Density Based Self Organizing Incremental Neural Network For Data Stream Clustering

Baile Xu, Furao Shen* and Jinxi Zhao
National Key Laboratory for Novel Software Technology
Department of Computer Science and Technology, Nanjing University, China
Email:mg1433069@smail.nju.edu.cn,frshen@nju.edu.cn,jxzhao@nju.edu.cn

***Abstract*—Clustering is an important technique widely used in many areas such as machine learning, pattern recognition, data analysis etc. Data stream clustering is a branch of clustering that draws much attention in recent years, where data objects are processed as an ordered sequence. In this paper, we propose an unsupervised learning neural network named Density Based Self Organizing Incremental Neural Network(DenSOINN) for data stream clustering tasks. DenSOINN is a self organizing competitive network that grows incrementally to learn suitable nodes to fit the distribution of learning data, combining online unsupervised learning and topology learning by means of competitive Hebbian learning rule [19]. By adopting a density-based clustering mechanism, DenSOINN can discover arbitrarily shaped clusters and diminish the negative effect of noise. In addition, we adopt a self-adaptive distance framework to obtain good performance for learning unnormalized input data. Experiments show that the DenSOINN can achieve high standard performance equally on both raw data and normalized data.**

I. Introduction

The main task of clustering is grouping similar data objects in a given data set into meaningful clusters. Data stream clustering is a special type of clustering that draws much attention in recent years, motivated by applications involving large data sets and real time learning tasks. A data stream is a set of data objects organized as an ordered sequence. Stream clustering models have many differences compared to traditional off-line algorithms [24].

First, stream clustering models can not access data objects randomly or iteratively learn the whole data set. Only a small amount of data objects can be stored in memory and need to be discarded after they are processed.

Second, underlying data distribution in the stream might be non-stationary. Stream clustering models should have the ability to learn new data distribution without forgetting previously learned knowledge.

Third, data set can be very large compared to time and storage space available to an algorithm. Stream clustering models should use concise representation of clusters, and be highly efficient in time.

Moreover, a problem remains unsolved in all existing stream clustering models. As a measure of dissimilarity between data objects, the distance metric used in a clustering algorithm plays an important role. In many real world data sets, the range of different features varies widely. Most distance metrics do not work well in this situation, because features with relatively wider range of value will be decisive in the final distance [3]. To make sure each feature contributes equally in the distance metric, data normalization (i.e. feature scaling) is used as an important preprocess technique to normalize the ranges of the features. However, most data normalization algorithms can not be realized under streaming constraints because they need to calculate the statistical characteristics of the whole data set, which is unfeasible when data access is sequential.

In this paper, we propose an innovate data stream clustering method named Density Based Self Organizing Incremental Network (DenSOINN), a competitive neural network based model with the following advantages:

(1) We adopt a self-adaptive distance metric to approximate the effect of data normalization, making the algorithm works on raw data as well as on normalized data.

(2) The network structure of DenSOINN is automatically built during learning procedure by means of Competitive Hebbian Learning [19].

(3) We provide a density based method to solve the problem of finding clusters in a Competitive Hebbian Learning based network .

The rest of the paper is organized as follows: section II reviews related works, section III describes details of DenSOINN, section IV presents experimental results on both artificial and real world data sets, section V concludes the paper.

II. Related Works

A. Data Stream Clustering Algorithms

In the past decades, many stream clustering algorithms have been proposed based on adaption of off-line algorithms, such as BIRCH [26], CluStream [2], DenStream [6], StreamKM++ [1], etc. These algorithms can be generally summarized into two steps: an online learning step that derives a data abstraction from input data stream, and an offline clustering step that generates the final result. During the online learning step, input data objects are summarized into specific data structures such that the algorithm can preserve data distribution without storing these data objects. For example, data abstraction are named "micro-clusters" in DenStream, and "coreset" in StreamKm++. When a clustering request is received, an off-line clustering is perform on the data abstraction. In this step, K-Means [18] is used by BIRCH, CluStream and StreamKM++, while DBSCAN [7] is used by DenStream.

978-1-5090-0620-5/16/$31.00

Competitive neural network is another type of models that can be used in stream clustering, like Self Organizing Maps [15] and Neural Gas [20]. During online learning step, each input data object would trigger a competition among neural nodes, and the winner will be given a reward. As learning procedure goes on, the nodes will finally become an abstraction of training data. However, how to get final clusters from the network remains a problem. An usual way is to appoint each node as a cluster, but most neural networks have a predefined structure and the number of nodes may be largely different from the number of underlying clusters.

Most stream clustering algorithms adopt K-Means or DBSCAN in their offline clustering step. However, both methods have their disadvantages. K-Means needs to decide the number of clusters k, and tends to generate similar sized spherical clusters. DBSCAN can detect a suitable number of arbitrary shaped clusters, but it has a distance relevant parameter ϵ, which is critical to clustering performance but sometimes difficult to find an appropriate value, especially on high dimensional data. Moreover, DBSCAN has difficulty in separating overlapped clusters.

B. Competitive Hebbian Learning And Relevant Models

Competitive Hebbian Learning is a topology learning method in competitive neural networks proposed by Martinetz in [19]. Its purpose is finding a topological structure that closely reflects the topology of the data distribution. The principle of Competitive Hebbian Learning is: for each incoming data object, connect the two nodes closest to it by a connection. Martinetz also proved that given a set of points distributed over a data manifold, this method can generate a perfect topology preserving map of the manifold in [19]. He proposed a topology representing network in [21]. In this network, connections are automatically learned by Competitive Hebbian Learning and weight of nodes are learned using the same rule as Neural Gas. This algorithm still requires an a priori decision about the size of the network.

Improved models based on Competitive Hebbian Learning have been proposed in later years, such as Growing Neural Gas (GNG) [8] and Self Organizing Incremental Neural Network(SOINN) [10]. In these algorithms the number of nodes is not predefined, but could grow as training procedure goes on. GNG adds a new node into the network at the position where accumulated error is the highest after learning a predefined number of data objects until its size grows to a limit. SOINN adds new nodes based on a distance threshold criterion.

As mentioned above, when using these models for clustering, a problem is how to analyze clusters in the network. To accurately represent the topology structure, the network must contain enough number of nodes, which might far exceed the number of underlying classes. SOINN appoint each connected component of the topological map as a cluster: starting from an unclassified node, mark all the nodes with a path to it with a specific label. However, this method can not separate overlapped clusters perfectly, and is quite sensitive to parameter setting. There are several improved models based on SOINN, including E-SOINN [11], Adjusted SOINN [12] and LB-SOINN [25]. These models made some improvements to detect overlapped areas in the network. E-SOINN adopts a density estimation based strategy: connected high density nodes are viewed as potential clusters and the connections between different potential clusters are removed. LB-SOINN improved the method of finding potential clusters. This method has been proven to be useful, but it might break the topological structure of the network.

III. Density Based Self Organizing Incremental Neural Network

DenSOINN is a single-layer competitive neural network that learns the density distribution of input data, represented by a graph $G = (N, C)$. N is the set of neural nodes. Each node $i \in N$ is a prototype with a weight vector $\boldsymbol{w_i} = (w_{i,1}, w_{i,2}, ..., w_{i,n})^T$ representing its position in n-dimensional input space. The set of connections C represents the topological structure. As a clustering algorithm, DenSOINN also outputs a node label vector $\boldsymbol{l}$ that separates N into subsets. Denote the size of N as $|N|$, $\boldsymbol{l} = (l_1, l_2, ..., l_{|N|})^T$, then two nodes i and j are in the same cluster if $l_i = l_j$.

A flowchart of DenSOINN is shown in Figure 1. Like many other stream clustering algorithms, DenSOINN can be summarized into two steps: an online training of network structure and an offline density based clustering of neural nodes. We will describe the detail of each step in following parts of this section.

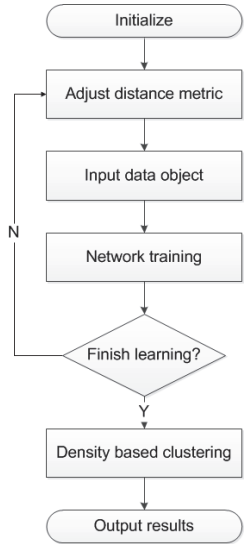


Fig. 1. Flowchart of DenSOINN.

A. Self-adaptive distance metric

We discussed that data normalization is unable to realize in stream clustering. To solve this problem, we adopt an self-adaptive weighted Euclidean distance as distance metric in

DenSOINN in order to make all features of input data objects contribute approximately proportionately to the final distance.

Denote the n-dimensional coefficient vector of weighted Euclidean distance as $\boldsymbol{c} = (c_1, \ldots, c_n)^T$, then for two data objects $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ and $\boldsymbol{y} = (y_1, \ldots, y_n)^T$,

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{n} c_i (x_i - y_i)^2} \tag{1}$$

We use this distance metric on raw data to approximate the Euclidean distance on the normalized data. $d(\boldsymbol{x}, \boldsymbol{y})$ will be used to denote the distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ calculated by this metric in the following parts of this paper. In the algorithms and experiments described in this paper, we adjust $\boldsymbol{c}$ according to the method of min-max normalization. Suppose the range of a feature in ith-dimension is $[min_i, max_i]$, and the target range of min-max normalization is $[0, 1]$,then for a data object $\boldsymbol{x}$ the formula is:

$$x_i^{'} = \frac{x_i - min_i}{max_i - min_i} \tag{2}$$

The Euclidean distance between two normalized n-dimensional vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is

$$d_{euc}(\boldsymbol{x}^{'}, \boldsymbol{y}^{'}) = \sqrt{\sum_{i=1}^{n} (\frac{x_i - y_i}{max_i - min_i})^2} \tag{3}$$

By comparing equations (1) and (3), we can get the following conclusion: when $max_i \neq min_i$, $c_i = \frac{1}{(max_i - min_i)^2}$; when $max_i = min_i$, all data objects have the same attribute value on dimension i, therefore all distances between data objects on dimension i is zero, $c_i = 0$.

A major problem of min-max normalization is that the maximum and minimum values of input data objects might be affected by noise. In DenSOINN, we use the maximum and minimum value of node weights instead. At the beginning of each learning round, the coefficient vector $\boldsymbol{c}$ is adjusted by Algorithm 1.

It is worth mentioning that there are many other data normalization algorithms besides min-max normalization, and no single algorithm works best on all data sets. Some algorithms whose relevant parameters have an online way of calculation can also fit in our self-adaptive distance framework, such as z-score normalization (i.e. standardization). We could choose from different methods of adjusting distance metric according to the characteristic of learning data, but in the following parts of this paper we still use the method described in Algorithm 1. Moreover, some data sets do not need data normalization. Although we ensemble the self-adaptive distance metric into our algorithm, user can choose not to realize it in their applications.

Algorithm 1 Self-adaptive Distance Metric

Input:
-N:set of nodes;
Output:
-$\boldsymbol{c}$:n-dimensional coefficient vector of weighted Euclidean distance;

for each dimension of input space i **do**
  $max_i = -\infty, min_i = \infty$;
  for each node $j \in N$ **do**
    if $w_{j,i} > max_i$ **then**
      $max_i = w_{j,i}$;
    else
      if $w_{j_i} < min_i$ **then**
        $min_i = w_{j,i}$;
      end if
    end if
  end for
  if $max_i = min_i$ **then**
    $c_i = 0$;
  else
    $c_i = \frac{1}{(max_i - min_i)^2}$;
  end if
end for
$\boldsymbol{c} = (c_1, ..., c_n)^T$;
return $\boldsymbol{c}$

B. Online network training

The nodes and connections in DenSOINN are iteratively trained during online learning step. For each node i, the following properties are preserved besides its weight $\boldsymbol{w_i}$: (1)an integer m_i denotes its times of winning in competition; (2)an activation threshold t_i. Denote the edge between node i and its neighbor j as (i, j), and the set of the neighbors of i as N_i, then t_i is the maximum distance between i and its neighbors:

$$t_i = \max_{j \in N_i} d(\boldsymbol{w_j}, \boldsymbol{w_i}) \tag{4}$$

If N_i is empty, then t_i is defined as the minimum distance between i and other nodes in N:

$$t_i = \min_{j \in N} d(\boldsymbol{w_j}, \boldsymbol{w_i}) \tag{5}$$

At the beginning of the algorithm, DenSOINN is initialized as an empty graph $G = (N, C)$, where $N = \phi$, $C = \phi$. When a training data object $\boldsymbol{x}$ arrived, if $|N| < 2$, then add a new node i at the position of $\boldsymbol{x}$ to N, i.e. $\boldsymbol{w_i} = \boldsymbol{x}$ and $m_i = 1$. Otherwise, adjust the distance metric by Algorithm 1, then trigger a competition among the nodes in N. The winner s_1 and runner-up s_2 are found by the following equations:

$$s_1 = arg \min_{i \in N} d(\boldsymbol{x}, \boldsymbol{w_i}) \tag{6}$$

$$s_2 = arg \min_{i \in N - \{s_1\}} d(\boldsymbol{x}, \boldsymbol{w_i}) \tag{7}$$

If $\boldsymbol{x}$ can activate both s_1 and s_2, i.e. $d(\boldsymbol{x}, \boldsymbol{w_{s_1}}) \leq t_{s_1}$ and $d(\boldsymbol{x}, \boldsymbol{w_{s_2}}) \leq t_{s_2}$, the input vector is judged as belonging to the same cluster of s_1, and s_1 is updated as following:

$$m_{s_1} = m_{s_1} + 1 \tag{8}$$

$$\boldsymbol{w_{s_1}} = \boldsymbol{w_{s_1}} + \frac{1}{m_{s_1}}(\boldsymbol{x} - \boldsymbol{w_{s_1}}) \tag{9}$$

If no edge exists between s_1 and s_2, connect them with a new edge, otherwise set the age of existing edge as 0. Then increase the age of all edges linked to s_1 by 1. If the age of any edge is greater than a predefined parameter δ, the edge is removed from the network.

If $\boldsymbol{x}$ can not activate s_1 or s_2, then $\boldsymbol{x}$ does not belong to any existing cluster, a new node is inserted into the network at the position of $\boldsymbol{x}$, and connected to s_1 with a new edge.

After learning λ data objects, the algorithm enters a denoising procedure. Here λ is a predefined parameter. The denoising procedure is controlled by two parameters c_1 and c_2. A node i is viewed as noise if one of the following conditions is met:

(a)$|N_i| = 0$

(b)$|N_i| = 1$ and $m_i < \frac{c_1}{|N|}\sum_{j\in N} m_j$

(c)$|N_i| = 2$ and $m_i < \frac{c_2}{|N|}\sum_{j\in N} m_j$

Noise nodes and their connections are removed from the network.

By adjusting λ, c_1 and c_2, user can control the frequency and intensity of denoising, which will affect the speed of network growth. We suggest setting $\lambda \in [50, 200]$, $c_1 \in [0.2, 1]$ and $c_2 \in [0, 0.1]$, with respect to the characteristic of training data. If a large amount of noises exist in the input data, user could try setting a smaller λ and larger c_1, c_2.

The online training step repeats until all data objects in the stream have been learned or a clustering request is received. The algorithm of online training of DenSOINN is given in Algorithm 2.

C. Density based clustering analysis

A clustering of nodes starts after the online training step. Each node of DenSOINN can be viewed as a small cluster of its nearby input data objects. These clusters are hyper spherical, like other competitive neural networks and K-Means. The target of node clustering is to combine these small clusters into arbitrary shaped clusters.

We adopt a density-based method by combining the "density peak" strategy proposed in [22] and online estimation of node density in E-SOINN. "density peak" strategy can be described as:"cluster centers are surrounded by neighbors with lower local density and they are at a relatively large distance from any points with a higher local density." [22]

In DenSOINN, node density is definable using the local accumulated number of data objects: the more data objects are near the node, the higher its node density is. It is a simple and nature way to use "times of being a winner" as definition of node density. However, there will be numerous nodes lies in high-density area. As a result, in high-density area, the winning times of a node will not be considerably higher than that in the low-density area. Therefore the distance between nearby nodes should be taken into consideration. For a node i, first calculate its mean distance from its neighbors:

$$d_i = \frac{1}{|N_i|}\sum_{j\in N_i} d(\boldsymbol{w_i}, \boldsymbol{w_j}) \tag{10}$$

Algorithm 2 Online Network Training of DenSOINN

Input:
- X:input data set, $X \subseteq R^n$;
- δ:connection age threshold;
- λ:frequency of denoising procedure;
- c_1, c_2:parameters for denoising;

Output:
- N:set of nodes;
- C:set of connections;

1: initialize the network with $N = \phi$, $C = \phi$;
2: **for** each data object $\boldsymbol{x} \in X$ **do**
3: **if** $|N| < 2$ **then**
4: create node r: $m_r = 1$, $\boldsymbol{w_r} = \boldsymbol{x}$, $N = N \bigcup \{r\}$;
5: goto step 2;
6: **end if**
7: adjust distance coefficients by Algorithm 1;
8: find winner s_1 and runner-up s_2 by (6) and (7);
9: calculate activation threshold t_{s_1} and t_{s_2} by (4) and (5);
10: **if** $d(\boldsymbol{x}, \boldsymbol{w_{s_1}}) > t_{s_1}$ or $d(\boldsymbol{x}, \boldsymbol{w_{s_2}}) > t_{s_2}$ **then**
11: create node r:$m_r = 1$, $\boldsymbol{w_r} = \boldsymbol{x}$, $N = N \bigcup \{r\}$;
12: build connection (r, s_1): $age_{r,s_1} = 0$, $C = C \bigcup \{(r, s_1)\}$;
13: Goto step 2;
14: **else**
15: $m_{s_1} = m_{s_1} + 1$;
16: $\boldsymbol{w_{s_1}} = \boldsymbol{w_{s_1}} + \frac{\boldsymbol{x} - \boldsymbol{w_{s_1}}}{m_{s_1}}$;
17: **if** (s_1, s_2) does not exist **then**
18: build connection (s_1, s_2): $age_{s_1,s_2} = 0$, $C = C \bigcup \{(s_1, s_2)\}$;
19: **else**
20: $age_{s_1,s_2} = 0$;
21: **end if**
22: **for** each neighbor i of s_1, i.e. $(s_1, i) \in C$ **do**
23: $age_{s_1,i} = age_{s_1,i} + 1$;
24: **if** $age_{s_1,i} > \delta$ **then**
25: $C = C - \{(s_1, i)\}$;
26: **end if**
27: **end for**
28: **end if**
29: **if** the number of learning rounds is an integer multiple of λ **then**
30: $m_{mean} = \frac{\sum_{j\in N} m_j}{|N|}$;
31: **for** each node $i \in N$ **do**
32: **if** $|N_i| = 0$ OR ($|N_i| = 1$ AND $m_i < c_1 m_{mean}$) OR ($|N_i| = 2$ AND $m_i < c_2 m_{mean}$) **then**
33: remove i from N and all its connections from C;
34: **end if**
35: **end for**
36: **end if**
37: **end for**
38: **return** N, C

Then the density of node i is calculated as following:

$$p_i = \frac{m_i}{1 + d_i^2} \tag{11}$$

We first cut the network $G = (N, C)$ into connected components, then find cluster centers on each subgraph. For a node i in a connected component $SG = (SN, SC)$, find the nearest node in SG that has a higher density than i, denote the node as s_i:

$$s_i = arg \min_{j \in SN \& p_i < p_j} d(\boldsymbol{w_i}, \boldsymbol{w_j}) \tag{12}$$

Then we assign i a "peak score":

$$v_i = p_i d(\boldsymbol{w_i}, \boldsymbol{w_{s_i}}) \tag{13}$$

Node i is chosen as a cluster center if it satisfies the following condition:

$$v_i \geq \frac{\alpha}{|SN|} \sum_{j \in SN} v_j \tag{14}$$

Here α is parameter defined by user. Otherwise i is grouped into the same cluster with s_i. To assure s_i is already grouped into a cluster when i is processed, SN is sorted in descend order of node density, thus s_i is always processed before i.

The method is described in Algorithm 3.

Algorithm 3 Clustering by node density

Input:
-N:set of nodes;
-C:set of connections;
-α:parameter for cluster center selection;
Output:
-$\boldsymbol{l}$:cluster labels of nodes;

for each node $i \in N$ **do**
  calculate node density p_i by (10) and (11);
end for
separate the graph $G = (N, C)$ into connected components $\{SG_1, SG_2, ...\}$;
$c = 1$;
for each connected component $SG_t = (SN_t, SC_t)$ **do**
  sort nodes in SN_t in descend order of node density;
  for each node $i \in SN_t$ **do**
    find nearest node with a higher density s_i by (12);
    calculate "peak score" v_i by (13);
  end for
  $v_{mean} = \frac{1}{|SN_t|} \sum_{i \in SN_t} v_i$;
  for each node $i \in SN_t$ **do**
    if $v_i \geq \alpha v_{mean}$ **then**
      i is chosen as a cluster center: $l_i = c, c = c + 1$;
    else
      group i into the same cluster with s_i:$l_i = l_{s_i}$;
    end if
  end for
end for
$\boldsymbol{l} = (l_1, ..., l_{|N|})^T$;
return $\boldsymbol{l}$

IV. Experiments

In this section, we present experimental results of our proposed algorithm on some artificial and real-world data sets. All experiments are performed on MATLAB platform. We compared our algorithm with another clustering neural network Adjusted SOINN, and widely used stream clustering algorithms StreamKM++ and DenStream.

A. Data sets and Validation Criteria

We performed our experiments on both artificial and real data sets to evaluate the clustering quality of our algorithm. The artificial data sets are 2-dimensional data sets in order that the results can be presented visually. First one is a simple data set combined two groups of data objects that satisfy 2-D Gaussian distribution, with different scales on X-axis and Y-axis. This data set is used to demonstrate the effect of self-adaptive distance metric. Another artificial data set is four groups of data objects sampled from four overlapped gaussian distributions. Most real-world data sets are available on UCI Machine Learning Repository [16], including Wine, Image Segmentation(Segment), Pendigits and Human Activity Recognition Using Smartphones(HAR) [5]. We also used another real world data set Usps [14] in our experiment. The details of the data sets are listed in Table I.

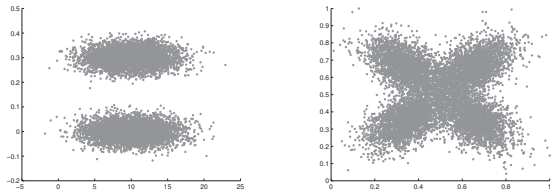
TABLE I
Data sets used in experiments

Data set	Classes	Instances	Attributes
Artificial I	2	10000	2
Artificial II	4	10000	2
Wine	3	178	14
Segment	7	2310	19
Pendigits	10	10992	16
HAR	6	10299	561
Usps	10	9298	256

We adopt two measures to evaluate result of clustering: Normalized Mutual Information (NMI) [4] and Adjusted Rand index (ARI) [13]. An algorithm is evaluated by calculating NMI and ARI between its result and the ground truth clusters in the data set. The value ranges of NMI and ARI are [0, 1] and [-1, 1] respectively. Better clustering performance results in higher NMI and ARI. We also reported the number of clusters found by each algorithm. Although this number is not a generally used evaluation measure, we consider that for an algorithm that could determine the number of clusters by itself, whether the number is reasonable can be used to judge its performance. Some widely used measures such as cluster purity tend to be high when the number of clusters is large, therefore we avoided using these measures in our experiments.

B. Experiments on artificial data sets

In this section, we show visual results of our experiments on 2-Dimensional artificial data sets. In the resulting figures, we use different colors to mark nodes grouped into different clusters. The first experiment on data set Artificial I is aimed at validating the effect of the self-adaptive distance metric



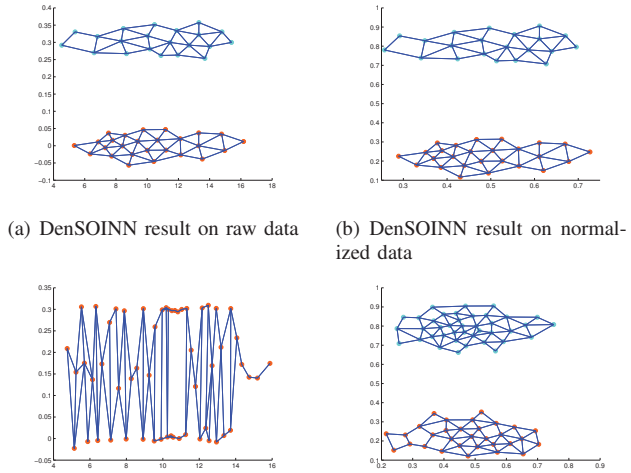
(a) Artificial I: 2 unnormalized Gaussian distributions

(b) Artificial II: 4 overlapped Gaussian distributions

Fig. 2. Artificial data sets

described in section III.A. The Artificial I data set consists of 10000 data objects sampled equally from two gaussian distributions as shown in Fig.2(a). In order to simulate unnormalized real world data set, value ranges are different on X axis and Y axis.

We compared DenSOINN with Adjusted SOINN on both raw data and data normalized by min-max normalization. Parameters of DenSOINN is set as $\delta = 50, \lambda = 100, c_1 = 1, c_2 = 0.05, \alpha = 5$; Adjusted SOINN used the default parameters suggested in [12]. The result is shown in Fig.3. In these figures, node and connections of neural networks are illustrated by colored points and lines between them. Same colored points denote nodes in the same cluster. We can see DenSOINN works fine in both environments and the results are approximately the same, while Adjusted SOINN fails to separate the clusters on the raw data. It indicates that the self-adaptive distance metric used in DenSOINN can work on raw data just like Euclidean distance works on normalized data.



(a) DenSOINN result on raw data

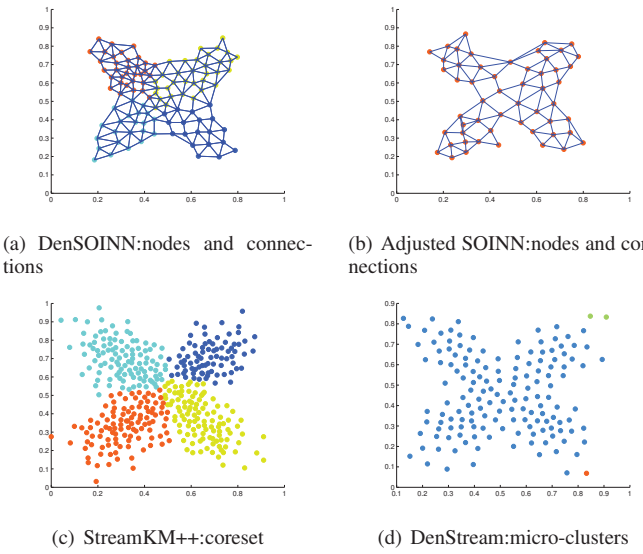
(b) DenSOINN result on normalized data

(c) Adjusted SOINN result on raw data

(d) Adjusted SOINN result on normalized data

Fig. 3. Results on Artificial I

The second experiment was performed on the data set Artificial II shown in Figure 2(b). This data set consists of 10000 data objects sampled equally from four gaussian distributions, which are overlapped near the central area.



(a) DenSOINN:nodes and connections

(b) Adjusted SOINN:nodes and connections

(c) StreamKM++:coreset

(d) DenStream:micro-clusters

Fig. 4. Results on Artificial II

In this experiment, we compared DenSOINN with Adjusted SOINN, DenStream and StreamKM++. The parameters of comparative algorithms are set as suggested in relevant paper. The parameter of DenSOINN is set as: $\delta = 25$,$\lambda = 100$,$c_1 = 1$,$c_2 = 0.05$,$\alpha = 5$. Figure 4 illustrates clusters generated by these algorithms. The points in these graphs illustrate nodes of DenSOINN and Adjusted SOINN, coreset of StreamKM++, micro-clusters of DenStream respectively. As illustrated in these figures, DenSOINN could separate the overlapped clusters but some nodes in overlapped area are misclassified. StreamKM++ can get also good results if the number of clusters k is appropriately set. Adjusted SOINN and DenStream failed to correctly separate clusters.

C. Experiments on real world data sets

In this section, we present experimental results on real-world data sets. In each experiment, each algorithm were repeatedly run 10 times on a given data set with random input sequence. Specifically, raw and normalized version of Wine and Segment were used as different data sets. Other data sets were each used only once because all their attributes have exactly the same range of value, therefore data normalization has no effect to clustering results. In these experiments, we compared DenSOINN with Adjusted SOINN, StreamKM++ and DenStream. The parameters of comparative algorithms were set as suggested in relevant paper. The parameter of DenSOINN was set as: $\delta = 25$,$\lambda = 200$,$c_1 = 0.5$,$c_2 = 0.05$,$\alpha = 5$.

The average value and standard deviation of the results are reported in Table II. Comparing results on raw and normalized versions of Wine and Segment, it is noticeable that DenSOINN got exactly the same results in both experiments, while other algorithms performed significantly worse on unnormalized data sets. The reason is that Wine and Segments have some attributes whose range of value far exceed other attributes, and

TABLE II
EXPERIMENTAL RESULTS ON REAL-WORLD DATA SETS

Data set		DenSOINN	Adjusted SOINN	StreamKM++	DenStream
Wine(Normalized)	Clusters	2.5000 ± 0.5270	1.5000 ± 0.8498	3^*	4.8000 ± 1.3166
	NMI	0.6742 ± 0.0974	0.1429 ± 0.2968	0.6961 ± 0.1461	$\mathbf{0.7088 \pm 0.0532}$
	ARI	0.5926 ± 0.1616	0.1196 ± 0.2820	0.6606 ± 0.2121	$\mathbf{0.6676 \pm 0.0926}$
Wine(Raw)	Clusters	2.5000 ± 0.5270	4.1000 ± 1.4907	3^*	4.5000 ± 0.7071
	NMI	$\mathbf{0.6742 \pm 0.0974}$	0.2197 ± 0.1528	0.4202 ± 0.0179	0.3932 ± 0.0131
	ARI	$\mathbf{0.5926 \pm 0.1616}$	0.1617 ± 0.1648	0.3157 ± 0.0316	0.2823 ± 0.0372
Segment(Normalized)	Clusters	7.5000 ± 1.1785	4.3000 ± 2.1108	7^*	29.6000 ± 2.7568
	NMI	0.6202 ± 0.0280	0.5466 ± 0.0701	$\mathbf{0.6275 \pm 0.0288}$	0.5944 ± 0.0153
	ARI	0.4511 ± 0.0483	0.2218 ± 0.0424	$\mathbf{0.4956 \pm 0.0370}$	0.3540 ± 0.0235
Segment(Raw)	Clusters	7.5000 ± 1.1785	3.0000 ± 1.4907	7^*	22.6000 ± 1.9551
	NMI	$\mathbf{0.6202 \pm 0.0280}$	0.2845 ± 0.1354	0.4685 ± 0.0417	0.5420 ± 0.0246
	ARI	$\mathbf{0.4511 \pm 0.0483}$	0.0825 ± 0.0430	0.3157 ± 0.0399	0.2659 ± 0.0439
Pendigits	Clusters	8.5000 ± 0.9718	3.5000 ± 0.9718	10^*	107.4000 ± 6.9952
	NMI	$\mathbf{0.7269 \pm 0.0384}$	0.3802 ± 0.1139	0.6341 ± 0.0305	0.6488 ± 0.0079
	ARI	$\mathbf{0.5530 \pm 0.0598}$	0.1154 ± 0.0684	0.4868 ± 0.0502	0.3812 ± 0.0387
HAR	Clusters	6.0000 ± 1.054	5.6000 ± 2.4129	6^*	31.6000 ± 2.8752
	NMI	$\mathbf{0.6257 \pm 0.0358}$	0.6166 ± 0.0471	0.5814 ± 0.0515	0.4875 ± 0.0089
	ARI	$\mathbf{0.4921 \pm 0.0573}$	0.4201 ± 0.0656	0.4263 ± 0.0654	0.2095 ± 0.0177
Usps	Clusters	5.2000 ± 0.9189	1.8000 ± 0.4216	10^*	16.8000 ± 2.1499
	NMI	$\mathbf{0.6153 \pm 0.0457}$	0.2111 ± 0.1150	0.5443 ± 0.0655	0.1730 ± 0.0201
	ARI	0.3914 ± 0.0743	0.0528 ± 0.0294	$\mathbf{0.4384 \pm 0.0882}$	0.0709 ± 0.0171

the distance metric in DenSOINN managed to cope with this problem.

By viewing the number of clusters found by each algorithm in Tabel II, we can see that the numbers of clusters DenSOINN are closest to the numbers of classes in all the experiments. Note that the number of classes is not necessarily equal to underlying clusters, but it can still be used as a reference. Compared to the numbers of clusters reported by Adjusted SOINN, DenSOINN can better separated overlapped clusters, and is more stable in different input sequences.

V. CONCLUSION

In this paper, we proposed an new competitive neural network named Density Based Self Organizing Incremental Neural Network(DenSOINN). This method aims at finding underlying clusters in data when the data is organized in the form of a stream. DenSOINN also applied Competitive Hebbian Learning in order to learn a topological expression of input data. By adopting a self-adaptive distance metric, DenSOINN can be applied on a raw data set as if it has been normalized, which solved the data normalization problem in data stream clustering tasks. By performing online density estimation and finding density peaks, DenSOINN can separate highly overlapped clusters while finding a suitable number of arbitrary shaped clusters.

However, there are still further work left to be done. In the current version of DenSOINN, we did not theoretically prove DenSOINN achieved an optimal solution of stream clustering problem by giving a cost function of such problem. Existing cost functions such as k-means cost function are only used by centroid models that use a mean vector to represent a cluster, and the number of clusters must be preset. Moreover, optimization of most existing cost functions is NP-hard, and it is more difficult to define and optimize a cost function under streaming constraints. We will try to provide a more stable theoretical basis for DenSOINN in future.

ACKNOWLEDGMENT

This work is supported in part by the National Science Foundation of China under Grant Nos. (61375064, 61373001) and Jiangsu NSF grant (BK20131279).

REFERENCES

[1] Ackermann, Marcel R., et al. *StreamKM++: A clustering algorithm for data streams* Journal of Experimental Algorithmics (JEA) 17 (2012): 2-4.

[2] Aggarwal, Charu C., et al. *A framework for clustering evolving data streams* Proceedings of the 29th international conference on Very large data bases-Volume 29. VLDB Endowment, 2003.

[3] Aksoy, Selim, and Robert M. Haralick, *Feature normalization and likelihood-based similarity measures for image retrieval*, Pattern recognition letters 22.5 (2001), pp. 563–582.

[4] Ana, L. N. F., and Anil K. Jain, *Robust data clustering*, Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on. Vol. 2. IEEE, 2003.

[5] Anguita, Davide, et al, *A public domain dataset for human activity recognition using smartphones*, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN. 2013.

[6] Cao, Feng, et al. *Density-Based Clustering over an Evolving Data Stream with Noise* SDM. Vol. 6. 2006.

[7] Ester, Martin, et al, *A density-based algorithm for discovering clusters in large spatial databases with noise*, Kdd. Vol. 96. No. 34. 1996.

[8] Fritzke, Bernd, *A growing neural gas network learns topologies*, Advances in neural information processing systems 7 (1995), pp. 625–632.

[9] Fukunaga, Keinosuke, and Larry D. Hostetler, *The estimation of the gradient of a density function, with applications in pattern recognition*, Information Theory, IEEE Transactions on 21.1 (1975), pp. 32–40.

[10] Furao, Shen, and Osamu Hasegawa, *An incremental network for on-line unsupervised classification and topology learning*, Neural Networks 19.1 (2006), pp. 90–106.

[11] Furao, Shen, Tomotaka Ogura, and Osamu Hasegawa, *An enhanced self-organizing incremental neural network for online unsupervised learning*, Neural Networks 20.8 (2007), pp. 893–903.

[12] Furao, Shen, and Osamu Hasegawa, *A fast nearest neighbor classifier based on self-organizing incremental neural network*, Neural Networks 21.10 (2008), pp. 1537–1547.

[13] Hubert, Lawrence, and Phipps Arabie, *Comparing partitions*, Journal of classification 2.1 (1985), pp. 193–218.
[14] Hull, Jonathan J. *A database for handwritten text recognition research* Pattern Analysis and Machine Intelligence, IEEE Transactions on 16.5 (1994): 550-554.
[15] Kohonen, Teuvo, *Self-organized formation of topologically correct feature maps*, Biological cybernetics 43.1 (1982), pp. 59–69.
[16] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
[17] Lloyd, Stuart P, *Least squares quantization in PCM*, Information Theory, IEEE Transactions on 28.2 (1982), pp. 129–137.
[18] MacQueen, James, *Some methods for classification and analysis of multivariate observations*, Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. Vol. 1. No. 14. 1967.
[19] Martinetz, Thomas, *Competitive Hebbian learning rule forms perfectly topology preserving maps*, ICANN93, Springer London, 1993, pp. 427–434.
[20] Martinetz, Thomas M., Stanislav G. Berkovich, and Klaus J. Schulten, *Neural-gas' network for vector quantization and its application to time-series prediction*, Neural Networks, IEEE Transactions on 4.4 (1993), pp. 558–569.
[21] Martinetz, Thomas, and Klaus Schulten, *Topology representing networks*, Neural Networks 7.3 (1994), pp. 507–522.
[22] Rodriguez, Alex, and Alessandro Laio, *Clustering by fast search and find of density peaks*, Science 344.6191 (2014), pp. 1492–1496.
[23] Sanjoy Dasgupta, *Topics in unsupervised learning. Lecture 6: Clusteringin an online/streaming setting*, Course notes, CSE 291. Section 6.2.3. In http://www-cse.ucsd.edu/.dasgupta/291/lec6.pdf, University of California, San Diego, Spring Quarter, 2008.
[24] Silva, Jonathan A., et al. *Data stream clustering: A survey.* ACM Computing Surveys (CSUR) 46.1 (2013): 13.
[25] Zhang, Hongwei, Xiong Xiao, and Osamu Hasegawa, *A Load-Balancing Self-Organizing Incremental Neural Network*, Neural Networks and Learning Systems, IEEE Transactions on 25.6 (2014), pp. 1096–1105.
[26] Zhang, Tian, Raghu Ramakrishnan, and Miron Livny. *BIRCH: A new data clustering algorithm and its applications.* Data Mining and Knowledge Discovery 1.2 (1997): 141-182.