The Role of Agents in Distributed Data Mining: Issues and Bene£ts

Matthias Klusch
Deduction and Multiagent Systems
German Research Centre for Arti£cial Intelligence
klusch@dfki.de

Stefano Lodi
Department of Electronics, Computer Science, and Systems
University of Bologna
slodi@deis.unibo.it

Gianluca Moro
Department of Electronics, Computer Science, and Systems
University of Bologna
gmoro@deis.unibo.it

Abstract

The increasing demand to extend data mining technology to data sets inherently distributed among a large number of autonomous and heterogeneous sources over a network with limited bandwidth has motivated the development of several approaches to distributed data mining and knowledge discovery, of which only a few make use of agents. We brie¤y review existing approaches and argue for the potential added value of using agent technology in the domain of knowledge discovery, discussing both issues and bene£ts. We also propose an approach to distributed data clustering, outline its agent-oriented implementation, and examine potential privacy violating attacks in which agents may incur.

1. Introduction

Mining information and knowledge from huge data sources such as weather databases, £nancial data portals, or emerging disease information systems has been recognized by industrial companies as an opportunity of major revenues from applications such as warehousing, process control, and customer services. *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 [4]. 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 classi£cation, generalization taking the data cube or attribute-oriented induction approach, and mining of association rules [2]. The increasing demand to scale up to massive data sets inherently distributed over a network with limited bandwidth and computational resources available motivated the development of methods for parallel (PKD) and distributed knowledge discovery (DKD) [8]. The related pattern extraction problem in DKD is referred to as *distributed data mining* (DDM). The main problems any approach to DDM is challenged to cope with concern issues of autonomy, privacy, and scalability.

Most of the existing DM techniques were originally developed for centralized data and need to be modi£ed for handling the distributed case. As a consequence, one of the most widely used approach 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 [11]. 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 infeasible.

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 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 warehouses may be prohibitively expensive in terms of costs of 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 [7]. This in turn leads us to the question whether there is any real added value of using concepts from agent technology [9, 17] 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, classi£cation (regression), and compression, but only a few of them make use of intelligent agents at all. In general, the inherent feature of software agents of being autonomous, capable of adaptive and deliberative reasoning seems to £t quite well with the requirements of coping with the above mentioned problems and challenges of DDM. Autonomous data mining agents as a special kind of information agents [9] 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, heterogeneous and massive data environments appear to be quite a natural vision for the near future to be realized.

In this paper we brie¤y review and classify existing DDM systems and frameworks according to some criteria in Section 2. This is followed by a brief discussion on the bene£ts 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 5 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 [8] 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 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 ¤ow between the agents.

PADMA [7] deals with the problem of DDM from homogeneous data sites. Partial data cluster models are £rst 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 classi£cation, and web based information visualization.

JAM [15] is a Java-based multi-agent system designed to be used for meta-learning DDM. Different learning classi£ers such as Ripper, CART, ID3, C4.5, Bayes, and WEPBLS 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 classi£cation model and different agents build classi£ers using different techniques. JAM also provides a set of meta-learning agents for combining multiple models learnt at different sites into a meta-classi£er that in many cases improves the overall predictive accuracy. Once the combined classi£ers 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 £nal 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 which is discovered out of data at different geographically distributed network sites with a minimum amount of network communication, and 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 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 [12].

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 £ltering agents in distributed data environments are encouraging [16].

Multi-strategy DDM. For some complex application settings an appropriate combination of multiple data mining technique may be more bene£cial than applying just one particular one. DM agents may learn in due course of their deliberative actions which one to choose depending on the type of data retrieved from different sites and mining tasks to be pursued. The learning of 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 [13].

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 bene£t from the additional opinions of other DM agents. Meta-learning techniques may be used to perform mining homogeneous, distributed data. 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-widely 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 no matter into 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 least privilege at a data source, that means endowing subjects with only enough permissions 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 con£dentiality 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. Not any of the existing DDM systems, agent-based or not, is capable of coping with this inference problem in the domain of secure DDM.

4. A Scheme for Distributed Data Clustering

4.1. Density Estimation Based Clustering

Cluster analysis is a 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 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 so-called probability density function from which the given data set is assumed to have arisen [3, 6, 14]. One important family of methods requires the computation of a non-parametric density estimate known as *kernel estimator*.

Let us assume a set $S = \{\vec{x}[i] \mid i = 1, \ldots, N\} \subseteq \mathbb{R}^n$ of data objects. Kernel estimators originate from the intuition that the higher the number of neighbouring data objects $\vec{x}[i]$ of some given space object $\vec{x} \in \mathbb{R}^n$, the higher the density at this object $\vec{x}$. However, there can be many ways of weighting the in¤uence of data objects. Kernel estimators use a so called *kernel function*, that is, real-valued, non-negative, non-increasing function $K(x)$ on $\mathbb{R}$ which has £nite integral over $\mathbb{R}$. Prominent examples of kernel functions are the

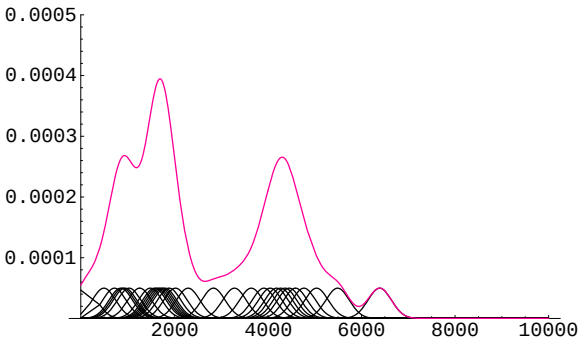

Figure 1. Kernel estimate showing Gaussian component kernels (h=250, N=32)

square pulse function $\frac{1}{4}(sign(x+1)-sign(x-1))$, and the Gaussian function $\frac{1}{\sqrt{2\pi}}\exp(-\frac{1}{2}x^2)$. A *kernel-based density estimate* $\hat{\varphi}_{K,h}[S](\cdot)\colon \mathbb{R}^n \to \mathbb{R}_+$ is de£ned, modulo a normalization factor, as the sum over all data objects $\vec{x}[i]$ in S of the distances $d(\vec{x},\vec{x}[i])$ between $\vec{x}[i]$ and $\vec{x}$, scaled by a factor h, called *window width*, and weighted by the kernel function K:

$$\hat{\varphi}_{K,h}[S](\vec{x}) = \sum_{i=1}^{N} K\left(\frac{d(\vec{x},\vec{x}[i])}{h}\right). \qquad (1)$$

The window width h controls the smoothness of the estimate, whereas K determines the decay of the in¤uence of a data object according to the distance. An example of kernel estimate in $\mathbb{R}$ showing the Gaussian component kernels is shown in Figure 1. Notice that in practice it is not necessary to compute N distances for calculating the estimate at a given object $\vec{x}$. In fact, the value of commonly used kernel functions is negligible or zero at distances larger than a few h units. In DE-clustering, the kernel estimate of a data set has been used for the discovery of many types of density-based clusters [3, 6, 14]. One simple type is the so-called *center-de£ned* 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 £nd the local maximum representing the object's cluster [6, 10, 14].

4.2. KDEC-based Distributed Data Clustering

We de£ne the problem of *homogeneous distributed data clustering* as follows. Let $S = \{\vec{x}[i] \mid i = 1,\ldots,N\} \subseteq \mathbb{R}^n$ be a data set of objects. Let $L_j, j = 1,\ldots,M$ be a £nite set of *sites*. Each site L_j stores one data set D_j of size N_j. It will be assumed that $S = \bigcup_{j=1}^{M} D_j$. Let $\mathcal{C} = \{C_k\} \subseteq 2^S$ be a clustering of S, whose elements are pairwise disjoint. The DDC problem is to £nd for $j = 1,\ldots,M$, a site clustering $\mathcal{C}_j$ residing in the data space of L_j, such that $\mathcal{C}_j = \{C_k \cap D_j \mid k = 1,\ldots,|\mathcal{C}|\}$ (*correctness requirement*), time and communications costs are minimized (*ef£ciency requirement*), and, at the end of the computation, the size of the subset of S 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 and ef£ciency. Therefore, in [10] 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 bene£ts of an agent-based implementation, we brie¤y review the KDEC scheme.

The KDEC scheme is based on three simple observations: density estimates are (i) additive for homogeneous distributed data sets, (ii) suf£cient 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 $\hat{\varphi}[S](\vec{x})$ for $\hat{\varphi}_{K,h}[S](\vec{x})$.

The global density estimate $\hat{\varphi}[S](\vec{x})$ can be decomposed into the sum of the site density estimates, one estimate for every data set D_j:

$$\hat{\varphi}[S](\vec{x}) = \sum_{j=1}^{M} \sum_{\vec{x}[i]\in D_j} K\left(\frac{d(\vec{x},\vec{x}[i])}{h}\right) = \sum_{j=1}^{M} \hat{\varphi}[D_j](\vec{x}). \qquad (2)$$

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, the hill-climbing technique of Section 4.1 to assign clusters to the local data objects. Note however that Equation (1) explicitly refers to the data objects $\vec{x}[i]$. 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 $\vec{x} \in \mathbb{R}^n$, let $x_1,\ldots,x_n$ be its components. Let $\vec{\tau} = [\tau_1,\ldots,\tau_n]^T \in \mathbb{R}^n$ be a vector of *sampling periods*, and let $\vec{z} \bullet \vec{\tau}$ denote $[z_1\tau_1,\ldots,z_n\tau_n]$, where $\vec{z} \in \mathbb{Z}^n$. A function $f\colon \mathbb{R}^n \to \mathbb{R}$ is *band-limited* to a bounded $B \subset \mathbb{R}^n$ if and only if the support of its Fourier transform is contained in B. If B is a subset of a rectangle $[-\pi/\tau_1,\pi/\tau_1) \times \cdots \times$

$[-\pi/\tau_n, \pi/\tau_n)$, it is well-known that the *sampling series*

$$\sum_{\vec{z}\in\mathbb{Z}^n} f(\vec{z}\bullet\vec{\tau})\, sinc(\frac{x_1}{\tau_1} - z_1) \cdots sinc(\frac{x_n}{\tau_n} - z_n), \quad (3)$$

where $sinc(x) = \frac{\sin \pi x}{\pi x}$, converges to f under mild conditions on f (see, e.g., [5, p.155]). If we let $f(\cdot) = \hat{\varphi}[D_j](\cdot)$ in (3), and truncate the series to a £nite n-dimensional rectangle $R(\vec{z}_1, \vec{z}_2)$ having diagonal $(\vec{z}_1, \vec{z}_2)$ we obtain an interpolation formula:

$$\sum_{\vec{z}\in R(\vec{z}_1,\vec{z}_2)} \sum_{j=1}^{M} \hat{\varphi}[D_j](\vec{z}\bullet\vec{\tau}) sinc(\frac{x_1}{\tau_1} - z_1) \cdots sinc(\frac{x_n}{\tau_n} - z_n). \quad (4)$$

Notice that the function represented by (4) is not extensionally equal to the kernel global estimate $\hat{\varphi}[S](\vec{x})$ both because kernel estimates are not band-limited on any region, and because of the truncation in the series. However, as argued in [10], 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 τ in such a way that the transform is negligible in $\mathbb{R}^n \setminus [-\pi/\tau_1, \pi/\tau_1) \times \cdots \times [-\pi/\tau_n, \pi/\tau_n)$, and by selecting $(\vec{z}_1, \vec{z}_2)$ so that the estimate is negligible in $\mathbb{R}^n \setminus R(\vec{z}_1, \vec{z}_2)$.

Therefore, (4) 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 $\{\hat{\varphi}[D_j](\vec{z}\bullet\vec{\tau}) : \vec{z} \in R(\vec{z}_1, \vec{z}_2)\}$ of the j-th local density estimate can be 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 (4) as global density estimate to which the hill-climbing technique is applied. 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 $\vec{\tau}$, $R(\vec{z}_1, \vec{z}_2)$, 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 [10] for more details).

Algorithm 1 KDEC: distributed clustering based on density estimation

funct *Interpolate*$(\vec{x}, \vec{\tau}, \vec{z}_i, \vec{z}_2, Sam) \equiv$
  foreach $\vec{z} \in R(\vec{z}_1, \vec{z}_2)$ **do**
    $r := r + Sam[\vec{z}] \prod_{i=1}^{n} Sinc(\frac{x_i}{\tau_i} - z_i)$ **od**; r.
proc *DataOwner*$(D[], H, Clus[]) \equiv$
  Negotiate$(H, \vec{\tau}, \vec{z}_1, \vec{z}_2, K, h)$;
  Send(*Sample*$(D, \vec{\tau}, \vec{z}_1, \vec{z}_2, K, h)$);
  Sam := *Receive*(H);
  for $i := 1$ **to** *Length*(D) **do**
   Clus$[i]$:= *Nearest*(
      FindLocalMax$(\vec{x}[i], \vec{\tau}, \vec{z}_1, \vec{z}_2, Sam, \nabla$*Interpolate*$(\,)))$;
  od.
  proc *Helper*$(L[]) \equiv$ *Negotiate*(L); $\vec{S} := \vec{0}$;
    for $j := 1$ **to** *Length*(L) **do** $\vec{S} := \vec{S} +$ *Receive*$(L[j])$ **od**;
    for $j := 1$ **to** *Length*(L) **do** *Send*$(\vec{S}, L[j])$ **od**.

4.3. Agents for KDEC-based DDC

There may be several ways to implement the KDEC scheme for agent-based distributed data clustering of which we only outline two of the most straightforward ones. Both assume a DDM system consisting of a set of networked, homogeneous data sites. Each data site respects its local autonomy by individually granting read-only access to external data mining agents. One option for an agent-based implementation of the KDEC based data clustering is to design a set of appropriate stationary DM agents, each of which is associated to one of multiple, networked local data sites. According to the KDEC scheme all site agents have to jointly agree on (a) which agent is taking the role of the helper site agent, (b) what kernel function to use for computing the local density estimate samples, and (c) the use of the DE-clustering algorithm for local clustering based on the global estimate. This agreement can be achieved as a result of a (application dependent) basic negotiation prior to the straightforward execution of the remaining steps of the KDEC scheme by each of the agents.

Another option is based on the idea that under certain constraints a mobile agent can perform the KDEC based DDC computation in the overall DDM system. In this case, the agent has to visit a sequence $\{L_n\}$ of data sites. At each site n the agent (a) carries in its data space, for every sampling point, the sum of all samples of every site L_m, $m < n$, at that point, (b) computes the sampled form of the density estimate of local data and sums the local samples, and then (c) returns the global estimate's sample to its owner and/or all sites for cluster analysis. Note that the role of the central helper site in the KDEC scheme is taken by the mobile DM agent. This agent may be initiated either by some distinguished central site, or by any of the local sites which then is in charge of coordinating the actions of its mobile agent. In any case, an appropriate mobile agent infrastructure, cooperation protocol between the mobile agent and site agents, as well as a proper security architecture has

to be implemented in such a system.

An agent should be allowed to exhibit autonomous behaviour inside procedure *Negotiate*, by refusing to participate to a KDEC-based DDC, e.g., if it deems the risk of disclosing sensitive information to be too high, when compared to potential bene£ts. Notice also that, both in the stationary and the mobile scenario, KDEC extends straightforwardly to a scheme in which two or more helpers are arranged into a directed tree, the leaves of which are the site agents. The selection of participating helpers and their arrangement into a tree structure can be carried out during the negotiation phase of the KDEC scheme, permitting each agent to decide its status autonomously. The summation of samples is carried out in a bottom-up fashion, yielding the samples of the global estimate at the root agent, which in turn propagates them down to the site agents. Such scheme may be useful to provide more protection against privacy infringements (see Section 4.4). As the samples of the global estimate are the largest, they could be propagated by the root agent directly to the site agents, without traversing the tree, in order to avoid high communication costs.

4.4. Inference and Trustworthiness in KDEC

Whether it is implemented by stationary or mobile agents, two crucial points must be considered when evaluating the robustness of the KDEC scheme: the inference problem for kernel density estimates and the trustworthiness of helpers.

In the following, by a *trustworthy* KDEC helper we mean a helper that acts in such a way that no bit of information written to memory by a process for the *Helper* procedure is sent to a system peripheral by a different process.

Inference Attacks on Kernel Density Estimates. The goal of an inference attack is to exploit information contained in a density estimate to infer the data objects. Formally let $g\colon \mathbb{R}^n \to \mathbb{R}$ be extensionally equal to a kernel estimate $\hat{\varphi}[S](\vec{x}) = \sum_{i=1}^{N} K\left(\frac{d(\vec{x},\vec{x}[i])}{h}\right)$, that is, $(\forall \vec{x} \in \mathbb{R}^n)\, g(\vec{x}) = \hat{\varphi}[S](\vec{x})$ holds, and let K and h be known. The problem is to compute $\vec{x}[i]$, $i = 1, \ldots, N$. In an inference attack to a KDEC-based clustering, $g(\vec{x})$ will be a reconstructed estimate effectively computed by the interpolation formula (4). General conditions on the estimate which permit to compute the objects, and to what extent this is possible, are currently under investigation. However, a few issues emerge.

A simple form of attack consists in searching the density estimate or its derivatives for discontinuities. For example, if $K(\vec{x})$ equals the square pulse $\frac{1}{4}(sign(x+1) - sign(x-1))$, 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 triangular pulse, allow for similar inferences.

In principle, estimates consisting of kernels that are in£nitely differentiable are also subject to inference attacks, although the attacker is likely to incur in signi£cant computational costs. In fact, since the integral is a linear functional, N can be computed as the integral of the global density divided by the integral of $K(\vec{x}/h)$. An attacker can select $n\,N$ space objects $\vec{y}_j$ and attempt to solve a system of equations

$$\sum_{i=1}^{N} K\left(\frac{d(\vec{y}_j, \vec{x}[i])}{h}\right) = g(\vec{y}_j), \quad j = 1, \ldots, n\,N \qquad (5)$$

Notice that, in general, the equations are non-linear.

Possible scenarios in which the above types of attack could be attempted include the following. In a *single-site attack*, one of the sites participating to the KDEC-based clustering may attempt to infer the data objects of the whole distributed data set using the global density estimate. In case of success, the site will however be unable to associate a speci£c data object to a speci£c site. In a *site coalition attack*, two or more sites agree to form a coalition and share their knowledge to the detriment of other sites. The sites fraudulently participate to a KDEC-based clustering only to the purpose of obtaining the sum of the density estimates of all the other sites which participate to the clustering. The density can be simply computed as a difference between the global density and the density of the sites in the coalition. Then the coalition attempts to infer the data objects of the other sites. Note that, in case the coalition includes all sites but one, the attack potentially reveals the data objects at the site.

Untrustworthy Helpers. There has recently been an increasing interest towards trustworthiness and referrals as a means to ascertain the degree of trustworthiness of an agent [18, 19]. 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 of 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

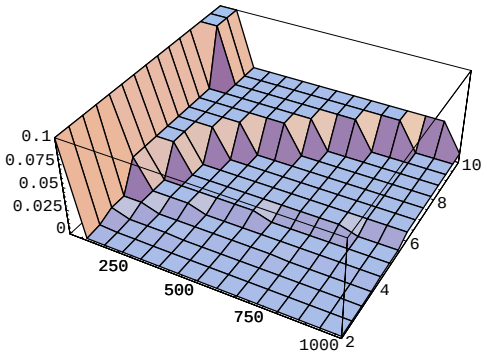

Figure 2. Graph of the upper bound on P(m,b) (p=0.2, M=1024)

are arranged to form a complete b-ary tree, it is not dif£cult to see that $P(m,b) \leq p^{\lceil \log_b(m+1) \rceil - 1}$. Such an upper bound decreases with m and increases with b according to intuition, (see Figure 2), however, it is worth noting that the lower b, the higher the chance that an agent could incur in 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 £nd a trade-off are under investigation.

5. 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 bene£ts 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 focuses in particular on the investigation of inference attacks on kernel density estimates exploiting recent advances in numerical methods for the solution of nonlinear systems of equations, and the analysis of risks of security and privacy violations in DDM environments. Finally, there are plans to develop an implementation of a multiagent system for KDEC-based DDC in a peer-to-peer network.

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*, page 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*, 8:866–883, 1996.

[3] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD-96*, pages 226–231.

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

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

[6] A. Hinneburg and D. A. Keim. An ef£cient approach to clustering in large multimedia databases with noise. In *Proc. KDD-98*, pages 58–65, New York City, 1998. AAAI Press.

[7] H. Kargupta, I. Hamzaoglu, and B. Stafford. Scalable, distributed data mining using an agent-based architecture. In *Proc. KDD-97*, pages 211–214. AAAI Press, 1997.

[8] 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. AAAI/MIT Press, 2000.

[9] M. Klusch. Information agent technology for the internet: A survey. *Data and Knowledge Engineering*, 36(3):337–372, 2001.

[10] M. Klusch, S. Lodi, and G. Moro. Distributed clustering based on sampling local density estimates. In *Proc. IJCAI-03*, Acapulco, Mexico, August 2003.

[11] G. Moro and C. Sartori. Incremental maintenance of multi-source views. In *Proc. ADC 2001*, pages 13–20, Brisbane, Australia, February 2001. IEEE Computer Society.

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

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

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

[15] 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 *Proc. KDD-97*, pages 74–81, Newport Beach, California, USA, 1997.

[16] W. Theilmann and K. Rothermel. Disseminating mobile agents for distributed information £ltering. In *Proc. 1st Int. Symp. on Mobile Agents*, pages 152–161. IEEE Press, 1999.

[17] M. Wooldridge. Intelligent agents: The key concepts. In *Multi-Agent-Systems and Applications*, volume 2322 of *LNCS*, pages 3–43. Springer-Verlag, 2002.

[18] P. Yolum and M. P. Singh. Locating trustworthy services. In *Proc. 1st Int. Workshop on Agents and Peer-to-Peer Computing (AP2PC 2002)*, Bologna, Italy, 2002. Springer-Verlag.

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