Inference on Distributed Data Clustering

Josenildo C. da Silva* and Matthias Klusch

German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
{jcsilva,klusch}@dfki.de

Abstract. In this paper we address confidentiality issues in distributed data clustering, particularly the inference problem. We present a measure of inference risk as a function of reconstruction precision and number of colluders in a distributed data mining group. We also present KDEC-S, which is a distributed clustering algorithm designed to provide mining results while preserving confidentiality of original data. The underlying idea of our algorithm is to use an approximation of density estimation such that it is not possible to reconstruct the original data with better probability than some given level.

1 Introduction

Data Clustering is a descriptive data mining task aiming to partition a data set into groups such that data objects in one group are similar to each other and are so different as possible from those in other groups. In distributed data clustering (DDC) the data objects are distributed among several sites. The traditional solution to (homogeneous) DDC problem is to collect all the distributed data sets into one centralized repository where the clustering of their union is computed and transmitted back to the sites.

This approach, however, may be impractical due to constraints on network bandwidth or secrecy issues, when the sites are not allowed to share data due to legal issues or because it is against some local security policy. Examples of such confidential data include medical information and marketing secrets. The main problem is that confidential information may be reconstructed even if it is not explicitly exchanged among the peers. This problem, known as *inference problem*, was first studied in statistical data bases and more recently has attracted the attention of the data mining community [7].

In this paper we address the problem of homogeneous DDC considering confidentiality issues, particularly the inference problem. Informally, the problem is to find clusters using distributed set of data ensuring that, at the end of the computation, each peer only knows his own dataset and the resulting cluster mapping. Our main objective is to propose a measure of how confidential one algorithm is, i.e. how vulnerable it is to inference attacks. Additionally, we present

* This work is partially supported by CAPES (Coord. de Aperfeicoamento do Pessoal de Nivel Superior) of Ministry for Education of Brazil, under Grant No. 0791/024.

a distributed algorithm for confidential DDC with an analysis of its confidentiality level.

The remaining of this paper is organized as follows. In section 2 we present our definitions of confidentiality and inference risk. In section 3 we present our algorithm. Related work are presented in section 4. Conclusions and remarks are presented in section 5.

2 Confidentiality in Distributed Data Clustering

We define the problem of confidential distributed data clustering as follows.

Definition 1. *Let $\mathcal{L} = \{L^j | 1 \leq j \leq P\}$ be a group of peers sites, each of them with a local set of data objects $D^j = \{\mathbf{x}_i \mid i = 1, \ldots, N\} \subset \mathbb{R}^n$, with $\mathbf{x}_i^{(d)}$ denoting the d-th component of $\mathbf{x}_i$. Let $\mathcal{A}$ be some DDC algorithm executed by the members of $\mathcal{L}$. We say that $\mathcal{A}$ is a Confidential DDC algorithm if the following holds: (a) $\mathcal{A}$ produce correct results (b) at the end of the computation L^j knows only the cluster mapping and its own data set D^j, with $1 \leq j \leq P$.*

Our objective in this paper is to analyze how much a given distributed clustering algorithm copes with the second requirement.

2.1 Confidentiality Measure

Our starting point is the definition of a confidentiality measure. One way to measure how much confidentiality an algorithm preserves, is to ask how close one attacker can get from the original data objects. In the following we define the notion of confidentiality of data w.r.t. reconstruction. Considering multidimensional data objects, we have to look at each dimension at time.

Definition 2. *Let $\mathcal{L}$ be a group of peer as in definition 1. Let $\mathcal{A}$ be some DDC algorithm executed by the members of $\mathcal{L}$. Denote by $R^k \subset \mathbb{R}^n$ a set of reconstructed data objects owned by some malicious peer L^k after the computation of the data mining algorithm, such that each $\mathbf{r}_i$ is a reconstructed version of $\mathbf{x}_i$. We define the confidence level of $\mathcal{A}$ with respect to dimension d as:*

$$Conf_{\mathcal{A}}^{(d)} = \min\{|\mathbf{x}_i^{(d)} - \mathbf{r}_i^{(d)}| : \mathbf{x}_i \in D^j, \mathbf{r}_i \in R^k, 1 \leq i \leq |D^j|\}$$

Definition 3. *We define the confidentiality level associated to some algorithm $\mathcal{A}$, as:*

$$Conf_{\mathcal{A}} = \min\{Conf_{\mathcal{A}}^{(d)} \mid 1 < d < n\}$$

Roughly speaking, our confidentiality measure, indicates the precision with which a data object $\mathbf{x}_i$ can be reconstructed.

In a distributed algorithm we have to consider the possibility of two of more peers forming a collusion group to disclose information owned by others. The next definition extends the confidentiality level to include this case.

Definition 4. *Let $\mathcal{A}$ be a distributed data mining algorithm. We define the function $Conf_{\mathcal{A}} : \mathbb{N} \to \mathbb{R}_+ \cup \{0\}$, representing $Conf_{\mathcal{A}}$ when c peers collude.*

Definition 5 (Inference Risk Level). *Let $\mathcal{A}$ be a DDC algorithm being executed by a group $\mathcal{L}$ with p peers, where c peers in $\mathcal{L}$ forms a collusion group. Then we define:*

$$\mathrm{IRL}_{\mathcal{A}}(c) = 2^{(-Conf_{\mathcal{A}}(c))}$$

One can easily verify that $\mathrm{IRL}_{\mathcal{A}}(c) \to 0$ when $Conf_{\mathcal{A}}(c) \to \infty$ and $\mathrm{IRL}_{\mathcal{A}}(c) \to 1$ when $Conf_{\mathcal{A}}(c) \to 0$. In other words, the better the reconstruction, the higher the risk. Therefore, we can capture the informal concepts of *insecure* algorithm ($\mathrm{IRL}_{\mathcal{A}} = 1$) and *secure* ($\mathrm{IRL}_{\mathcal{A}} = 0$) as well.

2.2 Confidential Density Estimation

Density-based clustering is a popular technique, which reduces the search for clusters to the search for dense regions. This is accomplished by estimating a so-called probability density function from which the given data set is assumed to have arisen. An important family of method is known as *kernel estimator* [8]. Let $D = \{\mathbf{x}_i \mid i = 1, \ldots, N\} \subset \mathbb{R}^n$ represent a set of data objects. Let K be a real-valuated, non-negative, non-increasing function on $\mathbb{R}$ with finite integral over $\mathbb{R}$. A *kernel-based density estimate* $\hat{\varphi}_{K,h}[S](\cdot) : \mathbb{R}^n \to \mathbb{R}_+$ is defined as follows:

$$\hat{\varphi}_{K,h}[D](\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} K\left(\frac{d(\mathbf{x}, \mathbf{x}_i)}{h}\right) \tag{1}$$

In [8] is presented an algorithm to find center-defined clusters using a density estimate. In [10] is presented the KDEC schema, a density-based algorithm for DDC. In density-based DDC each peer contributes to the mining task with a local density estimate of the local data set and not with data (neither original nor randomized). As shown in [4], in some cases, knowing the inverse of kernel function implies in the reconstruction of original (confidential) data. Therefore, we look for a more confidential way to build the density estimate, i.e. one which doesn't allow reconstruction of data.

Definition 6. *Let $f : \mathbb{R}_+ \cup \{0\} \to \mathbb{R}_+$ be a decreasing function. Let $\tau \in \mathbb{R}$ be a sampling rate and let $z \in \mathbb{Z}_+$ be an index. Denote by $\mathbf{v} \in \mathbb{R}^n$ a vector of iso-levels*[1] *of f, whose each component $v^{(i)}$, $i = 1, \ldots, n$, is built as follow:*

$$v^{(i)} = f(z \cdot \tau), \text{ if } f(z \cdot \tau) < f([z-1] \cdot \tau)$$

Moreover $0 < v^{(0)} < v^{(1)} \ldots < v^{(n)}$.

[1] One can understand $\mathbf{v}$ as iso-lines used to contour plots

Definition 7. *Let $f : \mathbb{R}_+ \cup \{0\} \to \mathbb{R}$ be a decreasing function. Let $\mathbf{v}$ be a vector of iso-levels of f. Then we define the function $\psi_{f,\mathbf{v}}$ as:*

$$\psi_{f,\mathbf{v}}(x) = \begin{cases} 0, \textit{ if } f(x) < v^{(0)} \\ v^{(i)}, \textit{ if } v^{(i)} \leq f(x) < v^{(i+1)} \\ v^{(n)}, \textit{ if } v^{(n)} \leq f(x) \end{cases} \tag{2}$$

Definitions 6 and 7 together define a step function based on the shape of some given function f. Figure 1 shows an example of $\psi_{f,\mathbf{v}}$ applied to a Gaussian[2] function with $\mu = 0$ and $\sigma = 2$, using four iso-levels.

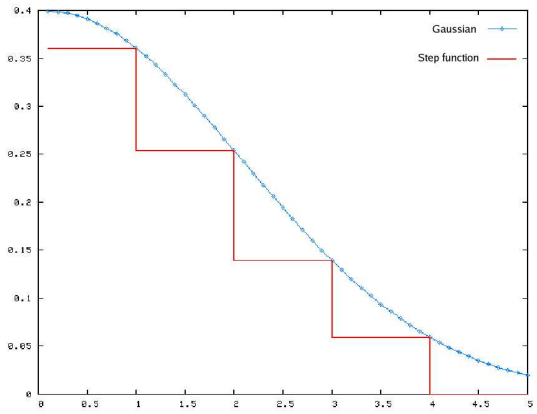

Fig. 1. $\psi_{f,\mathbf{v}}$ of the Gaussian function.

Lemma 1. *Let $\tau \in \mathbb{R}$ denote a sampling rate, and $z \in \mathbb{Z}_+$ be an index. Define $f_1 : \mathbb{R}_+ \to \mathbb{R}_+$, a decreasing function and $\mathbf{v}$, a vector of iso-levels. If we define a function $f_2 = f_1(x - k)$, then $\forall k \in (0, \tau), \forall z \in \mathbb{Z}_+$ we will have $\psi_{f_2,\mathbf{v}}(z \cdot \tau) = \psi_{f_1,\mathbf{v}}(z \cdot \tau)$.*

Proof. For $k = 0$ we get $f_2(x) = f_1(x - 0)$ and its is trivial to see that the assertion holds. For $0 < k < \tau$ we have $f_2 = f_1(x-k)$. Without loss of generality, let $z > 0$ be some integer. So, $f_2(z \cdot \tau) = f_1(z \cdot \tau - k) = f_1([z - k/\tau] \cdot \tau)$. If $f_1([z - 1] \cdot \tau) = a > f_1(z \cdot \tau) = b$ then we have $\psi_{f_1,\mathbf{v}}(z \cdot \tau) = a$. Since $z - 1 < z - k/\tau < z$, and since f_1 is decreasing, $f_1([z - 1] \cdot \tau) = a > f_1([z - k/\tau] \cdot \tau) > b = f_1(z \cdot \tau)$. By the definition 7 we can write $\psi_{f_1,\mathbf{v}}([z - k/\tau] \cdot \tau) = b = \psi_{f_1,\mathbf{v}}(z \cdot \tau)$

This lemma means that we have some ambiguity associated with the function $\psi_{f,\mathbf{v}}$, given some τ and $\mathbf{v}$, since two functions will issue the same values iso-levels around the points close than τ.

[2] Gaussian function is defined by $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$

With this definition we return to our problem of uncertainty of local density. We will substitute a kernel K by $\psi_{K,\mathbf{v}}$, Given a sample rate τ. According with the lemma 1, we should expect to localize the points in a interval not smaller than $|(0,\tau)|$, i.e. the confidentiality will be $Conf_{\mathcal{A}} \geq \tau$. So, we will compute a rough approximation of the local density estimate using:

$$\tilde{\varphi}[D^j](x) = \begin{cases} \sum_{x_i \in N_x} \psi_{K,\mathbf{v}}(\frac{d(x,x_i)}{h}) & \text{, if } (x \mod \tau) = 0 \\ 0 & \text{, otherwise.} \end{cases} \tag{3}$$

where N_x denotes the neighborhood of x.

Since $\psi_{K,\mathbf{v}}$ is a non-increasing function, we can use it as a kernel function. The global approximation can be computed by: $\tilde{\varphi}[D](x) = \sum_{j=1}^{p} \tilde{\varphi}[D^j](x)$

3 The KDEC-S Algorithm

KDEC-S is an extension of the KDEC Schema, which is a recent approach for kernel-based distributed clustering [10]. In KDEC each site transmits the local density estimate to a helper site, which builds a global density estimate and sends it back to the peers. Using the global density estimate the sites can execute locally a density-based clustering algorithm. KDEC-S works in a similar way, but replaces the original estimation by an approximated value. The aim is to preserve data confidentiality while maintaining enough information to guide the clustering process.

3.1 Basic definitions

Definition 8. *Given two vectors $\boldsymbol{z}_{low}, \boldsymbol{z}_{high} \in \mathbb{Z}^n$, which differ in all coordinates (called the sampling corners), we define a grid G as the filled-in cube in $\mathbb{Z}^n$ defined by $\boldsymbol{z}_{low}, \boldsymbol{z}_{high}$. Moreover for all $z \in G$, define $n_z \in \mathbb{N}$ as a unique index for z (the index code of z). Assume that z_{low} has index code zero.*

Definition 9. *Let G be a grid defined with some $\tau \in \mathbb{R}^n$. We define a sampling $\mathcal{S}^j$ of $\tilde{\varphi}[D^j]$ given a grid G, as:*

$$\mathcal{S}^j = \left\{ \tilde{\varphi}_z^j \mid \forall z \in G,\ \tilde{\varphi}_z^j > 0 \right\}$$

where $\tilde{\varphi}_z^j = \tilde{\varphi}[D^j](z \cdot \tau)$. Similarly, the global sampling set will be defined as: $\mathcal{S} = \{\tilde{\varphi}_z \mid \forall z \in G,\ \tilde{\varphi}_z > 0\}$

Definition 10 (Cluster-guide). *A cluster guide $CG_{i,\theta}$ is a set of index codes representing the grid points forming a region with density above some threshold θ:*

$$CG_{i,\theta} = \{n_z \mid \tilde{\varphi}_z \geq \theta\}$$

such that $\forall n_{z_1}, n_{z_2} \in CG_{i,\theta} : z_1$ and z_2 are grid neighbors and $\bigcap_{i=1}^{C} CG_{i,\theta} = \emptyset$. A complete cluster-guide is defined by: $CG_\theta = \{CG_{i,\theta} |\ i = 1, \ldots, C\}$ where C is the number of clusters found using a given θ.

A cluster-guide $CG_{i,\theta}$ can be viewed as a contour defining the cluster shape at level θ (a iso-line), but in fact it shows only the internal grid points and not the true border of the cluster, which should be determined using the local data set.

3.2 Detailed description

Algorithm 1 Local Peer

Input: D^j (local data set), $\mathcal{L}$ (list of peers), $\mathcal{H}$ (Helper);
Output: $clusterMap$;

```
1: negotiate(L, K, h, G, θ);
2: lde ← estimate(K, h, D^j, G, δ);
3: S^j ← buildSamplingSets(lde, G, θ, v);
4: send(H, S^j);
5: CG_θ ← request(H, θ);
6: clusterMap ← cluster(CG_θ, D^j, G);
7: return clusterMap

8: function cluster(CG_θ, D^j, G)
9:     for each x ∈ D^j do
10:        z ← nearestGridPoint(x, G);
11:        if n_z ∈ CG_{i,θ} then
12:            clusterMap(x) ← i;
13:        end if
14:    end for
15:    return clusterMap;
16: end function
```

Our algorithm has two parts: Local Peer and Helper. The local peer part of our distributed algorithm is density-based, since this was shown to be a more general approach to clustering [8].

Local Peer. The first step is the function negotiate(), which succeeds only if an agreement on the parameters is reached. Note that the helper doesn't take part on this phase. In the second step each local peer compute its local density estimate $\tilde{\varphi}[D^j](z \cdot \tau)$ for each $z \cdot \tau$, with $z \in G$. Using the definition 9 each local peer builds its local sampling set and sends it to the helper. The clustering step (line 6 in algorithm 1) is performed as a lookup in the cluster-guide CG_θ. The function cluster() shows the details of the clustering step. The data object $\mathbf{x} \in D^j$ will be assigned to the cluster i, the cluster label of the nearest grid point z, if $n_z \in CG_{i,\theta}$.

Helper. Given a θ, the helper sums up all samples sets and uses definition 10 to construct the cluster-guides CG_θ. Function buildClusterGuides() in algorithm 2 shows the details of this step.

Algorithm 2 Helper

1: $S^j \leftarrow$receive($\mathcal{L}$);
2: $\hat{\varphi}_z[D^j] = recover(S^j)$;
3: $\hat{\varphi}_z \leftarrow \sum \hat{\varphi}_z[D^j]$;
4: $CG_\theta \leftarrow$ buildClusterGuides($\hat{\varphi}_z, \theta$);
5: send($\mathcal{L}$, CG_θ);

6: **function** buildClusterGuides($\hat{\varphi}_z, \theta$)
7: $cg \leftarrow \{n_z | \hat{\varphi}_z > \theta\}$;
8: $n \in cg$;
9: $CG_{i,\theta} \leftarrow \{n\}$;
10: $i \leftarrow 0$;
11: **for** each $n \in cg$ **do**
12: **if** $\exists a((a \in neighbors(n)) \wedge (a \in cg))$ **then**
13: $CG_{i,\theta} \leftarrow \{n, a\} \cup CG_{i,\theta}$;
14: **else**
15: i++;
16: $CG_{i,\theta} \leftarrow \{n\}$;
17: **end if**
18: $cg \leftarrow cg \setminus CG_{i,\theta}$;
19: **end for**
20: $CG_\theta \leftarrow \{CG_{i,\theta} | i = 1, \ldots, C\}$;
21: **return** CG_θ
22: **end function**

3.3 Performance Analysis

Time. At the local site our algorithm has time complexity $O(|G|M^j + log(C)|D^j|)$, where $|G|$ is the grid size, M^j is the average size of the neighborhood, C is the number of clusters and D^j is the set of points owned by peer L^j. The first lines have complexity $O(|G|M^j)$, since the algorithm compute the density for each point z in the grid G using the subset of points in D^j which are neighbors from z, with average size M^j. Line 4 has complexity determined by the size of sampling set S^j, which is a subset of G, i.e., its complexity is $O(|G|)$. Line 5 has complexity $O(C)$. The last step (line 6) has to visit each point in D^j and for each point it has to decide its label by searching the corresponding index code in one of the cluster-guides. There are C cluster guides. Assuming the look-up time for a given cluster to be $log(C)$ we can say that $O(log(C)|D^j|)$ is the complexity of the last step.

Time complexity at the helper (algorithm 2) is mainly determined by the size of the total sampling set. The helper will receive from p peers at most $|G|$ sampling points. The local peer has to reconstruct and sum them up (lines 2 and 3), what takes in the worst case $O(p|G|)$ steps. Thus, the process of building the cluster-guides (line 4) will take $O(|G|)$ steps in worst case.

Communication. Each site will have at most $|S^j| < |G|$ sampling points (index-codes) to send to the helper site. The helper site has at most $|G|$ index-codes to inform back to local sites, but this size would be reduced if some com-

pression technique is used. Moreover, our algorithm uses few rounds of messages. Each site will send one message informing the local sampling $\mathcal{S}^j$ set to the helper and one (or more subsequent) message(s) requesting a cluster-guide with some desired θ. The helper will send a message informing the cluster-guides on demand.

3.4 Security Analysis

We will use two scenarios to analyze the inference risk level of KDEC-S (denoted $\mathrm{IRL}_{\text{KDEC-S}}$). First scenario we assume that the malicious peers doesn't form collusion group, i.e. $c = 1$, and the second scenario we assume that they can form collusion group, i.e., $c \geq 2$.

Lemma 2. *Let $\mathcal{L}$ be a mining group formed by $p > 2$ peers, one of them being the helper, and $c < p$ malicious peers form a collusion group in $\mathcal{L}$. Let $\tau \in \mathbb{R}$ be a sampling rate. We claim that $\mathrm{IRL}_{\text{KDEC-S}}(c) \leq 2^{-\tau}$ for all $c > 0$.*

Proof. Assume that $c = 1$, and that each peer has only its local data set and the cluster-guides he gets from the helper. The cluster-guides, which are produced by the helper, contains only code-index representing grid points where the threshold θ is reached. This is not enough to reconstruct the original global estimation. The Helper has all sampling points from all peers, but it has neither information on the kernel nor on sampling parameters. Hence, the attackers can not use the inverse of Kernel function to reconstruct the data. The best precision of reconstruction has to be based on the cluster guides. So, one attacker may use the width of the clusters in each dimension as the best reconstruction precision. This lead to $Conf_{\text{KDEC-S}}(1) = a\tau$, with $a \in \mathbb{N}$, since each cluster will have at least a points spaced by τ in each dimension. Hence, if $c = 1$ then $\mathrm{IRL}_{\text{KDEC-S}}(c) = 2^{-a\tau} \leq 2^{-\tau}$.

Assume $c \geq 2$. Clearly, any collusion group with at least two peers, including the helper, will produce a better result than a collusion which doesn't include the helper, since the helper can send to the colluders the original sampling sets from each peer. However, each sampling set $\mathcal{S}^j$ was formed based on the $\tilde{\varphi}[D^j]$ (cf. eq. (3)). Using lemma 1 we expect to have $Conf_{\text{KDEC-S}}(c) = \tau$. With more colluders, say $c = 3$, one of them being the helper, there are no new information which could improve the reconstruction. Therefore, $\mathrm{IRL}_{\text{KDEC-S}}(c) \leq 2^{-\tau}$, for all $c > 0$.

3.5 Comparison with KDEC

KDEC Scheme exploit statistical density estimation and information sampling to minimize communications cost among sites. Some of possibilities of inference attacks in KDEC were shown in[4]. Here we analyze it using our definition of inference risk.

Lemma 3. *Let $\tau \in \mathbb{R}$ be a sampling rate. Then $\mathrm{IRL}_{KDEC}(c) > 2^{-\tau}$, forall $c > 0$.*

Proof. Since KDEC uses $y = \hat{\varphi}(\mathbf{x})$ it can be used by a malicious peer inside the group to compute the distance $d = K^{-1}(y)h$, and consequently the true $\mathbf{x}^*$ with $\mathbf{x}^* = \mathbf{x} + d$. Errors in this method can arise due machine precision, but they are still much smaller than τ, which in KDEC is suggested to be $h/2$. Actually, this error is expected to be very small, since it is caused by machine precision. We remark that these results can be reached by one malicious peer alone, i.e. $Conf_{KDEC}(1) \ll \tau$. With collusion group this reconstruction may be more accurate. Therefore, $Conf_{KDEC}(c) \ll \tau$ for $c > 0$. Hence, $\mathrm{IRL}_{KDEC}(c) > 2^{-\tau}$, for all $c > 0$.

Theorem 1. *Let $\tau \in \mathbb{R}$ be a sampling rate parameter. Using the same τ we have $\mathrm{IRL}_{\text{KDEC-S}}(c) < \mathrm{IRL}_{\text{KDEC}}(c)$, for all $c > 0$.*

Proof. Using lemmas 2 and 3 we verify that the assertion holds.

4 Related Work

The question of how to protect confidential information from unauthorized disclosure has stimulated much research in the data base community. This problem, known as *the inference problem*, was first studied in statistical databases and secure multi-level databases and more recently in data mining [7].

Other works in privacy preserving data mining uses secure multi-party computation (SMC) [11, 12, 9], *sanitization* [3, 5] and *data randomization* [2, 13]. Some privacy measures were proposed in [6] and [1] to the case where the mining algorithm uses randomized data. The idea of randomization seems to be promising but in a distributed set the reconstruction of local probabilities densities would lead to errors in the global density, what would lead to erroneous clustering results.

5 Conclusions

Our contribution can be summarized as: a definition of inference levels for DDM and a distributed algorithm for clustering which is inference-proof at certain level. Our definition of confidentiality and inference levels make little assumptions, what allow comparison of a broad range of data mining algorithms with respect to the risk of data reconstruction, and consequently permit us to classify them in different security classes. On the other hand, this levels are currently defined just to distributed data clustering and doesn't include (up to date) the notion of discovery of data ownership in a mining group.

KDEC-S is based on a modified way of computing density estimation such that it is not possible to reconstruct the original data with better probability than some given level. Results of our analysis using our inference risk level showed that our algorithm presents better improved security level w.r.t. inference attacks to kernel density estimate, without compromising the clustering results. One can argue that KDEC-S has the disadvantage of using more parameters than KDEC.

However, KDEC-S is better noise resistance than KDEC, can find arbitrary-shape clusters (as any density-based clustering algorithm), and performs the clustering faster, since it uses a lookup table instead of hill climbing the density estimation.

As future work we plan to apply our definition of inference level to others DDM areas.

References

1. Dakshi Agrawal and Charu C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of 20th ACM Symposium on Principles of Database Systems*, pages 247–255, Santa Barbara, Califonia, May 2001.
2. Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000.
3. M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure limitation of sensitive rules. In *Proceedings of 1999 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'99)*, pages 45–52, Chicago,IL, November 1999.
4. Josenildo C. da Silva, Matthias Klusch, Stefano Lodi, and Gianluca Moro. Inference attacks in peer-to-peer homogeneous distributed data mining. In *16th European Conference on Artificial Intelligence (ECAI 04)*, Valencia, Spain, August 2004.
5. Elena Dasseni, Vassilios S. Verykios, Ahmed K. Elmagarmid, and Elisa Bertino. Hiding association rules by using confidence and support. *Lecture Notes in Computer Science*, 2137:369–??, 2001.
6. A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *In Proceedings of PODS 03.*, San Diego, California, June 9-12 2003.
7. Csilla Farkas and Sushil Jajodia. The inference problem: A survey. *ACM SIGKDD Explorations Newsletter*, 4(2):6–11, 2002.
8. Alexander Hinneburg and Daniel A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Knowledge Discovery and Data Mining*, pages 58–65, 1998.
9. Murat Kantarcioglu and Chris Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *The ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'02)*, June 2002.
10. Matthias Klusch, Stefano Lodi, and Gianluca Moro. Agent-based distributed data mining: the KDEC scheme. In Matthias Klusch, Sonia Bergamaschi, Pete Edwards, and Paolo Petta, editors, *Intelligent Information Agents: the AgentLink perspective*, volume 2586 of *Lecture Notes in Computer Science*. Springer, 2003.
11. Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. *Lecture Notes in Computer Science*, 1880:36–54, 2000.
12. Benny Pinkas. Cryptographic techniques for privacy-preserving data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):12–19, 2002.
13. Shariq J. Rizvi and Jayant R. Haritsa. Maintaining data privacy in association rule mining. In *Proceedings of the 28th VLDB – Very Large Data Base Conference*, pages 682–693, Hong Kong, China, 2002.