data. We determined that the network structure of SOINN contains information regarding the distribution underlying observed samples. KDESOINN is based on this idea for estimating the density function. KDE, which is the typical nonparametric density estimator, needs too many samples to improve its accuracy, and its computational complexity increases with increase in the number of samples. KDESOINN reduces its computational complexity by creating networks of sample prototypes along distributions. In our experiments, KDESOINN outperformed existing nonparametric density estimators with regard to robustness and training times, and achieved comparable performance with regard to estimation accuracy.

As future work, we plan to consider extensions to KDESOINN to better handle high-dimensional data sets, as discussed in Section V-B, as well as applications of KDESOINN for analyzing big data from real environments. We believe that KDESOINN has broad applicability, because nonparametrically estimating the density function from noisy data in an online fashion makes probabilistic prediction and inference more practical.

References

[1] V. Mayer-Schönberger and K. Cukier, *Big Data: A Revolution That Will Transform How We Live, Work, and Think*. London, U.K.: J. Murray, 2013.

[2] W. Fan and A. Bifet, "Mining big data: Current status, and forecast to the future," *SIGKDD Explorations Newslett.*, vol. 14, no. 2, pp. 1–5, Dec. 2012.
[3] J. K. Laurila *et al.*, "The mobile data challenge: Big data for mobile computing research," in *Proc. Workshop Nokia Mobile Data Challenge, Conjunction 10th Int. Conf. Pervasive Comput.*, 2012, pp. 1–8.
[4] D. Howe *et al.*, "Big data: The future of biocuration," *Nature*, vol. 455, pp. 47–50, Sep. 2008.
[5] D. Laney, "3D data management: Controlling data volume, velocity, and variety," META Group, Terni, Italy, Tech. Rep. 949, Feb. 2001.
[6] P. J. Huber, *Robust Statistics*. New York, NY, USA: Wiley, 1981.
[7] M. Hubert, P. J. Rousseeuw, and K. V. 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). New York, NY, USA: Springer-Verlag, 2013.
[9] J. Kim and C. D. Scott, "Robust kernel density estimation," *J. Mach. Learn. Res.*, vol. 13, pp. 2529–2565, Sep. 2012.
[10] Y. Nakamura and O. Hasegawa, "Robust fast online multivariate non-parametric density estimator," in *Proc. 20th Int. Conf. Neural Inf. Process.*, Daegu, Korea, Nov. 2013, pp. 180–187.
[11] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statist.*, vol. 33, no. 3, pp. 1065–1076, 1962.
[12] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. London, U.K.: Chapman & Hall, 1986.
[13] P. Hall, S. 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.
[14] M. C. Jones, J. S. Marron, and S. J. Sheather, "A brief survey of bandwidth selection for density estimation," *J. Amer. Statist. Assoc.*, vol. 91, no. 433, pp. 401–407, 1996.
[15] A. W. Bowman, "An alternative method of cross-validation for the smoothing of density estimates," *Biometrika*, vol. 71, no. 2, pp. 353–360, 1984.
[16] D. Chaudhuri, B. B. Chaudhuri, and C. A. Murthy, "A data driven procedure for density estimation with some applications," *Pattern Recognit.*, vol. 29, no. 10, pp. 1719–1736, Oct. 1996.
[17] J. M. Leiva and A. A. Rodríguez, "Algorithms for Gaussian bandwidth selection in kernel density estimators," in *Proc. Workshop Optim. Mach. Learn. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Jan. 2008, pp. 1–6.
[18] 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, Dec. 2008.
[19] 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, Oct. 2003.
[20] Z. Deng, F.-L. Chung, and S. Wang, "FRSDE: Fast reduced set density estimator using minimal enclosing ball approximation," *Pattern Recognit.*, vol. 41, no. 4, pp. 1363–1372, Apr. 2008.
[21] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. 19th Int. Conf. Comput. Statist.*, Paris, France, Aug. 2010, pp. 177–186.
[22] A. Declercq and J. H. Piater, "Online learning of Gaussian mixture models—A two-level approach," in *Proc. 3rd Int. Conf. Comput. Vis. Theory Appl.*, Funchal, Portugal, Jan. 2008, pp. 605–611.
[23] K. Tabata, M. Sato, and M. Kudo, "Data compression by volume prototypes for streaming data," *Pattern Recognit.*, vol. 43, no. 9, pp. 3162–3176, Sep. 2010.
[24] M. Kristan, A. Leonardis, and D. Skočaj, "Multivariate online kernel density estimation with Gaussian kernels," *Pattern Recognit.*, vol. 44, nos. 10–11, pp. 2630–2642, Oct./Nov. 2011.
[25] S. Furao and O. Hasegawa, "An incremental network for on-line unsupervised classification and topology learning," *Neural Netw.*, vol. 19, no. 1, pp. 90–106, Jan. 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, Jun. 2009.
[27] N. Makibuchi, F. Shen, and O. Hasegawa, "Online knowledge acquisition and general problem solving in a real world by humanoid robots," in *Proc. 20th Int. Conf. Artif. Neural Netw.*, Thessaloniki, Greece, Sep. 2010, pp. 551–556.
[28] S. Okada and T. Nishida, "Online incremental clustering with distance metric learning for high dimensional data," in *Proc. Int. Joint Conf. Neural Netw.*, San Jose, CA, USA, Jul./Aug. 2011, pp. 2047–2054.
[29] P. Kankuekul, A. Kawewong, S. Tangruamsub, and O. Hasegawa, "Online incremental attribute-based zero-shot learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Providence, RI, USA, Jun. 2012, pp. 3657–3664.
[30] A. Kawewong, R. Pimup, and O. Hasegawa, "Incremental learning framework for indoor scene recognition," in *Proc. 27th AAAI Conf. Artif. Intell.*, Bellevue, WA, USA, Jul. 2013, pp. 496–502.
[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, Mar. 2013.
[32] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems*, vol. 7. Cambridge, MA, USA: MIT Press, 1995, pp. 625–632.
[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, Oct. 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, Dec. 2008.
[35] V. N. Vapnik, *Statistical Learning Theory*. New York, NY, USA: Wiley, 1998.
[36] W. Härdle, *Nonparametric and Semiparametric Models* (Springer Series in Statistics). Heidelberg, Germany: Springer-Verlag, 2004.
[37] T. Oyama, S. G. Karungaru, S. Tsuge, Y. Mitsukura, and M. Fukumi, "Fast incremental algorithm of simple principal component analysis," *IEEJ Trans. Electron., Inf. Syst.*, 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, Jun. 2007.
[39] Y. Takeuchi, S. Ozawa, and S. Abe, "An efficient incremental kernel principal component analysis for online feature selection," in *Proc. Int. Joint Conf. Neural Netw.*, Orlando, FL, USA, Aug. 2007, pp. 2346–2351.
[40] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000.
[41] Z. Zhang and H. Zha, "Principal manifolds and nonlinear dimension reduction via local tangent space alignment," *SIAM J. Sci. Comput.*, vol. 26, no. 1, pp. 313–338, 2002.
[42] K. Yu and T. Zhang, "Improved local coordinate coding using local tangents," in *Proc. 27th Int. Conf. Mach. Learn.*, Haifa, Israel, 2010, pp. 1215–1222.
[43] B. Pelletier, "Kernel density estimation on Riemannian manifolds," *Statist. Probab. Lett.*, vol. 73, no. 3, pp. 297–304, Jul. 2005.
[44] G. Henry and D. Rodriguez, "Kernel density estimation on Riemannian manifolds: Asymptotic results," *J. Math. Imag. Vis.*, vol. 34, no. 3, pp. 235–239, Jul. 2009.

Yoshihiro Nakamura received the B.S. degree in computer engineering from the Nagoya Institute of Technology, Nagoya, Japan, in 2011, and the M.E. degree from the Tokyo Institute of Technology, Yokohama, Japan, in 2013, where he is currently pursuing the Ph.D. degree with the Department of Computational Intelligence and Systems Science.

His current research interests include machine learning.

Osamu Hasegawa (M'93) received the Engineering degree in electronics engineering from the University of Tokyo, Tokyo, Japan, in 1993.

He was a Research Scientist with the Electrotechnical Laboratory from 1993 to 1999, and the National Institute of Advanced Industrial Science and Technology, Tokyo, from 2000 to 2002. From 1999 to 2000, he was a Visiting Scientist with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. In 2002, he became a Faculty Member with the Imaging Science and Engineering Laboratory, Tokyo Institute of Technology, Yokohama, Japan. In 2002, he was jointly appointed as a Researcher with PRESTO, Japan Science and Technology Agency, Tokyo.

Dr. Hasegawa is a member of the IEEE Computer Society, the Institute of Electronics, Information and Communication Engineers, and the Information Processing Society of Japan.