Web Intelligence and Agent Systems: An international journal 3 (2006) 1–18
IOS Press

Privacy-preserving agent-based distributed data clustering

Josenildo Costa da Silva[a], Matthias Klusch[a], Stefano Lodi[b,*] and Gianluca Moro[c]
[a]*Deduction and Multiagent Systems, German Research Center for Artificial Intelligence, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany*
[b]*Department of Electronics, Computer Science, and Systems, University of Bologna, Viale Risorgimento 2, 40136 Bologna BO, Italy*
[c]*Department of Electronics, Computer Science and Systems, University of Bologna, Via Venezia 52, 47023 Cesena FC, Italy*

Abstract. A growing number of applications in distributed environment involve very large data sets that are inherently distributed among a large number of autonomous sources over a network. The demand to extend data mining technology to such distributed data sets has motivated the development of several approaches to distributed data mining and knowledge discovery, of which only a few make use of agents. We briefly review existing approaches and argue for the potential added value of using agent technology in the domain of knowledge discovery, discussing both issues and benefits. We also propose an approach to distributed data clustering, outline its agent-oriented implementation, and examine potential privacy violating attacks which agents may incur.

Keywords: Distributed data mining, data clustering, data security, Multi-Agent System, data privacy

1. Introduction

Mining information and knowledge from huge data sources such as weather databases, financial data portals, or emerging disease information systems has been recognized by industrial companies as an important area with an opportunity of major revenues from applications such as business data warehousing, process control, and personalised on-line customer services over the Internet and Web. *Knowledge discovery* (KD) is a process aiming at the extraction of previously unknown and implicit knowledge out of large databases which may potentially be of added value for some given application [7]. The automated extraction of unknown patterns, or *data mining* (DM), is a central element of the KD process. The large variety of DM techniques which have been developed over the past decade includes methods for pattern-based similarity search, cluster analysis, decision-tree based classification, generalization taking the data cube or attribute-oriented induction approach, and mining of association rules [2]. One classical application of data mining is the market-based or basket analysis of customer transactions, via offline methods for partitioning, discovery of sequential patterns, including tecniques to efficiently reduce the set of potential candidates for the selection of relevant items, such as hashing and sampling.

The increasing demand to scale up to massive data sets inherently distributed over a network, with limited bandwidth and available computational resources, motivated the development of methods for parallel (PKD) and distributed knowledge discovery (DKD) [17]. The related pattern extraction problem in DKD is referred to as *distributed data mining* (DDM). DDM is expected to perform partial analysis of data at individual sites and then to send the outcome as a partial result to other sites, where it is sometimes required to be aggregated to the global result. The principal problems any approach to DDM must cope with concern issues of autonomy, privacy, and scalability.

[*]Corresponding author. Tel.: +39 051 2093560; Fax: +39 051 2093540; E-mail: stefano.lodi@unibo.it.

Most of the existing DM techniques were originally developed for centralized data and need to be modified for handling the distributed case. As a consequence, one of the most widely used approaches to DDM in business applications is to apply traditional DM techniques to data which have been retrieved from different sources and stored in a central *data warehouse*, i.e., a collection of integrated data from distributed data sources in a single repository [23]. However, despite its commercial success, such a solution may be impractical or even impossible for some business settings in distributed environments. For example, when data can be viewed at the data warehouse from many different perspectives and at different levels of abstraction, it may threaten the goal of protecting individual data and guarding against invasion of privacy. Requirements to respect strict, or a certain degree of, autonomy of given data sources, as well as privacy restrictions on individual data, may make monolithic DM unfeasible.

Another problem arises with the need to scale up to massive data sets which are distributed over a large number of sites. For example, the NASA Earth Observing System (EOS) is a data collector for satellites producing 1450 data sets of about 350GB per day and per pair of satellites at a very high rate which are stored and managed by different systems, geographically located all over the USA. Any online mining of such huge and distributed data sets in a central data warehouse may be prohibitively expensive in terms of costs for both communication and computation.

To date, most work on DDM and PDM use distributed processing and the decomposability of data mining problems to scale up to large data sources. One lesson from the recent research work on DDM is that cooperation among distributed DM processes may allow effective mining even without centralized control [16]. This in turn leads us to the question of whether there is any real added value in using concepts from agent technology [18,35] for the development of advanced DDM systems. A number of DDM solutions are available using various techniques such as distributed association rules, distributed clustering, Bayesian learning, classification (regression), and compression, but only a few of them make use of intelligent agents at all. In general, the inherent feature of software agents, as being autonomous and capable of adaptive and deliberative reasoning, seems to fit quite well with the requirements of coping with the above mentioned problems and challenges of DDM. An autonomous data mining agent, as a special kind of information agent [18], may perform various kinds of mining operations on behalf of its user(s) or in collaboration with other agents. Systems of cooperative information agents for data mining tasks in distributed massive data environments, such as multidimensional peer-to-peer networks [22,25,28] and grid computing systems [9], appear to be quite a natural vision for the near future.

In this paper we briefly review and classify existing DDM systems and frameworks according to some criteria in Section 2. This is followed by a brief discussion on the benefits of using agents for DDM in Section 3. We introduce in Section 4 an agent-based, distributed data clustering scheme and discuss the threats to data privacy which potentially arise in its application. We conclude the paper in Section 6 with an outline of ongoing and future research work.

2. State of the art

In this section we provide a brief review of the most representative agent-based DDM systems to date, according to (a) the kind, type, and used means for security of data processed; (b) used DM techniques, implementation of the system and agents; and (c) the architecture with respect to the main coordination and control, execution of data processing, and transmission of agents, data, and models in due course of the DM tasks to be pursued by the system.

BODHI [17] has been designed according to a framework for collective DM tasks on heterogeneous data sites such as supervised inductive distributed function learning and regression. This framework guarantees a correct local and global data model with low network communication load. BODHI is implemented in Java; it offers message exchange and runtime environments (agent stations) for the execution of mobile agents at each local site. The mining process is distributed to the local agent stations and agents that are moving between them on demand each carrying its state, data and knowledge. A central facilitator agent is responsible for initializing and coordinating DM tasks to be pursued within the system by the agents and agent stations, as well as the communication and control flow between the agents. Inter-agent communication bases on KQML [8].

PADMA [16] deals with the problem of DDM from homogeneous data sites. Partial data cluster models are first computed by stationary agents locally at different sites. All local models are collected to a central site that performs a second-level clustering algorithm to generate the global cluster model. Individual agents perform hierarchical clustering in text document classification, and web based information visualization.

JAM [32] is a Java-based multi-agent system designed to be used for meta-learning DDM. Different learning classifiers such as Ripper, CART, ID3, C4.5, Bayes, and WPEBLS can be executed on heterogeneous (relational) databases by any JAM agent that is either residing on one site or is being imported from other peer sites in the system. Each site agent builds a classification model and different agents build classifiers using different techniques. JAM also provides a set of meta-learning agents for combining multiple models, learnt at different sites, into a meta-classifier that in many cases improves the overall predictive accuracy. Once the combined classifiers are computed, the central JAM system coordinates the execution of these modules to classify data sets of interest at all data sites simultaneously and independently.

Papyrus [1] is a Java-based system addressing wide-area DDM over clusters of heterogeneous data sites and meta-clusters. It supports different task and predictive model strategies including C4.5. Mobile DM agents move data, intermediate results, and models between clusters to perform all computation locally and reduce network load, or from local sites to a central root which produces the final result. Each cluster has one distinguished node which acts as its cluster access and control point for the agents. Coordination of the overall clustering task is either done by a central root site or distributed to the (peer-to-peer) network of cluster access points. Papyrus supports various methods for combining and exchanging the locally mined predictive models and metadata required to describe them by using a special markup language.

Common to all approaches, is that they aim at integrating the knowledge discovered from data at different geographically distributed network sites, with a minimum amount of network communication and a maximum of local computation.

3. Why agents for DDM?

Looking at the state of the art of agent-based DDM systems presented in the previous section we may identify the following arguments in favor or against the use of intelligent agents for distributed data mining.

Autonomy of data sources. A DM agent may be considered as a modular extension of a data management system to deliberatively handle the access to the data source in accordance with constraints on the required autonomy of the system, data and model. This is in full compliance with the paradigm of cooperative information systems [26].

Interactive DDM. Pro-actively assisting agents may drastically limit the amount a human user has to supervise and interfere with the running data mining process [39]. For example, DM agents may anticipate the individual limits of the potentially large search space and proper intermediate results particularly driven by their individual users' preferences with respect to the particular type of DM task at hand.

Dynamic selection of sources and data gathering. One challenge for intelligent DM agents acting in open distributed data environments in which, for example, the DM tasks to pursue, the availability of data sites and their content may change at any time, is to discover and select relevant sources. In such settings DM agents may be applied to adaptively select data sources according to given criteria such as the expected amount, type and quality of data at the considered source, actual network and DM server load [30]. Such DM agents may be used, for example, to dynamically control and manage the process of data gathering to support any OLAP (online analytical processing) and business data warehouse application.

Scalability of DM to massive distributed data. One option to reduce network and DM application server load may be to let DM agents migrate to each of the local data sites in a DDM system on which they may perform mining tasks locally, and then either return with or send relevant pre-selected data to their originating server for further processing. Experiments in using mobile information filtering agents in distributed data environments are encouraging [34].

Multi-strategy DDM. For some complex application settings an appropriate combination of multiple data mining techniques may be more beneficial than applying just one particular one. DM agents may learn in the due course of their deliberative actions which one to choose, depending on the type of data retrieved from the different sites and the mining tasks to be pursued. The learning process of the multi-strategy selection of DM methods is similar to the adaptive selection of coordination strategies in a multi-agent system as proposed, for example, in [27].

Collaborative DM. DM agents may operate independently on data they have gathered at local sites, and then combine their respective models. Or they may agree to share potential knowledge as it is discovered, in order to benefit from the additional opinions of other DM agents. Meta-learning techniques may be used to mine homogeneous, distributed data. However, naive approaches to local data analysis may produce an ambiguous and incorrect global data model if different heterogeneous data sites are involved which store data for different sets of features, possibly with some common features among the sites. Collaborative DM agents may negotiate among each other and jointly plan a solution for the above mentioned problems at hand. The need for DM agents to collaborate is prominent, for example, in cases where credit card frauds have to be detected by scanning, analysing, and partially integrating world-wide distributed data records in different, autonomous sources. Other applications of potential added value include the pro-active re-collection of geographically distributed patient records and mining of the corresponding data space on demand to infer implicit knowledge to support an advanced treatment of patients, regardless of which, and how many hospitals they have been taken into in the past. However, frameworks for agent-based collective data mining, such as BODHI, are still more than rare to date.

Security and trustworthiness. In fact, this may be an argument against the use of agents for DDM. Of course, any agent-based DDM system has to cope with the problem of ensuring data security and privacy. However, any failure to implement the minimal privilege at a data source, which means endowing subjects with only enough permission to discharge their duties, could give any mining agent unsolicited access to sensitive data. Moreover, any mining operation performed by agents of a DDM system lacking a sound security architecture could be subject to eavesdropping, data tampering, or denial of service attacks. Agent code and data integrity is a crucial issue in secure DDM: Subverting or hijacking a DM agent places a trusted piece of (mobile) software – thus any sensitive data carried or transmitted by the agent – under the control of an intruder. In cases where DM agents are even allowed to migrate to remote computing environments of the distributed data sites of the DDM system methods to ensure confidentiality and integrity of a mobile agent have to be applied. Regarding agent availability there is certainly no way to prevent malicious hosts from simply blocking or destroying the temporarily residing DM agents but selective replication in a fault tolerant DDM agent architecture may help. In addition, data integration or aggregation in a DDM process introduces concern regarding inference attacks as a potential security threat. Data mining agents may infer sensitive information even from partial integration to a certain extent and with some probability. This problem, known as the so called inference problem, occurs especially in settings where agents may access data sources across trust boundaries which enable them to integrate implicit knowledge from different sources using commonly held rules of thumb. None of the existing DDM systems, agent-based or not, is capable of coping with this inference problem in the domain of secure DDM.

In the following sections we investigate how agents may be used to perform a special kind of distributed data mining, that is clustering of data at different homogeneous data sites. For this purpose, we present an approach to distributed cluster analysis based on density estimation, and then briefly discuss issues of implementing the resulting scheme for distributed data clustering in an agent-based DDM system, including data privacy and trustworthiness.

4. A scheme for distributed data clustering

4.1. Density estimation based clustering

Cluster analysis is a descriptive data mining task which aims at partitioning a data set into groups such that the data objects in one group are similar to each other and are as different as possible from those in other groups. As dense regions of the data space are more likely to be populated by similar data objects, one popular clustering technique is based on reducing the search for clusters to the search for such regions. In *density estimation* (DE) based clustering the search for densely populated regions is accomplished by estimating a probability density function from which the given data set is assumed to have arisen [5,15,31]. One important family of methods requires the computation of a non-parametric density estimate known as *kernel estimator*.

Let us assume a set $D = \{\boldsymbol{x}_i | i = 1, \ldots, N\} \subseteq \mathbb{R}^d$ of data objects. Kernel estimators originate from the intuition that the higher the number of neighbouring data objects $\boldsymbol{x}_i$ of some given space object $\boldsymbol{x} \in \mathbb{R}^d$, the higher the density at this object $\boldsymbol{x}$. However, there can be many ways of computing the influence of neighbouring data objects. Kernel estimators use a so called *kernel function* $K(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ which integrates to unity over $\mathbb{R}^d$. A *kernel estimator* $\hat{\varphi}_{K,h}[D](\cdot) : \mathbb{R}^d \to \mathbb{R}_+$ is defined as the sum over all data objects $\boldsymbol{x}_i$ in D of the differences between $\boldsymbol{x}$ and $\boldsymbol{x}_i$, scaled by a factor h, called *window width*, and weighted by the kernel function K:

$$\hat{\varphi}_{K,h}[D](\boldsymbol{x}) = \frac{1}{Nh^d}\sum_{i=1}^{N} K\left(\frac{1}{h}(\boldsymbol{x}-\boldsymbol{x}_i)\right). \quad (1)$$

A kernel estimator can be considered as a sum of "bumps" placed at the observations. The window width h controls the smoothness of the estimate, whereas K determines the shape of the bumps.

Usually, K is radially symmetric and non-negative with a unique maximum in the origin; in this case, K formalizes the decay of the influence of a data object according to its distance. In the following, we will consider only kernels of this kind.

Prominent examples of kernel functions are the standard multivariate normal density $(2\pi)^{-d/2}\exp(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{x})$, the multivariate uniform kernel $w(\boldsymbol{x})$, defined by

$$w(\boldsymbol{x}) = \begin{cases} c_d^{-1} \text{ if } \boldsymbol{x}^T\boldsymbol{x} < 1, \\ 0 \quad \text{otherwise,} \end{cases}$$

and the multivariate Epanechnikov kernel $K_e(\boldsymbol{x})$, defined by

$$K_e(\boldsymbol{x}) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2) \text{ if } \boldsymbol{x}^T\boldsymbol{x} < 1, \\ 0 \qquad\qquad \text{otherwise,} \end{cases}$$

where c_d is the volume of the unit d-dimensional sphere. An example of kernel estimate in $\mathbb{R}$ showing the normal component kernels is shown in Fig. 1.

When $d \geqslant 2$, it may be advisable to linearly transform the data, in order to avoid large differences of spread in the dimensions, and transform inversely after applying the estimate [11]. In an equivalent manner, this is accomplished by the estimate:

$$\hat{\varphi}_{K,h}[D](\boldsymbol{x}) = \frac{(\det\boldsymbol{S})^{-1/2}}{Nh^d}\sum_{i=1}^{N} k \left(\frac{1}{h^2}(\boldsymbol{x}-\boldsymbol{x}_i)^T\boldsymbol{S}^{-1}(\boldsymbol{x}-\boldsymbol{x}_i)\right) \quad (2)$$

where $\boldsymbol{S}$ is the sample covariance matrix of D, and k satisfies $k(\boldsymbol{x}^T\boldsymbol{x}) = K(\boldsymbol{x})$.

The criteria to optimally choose K and h have been extensively dealt with in the literature on non-parametric density estimates, where K and h are deemed optimal if they minimize the expected value of the integrated squared pointwise difference between the estimate and the true density φ of the data, or *mean integrated square error* (MISE). It has been shown that the performance of the Epanechnikov kernel, measured by the MISE criterion, is optimal; however, the performances of commonly used kernels do not differ substantially from the performance of an optimal kernel. Therefore, the choice of a kernel may be based on computational or differentiability properties, rather than on its MISE. The optimal value of the window width h can be approximated as

$$h_{\text{opt}} = A(K)N^{-1/(d+4)} \quad (3)$$

where $A(K)$ depends also on the unknown true density φ. The value of $A(K)$ has been tabulated for commonly used kernels when φ is a unit multivariate normal density [31]. The resulting values of h_{opt} can be used directly in Eq. (2). Alternatively, if Eq. (1) is used, the value of h can be defined by

$$h = \left(d^{-1}\sum_i s_{ii}\right)^{1/2} h_{\text{opt}} \quad (4)$$

taking thus into account the average variance of the data over all dimensions [31].

In DE-clustering, the kernel estimate of a data set has been used for the discovery of many types of density-based clusters [5,15,31]. One simple type is the so-called *center-defined* cluster: every local maximum of $\hat{\varphi}$ corresponds to a cluster including all data objects which can be connected to the maximum by a continuous, uphill path in the graph of $\hat{\varphi}$. It is apparent that, for every data object, an uphill climbing procedure driven by the kernel estimate will find the local maximum representing the object's cluster [15,19,31].

4.2. KDEC-based distributed data clustering

We define the problem of *homogeneous distributed data clustering* (homogeneous DDC) as follows. Let $D = \{\boldsymbol{x}_i | i = 1,\ldots,N\} \subseteq \mathbb{R}^d$ be a data set of objects. Let $L_j, j = 1,\ldots,M$ be a finite set of *sites*. Each site L_j stores one data set D_j of size N_j. It will be assumed that $D = \bigcup_{j=1}^{M} D_j$. Let $\mathcal{C} = \{C_k\} \subseteq 2^D$ be a clustering of D, whose elements are pairwise disjoint. The homogeneous DDC problem is to find for $j = 1,\ldots,M$, a site clustering $\mathcal{C}$ residing in the data space of L_j, such that $\mathcal{C}_j = \{C_k \cap D_j | k = 1,\ldots,|\mathcal{C}|\}$ (*correctness requirement*), time and communications costs are minimized (*efficiency requirement*), and, at the end of the computation, the size of the subset of D which has been transferred out of the data space of any site L_j is minimized (*privacy requirement*). The traditional solution to the homogeneous DDC problem is to simply collect all the distributed data sets D_j into one centralized repository where the clustering of their union is computed and transmitted to the sites. Such an approach, however, does not satisfy our problem's requirements both in terms of privacy

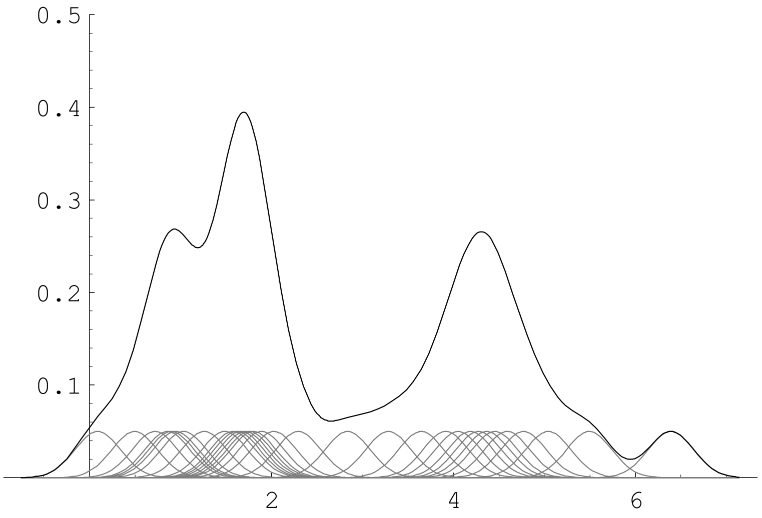


Fig. 1. Kernel estimate and its normal component kernels ($h = 0.25$, $N = 32$).

and efficiency. Therefore, in [19] a different approach has been proposed yielding a kernel density estimation based clustering scheme, called KDEC, which may be implemented by appropriately designed DM agents of an agent-based DDM system. Before examining the issues and benefits of an agent-based implementation, we briefly review the KDEC scheme.

The KDEC scheme is based on three simple observations: density estimates are (i) additive for homogeneous distributed data sets, (ii) sufficient for computing DE-clustering, and (iii) provide a more compact representation of the data set for the purpose of transmission. In the sequel, we tacitly assume that all sites L_j agree on using a global kernel function K and a global window width h. We will therefore omit K and h from our notation, and write $\hat{\varphi}[D](\boldsymbol{x})$ for $\hat{\varphi}_{K,h}[D](\boldsymbol{x})$.

The global density estimate $\hat{\varphi}[D](\boldsymbol{x})$ can be written as the sum of the site density estimates, one estimate for every data set D_j:

$$\begin{aligned}\hat{\varphi}[D](\boldsymbol{x}) &= \sum_{j=1}^{M} \sum_{\boldsymbol{x} \in D_j} K\left(\frac{1}{h}(\boldsymbol{x} - \boldsymbol{x}_i)\right) \\ &= \sum_{j=1}^{M} \hat{\varphi}[D_j](\boldsymbol{x}).\end{aligned} \tag{5}$$

Thus, the local density estimates can be transmitted to and summed up at a distinguished helper site yielding the global estimate which can be returned to all sites. Each site L_j then may apply to its local data space a hill-climbing technique to assign clusters to its local data objects. Note however that Eq. (1) explicitly refers to the data objects $\boldsymbol{x}_i$. Hence, transmitting a naive coding of the estimate entails transmitting the data objects which contradicts the privacy requirement. Multi-dimensional sampling provides an alternative extensional representation of the estimate which makes no explicit reference to the data objects.

For $\boldsymbol{x} \in \mathbb{R}^d$, let $x_1, \ldots, x_d$ be its components. Let $\boldsymbol{\tau} = [\tau_1, \ldots, \tau_d]^T \in \mathbb{R}^d$ be a vector of *sampling periods*, and let $\boldsymbol{z} \bullet \boldsymbol{\tau}$ denote $[z_1\tau_1, \ldots, z_d\tau_d]^T$, where $\boldsymbol{z} \in \mathbb{Z}^d$. A function $f : \mathbb{R}^d \to \mathbb{R}$ is *band-limited* to a bounded $B \subset \mathbb{R}^d$ if and only if the support of its Fourier transform is contained in B. If B is a subset of a rectangle $[-\pi/\tau_1, \pi/\tau_1) \times \cdots \times [-\pi/\tau_d, \pi/\tau_d)$, it is well-known that the *sampling series*

$$\begin{aligned}&\sum_{\boldsymbol{z} \in \mathbb{Z}^d} f(\boldsymbol{z} \bullet \boldsymbol{\tau}) \operatorname{sinc}\left(\frac{x_1}{\tau_1} - z_1\right) \\ &\cdots \operatorname{sinc}\left(\frac{x_d}{\tau_d} - z_d\right),\end{aligned} \tag{6}$$

where

$$\operatorname{sinc}(x) = \begin{cases} 1 & \text{if } x = 0, \\ \frac{\sin \pi x}{\pi x} & \text{otherwise,} \end{cases}$$

converges to f under mild conditions on f (see, e.g. [14, p.155]). If we let $f(\cdot) = \hat{\varphi}[D_j](\cdot)$ in Eq. (6), and truncate the series to a finite d-dimensional rectangle $R(\boldsymbol{z}_1, \boldsymbol{z}_2)$ having diagonal $(\boldsymbol{z}_1, \boldsymbol{z}_2)$ we obtain an interpolation formula:

$$\sum_{\boldsymbol{z} \in R(\boldsymbol{z}_1, \boldsymbol{z}_2)} \sum_{j=1}^{M} \hat{\varphi}[D_j](\boldsymbol{z} \bullet \boldsymbol{\tau}) \operatorname{sinc}\left(\frac{x_1}{\tau_1} - z_1\right)$$

$$\cdots \operatorname{sinc}\left(\frac{x_d}{\tau_d} - z_d\right) \quad (7)$$

Notice that the function represented by Eq. (7) is not extensionally equal to the kernel global estimate $\hat{\varphi}[D](\boldsymbol{x})$ both because kernel estimates are not band-limited on any region, and because of the truncation in the series. However, as argued in [19], the approximation introduces only a small error. In fact, both a density estimate and its Fourier transform vanish rapidly when the norm of the argument $\to \infty$; therefore, we may take $\boldsymbol{\tau}$ in such a way that the transform is negligible in $\mathbb{R}^d \setminus [-\pi/\tau_1, \pi/\tau_1) \times \cdots \times [-\pi/\tau_d, \pi/\tau_d)$, and by selecting $(\boldsymbol{z}_1, \boldsymbol{z}_2)$ so that the estimate is negligible in $\mathbb{R}^d \setminus R(\boldsymbol{z}_1, \boldsymbol{z}_2)$.

Therefore, Eq. (7) gives an approximation of the global density estimate that can be exploited to devise a distributed clustering scheme: For $j = 1, \ldots, M$, the samples $\{\hat{\varphi}[D_j](\boldsymbol{z} \bullet \boldsymbol{\tau}) : \boldsymbol{z} \in R(\boldsymbol{z}_1, \boldsymbol{z}_2)\}$ of the j-th local density estimate are transmitted to and summed up at a distinguished helper site yielding the samples of the global estimate which can be returned to all sites, which then use Eq. (7) as the global density estimate to which the hill-climbing technique is applied.

Algorithm 1 KDEC: distributed clustering based on density estimation

```
func Interpolate $(x, \tau, z_1, z_2, Sam[\,])$ ≡
  foreach $z \in R(z_1, z_2)$ do
    $r := r + Sam[z]\Pi_{i=1}^{d} Sinc\left(\frac{x_i}{\tau_i} - z_i\right)$
  od;
  $r$.
proc DataOwner$(D[\,], H, Clus[\,])$ ≡
  Negotiate $(H, \tau, z_1, z_2, K, h)$;
  Send (Sample $(D, \tau, z_1, z_2, K, h)$);
  Sam := Receive $(H)$
  for $i := 1$ to Length $(D)$ do
    Clus $[i]$ := Nearest(
      FindLocalMax $(x_i, \tau, z_1, z_2,$ Sam,
        $\nabla$ Interpolate( )));
  od.
proc Helper $(L[\,])$ ≡
  Negotiate $(L)$; SetZero(Sam);
  for $j := 1$ to Length $(L)$ do
    Sam := Sam + Receive $(L[j])$
  od;
  for $j := 1$ to Length $(L)$ do
    Send(Sam, $L[j]$)
  od.
```

A distributed implementation of the KDEC scheme is sketched as Algorithm 1. Local sites run *DataOwner*, whereas the helper site runs *Helper*, where H, $D[\,]$, $L[\,]$, reference the helper, the local data set, a list of local sites, respectively, and *Clus*[ ] is the result (an object-cluster look-up table). *Negotiate* sets up a forum where the local sites can reach an agreement on $\boldsymbol{\tau}$, $R(\boldsymbol{z}_1, \boldsymbol{z}_2)$, the kernel K and the window width h. Local sites send the samples of the local estimate of $D[\,]$ to H which sums them orderly. Finally, each local site receives the global samples and uses them in procedure *Interpolate* to compute the values of the global density estimate and applies the gradient-driven, hill-climbing procedure *FindLocalMax* to compute the corresponding local data clusters (see [19] for more details).

Example. Figure 2 shows a dataset of 100 2-dimensional objects containing two clusters, which were randomly generated with a bivariate normal distribution, and the contour plot of its kernel estimate, computed for a normal kernel with $h = h_{\text{opt}} = 1/\sqrt{5}$ by means of Eq. (2). Figure 3 illustrates the clusters computed by the KDEC scheme on the dataset, divided between two sites in such a way that each local dataset spans across both clusters. Objects plotted with equal symbol shapes belong to the same cluster and viceversa. The figure also shows the contours of the sampling series of the whole dataset, computed with $h = 1/\sqrt{5}$, $\boldsymbol{\tau} = [h, h]^T$. As the local maxima of the estimate are preserved, the clusters are correctly recovered. Figure 4 shows the output of KDEC and the contours of the series on the same datasets, setting $h = 1/\sqrt{5}$, $\boldsymbol{\tau} = [5h, 5h]^T$. In this case, the sampling series is clearly a poor approximation of the estimate. Nevertheless, the local maxima are preserved, and KDEC returns the correct clusters.

Algorithm 1 is abstract, in that it only specifies the flow of information between processes, and it ignores actual running locations in the network and the technology of distributed computation. In the following section we will describe various options to make the KDEC scheme more concrete, and some pitfalls to data privacy and security.

4.3. *Agents for KDEC-based homogeneous DDC*

The KDEC scheme may be modified in several ways to generate protocols for stationary or mobile agents. We assume a DDM system consisting of a set $\{L_n | 1 \leqslant n \leqslant M, M > 1\}$ of networked homogeneous data sites and a set $\{L_n | M+1 \leqslant n \leqslant M'\}$ of helpers. Both data sites and helpers provide support for local and external, visiting agents. Each site respects its local autonomy by individually granting read-only access to external data mining agents.

M' data mining agents A_n have been designed and deployed at the data sites and helpers. Agent A_n is assumed to be associated to site L_n, for $1 \leqslant n \leqslant M'$.

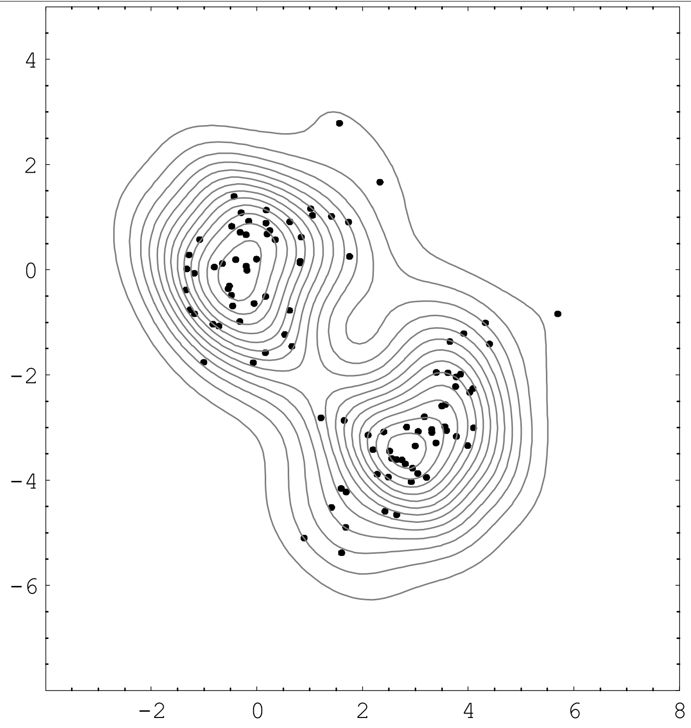


Fig. 2. Scatter plot of a 2-dimensional dataset and contour plot of its kernel estimate, $h = 1/\sqrt{5}$.

The data site agent interested in starting a data mining initiative, named the *initiator*, collects a list of possible participants by using search techniques commonly adopted in peer-to-peer systems, whose complexity is logarithmic in the number of peers in the network. The initiator agent searches for all peers able to interact according to a given data clustering service, which corresponds in this case to our algorithm, executed with a protocol specified by the initiator itself. When the list has been fixed, the initiator selects a *master* helper and, possibly, a set of auxiliary helpers, depending on the protocol. The initiator sends the master helper the list of peer data sites, the list of auxiliary helpers, and the protocol specifications. The helper takes care of arranging the data sites and the auxiliary helpers according to the topology specified in the protocol, and starts the mining activity. For $n \leqslant M'$, A_n carries out the initial negotiations of the protocol on behalf of L_n as stationary agent. Later, A_n, $n > M$, may proceed as either stationary or mobile agent, depending on the protocol.

To illustrate the negotiation and the protocols we introduce the following scenario.

Example. It has been announced that an international insurance company will soon enter the national market. As a consequence of the announcement, eight national insurers decide to combine and analyse their policy data by clustering. Their goal is to achieve a better understanding of the national market, and to plan strategies for keeping their share. Therefore, each insurer creates a view with the following dimensions: $Latitude$, $Longitude$, $Indemnity$, where $Latitude$ and $Longitude$ are computed from the address of the policyholder.

For simplicity we assume the datasets of the eight insurers follow three distribution patterns. Distribution 1 (sites 1–3): The policies have high indemnity and are held by inhabitants of large cities in the north-east; distribution 2 (site 4–6): The policies have very high indemnity, their holders live in the south; distribution 3 (sites 7 and 8): The policies have small indemnity and the holders live in the west. Figures 5(a), 5(b), and 5(c) show the scatter plots of the dataset, in the hyperplanes $(Longitude, Latitude)$, $(Longitude, Indemnity)$, and $(Latitude, Indemnity)$, respectively. Objects belonging to the first, second, and third distribution have been plotted in light gray, dark gray and black, respectively.

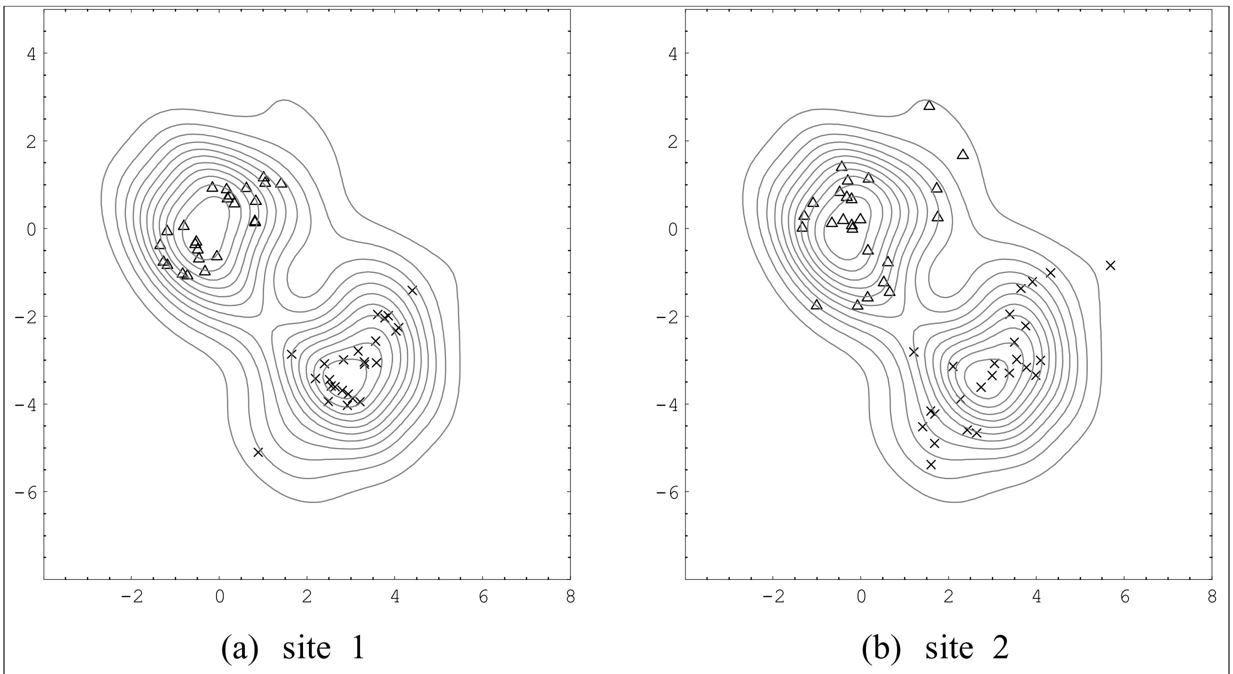


Fig. 3. Clusters computed by KDEC for $h = 1/\sqrt{5}$, $\boldsymbol{\tau}= [h, h]^T$.

4.3.1. *Negotiation of parameters*

According to the KDEC scheme, the master helper engages the other site agents in a negotiation to agree what kernel function to use for computing the local density estimate samples, an optimal value of the parameters h and $\boldsymbol{\tau}$, and the data space hyper-rectangle delimiting the data clustering area of interest.

The negotiation procedure is made up of three phases which can be implemented according to the communicative acts of ACL FIPA [10]. The interactions are based on the primitives of such a standard (e.g. *call-for-proposal, accept, inform, . . .*) which give a first level of semantics to messages exchanged among agents.

The helper agent in the first phase asks each participant, using the *query-ref* ACL primitive, for the number of its local objects, the linear sums and sums of squares of all its local objects along each dimension (i.e. object attribute), and finally the corners of the smallest hyper-rectangle covering its data space domain.

In the second phase, when all participant agents have replied with the requested data (using the primitive *inform*), the helper sends them a *call-for-proposal* containing (1) a set of possible kernel functions from which each participant should select its preferred one, (2) the corners $\boldsymbol{l}, \boldsymbol{r} \in \mathbb{R}^d$ of a hyper-rectangle containing the union of all hyper-rectangular data spaces, (3) for every kernel function K, a recommended value h_r for h, and the parameter β_K, which will be explained below, and (4) a possible value for $h^{-1}\,\tau$.

The helper computes the recommended values of the window width h for each kernel function using Eq. (4), the marginal variances of the whole dataset, and the values of $h_{\rm opt}$. The marginal variances of the data can be easily computed by the helper from the total number of objects in the dataset, their vector sum, the sums of squares of dimensions, which in turn can be computed as the sum of their local values at each site. The values of $h_{\rm opt}$ for each kernel function are given by Eq. (3). Although more sophisticated and accurate ways to choose $h_{\rm opt}$ automatically have been studied in the literature, the value given by Eq. (3) will be sufficiently accurate as a starting point for the negotiation. Moreover, it can be computed by the helper from the total number of objects only. The helper sets $\boldsymbol{l}, \boldsymbol{r}$ to the corners $\boldsymbol{l}', \boldsymbol{r}'$ of the smallest hyper-rectangle containing the union of all hyper-rectangular data spaces, randomly extended in order to prevent the disclosure of the coordinates of extremal data objects. That is, $\boldsymbol{l} = \boldsymbol{l}' + \boldsymbol{v}_l$, $\boldsymbol{r} = \boldsymbol{r}' + \boldsymbol{v}_r$, where the components of $\boldsymbol{v}_l$, $\boldsymbol{v}_r$ are uniformly distributed in $[-h_r, 0]$ and $[0, h_r]$, respectively. The parameter β_K is used to delimit how far beyond the border of the data space hyper-rectangle the estimate is not negligible and must be sampled. We assume β_K to be equal to the smallest multiple of 10^{-2} greater than $\sup\{\|\boldsymbol{x}\| : K(\boldsymbol{x}) > 10^{-2}\}$.

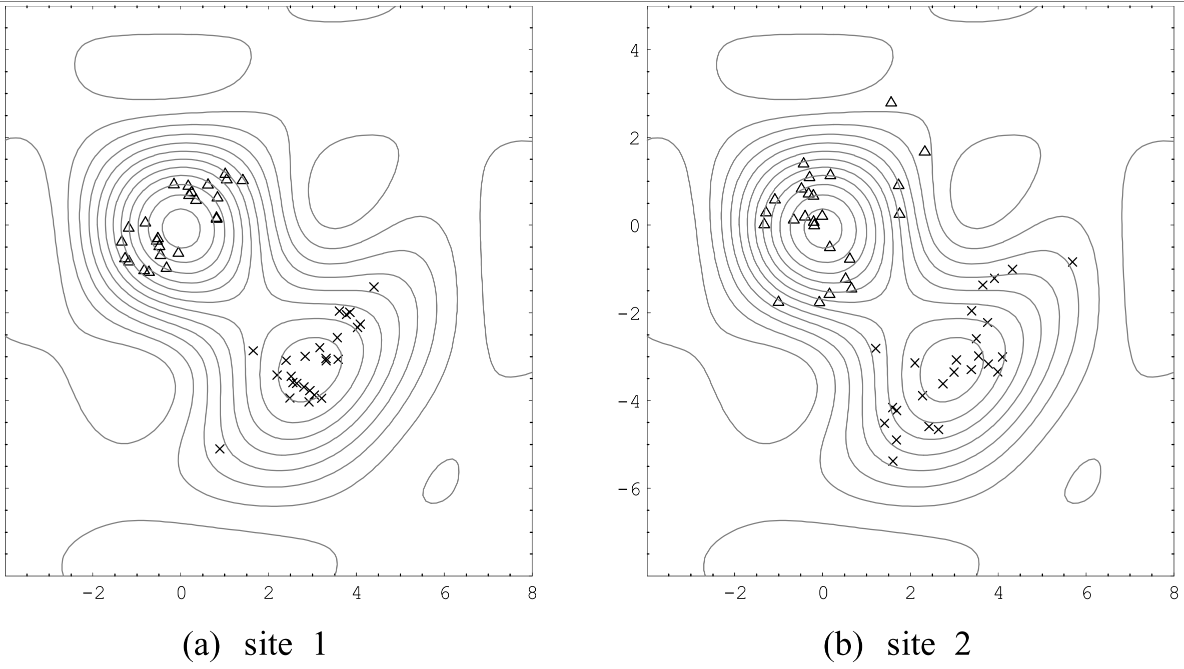


Fig. 4. Clusters computed by KDEC for $h = 1/\sqrt{5}$, $\boldsymbol{\tau}= [5h, 5h]^T$.

With the exception of the normal kernel, popular kernels have bounded support and their value is zero for $\|\boldsymbol{x}\| \geqslant 1$; thus $\beta_K = 1$. For the normal kernel, we have $\beta_K = 3.04$.

Each proposal returned to the master helper shall include a kernel function, an interval for h, and an interval for $h^{-1}\,\tau$. The choice of both the kernel and h have an impact on data privacy. The regularity of the kernel increases privacy (see Section 4.4.1). Isolated maxima centered at the data objects tend to become more and more evident in the density estimate as h decreases. On the other hand, choosing too large a value for h may result in an oversmoothed estimate and merging of some natural cluster. Each participant proposes an interval for $h^{-1}\,\tau$ such that the resulting time and transmission complexities are compatible with its application constraints. To this purpose, each agent assumes that the estimate is negligible outside the hyper-rectangle of corners $\boldsymbol{l} + [-\beta h_r, \ldots, -\beta h_r]^T, \boldsymbol{r} + [\beta h_r, \ldots, \beta h_r]^T$. The parameter β is equal to the largest β_K over all kernels K contained in the *call-for-proposal*.

In the third phase the helper site, after collecting all proposals from the interested participants, defines the final proposal including the most selected kernel function by participants, the best h and τ with respect to all possible counterproposal values and the definitive hyper-rectangular data space. The helper sends the final proposal and only agents who accept it will be involved in the distributed data clustering initiative.

All communications between the helper and participants are carried out using the digital sign mechanism, moreover each agent involved knows the digital identity of the helper, therefore no one may alter a message or impersonate the helper.

We illustrate the negotiation of parameters with our example.

Example. After sending a *query-ref*, the helper is informed of the aggregate statistics of the sites, computes the total count of data objects, the marginal variances of the data, and the corners of the hyper-rectangle covering the data space $\boldsymbol{l}' = [-29.11, -16.07, 1.89]^T$, $\boldsymbol{r}' = [105.49, 97.14, 161.99]^T$. The eligible kernels are the normal kernel and the Epanechnikov kernel, corresponding to $h_{\text{opt}} = 0.36$, $h_{\text{opt}} = 1.09$. The average marginal variance is $\sigma = 29.83$. As all sites are concerned about the potential disclosure of their data, their agents choose the most regular kernel, i.e., the normal kernel corresponding to a recommended value $h_r = \sigma h_{\text{opt}} = 10.80$. Sites L_1, L_2, L_3 propose the interval $[h_r/2, h_r]$, whereas the remaining sites $L_4, \ldots, L_8$ propose $[h_r, 2h_r]$. L_1, L_2, L_3 propose a smaller lower bound because the maximum distance between data objects in their datasets is smaller. Thus, for $h = h_r/2$, the density estimate has no isolated “bump” which would reveal the location of an

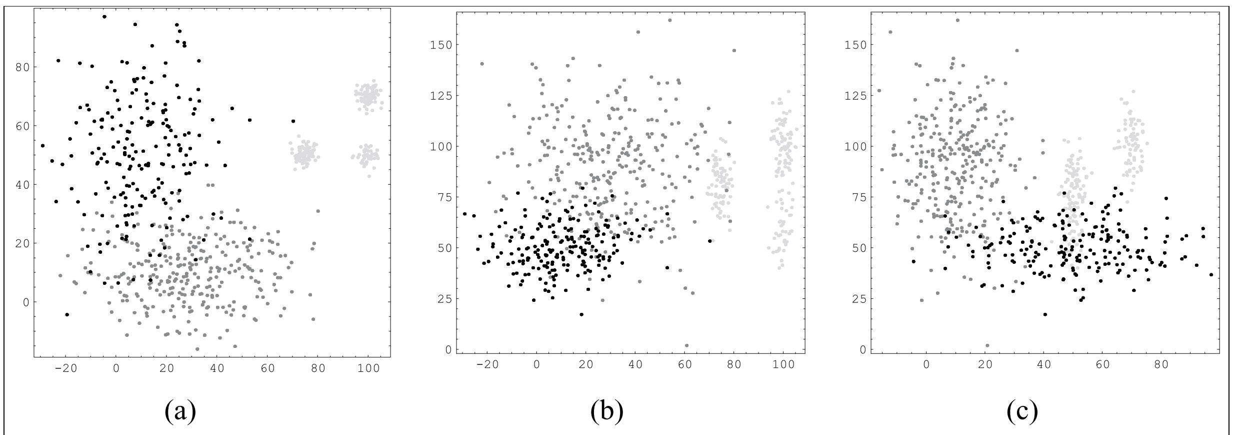


Fig. 5. Scatter plots of example data in the hyperplanes $(Longitude, Latitude)$, $(Longitude, Indemnity)$, and $(Latitude, Indemnity)$.

object, whereas the structure of the three clusters corresponding to their combined local datasets is visible. In contrast, for $h < h_r$, in the region where *Indemnity* is high, some maxima corresponding to objects stored at sites L_4 or L_5 are evident. The insurers owning L_4 and L_5 would be deeply concerned about the possibility that geographical information and high indemnity could be exploited to narrow the search for the identities of the policyholders. Figure 6(a) shows the marginal estimate in the hyperplane $(Longitude, Indemnity)$ for $h = h_r/2$, and Fig. 6(b) is an inset showing the local maxima. Assuming $\beta = 3.04$ (only the normal kernel and the Epanechnikov kernel are contained in the *call-for-proposal*), the number of required samples is about 5000 for $\tau = h_r/2$, which is modest and acceptable for all agents. Thus, all participants propose $h^{-1}\boldsymbol{\tau} = 1/2$. The final proposal of the helper contains the normal kernel, $h = h_r$ and the hyper-rectangle $\boldsymbol{l} + 3.04[-h_r, -h_r, -h_r]^T$, $\boldsymbol{r} + 3.04[h_r, h_r, h_r]^T$.

4.3.2. Protocols

We classify potential implementations and variations of the remaining steps of Algorithm 1 according to two directions: The path of samples, or partial sums of samples, in the network of sites, and the protocol to compute the sums. We consider the following three basic partial orders of the network of sites.

Sequence The data sites are arranged into a sequence $\{L_{\pi(n)}\}_{1 \leqslant n \leqslant M}$. For all $\boldsymbol{z} \in R(\boldsymbol{z}_1, \boldsymbol{z}_2)$, the value $s_{n-1}(\boldsymbol{z}) = \sum_{k=1}^{n-1} \hat{\varphi}[D_{\pi(k)}](\boldsymbol{z} \bullet \boldsymbol{\tau})$ is known at site $L_{\pi(n)}$, where $s_n(\boldsymbol{z}) = s_{n-1}(\boldsymbol{z}) + \hat{\varphi}[D_{\pi(n)}](\boldsymbol{z} \bullet \boldsymbol{\tau})$ is computed and moved to $L_{\pi(n+1)}$.

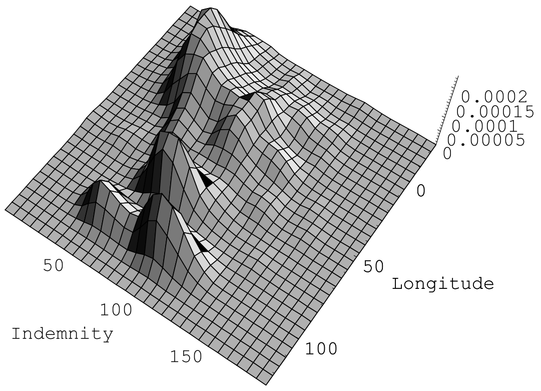
Star For all $\boldsymbol{z} \in R(\boldsymbol{z}_1, \boldsymbol{z}_2)$, at site L_n, the value $\hat{\varphi}[D_n](\boldsymbol{z} \bullet \boldsymbol{\tau})$ is computed and moved to a single helper.

Tree The master helper, the auxiliary helpers and the data sites are organized into a tree of order b, satisfying the following: the master helper is the root of the tree, the M data sites are exactly the leaves of the tree, the depths of any two leaves differ at most by one, and all internal nodes have exactly b children, except possibly one internal node of maximal depth, which has at least 2 and at most $b - 1$ children. A tree satisfying the properties above can be computed straightforwardly in linear time in M: Create a perfect tree of depth $\lfloor \log_b M \rfloor$, consisting of the master helper as root, $\lceil (M-1)/(b-1) \rceil - 1$ auxiliary helpers, and $\lfloor (b^{\lfloor \log_b M \rfloor + 1} - M)/(b-1) \rfloor$ data sites at depth $\lfloor \log_b M \rfloor$; then, add the remaining data sites (if any) as children of the auxiliary helpers at depth $\lceil \log_b M \rceil$, leaving at most one helper with less than b children. The minimum value of b is chosen by the initiator, which therefore knows in advance the maximum number of auxiliary helpers needed. At data site L_n, $1 \leqslant n \leqslant M$, for all $\boldsymbol{z} \in R(\boldsymbol{z}_1, \boldsymbol{z}_2)$, the value $\hat{\varphi}[D_n](\boldsymbol{z} \bullet \boldsymbol{\tau})$ is computed and moved to the parent. At any helper site (except the tree root) L_n, $n > M$, the value

$$s_n(\boldsymbol{z}) = \sum_{\substack{1 \leqslant k \leqslant M \\ L_k \in \mathrm{subtree}(L_n)}} \hat{\varphi}[D_k](\boldsymbol{z} \bullet \boldsymbol{\tau})$$

is computed.

In any of the partial orders above, the actual protocol among the sites can be implemented by stationary or mobile agents. In the following, assume A_{M+1} is the master helper's agent.



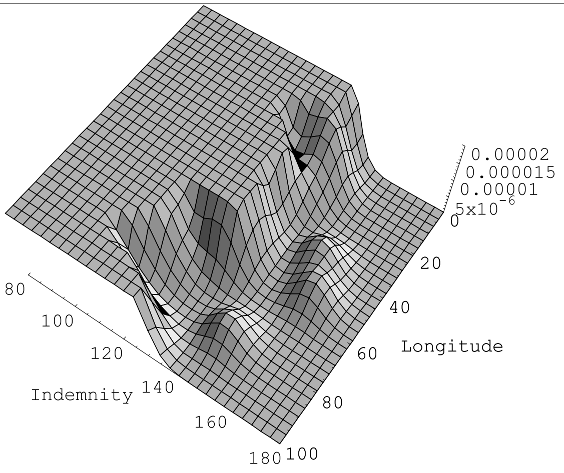


Fig. 6. Marginal kernel estimate ($h = h_r/2$) in the hyperplane $Longitude, Indemnity$ and inset showing isolated local maxima.

Sequence A_{M+1} selects an arrangement π such that $\pi(1), \ldots, \pi(M)$ is a random permutation of $1, \ldots, M$, and $\pi(M+1) = M+1$.

Stationary agents For $2 \leqslant n \leqslant M$, A_{M+1} sends $A_{\pi(n)}$ the addresses of sites $L_{\pi(n-1)}$, $L_{\pi(n+1)}$. To $A_{\pi(1)}$, the master sends its own address, the address of $L_{\pi(2)}$, and an empty list of samples. $A_{\pi(1)}$ waits for a list of samples from the master, adds its local list of samples to it and sends the result to $A_{\pi(2)}$. Each $A_{\pi(n)}$ waits for a list of samples from $A_{\pi(n-1)}$, $2 \leqslant n \leqslant M$, adds its local list of samples to it and sends the result to $A_{\pi(n+1)}$. The master waits for the list of samples from $A_{\pi(M)}$, and sends it to A_n, for $1 \leqslant n \leqslant M$.

Mobile agents Agent A_{M+1} initializes an empty list of samples and moves to $L_{\pi(1)}$. When residing at site $L_{\pi(n)}$ ($1 \leqslant n \leqslant M$), A_{M+1} requests the local agent $A_{\pi(n)}$ to locally send the list of samples, adds it to the sum in its data space, and moves to $L_{\pi(n+1)}$. Finally A_{M+1} sends the sum to all agents A_n, $1 \leqslant n \leqslant M$.

Star Assume agent A_{M+1} was designated as master.

Stationary agents A_{M+1} waits for the list of local samples from each of the A_n, $1 \leqslant n \leqslant M$, adds the lists and sends the result to A_n, for $1 \leqslant n \leqslant M$.

Mobile agents For $n = 1, \ldots, M$ agent A_{M+1} moves to site L_n, requests the local agent A_n to locally send the list of samples, adds it to the sum in its data space, and moves back to L_{M+1}.

Tree Assume site agents $A_{M+2}, \ldots, A_{M+H}$ were designated as auxiliary helpers. A_{M+1} receives the list $L_{M+2}, \ldots, L_{M+H}$ of helper sites and arranges it into a tree, as described earlier in this section. To minimize collusions, A_{M+1} sends each helper agent A_{M+k} only the references to its parent and its children. If the agents are mobile, A_{M+1} also sends each helper agent A_{M+k} the references to its siblings.

Stationary agents Every agent A_{M+k}, $1 \leqslant k \leqslant H$, waits for the list of local samples from all A_j residing at the children of L_{M+k}, and adds the lists. Then A_{M+k} sends the result to its parent, if $k > 1$; the master A_{M+1} sends the list to A_n, for $1 \leqslant n \leqslant M$.

Mobile agents For $k = 1, \ldots, H$, for every child L_j of L_{M+k}, agent A_{M+k} moves to L_j and requests the local agent A_j to locally send the list of samples. It adds the list to the sum in its data space, and moves back to L_{M+k}.

The transmission complexity of KDEC under any of the protocols above is $O(hops \cdot |\{\boldsymbol{z} \in \mathbb{Z}^n \cap R(\boldsymbol{z}_1, \boldsymbol{z}_2)\}|)$, where $hops$ is the total number of hops of the network.

4.4. Issues of data security in KDEC

The various KDEC protocols transmit only density estimates outside the physical data space of a site. However, avoiding network data transmission does not guarantee that data cannot be disclosed by appropriate techniques. In fact, a large body of research in statistical databases has been motivated by the observation that many types of aggregates contain implicit information that allows for data reconstruction, possibly obtained by posing carefully designed sequences of aggregate queries [6]. To the best of our knowledge, the problem of data reconstruction from kernel density estimates has not been investigated. Moreover, sites in open environments can be guaranteed neither to run provably correct code nor to be secure, and they can release information the protocol has been designed to hide. Finally, unsecured mobile agents are vulnerable to malicious servers which attempt to tamper with their code and data to their advantage [29,36]. Therefore, the security of the approach must be carefully evaluated. In particular, the following questions must be answered:

1. Are kernel density estimates secure, i.e., is it possible to disclose the original dataset from a kernel density estimate computed on that dataset?
2. Can a site improve the accuracy of the data disclosure by exploiting knowledge available to all sites taking part in the protocol?
3. Can a subset of the sites participating to an execution of the protocol improve the accuracy of their disclosure attempt by forming a coalition?

In the following, we will try to answer the questions above and discuss the strengths and weaknesses of potential countermeasures. Discussing security issues in agent platforms is outside the scope of this work. We assume authorization and authentication mechanisms to run mobile code are secure and authenticated and secure communication protocols between sites are used. Consequently, a site always knows the sender or receiver of any message.

4.4.1. Inference attacks on kernel density estimates

In any of the proposed protocol, all parties learn the global kernel density estimate. Even if all parties strictly follow the protocol, the privacy of data is not guaranteed unless it can be argued that density estimates do not contain enough implicit information to allow for reconstructing the data. Two techniques may apply: Iterative disclosure and non-linear system solving.

Iterative disclosure A simple form of attack consists in searching the density estimate or its derivatives for discontinuities. For example, if $K(\boldsymbol{x})$ equals $w(\boldsymbol{x})$, then the distance between discontinuities of the estimate on the same axis equals the distance between data objects on that axis. Therefore the relative positions of objects are known. If the window width h is known, then all objects are known since every discontinuity is determined by an object lying at distance h from the discontinuity, on the side where the estimate is greater. Kernels whose derivative is not continuous, such as the Epanechnikov kernel, allow for similar inferences. For the general case, in [4] a simple algorithm has been presented, which partially discloses the data by reconstructing one object at the time.

Non-linear system solving Let $g : \mathbb{R}^d \to \mathbb{R}$ be extensionally equal to a kernel estimate $\hat{\varphi}[D](\boldsymbol{x})$, that is, $(\forall \boldsymbol{x} \in \mathbb{R}^d)\ g(\boldsymbol{x}) = \hat{\varphi}[D](\boldsymbol{x})$ holds, and let K and h be known. The problem is to compute $\boldsymbol{x}_i$, $i = 1, \ldots, N$. In an inference attack to a KDEC-based clustering, $g(\boldsymbol{x})$ will be a reconstructed estimate effectively computed by the interpolation formula Eq. (7).

An attacker can select Nd space objects $\boldsymbol{y}_j$ and attempt to solve a system of equations

$$\sum_{i=1}^{N} K\left(\frac{1}{h}(\boldsymbol{y}_j - \boldsymbol{x}_i)^T(\boldsymbol{y}_j - \boldsymbol{x}_i)\right) = g(\boldsymbol{y}_i),$$
$$j = 1, \ldots, Nd. \qquad (8)$$

Although the resulting system of equations is non-linear (assuming the normal kernel) and contains a large number of unknowns, several efficient methods to solve non-linear systems of equations have been proposed in the literature [24]; therefore solving system Eq. (8) is not necessarily unfeasible even for large datasets. On the other hand, the accuracy and speed of convergence of such methods still depend on the structure of the problem at hand. Preliminary investigations have shown that an attacker is likely to incur slow speed of convergence or a large number of spurious solutions when trying to solve systems like Eq. (8).

4.4.2. Protocol attacks

Even if one of the agents could disclose all data objects, by inference alone it could not discover at which site a given reconstructed object resides. However, in each of the protocols of Section 4.3, semi-honest [12] or malicious behaviours by one agent or a coalition of agents could substantially reduce the uncertainty about the location of disclosed objects. In the following we describe potential attack scenarios in each of the protocols.

Sequence protocol In the sequence protocol with stationary agents, the n-th agent in the arrangement $A_{\pi(n)}$ knows the density estimate of the data at sites $L_{\pi(1)}, \ldots, L_{\pi(n-1)}$ and, by difference, the density estimate of the data at $L_{\pi(n+1)}, \ldots, L_{\pi(M)}$. Therefore a semi-honest agent can infer the data objects but the amount of information that is learned by $A_{\pi(n)}$ on the location of the objects depends on the position of $L_{\pi(n)}$ in the arrangement. In particular, if $\pi(n) = M - 1$ or $\pi(n) = 2$, objects can be assigned to $L_{\pi(M)}$ or $L_{\pi(1)}$. To refine their knowledge, two malicious agents $A_{\pi(n-p)}$, $A_{\pi(n+1)}$, $0 < p < n < M$ can collude and, by difference, calculate the estimate of the union of the datasets at $L_{\pi(n-p+1)}, \ldots, L_{\pi(n)}$. If $p = 1$, the reconstructed objects can be assigned to $L_{\pi(n)}$.

In the mobile agent case, the protocol is secure if agents are assumed semi-honest, as the partial density estimates are stored in the mobile agent's data space only. If local agents are potentially malicious, then data and code of the mobile agent can be tampered with and virtually all information that is learned by the mobile agent at any site could be learned by some malicious agent. For example, the malicious agent, say $A_{\pi(e)}$, could read the summation of samples of already visited sites $s_{e-1}(\boldsymbol{z}) = \sum_{1 \leqslant k < e} \hat{\varphi}[D_{\pi(k)}](\boldsymbol{z} \bullet \boldsymbol{\tau})$ for all $\boldsymbol{z} \in R(\boldsymbol{z}_1, \boldsymbol{z}_2)$. Worse, the agent's code could be modified to forward to $A_{\pi(e)}$ any local density estimate obtained after visiting its site.

Example. Suppose the sequence protocol with mobile agents is used. The master helper's agent A_9 randomly selects the permutation $5, 1, 4, 2, 6, 8, 3, 7$ and starts its navigation in the sequence. Assume A_1, A_2, A_3 are malicious, i.e., they actively attempt to read the data space of A_9 and tamper with its code, exploiting privileged accounts in their respective sites. A_1 reads the samples of L_5 from A_9's data. Then, it forwards the sum of its samples and L_5 to A_2. A_2 reads the cumulative samples of sites L_5, L_1, L_4 from A_9's data, and, by difference, learns the samples of L_4. Then, A_2 forwards to A_3 the sum of samples of L_5, L_1, L_4, L_2, and tampers with the code of the mobile agent, reprogramming it to keep a separate copy of the current sum of samples, before adding the samples of L_8. A_3 reads the sum of samples of preceding sites from A_9 and the separate copy, and learns the samples of L_6 and L_8. When the global sum is transmitted to all parties, A_3 learns the samples of L_7.

Star protocol In the stationary case, the helper has complete knowledge of the global density estimate and the local density estimates. One or more data sites could collude with the helper and have it send them the local density estimates of each of the remaining sites, from which local data could be inferred and correctly assigned. Without the collusion of the helper, the malicious sites can only obtain the density estimate of the remaining sites and infer the data without any assignment (unless there are only two data sites).

In the mobile agent case, code tampering is useless as the agent code is loaded once and returns to the originator immediately after collecting the samples. As communications are assumed authenticated, a malicious data site cannot impersonate the helper and send the agent to another data site to collect the samples of its local estimate. The attack scenario is therefore similar to the stationary case.

Tree protocol In the stationary case, a helper site and its children data sites form a star and the collusion scenario of the previous paragraph applies. To infer and assign data objects outside the local star, the malicious agents must collude with other helper agents. For instance, the malicious agents could collude with the master helper to learn the structure of the tree, which is not published, and thus attempt to collude with helpers having data sites as children of their sites. Alternatively, the malicious agents could attempt to set up a path of colluded agents until a helper agent having data sites as children of its site is encountered, by iteratively asking the last colluded agent a reference to the parent or a child. The scenarios in the mobile case are similar (see the paragraph on the Star protocol).

Example. Assume the initiator has set the minimum tree order b to 3, and, therefore, has collected 3 auxiliary helpers: L_{10}, L_{11}, and L_{12}. The master helper's agent A_9 computes the tree and finds out that one of the auxiliary helpers has exactly two children. As $b = 3$, reassigning any data site is useless. Therefere, A_9 sets the tree order b to 4 and recomputes the tree. The tree has now the following edges: (L_9, L_{10}), (L_9, L_{11}), (L_9, L_3), (L_9, L_7), (L_{10}, L_5), (L_{10}, L_1), (L_{10}, L_4), (L_{10}, L_2), (L_{11}, L_6), (L_{11}, L_8). Again, an auxiliary helper, L_{11}, has exactly two children. In this case, however, the parent of a data site, other than L_6, L_8, can be changed to L_{11}. Assume (L_{10}, L_2) is changed to (L_{11}, L_2). Let us suppose A_1, A_2, A_3 are malicious. If A_1 and A_2 collude with A_{10} and A_{11}, they learn the sample sums $\sum_{k \in \{1,4,5\}} \hat{\varphi}[D_k](\boldsymbol{z} \bullet \boldsymbol{\tau})$, $\sum_{k \in \{2,6,8\}} \hat{\varphi}[D_k](\boldsymbol{z} \bullet \boldsymbol{\tau})$. By difference, the sample sums $\sum_{k \in \{4,5\}} \hat{\varphi}(\boldsymbol{z} \bullet \boldsymbol{\tau})$, $\sum_{k \in \{6,8\}} \hat{\varphi}(\boldsymbol{z} \bullet \boldsymbol{\tau})$ are learned by malicious agents A_1 and A_2, respectively. When the global sum of samples is returned to every agent, malicious agent A_3 learns by difference the samples of L_7, i.e., $\hat{\varphi}[D_7](\boldsymbol{z} \bullet \boldsymbol{\tau})$, for all $\boldsymbol{z} \in R(\boldsymbol{z}_1, \boldsymbol{z}_2)$. Therefore, if the data objects can be reconstructed from the sample sums above, D_7, $D_4 \cup D_5$, and $D_6 \cup D_8$ are known to the malicious agents. The agents cannot however decide whether a reconstructed data object in $D_4 \cup D_5$ belongs to L_4 or L_5, and whether a reconstructed data object in $D_6 \cup D_8$ belongs to L_6 or L_8.

4.4.3. *Countermeasures*

The attacks in Sections 4.4.1 and 4.4.2 exploit two types of vulnerabilities: (i) given a kernel density estimate, data objects are reconstructible and (ii) partial density estimates are not secured against semi-honest or malicious agents. Countermeasures to these attacks which may apply have been investigated in the literature on both Secure Multiparty Computation and Mobile Cryptography.

Secure multiparty computation The goal of *Secure Multiparty Computation* (SMC) [12] is to allow two or more parties to compute the value of a function on an input which is distributed among the parties in such a way that at the end of the computation each party knows nothing except the value of the function and its own input. A simple SMC technique which can be applied to KDEC with the sequential protocol is *Secure Sum* [3]: For all $\boldsymbol{z} \in R(\boldsymbol{z}_1, \boldsymbol{z}_2)$, the helper agent A_{M+1} generates randomly $r(\boldsymbol{z}) \in [0,1]$ and sends $r(\boldsymbol{z})$ to $A_{\pi(1)}$. For $1 \leqslant n \leqslant M$, agent $A_{\pi(n)}$ receives $v_{n-1} = \big(r(\boldsymbol{z}) + s_{n-1}(\boldsymbol{z})\big) \bmod 1$ and sends to $A_{\pi(n+1)}$ the value $v_n = \big(r(\boldsymbol{z}) + s_n(\boldsymbol{z})\big) \bmod 1 = \big(r(\boldsymbol{z}) + s_{n-1}(\boldsymbol{z}) + \hat{\varphi}[D_{\pi(n)}](\boldsymbol{z} \bullet \boldsymbol{\tau})\big) \bmod 1 = \big(v_{n-1} + \hat{\varphi}[D_{\pi(n)}](\boldsymbol{z} \bullet \boldsymbol{\tau})\big) \bmod 1$. Finally, A_{M+1} sends to $A_{\pi(n)}$, $1 \leqslant n \leqslant M$, the difference $v_M - r(\boldsymbol{z})$. Both in the stationary and mobile protocol, $A_{\pi(n)}$, $1 \leqslant n \leqslant M+1$, learns nothing about the density estimate of the other sites or group of sites, even if agents are malicious, since the samples of partial estimates that are moved between sites are uniformly distributed in $[0,1]$. (The list of samples of the global density estimate is the result of the computation and is known by all sites, independent of the way it is computed.) Consequently, in the stationary case a semi-honest agent cannot assign reconstructed objects, independent of its position in the arrangement.

The secure sum is vulnerable to collusion attacks. Agents can easily learn their relative positions in the arrangement by comparing successors and predecessors, and colluded agents $A_{\pi(n-1)}$ and $A_{\pi(n+1)}$ can easily compute $\hat{\varphi}[D_{\pi(n)}](\boldsymbol{z} \bullet \boldsymbol{\tau})$ by comparing v_n an v_{n-1}. Dividing each $r(\boldsymbol{z})$ into shares and permuting the arrangement for each share prevents malicious agents from having the same neighbour [3]. Obviously, since the crucial secrets $r(\boldsymbol{z})$ are generated by the master helper's agent, any security failure of the master helper or collusion of its agent makes the Secure Sum approach useless.

In the mobile agent case, the above type of collusion is not possible as long as the mobile agent keeps its path secret. However, when the mobile agent moves to a new site, the destination address must be shared with the host.

Mobile cryptography In [29] Sander and Tschudin introduce the term "mobile cryptography" to denote fully software based cryptographic solutions to the problem of designing secure mobile programs, and define two basic scenarios: Computing with Encrypted Data (CED) and Computing with Encrypted Functions (CEF). In the CED scenario, Alice wants to know the output of Bob's private algorithm for function f on her private input x; nothing else must be learned by Alice or Bob. In the CEF scenario, Alice wants to know the output of her private algorithm for function f at Bob's private input x; nothing else must be learned by Alice or Bob. It turns out that, when f is a polynomial, both CED and CEF can be implemented using a *Homomorphic Encryption Scheme* (HES). An algebraic HES is an encryption function $E: R \to S$, where R and S are rings, such that $E(x \Diamond y)$ can be efficiently computed from $E(x)$ and $E(y)$, where $\Diamond$ is a ring operation.

As density estimate samples are summed, in the sequel we assume E to be only additively homomorphic, that is, there exists an efficient algorithm `Plus` to compute $E(x+y)$ from $E(x)$ and $E(y)$. A CED approach in KDEC applies naturally both to the sequential protocol with mobile agents and the star protocol with stationary agents, as follows. Agent A_n, $1 \leqslant n \leqslant M$, sends $E\big(\hat{\varphi}[D_n](\boldsymbol{z} \bullet \boldsymbol{\tau})\big)$ to the agent which computes the partial sum of samples, i.e. A_{M+1}. Algorithm `Plus` must be made known to A_{M+1} in advance, e.g. by A_1. The helper A_{M+1} uses `Plus` to sum encrypted samples: $E\big(s_M(\boldsymbol{z})\big) = E\big(\sum_{k=1}^{M} \hat{\varphi}[D_k](\boldsymbol{z} \bullet \boldsymbol{\tau})\big) = \texttt{Plus}\big(E(\hat{\varphi}[D_1](\boldsymbol{z} \bullet \boldsymbol{\tau})), \texttt{Plus}(\ldots, E(\hat{\varphi}[D_M](\boldsymbol{z} \bullet \boldsymbol{\tau})))\big)$. Finally A_{M+1} sends $E\big(s_M(\boldsymbol{z})\big)$ to all agents A_n, $1 \leqslant n \leqslant M$. Such implementation of CED in KDEC effectively hides the samples from the helper agent, however it is not effective against collusions between the helper and malicious agents, as E must be known by all A_n, which could decrypt any set of samples maliciously forwarded by A_{M+1}.

4.4.4. *Untrustworthy helpers*

Recently, we have witnessed increasing interest towards trustworthiness and referrals as a means to ascer-

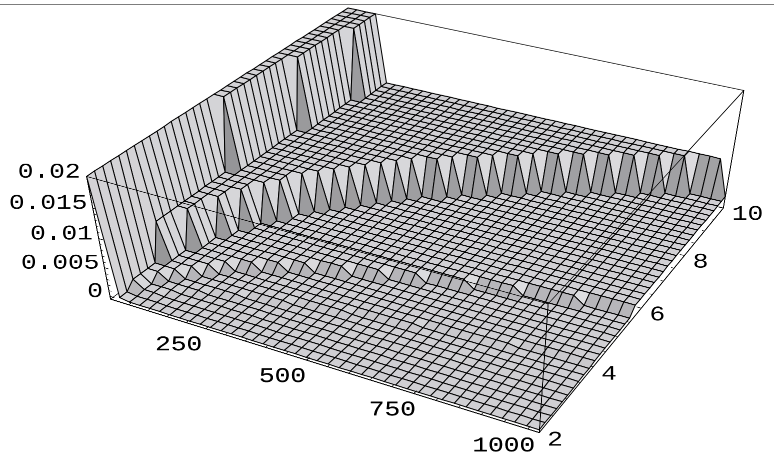


Fig. 7. Graph of the upper bound on $P(m, b)$ ($p = 0.2$, $M = 1024$).

tain the degree of trustworthiness of an agent [37,38]. In the following we assume the agent community supports referrals about an agent's reputation as a helper. We model a helper's *reputation* as a binary random variable with probability p and $1 - p$, where p is the probability that the helper will behave as untrustworthy in the forthcoming interaction. We assume that p can be derived from referrals during the initial negotiation phase.

One way to measure the risk of data privacy infringement in KDEC is the probability $P(m)$ that an agent's local data are not protected against at least m other participating agents. If only one helper is used, it is apparent that $P(m) = p$, for every m. If the helpers and the site agents are arranged to form a complete b-ary tree, it is not difficult to see that $P(m, b) \leqslant p^{\lceil \log_b(m+1) \rceil}$. Such an upper bound decreases with m and increases with b according to intuition (see Fig. 7), however, it is worth noting that the lower b, the higher the chance that an agent could incur coalition attacks. Notably, the best performance against untrustworthiness is obtained by a binary tree, which should always be rejected by any site agent since it gives complete information to each member of any pair of siblings in the tree about the other member's density estimate. Techniques to find a trade-off are under investigation.

5. Related work

Only a few approaches to solve the problem of homogeneous distributed data clustering are available to date.

In [21] a solution to the problem of homogeneous distributed clustering under an information theoretic privacy constraint is proposed.

A global probabilistic model of the data is built by combining the parameters of the models computed at the different sources. The approach differs from ours in that a particular parametric family of models must be assumed, whereas the KDEC scheme is non-parametric.

In [33] the k-windows approach to data clustering is extended to handle the distributed case. Initially, the k-windows algorithm is executed locally at every data site. Then the generated windows are sent to a central node which is responsible for the final merging of the windows and the construction of global clusters.

In [19] the abstract KDEC scheme is described in more detail. However, its agent implementations and their security issues are not discussed. A shorter version of the present paper is contained in [20]. Possible agent implementations of the KDEC scheme are briefly sketched and the possibility of inference attacks on kernel estimates is suggested. The impact of untrustworthy helpers is also considered. However, no countermeasures have been proposed. An algorithm to perform an inference attack on a kernel estimate is presented in [4]. It has been experimentally proven that the algorithm effectively recovers objects located in the tails of a kernel estimate.

6. Conclusion and future work

The ever growing amount of data that are stored in distributed form over networks of heterogeneous and autonomous sources poses several problems to research in knowledge discovery and data mining, such as communication minimization, autonomy preservation, scalability, and privacy protection. In this paper, we have reviewed prominent approaches in the literature and discussed the benefits that agent-based data mining architectures provide in coping with such problems, and the related issues of data security and trustworthiness. We have presented a scheme for agent-based distributed data clustering based on density estimation, which exploits information theoretic sampling to minimize communications between sites and protect data privacy by transmitting density estimation samples instead of data values outside the site of origin. Potential privacy violations due to inference and coalition attacks and issues of trustworthiness have been discussed. Ongoing research will focus in particular on the investigation of inference attacks on kernel density estimates exploiting recent advances in numerical methods for the solution of non-linear systems of equations, and the analysis of risks of security and privacy violations in DDM environments. Finally, it is planned to implement a multiagent system for KDEC-based homogeneous DDC able to work in peer-to-peer networks and grid computing systems.

References

[1] S. Bailey, R. Grossman, H. Sivakumar and A. Turinsky, *Papyrus: a system for data mining over local and wide area clusters and super-clusters*, In Proc. Conference on Supercomputing, 63, ACM Press, 1999.

[2] M.-S. Chen, J. Han and P.S. Yu, Data mining: an overview from a database perspective, *IEEE Trans. On Knowledge And Data Engineering* **8** (1996), 866–883.

[3] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin and M.Y. Zhu, Tools for privacy preserving distributed data mining, *ACM SIGKDD Explorations Newsletter* **4**(2) (2002), 28–34.

[4] J. Costa da Silva, M. Klusch, S. Lodi and G. Moro, Inference attacks in peer-to-peer homogeneous distributed data mining, in: *Proceedings of the 16th European Conference on Artificial Intelligence* (*ECAI 2004*), R. López de Mántaras and L. Saitta, eds, Valencia, Spain, 22–27 August 2004, pp. 450–454, IOS Press.

[5] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise*, in Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, OR, 1996, 226-231.

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

[7] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*, AAAI Press/MIT Press, 1996.

[8] T. Finin, Y. Labrou and J. Mayfield, KQML as an agent communication language, in: *Software Agents*, J. Bradshaw, ed., MIT Press, 1997, pp. 291–316.

[9] I.T. Foster, N.R. Jennings and C. Kesselman, *Brain meets brawn: Why grid and agents need each other*, In 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), 19–23 August 2004, 8–15, New York, NY, USA, 2004. IEEE Computer Society.

[10] Foundation for Intelligent Physical Agents. FIPA Communicative Act Library Specification, Aug. 2001. Published on August 10th, 2001, http://www.fipa.org.

[11] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, New York, NY, USA, 1972.

[12] O. Goldreich, Secure multi-party computation. http://www.wisdom.weizmann.ac.il/∼oded/pp.html October 2002, *Final* (*incomplete*) *Draft*, version 1. 4.

[13] D. Heckerman, H. Mannila, D. Pregibon and R. Uthurusamy, eds, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, Newport Beach, California, USA, 1997. AAAI Press.

[14] J.R. Higgins, *Sampling Theory in Fourier and Signal Analysis*, Clarendon Press, Oxford, 1996.

[15] A. Hinneburg and D.A. Keim, *An efficient approach to clustering in large multimedia databases with noise*, In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), 58–65, New York City, New York, USA, 1998, AAAI Press.

[16] H. Kargupta, I. Hamzaoglu and B. Stafford, *Scalable, distributed data mining using an agent-based architecture*, In Heckerman et al. [13], 211–214.

[17] H. Kargupta, B. Park, D. Hershberger and E. Johnson, *Advances in Distributed and Parallel Knowledge Discovery*, chapter 5, Collective Data Mining: A New Perspective Toward Distributed Data Mining, 131–178. AAAI/MIT Press, 2000.

[18] M. Klusch, Information agent technology for the internet: A survey. Data and Knowledge Engineering, Special Issue on Intelligent Information Integration, *Elsevier Science* **36**(3) (2001), 337–372.

[19] M. Klusch, S. Lodi and G. Moro, *Distributed clustering based on sampling local density estimates*, In Proceedings of the 19th International Joint Conference on Artificial Intelligence, IJCAI-03, 485–490, Acapulco, Mexico, August 2003. AAAI Press.

[20] M. Klusch, S. Lodi and G. Moro, *The role of agents in distributed data mining: Issues and benefits*, In Proceedings of the 2003 IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003), 211–217, Halifax, Canada, October 2003. IEEE Computer Society -Web Intelligence Consortium (WIC), IEEE Computer Society Press.

[21] S. Merugu and J. Ghosh, Privacy-preserving distributed clustering using generative models, In Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), 19–22 December 2003, Melbourne, Florida, USA, *IEEE Computer Society* (2003).

[22] G. Moro, G. Monti and A.M. Ouksel, Merging G-Grid P2P systems while preserving their autonomy, in: *Proceedings of the MobiQuitous '04 Workshop on Peer-to-Peer Knowledge Management* (*P2PKM 2004*), I. Zaihrayeu and M. Bonifacio, eds, August 22, 2004, volume 108 of CEUR Workshop Proceedings, Boston, MA, USA, 2004. CEUR-WS. org.

[23] G. Moro and C. Sartori, Incremental maintenance of multi-source views, in: *Proceedings of 12th Australasian Database Conference, ADC 2001, Brisbane, Queensland*, M.E. Orlowska and J. Roddick, eds, Australia, IEEE Computer Society, February 2001, pp. 13–20.

[24] J. Nocedal and S.J. Wright, *Numerical Optimization*, Springer, 2000.

[25] A.M. Ouksel and G. Moro, G-Grid: A class of scalable and self-organizing data structures for multi-dimensional querying and content routing in P2P networks, in: *Agents and Peer-to-Peer Computing, Second International Workshop, AP2PC 2003*, G. Moro, C. Sartori and M.P. Singh, eds, July 14, 2003, Revised and Invited Papers, volume 2872 of Lecture Notes in Computer Science, Melbourne, Australia, 2004. Springer, pp. 123–137.

[26] M.P. Papazoglou and G. Schlageter, *Cooperative Information Systems – Trends and Directions*, Academic Press Ltd, London, UK, 1998.

[27] M. Prasad and V. Lesser, *Learning situation-specific coordinating in cooperative multi-agent systems*, Autonomous Agents and Multi-Agent Systems, 1999.

[28] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Schenker, *A scalable content-addressable network*, In SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, New York, NY, USA, 2001. ACM Press, 161–172.

[29] T. Sander and C.F. Tschudin, Protecting mobile agents against malicious hosts, in: Mobile Agents and Security, G. Vigna, ed., volume 1419 of Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, 1998, pp. 44–60.

[30] S. Sen, A. Biswas and S. Gosh, *Adaptive choice of information sources*, In Proc. 3rd International workshop on Cooperative Information Agents. Springer, 1999.

[31] B.W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London, 1986.

[32] S.J. Stolfo, A.L. Prodromidis, S. Tselepis, W. Lee, D.W. Fan and P.K. Chan, *JAM: Java agents for meta-learning over distributed databases*, In Heckerman et al. [13], 74–81.

[33] D.K. Tasoulis and M.N. Vrahatis, *Unsupervised distributed clustering*, In Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks, Innsbruck, Austria, 2004.

[34] W. Theilmann and K. Rothermel, *Disseminating mobile agents for distributed information filtering*, In Proc. of 1st International Sympos. on Mobile Agents, 152–161, IEEE Press, 1999.

[35] M. Wooldridge, Intelligent agents: The key concepts, in: *Multi-Agent-Systems and Applications*, (vol. 2322 of LNCS), V. Marík, O. Stepánková, H. Krautwurmova and M. Luck, eds, Springer-Verlag, 2002, pp. 3–43.

[36] B.S. Yee, *A sanctuary for mobile agents*, In Secure Internet Programming, 1999, 261–273.

[37] P. Yolum and M.P. Singh, An agent-based approach for trustworthy service location, in: *Agents and Peer-to-Peer Computing, First International Workshop, AP2PC 2002, July, 2002, Revised and Invited Papers*, (vol. 2530 of Lecture Notes in Computer Science), G. Moro and M. Koubarakis, editors, Bologna, Italy, 2003. Springer, pp. 45–56.

[38] B. Yu and M.P. Singh, *Emergence of agent-based referral networks*, In Proc. AAMAS 2002, ACM Press, 2002, 1268–1269.

[39] N. Zhong, Y. Matsui, T. Okuno and C. Liu, Framework of a multi-agent kdd system, in: *Proc. of Intelligent Data Engineering and Automated Learning – IDEAL 2002*, H. Yin, N.M. Allinson, R. Freeman, J.A. Keane and S.J. Hubbard, eds, Third International Conference, Manchester, UK, (vol. 2412 of Lecture Notes in Computer Science), Springer-Verlag, 2002, pp. 337–346.