Inference in Distributed Data Clustering

Josenildo Costa da Silva *, Matthias Klusch

German Research Center for Artificial Intelligence (DFKI)
Deduction and Multiagents Systems
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 KDEC-S algorithm for distributed data clustering, which is shown to provide mining results while preserving confidentiality of original data. We also present a confidentiality framework with which we can state the confidentiality level of KDEC-S. The underlying idea of KDEC-S is to use an approximation of density estimation such that the original data cannot be reconstructed to a given extent.

Key words: Privacy-preserving data mining, distributed data mining, data clustering, inference problem.

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 as different as possible from those in other groups. In distributed data clustering (DDC) the data set 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 if there are constraints on network bandwidth or restrictive security policies. For instance, the sites may not be allowed to share data due to legal imposition, e.g. medical records and

* Corresponding author. Tel: +49 681 302 51 55, Fax: +49 681 302 22 35

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 [1].

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. We present a solution to this problem by means of a distributed algorithm for DDC, the KDEC-S, which is designed to preserve confidentiality of local data. In order perform the confidentiality analysis of KDEC-S we propose a confidentiality framework which address scenarios where parties colludes and use inference attacks to disclose data from its mining peers. This paper is an extended version of our work presented at the International Conference on Data Mining and Machine Learning MLDM 2005 [2].

The remaining of this paper is organized as follows. In section 2, we present our confidentiality framework. KDEC-S algorithm is presented in section 3. Experiments concerning the trade-off between confidentiality and mining results are presented in section 4. Sections 5 and 6 present related works and conclusions, respectively.

2 Confidentiality in Distributed Data Clustering

In the following sections, we define the problem of confidential distributed data clustering and present a technique to compute density estimate while preserving data confidentiality to a given extent.

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 a 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 $1 \leq j \leq P$.*

To analyze how much confidentiality a given distributed clustering algorithm manages to keep (the second requirement in the above stated problem), we have to introduce a confidentiality framework, which we discuss in the following section.

2.1 Confidentiality Measure

We start off by defining 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 with respect to data reconstruction.

Definition 2 *Let $\mathcal{L}$ be a group of peers and $\mathcal{A}$ an algorithm, as in definition 1. 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 $\mathcal{A}$, such that each $\mathbf{r}_i$ is a reconstructed version of $\mathbf{x}_i \in D^j$. We define the confidence level of $\mathcal{A}$ with respect to dimension d as:*

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

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

$$Conf_{\mathcal{A}} = \min\{Conf_{\mathcal{A}}^{(d)} \mid 1 < d < n\} \tag{2}$$

Roughly speaking, our confidentiality measure, indicates the precision with which a data object $\mathbf{x}_i$ may 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:*

$$\mathrm{IRL}_{\mathcal{A}}(c) = 2^{(-Conf_{\mathcal{A}}(c))} \tag{3}$$

It turns out 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, this definition 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* [3]. 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:

$$\hat{\varphi}_{K,h}[D](\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} K\left(\frac{d(\mathbf{x}, \mathbf{x}_i)}{h}\right) \tag{4}$$

In [4] is presented the KDEC schema, a density-based algorithm for DDC, which is based on [3]. 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 [5], 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:*

$$v^{(i)} = f(z \cdot \tau), \text{ if } f(z \cdot \tau) < f([z-1] \cdot \tau) \tag{5}$$

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

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:*

$$\psi_{f,\mathbf{v}}(x) = \begin{cases} 0, \text{ if } f(x) < v^{(0)} \\ v^{(i)}, \text{ if } v^{(i)} \leq f(x) < v^{(i+1)} \\ v^{(n)}, \text{ if } v^{(n)} \leq f(x) \end{cases} \tag{6}$$

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.

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 it is trivial to see that the assertion holds. For $0 < k < \tau$ we have $f_2 = f_1(x - k)$. Without loss of

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

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

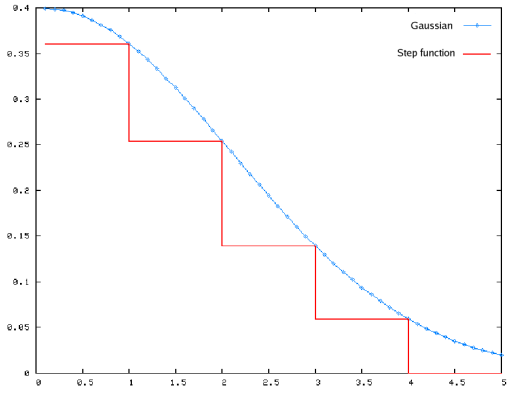

Fig. 1. $\psi_{f,\mathbf{v}}$ of the Gaussian function.

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.

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

$$\tilde{\varphi}[D^j](x) = \begin{cases} \sum_{x_i \in N_x} \psi_{K,\mathbf{v}}(\frac{d(x,x_i)}{h}) & \text{, if } (x \mod \tau) = 0 \\ 0 & \text{, otherwise.} \end{cases} \tag{7}$$

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:

$$\tilde{\varphi}[D](x) = \sum_{j=1}^{p} \tilde{\varphi}[D^j](x) \tag{8}$$

3 The KDEC-S Algorithm

KDEC-S is an extension of the KDEC scheme, which is a recent approach for kernel-based distributed clustering [4]. 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:*

$$\mathcal{S}^j = \left\{ \tilde{\varphi}_z^j \mid \forall z \in G,\ \tilde{\varphi}_z^j > 0 \right\} \tag{9}$$

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:*

$$CG_{i,\theta} = \{n_z \mid \tilde{\varphi}_z \geq \theta\} \tag{10}$$

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 (an 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

KDEC-S algorithm is structured in two parts, as discussed in the following.

Local Peer. (Algorithm 1) 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

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
```

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. (Algorithm 2) For a given value of θ, the helper sums up all samples sets and, using definition 10, computes the cluster-guides CG_θ. Function buildClusterGuides() in algorithm 2 shows the details of this step.

3.3 Performance Analysis

Time complexity at Local Peer (algorithm 1) is $O(|G|M^j + log(C)|D^j|)$, 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.

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**

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. The overall communication cost is $O(|G|)$. 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. Moreover, our algorithm uses only few rounds. In the first round each site sends 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, then, sends messages informing the cluster-guides as requested by local peers.

3.4 Confidentiality Analysis

We used two scenarios to analyze the inference risk level of KDEC-S (denoted $\text{IRL}_{\text{KDEC-S}}$). 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 no information about the kernel or about the 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. (7)). 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 exploits statistical density estimation and information sampling to minimize communication costs among sites. Some possibilities of inference attacks in KDEC were shown in[5]. 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. KDEC requires the peers to exchange samples of the local density estimate. Let $y = \hat{\varphi}(\mathbf{x})$ be a sample point. Assuming that a malicious peer L^k inside the group knows all parameter used during the computation, L^k may use y to compute the distance $d = K^{-1}(y)h$, and consequently find the true $\mathbf{x}^*$ which lies at $\mathbf{x}^* = \mathbf{x}+d$. Errors in this method can arise due machine precision,

but they are still much smaller than τ, which in KDEC is suggested to be $h/2$. 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, $\text{IRL}_{KDEC}(c) > 2^{-\tau}$, for all $c > 0$.

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

Proof. By lemma 2 we know that $\text{IRL}_{\text{KDEC-S}}(c) < 2^{\tau}$ and by lemma 3 we have $2^{\tau} < \text{IRL}_{\text{KDEC}}(c)$, for all $c > 0$. Therefore, the assertion holds.

4 Experimental Evaluation

In the experiments reported here we focus on the trade-off between the privacy and the clustering results in KDEC-S. We used two synthetic data sets. The first one consists of 500 points, generated from a mixture model with four Gaussians, each one with σ^2 =1 in all dimensions. For the second data set we produced 400 points. We generated 200 points from a Gaussian with $\mu = 0$ and $\sigma^2 = 5$. Additionally we generated 200 points around the center with radius $R \sim N(20, 1)$ and angle uniformly distributed from $\sim U(0, 2\pi)$.

We applied KDEC-S to both data sets with the following parameters: bandwidth $h = 1$, neighborhood radius fixed in 4, reference tau set to $\tau_{ref} = h/2$, and value of τ ranging from 0.5 to 3.0 with step 0.1. For the Gaussian data set we used grid corners ((-15,-15), (15, 15)) and threshold $\theta = 1.0$. For the polar data set we used grid corners ((-30, -30),(30,30)) and threshold $\theta = 0.1$.

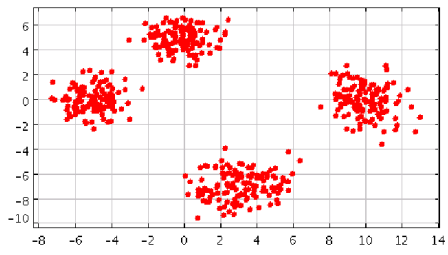

Fig. 2. Gaussian Data: four clusters generated from a Gaussians mixture.

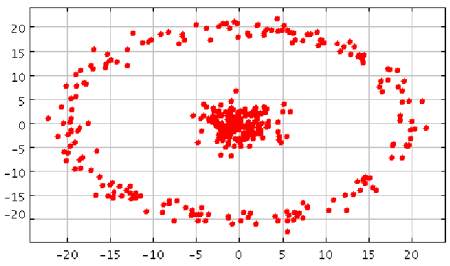

Fig. 3. Polar Data: two clusters with arbitrary shape.

We counted the mislabeling error, considering as correct mapping the clustering obtained with τ_{ref} = h/2. We follow [6] and compute the clustering error as follows:

$$E = \frac{2}{|D|(|D|-1)} \sum_{i,j \in R^2, i<j} e_{ij} \tag{11}$$

where $|D|$ is the size of the data set and e_{ij} is defined as:

$$e_{ij} = \begin{cases} 0 & \text{if } (c(x_i) = c(x_j) \wedge c'(x_i) = c'(x_j)) \quad \vee \\ & (c(x_i) \neq c(x_j) \wedge c'(x_i) \neq c'(x_j)) \\ 1 & \text{otherwise} \end{cases} \tag{12}$$

with $c(x)$ denoting the reference cluster label assigned to object x, i.e. the label found using τ_{ref}, and $c'(x)$ denoting the new label found with $\tau > \tau_{ref}$.

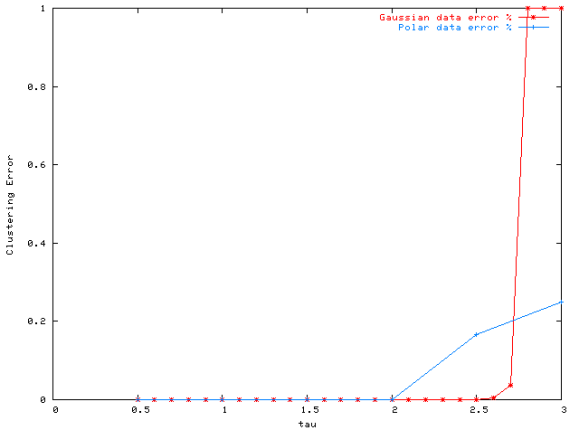

Fig. 4. Clustering error bandwidth $h = 1$ and τ ranging from 0.5 to 3.

From the results of our experiments we conclude that we may set our value of τ up to h with no error in the clustering results. In the Gaussian data set error appears just after $\tau = 2.5$ and in the polar data sets error becomes large after $\tau = 2$. In general, we observed that as the tau goes to values beyond the kernel bandwidth we have an increase in the error because more points are considered as outliers. With the Gaussian kernel we know that the kernel goes to zero around $3 * h$, therefore the iso levels summation does not reach the given threshold. Consequently the correspondent grid point is left out from the cluster guides, i.e. is considered as an outlier. A possible solution is to use adaptive thresholds or even adaptive isolines. We are working on these issues.

5 Related Work

5.1 Inference and privacy preserving data mining

The question of how to protect confidential information from unauthorized disclosure has stimulated much research in the data base community. This prob-

lem, known as *the inference problem*, was first studied in statistical databases and secure multi-level databases and more recently in data mining. The reader is referred to [1] for a survey on *the inference problem* including its instance on data mining. Proposed solutions to privacy preserving data mining can be organized into three main groups: *secure multi-party computation* (SMC) [7–9], *sanitization* [10,11] and *data randomization* [12,13]. Privacy measures can be found in [14] and [15] to the case where the mining algorithm uses randomized data.

5.2 Privacy preserving data clustering

Merugu and Ghosh [16] presents an algorithm for computing a global model F from a predefined fixed family of models. The global model approximates the underlying probability model that generated the global dataset Z. They define privacy as the reciprocal of the average log-likelihood of Z given the model F. Intuitively, the larger the likelihood that the data was generated by the global model, the less privacy is retained. If the predefined family has a very large number of mixture components then the privacy is likely to be low.

Vaidya and Clifton [17] propose a privacy-preserving K-means algorithm on heterogeneously distributed data using cryptographic techniques. They offer a proof that each site does not learn anything beyond its part of each cluster centroid and the cluster assignment of all points at each iteration. The key problem faced at each iteration is securely assigning each point to its nearest cluster. Since each site owns a part of each tuple (which must remain private), this problem is non-trivial. They proposed an algorithm which is based on homomorphic encryption to achieve security.

Oliveira and Zaiane [18] present a family of geometric data transformation is introduced, which aims to provide privacy while maintaining the statistical features of the data in order perform a clustering algorithm. They discussed the efficiency of this method using misclassification error measure and included a comparison with additive noise approach (randomization) with respect to the amount of privacy provided.

6 Conclusions

Our contribution can be summarized as: a confidentiality framework for distributed data clustering together with an algorithm which was showed to be privacy preserving. Our definition of confidentiality and inference levels make little assumptions, what allow for comparing 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 showed that KDEC-S indeed represents an improvement with respect to inference attacks on kernel density estimate, without compromising the clustering results. KDEC-S suffers from using more parameters than KDEC, however, it has a lot of nice properties, e.g. better noise resistance than KDEC, may find arbitrarily shaped clusters, and performs the clustering faster, due to use of lookup table instead of hill climbing the density estimate.

7 Acknowledgments

The authors thank German Ministry of Education and Research for support through grant BMBF 01-IW-D02-SCALLOPS and the Brazilian Ministry for Education for support through grant CAPES 0791/024.

References

[1] C. Farkas, S. Jajodia, The inference problem: A survey, ACM SIGKDD Explorations Newsletter 4 (2) (2002) 6–11.

[2] J. C. da Silva, M. Klusch, Inference and distributed data clustering, in: P. Perner, A. Imiya (Eds.), Machine Learning and Data Mining in Pattern Recognition, no. 3587 in LNAI, Springer Verlag, Leipzig/Germany, 2005, pp. 42–52.

[3] A. Hinneburg, D. A. Keim, An efficient approach to clustering in large multimedia databases with noise, in: Knowledge Discovery and Data Mining, 1998, pp. 58–65.

[4] M. Klusch, S. Lodi, G. Moro, Agent-based distributed data mining: the KDEC scheme, in: M. Klusch, S. Bergamaschi, P. Edwards, P. Petta (Eds.), Intelligent Information Agents: the AgentLink perspective, Vol. 2586 of Lecture Notes in Computer Science, Springer, 2003.

[5] J. C. da Silva, M. Klusch, S. Lodi, G. Moro, Inference attacks in peer-to-peer homogeneous distributed data mining, in: 16th European Conference on Artificial Intelligence (ECAI 04), Valencia, Spain, 2004.

[6] N. Labroche, N. Monmarché, G. Venturini, A new clustering algorithm based on the chemical recognition system of ants, in: F. Harmelen (Ed.), Proceedings of the 15th European Conference on Artificial Intelligence, IOS Press, Lyon, France, 2002, pp. 345–349.

[7] Y. Lindell, B. Pinkas, Privacy preserving data mining, Lecture Notes in Computer Science 1880 (2000) 36–54.

[8] B. Pinkas, Cryptographic techniques for privacy-preserving data mining, ACM SIGKDD Explorations Newsletter 4 (2) (2002) 12–19.

[9] M. Kantarcioglu, C. 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), 2002.

[10] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, V. Verykios, Disclosure limitation of sensitive rules, in: Proceedings of 1999 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'99), Chicago,IL, 1999, pp. 45–52.

[11] E. Dasseni, V. S. Verykios, A. K. Elmagarmid, E. Bertino, Hiding association rules by using confidence and support, Lecture Notes in Computer Science 2137 (2001) 369–??

[12] R. Agrawal, R. Srikant, Privacy-preserving data mining, in: Proc. of the ACM SIGMOD Conference on Management of Data, ACM Press, 2000, pp. 439–450.

[13] S. J. Rizvi, J. R. Haritsa, Maintaining data privacy in association rule mining, in: Proceedings of the 28th VLDB – Very Large Data Base Conference, Hong Kong, China, 2002, pp. 682–693.

[14] A. Evfimievski, J. Gehrke, R. Srikant, Limiting privacy breaches in privacy preserving data mining, in: In Proceedings of PODS 03., San Diego, California, 2003.

[15] D. Agrawal, C. C. Aggarwal, On the design and quantification of privacy preserving data mining algorithms, in: Proceedings of 20th ACM Symposium on Principles of Database Systems, Santa Barbara, Califonia, 2001, pp. 247–255.

[16] Merugu S. and Ghosh J., Privacy-Preserving Distributed Clustering Using Generative Models, in: Proceedings of the IEEE Conference on Data Mining (ICDM), 2003.

[17] Vaidya J. and Clifton C., Privacy-Preserving K-means Clustering Over Vertically Partitioned Data, in: Proceedings of the SIGKDD, 2003, pp. 206–215.

[18] S. Oliveira, O. R. Zaiane, Privacy preserving clustering by data transformation, in: Proc. of SBBD 2003, Manaus, AM, Brasil, 2003.