PRIVACY-PRESERVING CLUSTERING WITH DISTRIBUTED EM MIXTURE MODELING

by Xiaodong Lin, Chris Clifton, Michael Zhu

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

DOI 10.1007/s10115-004-0148-7
Springer-Verlag London Ltd. © 2004
Knowledge and Information Systems (2005) 8: 68–81

Knowledge and
Information Systems

Privacy-preserving clustering with distributed EM mixture modeling

Xiaodong Lin[1], Chris Clifton[2], Michael Zhu[3]

[1]Department of Mathematical Sciences, University of Cincinnati, Cincinnati, OH, USA
[2]Department of Computer Science, Purdue University, West Lafayette, IN, USA
[3]Department of Statistics, Purdue University, West Lafayette, IN, USA

Abstract. Privacy and security considerations can prevent sharing of data, derailing data mining projects. Distributed knowledge discovery can alleviate this problem. We present a technique that uses EM mixture modeling to perform clustering on distributed data. This method controls data sharing, preventing disclosure of individual data items or any results that can be traced to an individual site.

Keywords: Privacy; Security; Clustering

1. Introduction

Generally distributed knowledge discovery is viewed as an optimization—data is distributed and the traditional method of building a centralized data warehouse is costly. Distributed data mining methods can offer savings in processing time through use of the inherent parallelism in a distributed system, storage cost because the data doesn't need to be copied and the human cost of integrating data into a warehouse.

Privacy and security concerns provide another motive for distributed knowledge discovery. Often, data is distributed because it has been collected or produced by different parties. Contractual or regulatory controls on privacy can prevent release of the data. Trade secrecy concerns may outweigh the perceived benefit of global data mining. In these cases, building a centralized data warehouse is impossible, as no single party can be trusted by all of the distributed data sources. Distributed data mining is the *only* alternative.

For example, a corporation may cluster its customers to identify different groups to target in marketing campaigns. Now imagine that a multinational corporation

Received 18 January 2003
Revised 16 August 2003
Accepted 28 October 2003
Published online 23 December 2004

would like to develop a global advertising campaign. Building a data warehouse of worldwide customers would enable the desired clustering, but privacy laws may prevent transferring customer data across borders (Pri 2001; Blackmer and Wilmer, Cutler, Pickering 1998). Clustering within each country doesn't give the knowledge needed to develop a *global* campaign. Distributed clustering is the only solution, provided it can be done without violating the privacy laws restricting transborder flow of customer data.

This example demonstrates the three constraints that define a privacy preserving distributed data mining problem:

1. What is the data mining task? Perhaps the easiest way to answer this question is to ask what would be done if a centralized warehouse *could* be built.
2. How is the data distributed? Simple examples include *horizontal partitioning*, where each entity is represented entirely at a single site, and *vertical partitioning*, where the attributes are divided across sites and the sites must be joined to obtain complete information on any entity.
3. What are the security constraints? Privacy regulations may prevent disclosure of individually identifiable data. Corporate secrecy may allow release of individual data, but need to protect higher level rules and summaries that could be gleaned from that data.

This paper addresses (1) the task of clustering, where (2) the distributed data is horizontally partitioned, and (3) security constraints prevent sharing any information that can be traced to an individual site.

Specifically, we present a secure method for generating an expectation maximization (EM) mixture model from distributed sources. EM mixture clustering (Dempster, Laird, and Rubin 1977) iterates over the data, producing a new set of cluster centroids at each iteration. Over time, these converge to good cluster centers. We show that this can be done without revealing individual data points and without revealing which portion of the model came from which site. We assume only that their is a majority of sites that do not collude to violate privacy; any individual site (or a minority acting together) may actively try to defeat the method and will still not be able to learn individual data points or which portion of the model came from which site.

The basic idea is that each iteration can be broken into a sum of values corresponding to partitions of the data. Each partition can be computed locally, based on the local data points and global information from the previous iteration. The global sum is then computed without revealing the individual values. This provides sufficient information to compute the global information needed for the next iteration. Once this process converges, the individual sites can use the resulting model to determine in which cluster their data values lie.

The next section discusses background and related work. In Sect. 3, we show how to distribute EM mixture modeling and use this to produce a secure clustering methodology. Section 3.2 addresses evaluating the stopping criterion for the EM algorithm in a distributed setting. Section 4 provides an analysis of what is disclosed by the method.

2. Related work

First, we give some background on secure multiparty computation and discuss other approaches to privacy-preserving data mining. This is followed with a brief introduction to EM mixture modeling.

2.1. Secure multiparty computation

Yao's millionaire's problem (Yao 1986) succinctly captures the problem of secure multiparty computation. Suppose two millionaires want to find out who is worth more, but neither want to reveal their worth? More generally, the goal is to compute a global function, without any party learning anything except their own input and the global result. Yao presented a general solution to this problem for two parties, since extended to the multiparty case by Goldreich, Micali, and Wigderson (1987).

One key issue in the above definition is what is meant by *learning anything*. At first glance, this would seem to imply that no communication is allowed because wouldn't any communication tell the parties something? The formal definition is based on distributions of random data. Each party should be able to construct a data distribution from a random distribution, its own input and the global output that is *computationally indistinguishable* from the data distribution of messages exchanged in runs of the secure multiparty computation. This shows that what is learned in the actual run can be modeled with only local information and the global result, therefore nothing has been *learned* by the data exchanged in the computation.

Another issue is what the parties may do to defeat the protocol. The two models used in secure multiparty computation are semihonest (also called honest but curious) and malicious. A semihonest party is required to follow the protocol but may try to deduce private information from what it sees during execution of the protocol. This is insufficient for most practical purposes. The malicious model solves this, requiring that, regardless of the actions of the malicious party, it learns nothing but the result and that the honest party either sees a result that could have come from *some* input from the malicious party or knows that the other party is malicious. To elaborate, there is no way to detect if the malicious party just gives wrong input, but otherwise behaves honestly—this is equivalent to an honest party that really had that input. There is also no way to prevent a malicious party from stopping the protocol at any point (say, after learning the final result but before the honest party learns the result). However, in this case, the honest party knows the other is malicious. This is more than is needed to preserve privacy. We instead define a *noncolluding majority* standard. Parties may arbitrarily cheat, but as long as at most a minority collaborate as part of the cheating, they cannot violate the privacy of the honest parties. The honest parties may obtain bad results and not be able to detect the malicious parties, but their privacy is not violated. We believe this is sufficient for practical use and enables more efficient solutions than the secure multiparty computation malicious model.

There are still two issues to address before applying secure multiparty computation to a practical problem.

1. Is it sufficient? Secure multiparty computation assumes that every site learns the global result. However, we still must ensure that this global result does not reveal protected information. Just because the *computation* is secure doesn't mean the result is.
2. Is it necessary? The general method (Yao 1986; Goldreich et al. 1987) is not practical for large inputs. However, relaxing the security constraints may enable efficient solutions that still meet practical privacy and security requirements.

Section 4 addresses these issues for the method presented in this paper. We show that the EM mixture model generated does not disclose individual data values or release information that can be traced to a specific site. The method does not meet the definition of secure multiparty computation, as each iteration reveals some information. However, because this information does not reveal individual values or specific site information, it does meet the security constraints of our problem.

There has been other work addressing distributed data mining in the face of security constraints. Lindell and Pinkas showed how to construct decision trees under secure multiparty computation constraints (Lindell and Pinkas 2000). More recently, association rule mining has been addressed in both horizontally partitioned (Kantarcıoğlu and Clifton to appear) and vertically partitioned (Vaidya and Clifton 2002) data. Secure K-means clustering has been addressed, but only where the data is vertically partitioned, i.e. each dimension is completely contained at one site (Vaidya and Clifton 2003). To our knowledge, this is the first work to address secure distributed Clustering, where the data is horizontally partitioned (each site contains complete information about a set of entities.)

2.2. EM mixture modeling and secure data mining

The expectation maximization (EM) algorithm is an iterative method based mainly on the maximum likelihood principle. Since Dempster, Laird, and Rubin's celebrated paper on the EM algorithm (Dempster et al. 1977), it has become a very popular method in the AI and statistics community. More details on the EM algorithm and mixture models can be found in McLachlan and Basford (1988); McLachlan and Krishnan (1997); McLachlan and Peel (2000).

The idea behind the EM algorithm is as follows. Assume i.i.d. data $\mathbf{y} = \{y_1, \cdots, y_n\}$ drawn from a population with density function $f(\mathbf{y}; \Psi)$. Ψ is a vector of the unknown parameters. The observed data log likelihood is

$$\log L(\Psi) = \log f(\mathbf{y}; \Psi).$$

The maximum likelihood principle says that the estimators that maximize the data likelihood are consistent estimators of the true parameters. However, it is virtually impossible to find analytical solutions. The EM algorithm is an iterative procedure to find the Ψ that maximizes $\log L(\Psi)$ by data augmentation. The observed data $\mathbf{y}$ are augmented by the missing value $\mathbf{z}$ that contains group information of the observed data. More specifically, $\mathbf{z} = (Z_1, \cdots, Z_n)$, where $Z_j = (Z_{j1}, Z_{j2}, \cdots, Z_{jk})$. $Z_{ji} = 1$ means data point j belongs to the ith component. For instance, $Z_j = (1, 0, 0, 0, 0)$ means that the jth data point belongs to component 1. $\mathbf{x} = \langle \mathbf{y}, \mathbf{z} \rangle$ becomes complete data with density function $f_c(\mathbf{x}; \Psi)$. The complete-data log likelihood is

$$\log L_c(\Psi) = \log f_c(\mathbf{x}; \Psi).$$

Typically, the complete-data likelihood has a simpler structure and its expected likelihood can be maximized analytically. Dempster et al. (1977) proved that by maximizing $G(\Psi; \Psi^{(t)}) = E_{\Psi^{(t)}}\{\log L_c(\Psi)|\mathbf{y}\}$, the observed log likelihood is nondecreasing for each iteration step, which guarantees convergence of the algorithm. The EM algorithm takes advantage of this and solves the maximum likelihood problem iteratively. The algorithm contains two steps:

E-Step: On the $(t + 1)$st step, calculate the expected complete-data log likelihood given observed data values: $G(\Psi; \Psi^{(t)})$.
M-Step: Find $\Psi^{(t+1)}$ to maximize $G(\Psi; \Psi^{(t)})$.

The algorithm stops when $\log L(\Psi^{(t+1)}) - \log L(\Psi^{(t)})$ is less than a preselected threshold.

In this paper, we focus on an EM algorithm for finite normal mixtures, as is widely used in the data mining community. Assume a mixture of k components (clusters), with a d-dimensional data set $\mathbf{y}$ of size n. Assume further that the unknown parameters are $\Psi = (\mu_1, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k, \pi_1, \ldots, \pi_k)^T$. The normal mixture model is

$$f(Y; \Psi) = \sum_{i=1}^{k} \pi_i f_i(Y; \mu_i, \Sigma_i),$$

where $f_i(Y; \theta_i)$ is the normal density:

$$f_i(Y; \mu_i, \Sigma_i) = (2\pi_i)^{-d/2} |\Sigma_i|^{-1/2} \exp\left\{ -\frac{1}{2}(Y - \mu_i)^T \Sigma_i^{-1} (Y - \mu_i) \right\}.$$

Let component information $Z_{i,j}$, representing that the jth data point belongs to component i, be missing data. The complete-data log likelihood is

$$\log(L_C(\Psi)) = \sum_{i=1}^{k} \sum_{j=1}^{n} Z_{ij} \log f_i(y_j; \mu_i, \Sigma_i) \tag{1}$$

$$= -\frac{1}{2} n \log(2\pi) - \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{n} Z_{ij} \left\{ \log |\Sigma_i| + (y_j - \mu_i)' \Sigma_i^{-1} (y_j - \mu_i) \right\}. \tag{2}$$

It is clear that, for the $(t + 1)$st iteration, $G(\Psi; \Psi^{(t)})$ can be calculated by simply computing $z_{ij}^{(t)} = E_{\Psi^{(t)}}(Z_{ij})$. The expected complete-data likelihood can thus be maximized in each iteration step to obtain a good estimator for Ψ.

3. Privacy-preserving expectation maximization

Based on the framework of the previous section, we present a privacy-preserving EM algorithm that utilizes the linearity of parameter estimators in each iteration step during the EM algorithm. Section 3.1 gives the general algorithm. Section 3.2 discusses evaluation of stopping criteria and Sect. 3.3 explains the connection between the algorithm and privacy. For clarity, the conventions for notation of the paper are given in Table 1. Table 2 defines the meaning of indexes.

3.1. The algorithm

Different versions of EM algorithms have been proposed during the past 25 years. Examples include Meng's SEM (Meng and Rubin 1991), a stochastic version of the EM algorithm by Celeux (Celeux, Chauveau and Diebolt 1996), and Mclust by Banfield (Banfield and Raftery 1993; Fraley and Raftery 1998). In this section, we follow the classical EM steps and relate the results to privacy-preserving clustering.

Table 1. Convention for symbols

k	Total number of mixture components (clusters).
s	Total number of distributed sites.
n	Total number of data points.
n_l	Total number of data points for site l.
y_j	Observed data.
μ_i	Vector of mean values for cluster i.
Σ_i	Covariance matrix for cluster i.
π_i	Estimate of proportion of items in cluster i.
Z_{ij}	Cluster membership. If $y_i \in$ component j, $Z_{ij} = 1$, otherwise $Z_{ij} = 0$.

Table 2. Meaning of indexes

Index	Range	Meaning
i	1 to k	Index for mixture components (clusters).
j	1 to n	Index for data points.
l	1 to s	Index for distributed sites.
t	1 to number of iterations needed	Index for iteration steps.

Following the complete-data log likelihood defined at (2), for step $(t+1)$, $\Psi^{(t+1)}$ can be found by computing the zeros of $\partial G(\Psi; \Psi^{(t)})/\partial\Psi$. This leads to the parameter updates at the $(t+1)$th iteration:

$$\mu_i^{(t+1)} = \sum_{j=1}^{n} z_{ij}^{(t)} y_j \Big/ \sum_{j=1}^{n} z_{ij}^{(t)} \tag{3}$$

$$\Sigma_i^{(t+1)} = \sum_{j=1}^{n} z_{ij}^{(t)} \left(y_j - \mu_i^{(t+1)}\right)\left(y_j - \mu_i^{(t+1)}\right)' \Big/ \sum_{j=1}^{n} z_{ij}^{(t)} \tag{4}$$

$$\pi_i^{(t+1)} = \sum_{j=1}^{n} z_{ij}^{(t)} / n. \tag{5}$$

Assume the data (y_j) are partitioned across s sites $(1 \le l \le s)$. Each site has n_l data items; the total number of items $n = \sum_{l=1}^{s} n_l$. To obtain global estimators for $\mu_i^{(t+1)}$, $\Sigma_i{}^{(t+1)}$ and $\pi_i^{(t+1)}$ (the E step) requires only the global values n and

$$\sum_{j=1}^{n} z_{ij}^{(t)} y_j = \sum_{l=1}^{s} A_{il} \tag{6}$$

$$\sum_{j=1}^{n} z_{ij}^{(t)} = \sum_{l=1}^{s} B_{il} \tag{7}$$

$$\sum_{j=1}^{n} z_{ij}^{(t)} \left(y_j - \mu_i^{(t+1)}\right)\left(y_j - \mu_i^{(t+1)}\right)' = \sum_{l=1}^{s} C_{il}, \tag{8}$$

where

$$A_{il} = \sum_{j=1}^{n_l} z_{ijl}^{(t)} y_j \tag{9}$$

$$B_{il} = \sum_{j=1}^{n_l} z_{ijl}^{(t)} \tag{10}$$

$$C_{il} = \sum_{j=1}^{n_l} z_{ijl}^{(t)} \big(y_j - \mu_i^{(t+1)}\big)\big(y_j - \mu_i^{(t+1)}\big)'. \tag{11}$$

Clearly, these A, B, C values can be computed locally at each site. In Sect. 4, we will show that these items do not reveal individual data values or the respective grouping information. Furthermore, it is not necessary to share n_l, A_{il}, B_{il} and C_{il} from site to site as the parameter updates only require the global sums over different sites. Appendix A shows how to compute these summations securely, in the secure multiparty computation sense.

After the global parameters ($\mu_i^{(t+1)}$, $\Sigma_i^{(t+1)}$, $\pi_i^{(t+1)}$) are obtained and shared site-wise, z_{ijl} can be computed locally as

$$z_{ijl} = \frac{\pi_i^{(t+1)} f_i\big(y_{jl}; \mu_i^{(t+1)}, \Sigma_i{}^{(t+1)}\big)}{\sum_i \pi_i^{(t+1)} f_i\big(y_{jl}; \mu_i^{(t+1)}, \Sigma_i{}^{(t+1)}\big)}, \tag{12}$$

where y_{jl} is a data point at site l. Algorithm 1 summarizes the method.

Algorithm 1 Secure EM algorithm

```
At each site $l$, $\forall_{i=1...n_l, j=1...k}$ randomly initialize $z_{ijl}$ to 0 or 1.
Use secure sum of Appendix A to compute $n = \sum_{l=1}^{s} n_l$
$t = 0$
while Threshold criterion of Sect. 3.2 not met do
    for all $i = 1...k$ do
        At each site $l$, calculate $A_{il}^{(t+1)}$ and $B_{il}^{(t+1)}$ using equations (9) and (10).
        Use secure sum to calculate $A_i^{(t+1)}$ and $B_i^{(t+1)}$.
        Site 1 uses these to compute $\mu_i^{(t+1)}$ by equation (3) and broadcasts it to all sites.
        Each site $l$ calculates $C_{il}^{(t+1)}$ using equation (11).
        Use secure sum to calculate $C_i^{(t+1)}$.
        Site 1 calculates $\Sigma_i{}^{(t+1)}$ and $\pi_i^{(t+1)}$ by equations (4) and (5), and broadcasts them to all sites.
        At each site $l$, $\forall_{j=1...n_l}$ update $z_{ijl}^{(t+1)}$ using equation (12).
    end for
    $t = t + 1$
    Calculate the log likelihood difference as described in Sect. 3.2.
end while
```

The number of values communicated at each step is $3 * k * s + 2 * (k - 1) * s$. This is quite reasonable, particularly as it is constant in n and thus scales well with data size.

3.2. Analysis of stopping criterion

Usually convergence in an EM mixture algorithm is defined as

$$|\log L^{(t+1)}(\Psi^{(t+1)}|\mathbf{y}) - \log L^{(t)}(\Psi^{(t)}|\mathbf{y})| \leq \epsilon, \tag{13}$$

where

$$\log L^{(t)}(\Psi^{(t)}|\mathbf{y}) = \sum_{j=1}^{n}\sum_{i=1}^{k}\{[\log \pi_i f_i(y_j|\Psi^{(t)})]\} \tag{14}$$

and ϵ is a predetermined threshold. The sum can be partitioned among the sites,

$$\log L^{(t)}(\Psi^{(t)}|\mathbf{y}) = \sum_{l=1}^{s} D_l, \tag{15}$$

where

$$D_l = \sum_{j=1}^{n_l}\sum_{i=1}^{k}\{\log \pi_i f_i(y_{jl}|\Psi^{(t)})\}. \tag{16}$$

Using the secure sum protocol in Appendix A, D_l can be computed as the sum of the locally computed $\log L_l^{(t)}$. The master site can then check the stopping criterion, i.e.

$$|\log L^{(t+1)} - \log L^{(t)}| \leq \epsilon \tag{17}$$

to see whether the algorithm has converged. Once the stopping criterion is met at the $(t+1)$st step, each site clusters their own data using the following principle:

$$y_j \in \mathit{cluster}\ h, \quad \mathit{if}\ z_{hj}^{(t+1)} = \max_{1\leq i\leq k} z_{ij}^{(t+1)}, \quad 1 \leq h \leq k.$$

3.3. Linearity of estimators and privacy

Through our analysis, linearity of the $G(\Psi; \Psi^{(t)})$ plays an important role for the algorithm. Because of the linearity, we can calculate the required statistics locally at each site, then combine them through a secure summation.

Also of note is that partitioning $G(\Psi; \Psi^{(t)})$ to local sites doesn't change the EM step. Thus, the general properties of a finite mixture model using the EM algorithm still hold, which guarantees the convergence of the algorithm. We have also empirically validated that the results generated by this method are comparable to a well-known EM mixture model, FastMix (Moore 1999).

An important issue for clustering is to select the right number of clusters. We have assumed a priori knowledge of the number of clusters, k. Researchers have been using criteria such as Bayesian information criteria and minimum description length to select this value. The basic idea is to fit data into mixtures with different numbers of normal components (clusters), then choose the one with the largest convergent log likelihood. This is not a problem for secure clustering because the sum of the convergent log likelihood across the sites can be computed and compared under different model assumptions.

4. Security analysis

The goal of this method is to develop an EM mixture model without:

1. Disclosing individual data values beyond the site containing those items or
2. Revealing any information that can be traced to a specific site

Without applying the secure summation algorithms, values of n_l, A_{il}, B_{il}, C_{il} and L are revealed for each component l. It is then possible for the end user to derive sample mean and covariance matrices at site l by

$$\mu_{il} = \frac{\sum_{j=1}^{n_l} z_{ijl} y_j}{\sum_{j=1}^{n_l} z_{ijl}} = \frac{A_{il}}{B_{il}} \tag{18}$$

$$\Sigma_{il} = \frac{\sum_{j=1}^{n_l} z_{ijl}(y_j - \mu_i)(y_j - \mu_i)'}{n_l}. \tag{19}$$

With these quantities, confidence intervals for the true mean and covariance matrix can by derived for each site. However, by using the secure summation protocol of Appendix A, only the global values A_i, B_i, C_i, n and L are revealed. This results in revealing only the global values μ_i and Σ_i. From these values, it is not possible to deduce confidence intervals w.r.t. the mean and covariance matrix for component i at the local sites.

Furthermore, if μ_{il} and Σ_{il} for every site l are shared across all the sites, the probability that a data point y belong to a specified interval can be calculated at each site l as

$$P(y \in I(\mu_{il} - a, \mu_{il} + a)) = \Phi(\mu_{il} + a|\mu_{il}, \Sigma_{il}) - \Phi(\mu_{il} - a|\mu_{il}, \Sigma_{il}), \tag{20}$$

where Φ is the usual cumulative function of normal distribution. By comparing the probabilities across every site l, it is possible to deduce to which site the data point y belongs. Revealing this would violate the privacy constraints. When only the global values of μ_i and Σ_i are disclosed, these inductions are not possible.

We now address whether the revealed values themselves can be used to deduce any information on individual data items.

n is the global count of data items. It clearly does not reflect individually identifiable information. Provided there are more than two sites, it reveals only an upper bound on the number of items at any given site—most likely innocuous information.

A_i is based on the data values at each site. However, it distills these into a single number for each component, independent of the number of data values or sites. Thus, by itself, it does not reveal restricted information. Even if a component contains only a single data item (and thus the cluster center converges to that item), no site can *know* that this is the case.

B_i is constructed from data from a previous iteration, along with knowledge of the local number of items n_l. Because it is a single value for each component, n_l is not revealed.

C_i use individual data values, but again these are distilled into a single value for each component, as with A_i.

L is the global log likelihood of the data, again a single scalar value that is not tied to an individual site or data item.

Thus, a single iteration reveals no restricted information.

However, can the values we reveal in the previous steps be used to reveal values that should not be disclosed? To address this question, assume, without loss of generality, that s new data points are assigned to component i. From the mean and variance of steps t and $t+1$, we have

$$\Sigma_i^{(t)} = \frac{\sum_{j=1}^{n_i} \left(y_j - \mu_i^{(t)}\right)\left(y_j - \mu_i^{(t)}\right)'}{n_i - 1}$$

$$\Sigma_i^{(t+1)} = \frac{\sum_{j=1}^{n_i} \left(y_j - \mu_i^{(t+1)}\right)\left(y_j - \mu_i^{(t+1)}\right)' + \sum_{j=n_i+1}^{n_i+s} \left(y_j - \mu_i^{(t+1)}\right)\left(y_j - \mu_i^{(t+1)}\right)'}{n_i + s - 1}.$$

Clearly, when $s > 1$, these two equations have infinite solutions for $y_{n_i+1}, \dots, y_{n_i+s}$. In other words, values from previous iterations will not reveal any information that is not already revealed by step $t+1$ alone.

A second reason that multiple iterations do not release data is that the secure summation prevents us from knowing *which* site is responsible for a change in values between iterations or even how many sites are responsible.

These arguments assume three or more parties and no collusion. With two parties, while no individual values are disclosed, a dishonest party could learn how data clusters on the honest party. Simple input modification allows this—set $n_{\text{dishonest}} = 0$ and participate in the protocol normally. The result is an EM mixture model based only on the honest party's data, with π_i, σ_i and μ_i known to both. Collusion can result in the same problem. However, the method can be extended to be secure with an honest majority as described in Appendix A.

5. Conclusions and further work

We have presented a clustering method based on expectation maximization that limits the disclosure of data between sites in a distributed environment. Specifically,

1. The values of individual data items are not disclosed.
2. No information can be traced to a specific site.

These properties are sufficient for many practical privacy problems, enabling clustering even when data sharing is constrained.

This method is also quite efficient. The only communication needed by this method is to generate the values $A_i^{(t+1)}$, $B_i^{(t+1)}$, $C_i^{(t+1)}$ and n that are used to calculate $\mu_i^{(t+1)}$, $\Sigma_i{}^{(t+1)}$ and $\pi_i^{(t+1)}$, the local log likelihood estimates $L_l^{(t+1)}$ and sending μ_i, Σ_i and π_i from the central site to the distributed sites. Thus, the communication cost for each iteration scales as $O(ks)$, where k is the number of clusters and s is the number of sites—in particular, this is independent of the size of the data. The rate of convergence, and thus the number of iterations, is independent of the distributed aspect of the problem.

The field of privacy-preserving data mining is wide open for research. One question is if the method presented here can be *efficiently* extended to be secure in the secure multiparty communication sense. This requires showing how the iteration can be performed without learning the results of any but the last iteration. This would enable a more formal proof than the arguments given in Sect. 4. However, the practical benefit is questionable, as secure multiparty computation is neither

necessary, as it prevents the disclosure of innocuous information that could enable a more efficient algorithm, or

sufficient, as the result combined with a site's own input may disclose restricted information.

This comes back to the problems described at the beginning of Sect. 4. Without a formal definition of what constitutes acceptable and unacceptable disclosure, a formal proof is meaningless. Security policy is today specified with informal descriptions, a practical formal security semantics is still an open problem.

One approach is to reduce the final result to a minimal model that does not disclose *any* unneeded information. Assuming that the goal of clustering is that each site should be able to determine which of its items fall into which cluster, the EM mixture model result (which represents cluster centers) provides unneeded information. A minimal result would simply provide each party with a mapping from their own items to cluster numbers. While sufficient, such a solution is unlikely to be necessary and the benefit must be weighed against the likely extra communication, computation and complexity costs. This would address the two-party case: A solution secure under such a definition would not reveal anything to a dishonest party simulating no input, although other attacks may be possible.

There are many open problems in privacy-preserving data mining. Section 2.1 discusses some of the problems that have been addressed, but there are many more that have not. In addition to distributed privacy issues, there has also been work on preserving privacy by distorting the data values, while still allowing data mining. This has as of yet only been applied to decision trees (Agrawal and Srikant 2000; Agrawal and Aggarwal 2001) and association rules (Rizvi and Haritsa 2002).

Distributed knowledge discovery has many benefits. Enabling data mining that would otherwise be prevented due to privacy and security constraints is a key benefit and is worthy of further exploration.

A. Secure summation

This method frequently needs to calculate the sum of values from individual sites. Assuming three or more parties and no collusion, the following method (from Benaloh (1986)) securely computes such a sum.

Assume that the value $v = \sum_{l=1}^{s} v_l$ to be computed is known to lie in the range $[0 \ldots n)$.

One site is designated the *master* site, numbered 1. The remaining sites are numbered $2 \ldots s$. Site 1 generates a random number R, uniformly chosen from $[0 \ldots n)$. Site 1 adds this to its local value v_1 and sends the sum $(R + v_1) \bmod n$ to site 2. Because the value R is chosen uniformly from $[0 \ldots n)$, the number $(R + v_1) \bmod n$ is also distributed uniformly across this region, independent of the value of v_1. Therefore, site 2 learns nothing about the actual value of v_1.

For the remaining sites, $l = 2 \ldots s-1$, the algorithm is as follows: Site l receives

$$V = \left(R + \sum_{j=1}^{l-1} v_j \right) \bmod n.$$

Again, this value is uniformly distributed across $[0 \ldots n)$, regardless of the values of v_j, so l learns nothing about the values of v_j. Site l then computes

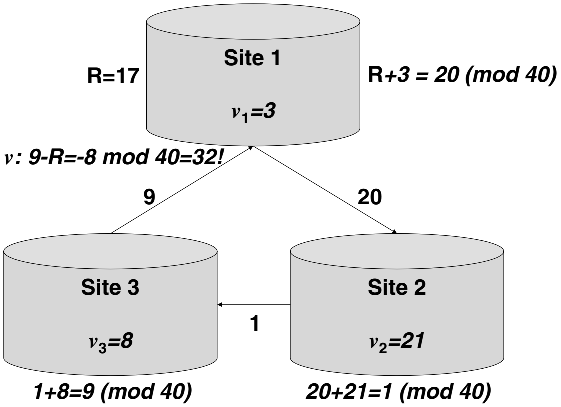

Fig. 1. Values computed at each site during secure computation of a sum initiated by site 1 (all arithmetic modulo $n = 40$)

$$\left(R + \sum_{j=1}^{l} v_j\right) \bmod n = (v_l + V) \bmod n$$

and passes it to site $l + 1$.

Site s performs the above step and sends the result to site 1. Site 1, knowing R, can subtract R to get the actual result. Note that site 1 can also determine $\sum_{l=2}^{s} v_l$ by subtracting v_1. However, given only the final sum, any site can determine the sum of the v_j at all sites other than itself. Because this is obtained from the result and one's own input, it does not represent an information leak from the algorithm. Figure 1 depicts how this method operates.

This method faces an obvious problem if sites collude. Sites $l - 1$ and $l + 1$ can compare the values they send/receive to determine the exact value for v_l. The method can be extended to work for an honest majority. Each site divides v_l into shares. The sum for each share is computed individually. However, the path used is permuted for each share such that no site has the same neighbor twice. To compute v_l, the neighbors of l from each iteration would have to collude. Varying the number of shares varies the number of dishonest (colluding) parties required to violate security.

References

Agrawal D, Aggarwal CC (2001) On the design and quantification of privacy preserving data mining algorithms. In: Proceedings of the twentieth ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems. ACM, Santa Barbara, CA, pp 247–255
*http://doi.acm.org/10.1145/375551.375602

Agrawal R, Srikant R (2000) Privacy-preserving data mining. In: Proceedings of the 2000 ACM SIGMOD conference on management of data. ACM, Dallas, TX, pp 439–450
*http://doi.acm.org/10.1145/342009.335438

Banfield JD, Raftery AE (1993) Model-based Gaussian and non-Gaussian clustering. Biometrics 49:803–821

Benaloh JC (1986) Secret sharing homomorphisms: keeping shares of a secret secret. In: Odlyzko A (ed) Advances in cryptography—CRYPTO86: proceedings (Lecture notes in computer science), vol 263. Springer, Berlin Heidelberg New York pp 251–260
*http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=263&spage=251

Blackmer S and Wilmer, Cutler, Pickering (1998) Transborder personal data flows: administrative practice. In: The privacy and American business meeting on model data protection contracts and laws. Washington, DC
*http://www.privacyexchange.org/tbdi/pdataflow.html

Celeux G, Chauveau D, Diebolt J (1996) Stochastic versions of the EM algorithm: an experimental study in the mixture case. J Stat Comput Simul 55:287–314

Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm (with discussion). J Roy Stat Soc B 39:1–38

Fraley C, Raftery AE (1998) How many clusters? Which clustering method? Answers via model based cluster analysis. Comput J 41:578–588

Goldreich O, Micali S, Wigderson A (1987) How to play any mental game—a completeness theorem for protocols with honest majority. In: 19th ACM symposium on the theory of computing, pp 218–229
*http://doi.acm.org/10.1145/28395.28420

Kantarcıoğlu M, Clifton C (to appear) Privacy-preserving distributed mining of association rules on horizontally partitioned data. IEEE Trans Knowl Data Eng

Lindell Y, Pinkas B (2000) Privacy preserving data mining. In: Advances in cryptology—CRYPTO 2000. Springer, Berlin Heidelberg New York, pp 36–54
http://link.springer.de/link/service/series/0558/bibs/1880/18800036.htm

McLachlan GJ, Basford KE (1988) Mixture models: inference and applications to clustering. Dekker, New York

McLachlan GJ, Krishnan T (1997) The EM algorithm and extensions. Wiley, New York

McLachlan G, Peel D (2000) Finite mixture models. Wiley, New York

Meng XL, Rubin DB (1991) Using EM to obtain asymptotic variance covariance matrices: the SEM algorithm. J Am Stat Assoc 86:899–909

Moore AW (1999) Very fast EM-based mixture model clustering using multiresolution kd-trees. Adv Neur Inf Process Syst 11

Pri (2001) National omnibus laws, http://www.privacyexchange.org/legal/nat/omni/nol.html
*http://www.privacyexchange.org/legal/nat/omni/nol.html

Rizvi SJ, Haritsa JR (2002) Maintaining data privacy in association rule mining. In: Proceedings of 28th international conference on very large data bases. VLDB, Hong Kong, pp 682–693
*http://www.vldb.org/conf/2002/S19P03.pdf

Vaidya J, Clifton C (2002) Privacy preserving association rule mining in vertically partitioned data. In: The eighth ACM SIGKDD international conference on knowledge discovery and data mining. Edmonton, Alberta, Canada, pp 639–644
*http://doi.acm.org/10.1145/775047.775142

Vaidya J, Clifton C (2003) Privacy-preserving *k*-means clustering over vertically partitioned data. In: The ninth ACM SIGKDD international conference on knowledge discovery and data mining. Washington, DC

Yao AC (1986) How to generate and exchange secrets. In: Proceedings of the 27th IEEE symposium on foundations of computer science. IEEE, pp 162–167

Author biographies

Xiaodong Lin is an assistant professor of mathematics at University of Cincinnati. He is on an academic leave at the Statistics and Applied Mathematics Science Institute during 2003–2004. He has a Ph.D. and M.S. from Purdue University and a bachelor's degree from the University of Science and Technology of China. His research interests include data mining, statistical learning, machine learning and privacy-preserving data mining.

Chris Clifton is an associate professor of computer science at Purdue University. He has a Ph.D. from Princeton University and bachelor's and master's degrees from the Massachusetts Institute of Technology. Prior to joining Purdue in 2001, Chris had served as a principal scientist at The MITRE Corporation and as an assistant professor of computer science at Northwestern University. His research interests include data mining, data security, database support for text, and heterogeneous databases.

Michael Zhu is an assistant professor of statistics at Purdue University. He has a Ph.D. from the University of Michigan and bachelor's and master's degrees from Hsinghua University. His research interests include data mining, statistical learning, machine learning and experimental designs.

Correspondence and offprint requests to: Xiaodong Lin, Department of Mathematics, University of Cincinnati, Cincinnati, OH 45221-0025, USA. Email: linxd@math.uc.edu