Improved Kernel Density Estimation Self-organizing Incremental Neural Network to Perform Big Data Analysis

Wonjik Kim[1(✉)] and Osamu Hasegawa[1,2]

[1] Department of Systems and Control Engineering, Tokyo Institute of Technology, Tokyo, Japan
{kim.w.ab,hasegawa.o.aa}@m.titech.ac.jp
[2] Inc.SOINN, Cureindo-Building 405, Turuma8-4-30, Tamachi, Tokyo, Japan
oh@soinn.com

Abstract. Plenty of data are generated continuously due to the progress in the field of network technology. Additionally, some data contain substantial noise, while other data vary their properties in according to various real time scenarios. Owing to these factors, analyzing big data is difficult. To address these problems, an adaptive kernel density estimation self-organizing neural network (AKDESOINN) has been proposed. This approach is based on the kernel density estimation self-organizing incremental neural network (KDESOINN), which is an extension of the self-organizing incremental neural network (SOINN). An SOINN can study the distribution using the input data online, while KDESOINN can estimate the probability density function based on this information. The AKDESOINN can adapt itself to the changing data properties by estimating the probability density function. Further, the experimental results depict that AKDESOINN succeeds in maintaining the performance of KDESOINN, while depicting an ability to adapt to the changing data.

Keywords: Neural network · Kernel density estimation · Data analysis
Self-organizing incremental neural network

1 Introduction

Due to the expansion of network communications, data are generated continuously. Such data are called big data, and there have been many attempts to analyze and apply them to various research fields [1, 2, 11, 16, 24].

Laney derived three concepts of the big data characteristics, which are as follows [18]:

(1) Data Volume: Massive amounts of data that continue to grow after being generated.
(2) Data Velocity: Increasing numbers of networks are generating data continuously, which means that the data generation velocity is very high.
(3) Data Variety: Data in a pool can be of different types, such as time-series, real environment, artificial environment, textual, and image data.

L. Cheng et al. (Eds.): ICONIP 2018, LNCS 11302, pp. 3–13, 2018.
https://doi.org/10.1007/978-3-030-04179-3_1

These three characteristics that are considered by [18] are known as the 3Vs, and are taken into consideration while dealing with big data.

In machine learning and data analysis research, it is necessary to estimate the probability density. However, it is difficult to estimate the probability density of big data due to the three reasons [22].

First, the density estimator for big data must be nonparametric because of the data volume. Further, we observe that parametric methods are effective for handling fixed data, because it is possible to tune the parameters of the method to obtain an optimal performance. However, the volume of big data is not observed to be constant. Therefore, the volume of big data cannot be analyzed in advance in order to obtain optimal parameters for the density estimator. However, we observe that the nonparametric density estimator is not troublesome, since analyzing and constructing a big data model beforehand is not necessary for a nonparametric density estimator.

Second, the density estimator for big data must use online learning methods due to the observed data velocity. In big data, massive amounts of data grow quickly until the total size of data becomes gigantic. Online learning methods can be sequentially updated using the growing data.

Third, the density estimator for big data must be robust. Data that are collected from real environments often contain noise, which could cause overfitting and decrease performance. Thus, robust methods are required to deal with data that contain noise.

Further, we observe that robustness is defined differently across various fields [9, 13]. In this study, we define robustness as 'a function that provides almost the same results as learning data without noise when learning with noisy data.' [22]. Further, we observe that there are two types of noise. The first type is the noise that is generated by the environment, but that is not related to the objective distribution. Thus, this type of noise needs to be eliminated. The second type is observed to be related to variance and fluctuation. Therefore, this type of noise must be preserved.

The kernel density estimation self-organizing incremental neural network (KDESOINN) method [22] satisfies all the three conditions for dealing with big data and is further observed to be robust to noise. However, it cannot adapt to a changing environment. Due to the variety of big data, the structure of data is likely to vary at any instance. Therefore, an ability to adapt to the observed variation of data is required. In this study, we propose a revised KDESOINN method to solve this problem. Further, our proposed method has been termed adaptive KDESOINN (AKDESOINN) in this paper.

2 Related Works

2.1 Kernel Density Estimation

Kernel Density Estimation (KDE) is a typical nonparametric density estimation approach [23]. The methodology of KDE process is presented in Algorithm 1

Algorithm 1. Kernel Density Estimation

(1) **Require**: training samples $\{x_i | x_i \in \mathbb{R}^d, i = 1, 2, \ldots, N\}$, K : kernel function, H : bandwidth matrix

(2) $\hat{p}(x) = \frac{1}{N}\sum_{i=1}^{N} K_H(x - x_i)$

For the kernel function K, the Gaussian kernel is often used in an identical manner as that in (1)

$$K_H(x-\mu) = \left(1/\sqrt{(2\pi)^d |H|}\right) * \exp\left(-(x-\mu)^T H^{-1}(x-\mu)/2\right) \tag{1}$$

H in algorithm 1 is a parameter, which influences the performance of the estimation function. Further, attempts have been made to optimize the estimation function [8, 10]. KDE has been investigated using several methods such as by method of setting the number of kernels [3], gradient descent method [19], and online clustering method [17].

2.2 Self-organizing Incremental Neural Network

In the field of artificial intelligence, artificial neural networks have been recently proposed. They are usually classified into two groups, namely, supervised and unsupervised learning [25].

SOINN is an unsupervised learning method that is driven by growing neural gas [4]. There are several kinds of SOINN, including two-layer [5], enhanced [6], and adjusted SOINN [7]. Since the adjusted SOINN has less parameters than that of the other SOINNs, it is generally used in applied research [12, 14, 15].

While SOINN learns from the training data, it constructs a data network through competitive learning. Various nodes are added or deleted from the network or they may update their location. Further, the edges are added or deleted in a similar manner as the nodes. Thus, the SOINN network is updated in order to approximate the distribution using the added input data.

The flowchart of the adjusted SOINN is depicted in Fig. 1, and its procedural flow is presented in Algorithm 2

Algorithm 2. Adjusted SOINN process

(1) **Require**: A: set of all neurons. C ⊂ A × A: set of all edges. N_i: set of all neighbors of neuron i. W_i: weight of neuron i. λ: time period to delete redundant neurons. age_{max}: parameter to delete edges.

(2) **if** first time of input **then**

(3) A ← c_1, c_2; randomly pick up two vectors from training data to initialize the neuron set.

(4) $C \leftarrow \emptyset$

(5) **end if**

(6) **while** input data ξ exist **do**

(7) $s_1 \leftarrow argmin_{c \in A}\|\xi - W_c\|$: find out the winner.

(8) $s_2 \leftarrow argmin_{c \in A \setminus s_1}\|\xi - W_c\|$: find out second winner.

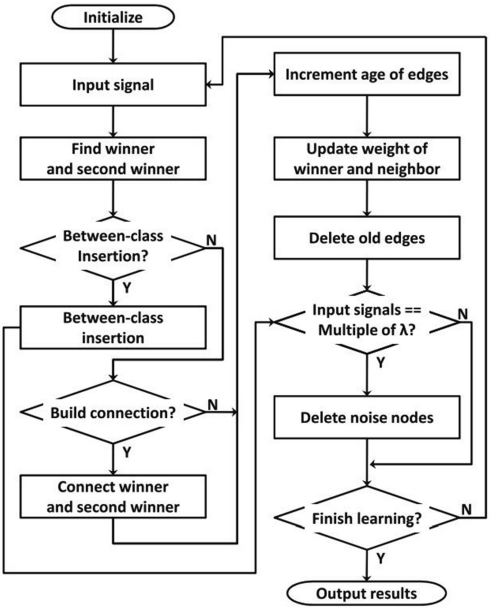

Fig. 1. Flowchart of SOINN

(9) calculate similarity thresholds T_{s_1}, T_{s_2}. If i got neighbors, T_i is the distance to the farthest neighbor, else the distance to the nearest neuron.
(10) **if** $\|\xi - W_{s_1}\| > T_{s_1}$ or $\|\xi - W_{s_2}\| > T_{s_2}$ **then**
(11) $\mathrm{A} \leftarrow \mathrm{A} \cup \xi$: insert ξ as a new neuron.
(12) **else**
(13) **if** $(s_1, s_2) \notin \mathrm{C}$: there is no edge between the winner and second winner, **then**
(14) $\mathrm{C} \leftarrow \mathrm{C} \cup (s_1, s_2)$: add new edge into the network
(15) **end if**
(16) $age_{(s_1,s_2)} \leftarrow 0$: reset the age of (s_1, s_2)
(17) $age_{(s_1,i)} \leftarrow age_{(s_1,i)} + 1(\forall i \in N_{s_i})$: increase age of edges connected with the winner by 1.
(18) $\Delta W_{s_i} = \epsilon(t_{s_1})(\xi - W_{s_1}), \Delta W_i = \epsilon(100t_i)(\xi - W_i)(\forall i \in N_{s_i}), \epsilon(t) = \frac{1}{t}$
(19) using $vartriangle W_{s_i}, \Delta W_i$ to adjust the winner and its neighbors
(20) delete edges whose age is larger than age_{max}
(21) among these neurons which the edge deleted in last step connected to, delete neurons having no neighbors.
(22) **end if**
(23) **if** input data number becomes $\mathrm{n} \times \lambda (\mathrm{n} \in N^+)$ **then**
(24) Delete neurons having less than one neighbor
(25) **end if**
(26) **end while**

2.3 KDESOINN

KDESOINN is an extended version of the adjusted SOINN [22]. It determines the structure of the network using each kernel in the node of a local network that is located near the node. Additionally, it estimates the probability function using the sum of the kernels. In the adjusted SOINN, only the Euclidean distance is used for calculating the similarity thresholds. Conversely, KDESOINN calculates the threshold using Algorithm 3.

Algorithm 3. KDESOINN threshold calculation

(1) **Require**: A: set of all neurons. ξ: new sample data. P_i: set of nodes connected to node i. ρ: parameter for threshold. $\varpi_i \in \mathbb{R}^d$: positional vector of node i. t_i: number of wins of node i in competitive learning. I: identity matrix. Θ_i: threshold region of node i.

(2) calculate $\gamma_i = \begin{cases} \min_{p \in P_i} \left\| w_p - w_i \right\| & (P_i \neq \phi) \\ \min_{p \in A\{i\}} \left\| w_p - w_i \right\| & (otherwise) \end{cases}$

(3) $T_{P_i} \leftarrow \sum_{i \in P_i} t_i$

(4) $C_i \leftarrow \frac{1}{T_{P_i}} \sum_{p \in P_i} t_p \left(w_p - w_i\right)\left(w_p - w_i\right)^T$

(5) $M_i \leftarrow C_i + \rho \gamma_i I$

(6) threshold region $\Theta_i = (\xi - w_i)^T M_i^{-1} (\xi - w_i) \leq 1$

KDESOINN can divide clusters more effectively than the adjusted SOINN. The entire process of KDESOINN is presented in Algorithm 4

Algorithm 4. KDESOINN process

(1) **Require**: A: set of all neurons. $C \subset A \times A$: set of all edges. N_i: set of all neighbors of neuron i. W_i: weight of neuron i. λ: time period to delete redundant neurons. age_{max}: parameter to delete edges. P_i: set of nodes connected to node i. ρ: parameter for threshold. t_i: number of wins of node i in competitive learning. I: identity matrix. E(G): set of edges in graph G.

(2) **if** first time of input **then**

(3) $A \leftarrow c_1, c_2$; randomly pick up two vectors from training data to initialize the neuron set.

(4) $C \leftarrow \emptyset$

(5) **end if**

(6) **while** input data ξ exist **do**

(7) $s_1 \leftarrow argmin_{c \in A} \|\xi - W_c\|$: find out the winner.

(8) $s_2 \leftarrow argmin_{c \in A \setminus s_1} \|\xi - W_c\|$: find out second winner.

(9) calculate similarity thresholds $\Theta_{s_1}, \Theta_{s_2}$ by algorithm 3.

(10) **if** $(\xi - W_{s_1})^T M_{s_1}^{-1} (\xi - W_{s_2}) > 1$ or $(\xi - W_{s_2})^T M_{s_2}^{-1} (\xi - W_{s_2}) > 1$ **then**

(11) $A \leftarrow A \cup \xi$: insert ξ as a new neuron.

(12) **else**

(13) **if** $(s_1, s_2) \notin C$: there is no edge between the winner and second winner, **then**

(14) $C \leftarrow C \cup (s_1, s_2)$: add new edge into the network

(15) **end if**

(16) $age_{(s_1,s_2)} \leftarrow 0$: reset the age of (s_1, s_2)
(17) $age_{(s_1,i)} \leftarrow age_{(s_1,i)} + 1(\forall i \in N_{s_i})$: increase age of edges connected with the winner by 1.
(18) $\Delta W_{s_i} = \epsilon(t_{s_1})(\xi - W_{s_1}), \Delta W_i = \epsilon(100t_i)(\xi - W_i)(\forall i \in N_{s_i}), \epsilon(t) = \frac{1}{t}$
(19) using $vartriangle W_{s_i}, \Delta W_i$ to adjust the winner and its neighbors
(20) delete edges whose age is larger than age_{max}
(21) among these neurons which the edge deleted in last step connected to, delete neurons having no neighbors.
(22) **end if**
(23) **if** input data number becomes $\mathrm{n} \times \lambda(\mathrm{n} \in N^+)$ **then**
(24) delete neurons having no neighbor
(25) create a k-NN graph G whose set of nodes is A.
(26) $\mathrm{C} \leftarrow \mathrm{C} \cup \{(i,j)|(i,j) \in E(G), (j,i) \in E(G)\}$
(27) **end if**
(28) **end while**
(29) create a k-NN graph G whose set of nodes is A.
(30) $\mathrm{C} \leftarrow \mathrm{C} \cup \{(i,j)|(i,j) \in E(G), (j,i) \in E(G)\}$

3 Proposed Method

To improve KDSOINNs ability of adapting to the changing data, algorithm 5 was used after line 10 of Algorithm 4

Algorithm 5. Adaptive step

(1) **Require**: s_1: first winner. s_2: second winner. η: parameter for adapting. ξ: new sample data.
(2) $D_{s_1} \leftarrow |s_1 - \xi|, D_{s_2} \leftarrow |s_2 - \xi|$
(3) update $s_1 \leftarrow s_1 + \frac{\eta D_{s_2}}{D_{s_1} + D_{s_2}}(\xi - s_1)$
(4) update $s_2 \leftarrow s_2 + \frac{\eta D_{s_1}}{D_{s_1} + D_{s_2}}(\xi - s_2)$

By applying algorithm 5, SOINN can adapt to the data as they change with time. η is the adaptation parameter. Further, if η is observed to be equal to 0, the performance of AKDESOINN is observed to be exactly the same as that of KDESOINN. If η is observed to be bigger than 1, it is possible that it can fit over ξ. To avoid overfitting and low performance, it is recommended to set η within the range of 0 to 1. The entire process of AKDESOINN is presented in algorithm 6.

Algorithm 6. AKDESOINN process

(1) **Require**: A: set of all neurons. $\mathrm{C} \subset \mathrm{A} \times \mathrm{A}$: set of all edges. N_i: set of all neighbors of neuron i. W_i: weight of neuron i. λ: time period to delete redundant neurons. age_{max}: parameter to delete edges. P_i: set of nodes connected to node i. ρ: parameter for threshold. η: parameter for adapting. t_i: number of wins of node i in competitive learning. I: identity matrix. E(G): set of edges in graph G.

(2) **if** first time of input **then**
(3) A $\leftarrow c_1, c_2$; randomly pick up two vectors from training data to initialize the neuron set.
(4) $C \leftarrow \emptyset$
(5) **end if**
(6) **while** input data ξ exist **do**
(7) $s_1 \leftarrow argmin_{c \in A} \xi - W_c$: find out the winner.
(8) $s_2 \leftarrow argmin_{c \in A \setminus s_1} \xi - W_c$: find out second winner.
(9) calculate similarity thresholds $\Theta_{s_1}, \Theta_{s_2}$ by Algorithm 3.
(10) **if** $(\xi - W_{s_1})^T M_{s_1}^{-1} (\xi - W_{s_2}) > 1$ or $(\xi - W_{s_2})^T M_{s_2}^{-1} (\xi - W_{s_2}) > 1$ **then**
(11) $D_{s_1} \leftarrow |s_1 - \xi|, D_{s_2} \leftarrow |s_2 - \xi|$
(12) update the location of $s_1 \leftarrow s_1 + \frac{\eta D_{s_2}}{D_{s_1} + D_{s_2}} (\xi - s_1)$
(13) update the location of $s_2 \leftarrow s_2 + \frac{\eta D_{s_1}}{D_{s_1} + D_{s_2}} (\xi - s_2)$
(14) A $\leftarrow$ A $\cup \xi$: insert ξ as a new neuron.
(15) **else**
(16) **if** $(s_1, s_2) \notin$ C: there is no edge between the winner and second winner, **then**
(17) C $\leftarrow$ C $\cup (s_1, s_2)$: add new edge into the network
(18) **end if**
(19) $age_{(s_1,s_2)} \leftarrow 0$: reset the age of (s_1, s_2)
(20) $age_{(s_1,i)} \leftarrow age_{(s_1,i)} + 1 (\forall i \in N_{s_i})$: increase age of edges connected with the winner by 1.
(21) $\Delta W_{s_i} = \epsilon(t_{s_1})(\xi - W_{s_1}), \Delta W_i = \epsilon(100 t_i)(\xi - W_i)(\forall i \in N_{s_i}), \epsilon(t) = \frac{1}{t}$
(22) using $vartriangle W_{s_i}, \Delta W_i$ to adjust the winner and its neighbors
(23) delete edges whose age is larger than age_{max}
(24) among these neurons which the edge deleted in last step connected to, delete neurons having no neighbors.
(25) **end if**
(26) **if** input data number becomes n $\times \lambda$(n $\in N^+$) **then**
(27) delete neurons having no neighbor
(28) create a k-NN graph G whose set of nodes is A.
(29) C $\leftarrow$ C $\cup \{(i,j) | (i,j) \in E(G), (j,i) \in E(G)\}$
(30) **end if**
(31) **end while**
(32) create a k-NN graph G whose set of nodes is A.
(33) C $\leftarrow$ C $\cup \{(i,j) | (i,j) \in E(G), (j,i) \in E(G)\}$

4 Experimental Study

In order to compare AKDESOINN with other methods, experimental evaluations were performed to evaluate the robustness, calculation time, accuracy, and adaptation ability of SOINN, KDESOINN, and AKDESOINN. The experimental environment comprised MATLAB2017b that was used on a personal computer having an eight-core CPU at 3.40 GHz and 16.0 GB RAM.

4.1 Fixed Gaussian Distribution

Initially, we evaluated the performance of the proposed method using a fixed Gaussian distribution. Specific details regarding the experiment are described in Table 1.

Table 1. Information of experiment 1

Description	Details
Gaussian distribution 1	$\mu = (1, 1), \sigma = 0.25$
Gaussian distribution 2	$\mu = (0, 0), \sigma = 0.25$
Gaussian distribution 3	$\mu = (-1, -1), \sigma = 0.25$
Uniform distribution	Range of $(-2, -2)$ to $(2, 2)$
Number of data	1000 in each distribution Total number: 4000
λ	200
age_{max}	50
ρ	0.1
H	1

The experiment was repeated 100 times. Further, the Jensen-Shannon divergence was used to compare the accuracy [20], and the results are presented in Table 2.

Table 2. Result of the 100 trials of experiment 1

Description	Mean	Variance	Total computation time [s]
SOINN	4.40E−02	3.63E−05	4.42E01
KDESOINN	1.03E−02	8.68E−07	1.09E02
AKDESOINN	1.10E−02	1.71E−06	1.27E02

According to Table 2, KDESOINN is observed to depict the most effective performance in this experiment. Further, AKDESOINN depicts better performance than SOINN.

4.2 Changing Gaussian Distribution

To evaluate the adaptation performance to the changing data, a Gaussian distribution was used in the experiment. Specific details of the experiment are provided in Table 3.

The experiment was repeated 100 times, and the results were compared using the Jensen-Shannon divergence, in a similar manner as that in experiment 1. The results of the comparison are presented in Table 4.

According to Table 4, AKDESOINN was observed to be the most effective in experiment 2 in terms of mean value

Table 3. Information of experiment 2

Description	Details
Gaussian distribution 1	$\mu = (1, 1)$
Gaussian distribution 2	$\mu = (0, 0)$
Gaussian distribution 3	$\mu = (-1, -1)$
Sigma	$\sigma : 0.25 \rightarrow 0.15$with 0.01interval
Uniform distribution	Range of $(-2, -2)$ to $(2, 2)$
Number of data	100 in each distribution Total number 4400
Λ	200
age_{max}	50
ρ	0.1
H	1

Table 4. Result of the 100 trials of experiment 2

Description	Mean	Variance	Total calculation time [s]
SOINN	6.88E−02	7.72E−05	52.2E01
KDESOINN	2.04E−02	2.10E−06	1.33E02
AKDESOINN	1.88E−02	3.13E−06	1.73E02

5 Conclusion

In this study, AKDESOINN was proposed, not only as a robust fast online nonparametric density estimator, but also as an adaptive method to the changing data. KDESOINN is a method combining both KDE and SOINN and is known to outperform the existing nonparametric density estimators in terms of robustness, calculation cost, and accuracy. The revised KDESOINN algorithm was successful in adapting to the changing data without depicting any performance loss.

For future studies, we could analyze the application of AKDESOINN to meteorological data. Because of the extraordinary climatic conditions, analysis models need to be updated constantly [21]. Because AKDESOINN can adapt its model online with analyzing the data, AKDESOIN may be effective in addressing this problem of constant change of large amounts of climatic data. Apart from the meteorological data, AKDESOINN could be extensively applicable to other fields that require nonparametric density estimation with a suitable adaptation ability to the changing data.

References

1. Amimi, S., Ilias Gerostathopoulos, I,. Prehofer, C.: Big data analytics architecture for real-time traffic control. In: 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems, pp. 710–715 (2017)
2. Anker, S., Asselbergs, F.W., Brobert, G., Vardas, P., Grobbee, D.E., Cronin, M.: Big data in cardiovascular disease. Eur. Hear. J. **38**(24), 1863–1865 (2017)
3. Deng, Z., Chung, F.L., Wang, S.: FRSDE: fast reduced set density estimator using minimal enclosing ball approximation. Pattern Recognit. **41**(4), 1363–1372 (2008)
4. Fritzke, B.: A growing neural gas network learns topologies. In: Advances in Neural Information Processing Systems, vol. 7, pp. 625–632, MIT Press, USA (1995)
5. Furao, S., Hasegawa, O.: An incremental network for on-line unsupervised classification and topology learning. Neural Netw. **19**(1), 90–106 (2006)
6. Furao, S., Ogura, T., Hasegawa, O.: An enhanced self-organizing incremental neural network for online unsupervised learning. Neural Netw. **20**(8), 893–903 (2007)
7. Furao, S., Hasegawa, O.: A fast nearest neighbor classifier based on self-organizing incremental neural network. Neural Netw. **211**(10), 1537–1547 (2008)
8. Hall, P., Sheather, S.J., Jones, M.C., Marron, J.S.: On optimal data-based bandwidth selection in kernel density estimation. Biometrika **78**(2), 263–269 (1991)
9. Huber, P.J., Ronchetti, E.M.: Robust Statistics. International Encyclopedia of Statistical Science, pp. 1248–1251. Springer Press, Berlin (2011)
10. Jones, M.C., Marron, J.S., Sheather, S.J.: A brief survey of bandwidth selection for density estimation. J. Am. Stat. Assoc. **91**(433), 401–407 (1996)
11. John, W.S.: Big data: a revolution that will transform how we live, work, and think. Int. J. Advert. **33**(1), 181–183 (2014)
12. Kawewong, A., Pimup, R., Hasegawa, O.: Incremental learning framework for indoor scene recognition. In: Proceedings of the 27th AAAI Conference on Artificial Intelligence, pp. 496–502. Bellevue (2013)
13. Kim, J., Scott, C.D.: Robust kernel density estimation. J. Mach. Learn. Res. **13**, 2529–2565 (2012)
14. Kim, W., Hasegawa, O.: Prediction of tropical storms using self-organizing incremental neural networks and error evaluation. In: Liu, D., Xie, S., Li, Y., Zhao, D., El-Alfy, E.S. (eds.) Neural Information Processing. ICONIP 2017. LNCS, vol. 10636, pp. 846–855. Springer Press, Cham (2017). https://doi.org/10.1007/978-3-319-70090-8_86
15. Kim, W., Hasegawa, O.: Time series prediction of tropical storm trajectory using self-organizing incremental neural networks and error evaluation. J. Adv. Comput. Intell. Intell. Inform. **22**(4), 465–474 (2018)
16. Rob, K., Gavin, M.: What makes Big Data, Big Data? Exploring the ontological characteristics of 26 datasets. Big Data & Society **3**(1) (2016)
17. Kristan, M., Leonardis, A., Skočaj, D.: Multivariate online kernel density estimation with Gaussian kernels. Pattern Recognit. **44**, 2630–2642 (2011)
18. Laney, D.: 3D data management: controlling data volume, velocity, and variety. META group research note 6, 1 (2001)
19. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Lechevallier, Y., Saporta, G. (eds,) Proceedings of COMPSTAT 2010, pp. 177–186. Springer Press, Heidelberg (2010). https://doi.org/10.1007/978-3-7908-2604-3_16
20. Lin, J.: Divergence measures based on the Shannon entropy. IEEE Trans. Inf. Theory **37**(1), 145–151 (1991)

21. Lobell, D.B., et al.: Prioritizing climate change adaptation needs for food security in 2030. Science **319**(5863), 607–610 (2008)
22. Nakamura, Y., Hasegawa, O.: Nonparametric density estimation based on self-organizing incremental neural network for large noisy data. IEEE Trans. Neural Netw. Learn. Syst. **28**(1), 8–17 (2017)
23. Parzen, E.: On estimation of a probability density function and mode. Ann. Math. Stat. **33**, 1065–1076 (1962)
24. Chris, P.P.: Big knowledge from big data in functional genomics. Emerg. Top. Life Sci. **1**(3), 245–248 (2017)
25. Zurada, J.M.: Introduction to Artificial Neural Systems. West, St. Paul (1992)