Distributed Clustering with Limited Knowledge Sharing

Joydeep Ghosh and Srujana Merugu
Department of Electrical and Computer Engineering,
The University of Texas at Austin
{ghosh,merugu}@ece.utexas.edu

Abstract

While data mining algorithms invariably operate on centralized data, in practice related information is often acquired and stored at geographically distributed locations due to organizational or operational constraints. Centralization of such data before analysis may not be desirable because of computational or bandwidth costs. In some cases, it may not even be possible due to variety of real-life constraints including security, privacy, proprietary nature of data/software and the accompanying ownership and legal issues. This paper briefly describes how one can achieve distributed clustering in two different settings that impose severe constraints on the data or knowledge that can be shared among (local) data sites. The first allows only the cluster labels of individual objects to be shared, but not their attributes. The second disallows sharing of the attributes or cluster labels of individual objects altogether. In this case generative (probabilistic) models of local data are used to generate "virtual samples" that are then used to obtain a "global" solution. Applications are identified for both of these settings.

1. Introduction

Extracting useful knowledge from large, distributed data repositories can be a very difficult task when such data cannot be directly centralized or unified as a single file or database due to a variety of constraints. As in much of parallel processing, early work on distributed data mining largely focused more on technical constraints such as narrow communication bandwidths or limited central storage. More recently, there has been an emphasis on how to obtain high quality information from distributed sources while simultaneously adhering to restrictions on the *nature* of the data to be shared, due to data ownership or privacy issues. Much of this work is appearing under the moniker of "privacy-preserving data mining".

Privacy-preserving data mining techniques so far have largely taken one of three approaches: (i) query restriction to solve the inference problem in databases[10] (ii) subjecting individual records or attributes to a "privacy preserving" randomization operation and subsequently recovering the original data [2], (iii) using cryptographic techniques for two-party or multi-party communications [16]. The first method is difficult and manually intensive, while the latter two approaches are largely restricted to vector data and are applicable only in settings where a central party is collecting individual records that need to be protected.

Most of the privacy-aware distributed data mining techniques developed so far have focused on classification or on association rules [1, 5, 9, 16]. There has also been some work on distributed clustering for *vertically partitioned data* (different sites contain different attributes/features of a common set of records/objects) [12], and on parallelizing clustering algorithms for *horizontally partitioned data* (i.e. the objects are distributed amongst the sites, which record the same set of features for each object) [8]. These techniques, however, do not specifically address privacy issues, other than through encryption [19]. This is also true of earlier data-parallel methods [8] that are susceptible to privacy breaches, and also need a central planner that dictates what algorithm runs on each site.

In this paper, we summarize two recent approaches towards clustering distributed data taking into account various privacy restrictions [18, 13]. They correspond to two different settings that cater to different notions of privacy. In both the cases, the prototypical application scenario is one in which there are multiple parties with confidential databases and the goal is to cluster the entire distributed data, without actually first pooling this data. For example, the parties can be a group of banks, with their own sets of customers, who would like to have a better insight into the behavior of the entire customer population without compromising the privacy of their individual customers. In the first setting, the objective is to obtain a single "consensus" clustering of all the objects owned by the different parties without revealing any information about the objects' attributes or even the clustering algorithms used by these parties. In the second setting, one strives to characterize the distributed data distribution *via* clustering using high-level information that provides a trade-off between privacy and quality. The overall objective of this paper is to show that there are several types of privacy constraints and that even though such constraints can be quite restrictive, surprisingly good results

can be achieved by appropriate algorithm design.

2. Cluster Ensembles

In this section, we summarize some results from our recent work [18] on the *combination* of multiple *partitionings* of the same underlying set (or subset) of objects *without accessing the original features*. Specifically, each clusterer only provides the cluster labels for the data that it has examined, to a central "combiner". It does not however share information about what features were used for clustering or what clustering technique was employed. Note that cluster labels are symbolic, i.e., cluster "A" of one solution may not correspond to cluster "A" of another solution at a different site. Also, the number of clusters may differ from site to site, and these may not even be examining identical sets of objects or looking at the same feature space. Thus this is a much more difficult problem than the well-studied problem of combining the results of multiple classifiers [11].

The setting sketched above is natural to several application scenarios that involve *knowledge reuse* [3]. For clustering, this means that a variety of clusterings for the objects under consideration *may already exist*, and one desires to either integrate these clusterings into a single solution, or use this information to influence a new clustering (perhaps based on a different set of features) of these objects. We encountered such a situation when clustering visitors to an e-tailing website based on market basket analysis, in order to facilitate a direct marketing campaign [17]. The company already had a variety of legacy customer segmentations based on demographics, credit rating, geographical region and purchasing patterns in their retail stores, etc. They were obviously reluctant to throw out all this domain knowledge, and instead wanted to reuse such pre-existing knowledge to create a single consolidated clustering. Note that since the legacy clusterings were largely provided by human experts or by other companies using proprietary methods, the information in the legacy segmentations had to be used *without* going back to the *original features* or the 'algorithms' that were used to obtain these clusterings.

Suppose we are given r clusterings, with the ith clustering having k_i clusters, and we want to obtain a single consensus clustering with a pre-specified number of clusters, k. The first issue to address is how to evaluate this consensus clustering. Note that there is no ground truth to measure against, unlike in a classification setting. Intuitively, if there is no other apriori knowledge, then the best one can do is to extract the commonalities among the different clusterings. If we consider the cluster labels produced by a solution as the values taken by a random variable, then a natural measure of the similarity between two clusterings is the mutual information (MI) between the two corresponding random variables. Note that this measure does not require that the number of clusters in each solution to be the same, or that a correspondence between the two sets of clusters be pre-established. Based on these observations, we proposed that the optimal combined clustering be defined as the one that has maximal average (pairwise) mutual information with each individual labeling [18]. Moreover, to cater to solutions with widely differing numbers of clusters, each MI term was normalized by the geometric mean of the entropies of the two random variables, so as to lie within the interval [0,1].

2.1 Efficient Consensus Functions

This average normalized mutual information (ANMI) thus serves as the objective function for the cluster ensemble problem. Direct optimization of this objective function is intractable, as the search space is super-exponential, and simple greedy approaches resulted in poor solutions. So, we instead examined three efficient heuristics to obtain high values for the objective. All algorithms approach the problem by first transforming the set of clusterings into a hypergraph representation. Simply put, each cluster is considered as a hyperedge connecting all its members (vertices). The hyperedges obtained from different clusterings are all added to a common graph, which has n vertices and $\sum_{i=1}^{r} k_i$ hyperedges where n is the number of datapoints.

The simplest heuristic is to define a similarity measure between two objects as the fraction of clusterings in which these objects are in the same cluster. The resulting matrix of pairwise similarities can be used to recluster the objects using any reasonable similarity-based clustering algorithm. The second heuristic looks for a hyperedge separator that partitions the hypergraph into k unconnected components of approximately the same size, using a suitable hypergraph partitioning package such as HMETIS. The idea behind the third heuristic is to group and collapse related hyperedges into k meta-hyperedges. The hyperedges that are considered related for the purpose of collapsing are determined by a graph-based clustering of hyperedges. Finally, each object is assigned to the collapsed hyperedge in which it participates most strongly.

It turns out that the first and third approaches typically do better than the second. Since the third approach is much faster than the first, it is preferred. However, note that our objective function has an added advantage that it allows one to add a stage that selects the best consensus function without any supervision information, by simply selecting the one with the highest NMI. So, for the results reported later, we simply use this 'supra'-consensus function obtained by running *all three* algorithms, and selecting the one with the greatest score.

2.2 Applications and Results

For brevity, we just illustrate one application of cluster ensembles, namely how it can be used to boost the quality of

data	subspace #dims	#models r	**quality of consensus** $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$	max. individual quality $\max_q \phi^{(\mathrm{NMI})}(\kappa, \lambda^{(q)})$	ave. individual quality $\mathrm{avg}_q \phi^{(\mathrm{NMI})}(\kappa, \lambda^{(q)})$
PENDIG	4	10	**0.638**	0.479	0.420
YAHOO	128	20	**0.410**	0.202	0.160

Table 1: Effectiveness of consensus clustering for integrating multiple clusterings based on partial feature views.

results by combining a set of clusterings obtained from partial views of the data. This scenario is motivated by certain distributed data mining situations in which it is not feasible to collect all the features at one central location. Results on a wider range of application scenarios and data sets can be found in [18].

For our experiments, we simulate such a scenario by running several clusterers, each having access to only a restricted, small subset of features. The clusterers find groups in their views/subspaces. In the combining stage, individual results are integrated to recover the full structure of the data (without access to any of the original features). Results are provided on two real data sets: (i) PENDIG from the UCI ML repository, is for pen-based recognition of handwritten digits from 16 spatial features. There are ten classes of roughly equal size corresponding to the digits 0 to 9. (ii) YAHOO represents 2340 documents from 20 news categories, and is available from ftp://ftp.cs.umn.edu/dept/users/boley/. After standard preprocessing, each document is represented by a 2903-dimensional vector. These two data sets are partitioned into 10 and 40 clusters respectively by each clustering algorithm. For this experiment, the individual clusterers are all graph-partitioning based (as they are quite robust and give comparable sized clusters), using a domain-appropriate similarity function, namely, Euclidean distance for PENDIG and cosine similarity for YAHOO. Table 1 summarizes the results, averaged over 10 runs. For example, in the YAHOO case, 20 clusterings were performed in 128-dimensions (occurrence frequencies of 128 randomly chosen words) each. The average quality amongst the results was 0.160 and the best quality was 0.202. Using the supra-consensus function to combine all 20 labelings results in a quality of 0.410, or 156% higher mutual information than the average individual clustering. The results indicate that, when processing on the all features is not possible but multiple, limited views exist, a cluster ensemble can significantly boost results compared to the individual clusterings.

3. Distributed Model-based Clustering

Now consider a different privacy preserving scenario where the over-riding constraint is that information about individual records, be it the feature values or the cluster label, cannot be shared with another site. Applications abound, specially where each record contains various attributes of an individual. So now we have to describe the data by modeling of feature distributions across multiple records in such a way that the specifics of a particular record are obscured. To make this problem tractable, we consider the case where the records have the same sets of features at each site. This points to an approach of building models locally and then combining them at a central location to obtain a more accurate model [5, 20]. This approach enables easy analysis of privacy and communication costs in terms of the local model that is shared with the central location. The key is to characterize the data at each site using a suitable probabilistic (generative) model, transmit only the model parameters to a central site, where "virtual samples" can be now generated using Monte Carlo Markov Chain sampling techniques and used to form a combined model. Since generative models are available for a wide range of data types, from vectors to variable length sequences and graphs [4, 21], this approach is quite general.

Since another goal of data mining is to obtain highly interpretable results, let the global model be specified as a mixture model based on a given parametric family (e.g., mixture of Gaussians). We call the resulting search problem of finding the highest quality global model within this family of models the **Distributed Model-based Clustering** (DMC) problem and state it more formally below.

Let $\{\mathcal{X}_i\}_{i=1}^n$ be n horizontally partitioned data sources generated by a common underlying model, λ^0. Let $\{\lambda_i\}_{i=1}^n$ be the local models obtained by applying clustering algorithms to these data sources and $\{\nu_i\}_{i=1}^n$ be non-negative weights associated with the local models based on their importance or on the size of the corresponding data sources. The objective of the DMC problem is to obtain the optimal global model λ_c^* belonging to a given family of models $\mathcal{F}$, i.e., $\lambda_c^* = \operatorname{argmin}_{\lambda_c \in \mathcal{F}} \mathcal{Q}(\lambda_c)$, where $\mathcal{Q}(\cdot)$ is the model quality cost. A natural definition of the quality cost is the " distance" from the underlying true model λ^0, but since λ^0 is unknown, we instead consider the different local models $\{\lambda_i\}_{i=1}^n$ as estimators of λ^0 and define the quality cost function in terms of the average distance from the local models, i.e., $\mathcal{Q}(\lambda_c) = \sum_{i=1}^n \nu_i D(\lambda_i, \lambda_c)$. where $D(\cdot, \cdot)$ is a suitable distance measure for models such as the L_1 distance, the squared L_2 distance and KL-divergence. Of these, KL-divergence is the most appropriate measure since it is linearly related to the average log-likelihood of the data generated by one model with respect to the other and hence, we optimize the quality cost function based on the KL-divergence and use other measures only for secondary eval-

uation of the experimental results.

3.1 DMC Algorithm

It can be shown [13] that the DMC problem, i.e., finding the global model of a specified parametric family that has the lowest average KL-divergence to a set of local models, is exactly equivalent to that of finding the model in the specified family that is closest to the mean of the local models in terms of KL-divergence. This follows from the result that the average KL-divergence of a set of models to any model, λ_c can be written as the sum of the average KL-divergence to the mean model and the KL-divergence between the mean model and λ_c. In the absence of any constraints, i.e., when no parametric family is specified, the optimal solution is just the mean model $\bar{\lambda}$ such that $p_{\bar{\lambda}}(x) = \sum_{i=1}^{n} \nu_i p_{\lambda_i}(x)$.

The mean model also has the nice property that its distance from the true model is always less than or equal to the average distance of the set of local models from the true model for all distance measures that are convex in the second argument[1]. Since the true model λ^0 is unknown, it is not possible to find out which of the models $\{\lambda_i\}_{i=1}^n$ is more accurate in terms of distance to the true model, but the mean model is guaranteed to provide an improvement over the average quality of the available models. The mean model is thus a good choice in terms of the quality, but it will in general have a large number of overlapping components and not belong to the specified parametric family $\mathcal{F}$. For the sake of interpretability, we look at the simplified version of the DMC problem, i.e., finding the model in $\mathcal{F}$ that is closest to the mean model in terms of KL-divergence, i.e.,

$$\lambda_c^* = \underset{\lambda_c \in \mathcal{F}}{\operatorname{argmin}} \; D_{KL}(\bar{\lambda}, \lambda_c) \tag{1}$$

This new optimization problem (1) is difficult to solve directly using gradient descent techniques. Therefore, we pose an approximate version of the above problem based on a dataset $\bar{\mathcal{X}} = \{x_j\}_{j=1}^m$ sampled from the mean model. Consider the problem of finding the model $\lambda_c^a \in \mathcal{F}$ that maximizes the average log-likelihood of the dataset $\bar{\mathcal{X}}$. As the size of the dataset $\bar{\mathcal{X}}$ goes to ∞, the average log-likelihood converges to the cross entropy between the densities $p_{\bar{\lambda}}$ and p_{λ_c}. Now, the cross entropy between any two densities is linearly related to the KL-divergence between them, and hence, maximizing the cross entropy with respect to the mean model is equivalent to minimizing the KL-divergence with respect to the mean model. The approximate problem, therefore, converges to the unsupervised DMC problem (1) as the size of $\bar{\mathcal{X}}$ goes to ∞.

Viewing this approximate problem as a maximum-likelihood parameter estimation problem leads to the DMC algorithm 1. The main idea is to first generate a dataset $\bar{\mathcal{X}}$ following the mean model $\bar{\lambda}$, using Markov Chain Monte Carlo (MCMC) sampling techniques [14] and then, apply the EM algorithm [7] to this dataset to obtain the clustering model $\lambda_c^a \in \mathcal{F}$ that maximizes its likelihood of being observed. The resulting model λ_c^a is a local minimizer of the approximate problem and not necessarily the same as the solution λ_c^* of the original unsupervised DMC problem. However, it is guaranteed to asymptotically converge to a locally optimal solution as the size of $\bar{\mathcal{X}}$ goes to ∞. In practice, one can use multiple runs of the EM algorithm and pick the best solution among these so that the obtained model is reasonably close to the globally optimal model.

Algorithm 1 DMC Algorithm

Input: Set of clustering models $\{\lambda_i\}_{i=1}^n$ with weights $\{\nu_i\}_{i=1}^n$ summing to 1, Mixture model family $\mathcal{F}$.

Output: $\lambda_c^a \simeq \underset{\lambda_c \in \mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^n \nu_i D_{KL}^{\text{clus}}(\lambda_i, \lambda_c)$

Method:

1. Obtain mean model $\bar{\lambda}$ such that

$$p_{\bar{\lambda}}(x) = \sum_{i=1}^{n} \nu_i p_{\lambda_i}(x).$$

2. Generate $\bar{\mathcal{X}} = \{x_j\}_{j=1}^m$ from mean model, $\bar{\lambda}$ using MCMC sampling.

3. Apply EM algorithm to obtain the optimal model, λ_c^a, such that

$$\lambda_c^a = \underset{\lambda_c \in \mathcal{F}}{\operatorname{argmax}} \; L(\bar{\mathcal{X}}, \lambda_c) = \underset{\lambda_c \in \mathcal{F}}{\operatorname{argmax}} \; \frac{1}{m} \sum_{j=1}^{m} \log(p_{\lambda_c}(x_j)).$$

3.2 Privacy Costs

In this section, we quantify the privacy cost using ideas from information theory and also show that there is an inverse relation between the privacy of the local models and the quality of the mean model. In order to quantify privacy, we need a measure that indicates the uncertainty in predicting the original dataset from the model. We propose that the privacy, $\mathcal{P}(x, \lambda)$ of an object x given a model λ be defined in terms of the probability of generating the data object from the model. The higher the probability, the lower the privacy. More specifically, noting that the reciprocal of the probability is related to uncertainty [6], we have $\mathcal{P}(x, \lambda) = (p_\lambda(x))^{-1}$.

For vector data, $\mathcal{P}(x, \lambda) = 1$ implies that x can be predicted with the same accuracy as a random variable with a uniform distribution on a ball of unit volume. We can now define the privacy, $\mathcal{P}(\mathcal{X}, \lambda)$ of a dataset $\mathcal{X}$ with respect to the model as some function of the privacy of the individual data objects. The geometric mean has a nice interpretation as the reciprocal of the average likelihood of the dataset being generated by the model, assuming that the individual

[1] Examples of distance functions that are convex in the density function of the second argument include KL-divergence, L_1 distance and squared L_2 distance.

samples are i.i.d., i.e.,

$$\mathcal{P}(\mathcal{X}, \lambda) = \left(\prod_{x \in \mathcal{X}} p_\lambda(x) \right)^{\frac{-1}{|\mathcal{X}|}} = 2^{(-\frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \log_2 p_\lambda(x))}.$$

A higher likelihood of generating the dataset from the model implies a lower amount of privacy. For example, let us consider vector space data being modeled by a mixture of Gaussians. A highly detailed model with Gaussians of vanishing variance, centered at each of the data objects gives away the entire dataset and has no privacy. This is to be expected as the probability density $p_\lambda(x)$ goes to ∞, for all data objects $x \in \mathcal{X}$ making the privacy measure go to 0^+. On the other hand, a very coarse model, say with a single Gaussian of high variance has a low likelihood of generating the data and hence, has a high privacy.

Intuitively, if the local models are more detailed, the combined model can be improved at the cost of decreased privacy. In particular, there is a linear relation between the average logarithm of privacy (log-privacy) of the local models and the quality of the optimal mean model. Using the weak law of large numbers and Chebyshev inequality [15], it can be shown that the log-privacy of the local models, λ_i converges to the cross-entropy between p_{λ_i} and the true distribution, p_{λ^0}, when the size of the individual data sources tend to ∞. As a result, the average log-privacy of the local models converges to their average cross-entropy, which differs from the KL-divergence between the mean model and the true model only by a constant. As the privacy of the local models increases, the ideal quality cost of the mean model, which is the optimal model with no constraints, also goes up. On the other hand, when the privacy of the local models decreases, the mean model tends to be more accurate.

3.3. Experimental Evaluation

In this section, we provide empirical evidence that for a reasonable global sample size and privacy level, the global model obtained through the DMC algorithm is as good as or better than the best local model for different types of data not only in terms of KL-divergence but also for other distance measures. We performed experiments on four different types of data, namely, vectors in Euclidean space, high dimensional directional vectors on a hypersphere (a popular representation of text, for example), discrete symbol sequences and continuous sequences. For this purpose, we generated synthetic datasets based on mixtures of Gaussian, von Mises-Fisher(VMF), discrete and continuous Hidden Markov models(HMMs) respectively using MCMC sampling techniques. These datasets were then used as the distributed data sources for obtaining the local clustering models using the appropriate form of the EM algorithms. Details of the datasets and algorithms can be found in [13]. For

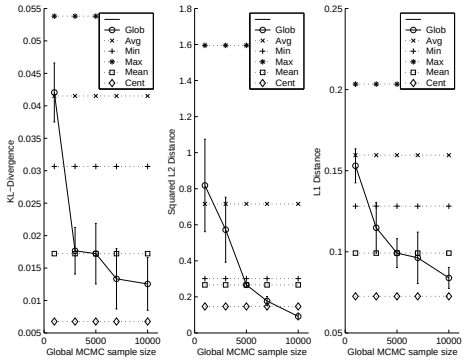

Figure 1: Variation of global model quality with sample size.

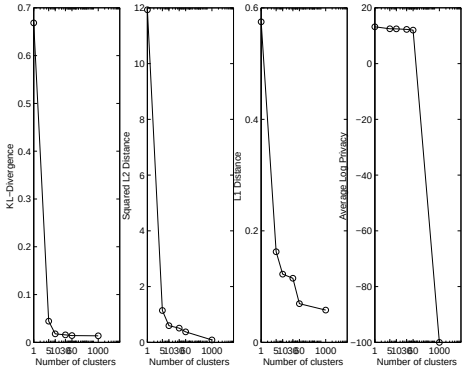

Figure 2: Variation of privacy and cluster quality w.r.t. base model resolution.

each experiment, we computed the privacy costs of the local models and the ideal quality costs, based on the L_1 distance, squared L_2 distance and the KL-divergence measures.

We first studied the performance of the DMC algorithm on the Euclidean vector datasets for different choices of global MCMC sample size and local model resolution. Based on this, we chose good values for the global sample size and model resolution and applied it to different data types.

An important step in the DMC algorithm is choosing the global MCMC sample size. Figure 1 shows that the quality of the global model improves with the number of artificially generated samples, with diminishing returns after a point. When the sample size increases to that of the combined size of all the data sources, the global model is better than even the best of the local models. Another significant aspect of our framework is the trade-off between privacy restrictions and the quality of the combined model obtained, which can be controlled by picking a suitable model resolution. From figure 2, we note that the average log-privacy as well as the quality costs decrease as the number of clusters increases. Figure 3 shows the quality of the different models for different data types with the global sample size chosen to be equal to the combined size of all the data sources and the

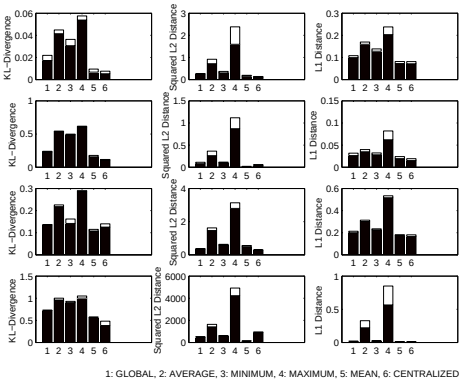

Figure 3: Global model quality for different types of data. The rows 1-4 correspond to the results on Gaussian, directional, discrete and continuous sequence data respectively. The black bar represents the average value and the white bar represents the standard deviation.

model resolution of the local models chosen to be the same as that of the true model. In all the cases, the global model performs better than the best local model and is in general closer to the the quality of the centralized model.

4. Concluding Remarks

We described two settings for distributed clustering under restrictions on what type of information can be shared among the different local sites. The first setting allowed only cluster labels to be shared, but not individual features or specifics of the algorithms employed. This setting does not need the same set of features to be present at each site, and can also cater to missing records. A solution was provided by defining and obtaining a single consensus clustering. The second setting did not allow even cluster labels to be shared. However the sets of features at each site need to be the same, since that allows a way of getting a unified solution out of generative models developed for each local data set. Surprisingly good results are obtained in each of these settings, even though the sharing restrictions are rather severe. Further studies of distributed data mining for a wider range of data analysis goals/procedures and information sharing restrictions are warranted in order to unearth the potential of this emerging pattern recognition area.

Acknowledgment. We would like to acknowledge support from the NSF under grants ECS-9900353 and IIS-0307792 and from Intel Corp. Alexander Strehl is a co-author on the work on cluster ensembles.

References

[1] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *PODS*, pages 247–255, 2001.

[2] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *ACM SIGMOD*, pages 439–450, 2000.

[3] K. D. Bollacker and J. Ghosh. Effective supra-classifiers for knowledge base construction. *Pattern Recognition Letters*, 20(11-13):1347–52, 1999.

[4] I. V. Cadez, S. Gaffney, and P. Smyth. A general probabilistic framework for clustering individuals and objects. In *KDD*, pages 140–149, 2000.

[5] P. Chan, S. Stolfo, and D. Wolpert. Integrating multiple learned models for improving and scaling machine learning algorithms. *Machine Learning*, 36(1-2), 1996.

[6] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[8] I. S. Dhillon and D. S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In M. Zaki and C. Ho, editors, *Large Scale Parallel Data Mining*, pages 245–260. LNCS vol 1759. Springer, 2000.

[9] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *KDD*, pages 217–228, 2002.

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

[11] J. Ghosh. Multiclassifier systems: Back to the future. In F. Roli and J. Kittler, editors, *Multiple Classifier Systems*, pages invited paper, 1–15. LNCS Vol. 2364, Springer, 2002.

[12] E. Johnson and H. Kargupta. Collective, hierarchical clustering from distributed, heterogeneous data. In M. Zaki and C. Ho, editors, *Large-Scale Parallel KDD Systems*, pages 221–244. LNCS Vol. 1759, Springer, 1999.

[13] S. Merugu and J. Ghosh. Privacy perserving distributed clustering using generative models. In *ICDM*, 2003.

[14] R. M. Neal. Probabilistic inference using Markov Chain Monte Carlo methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, Univ. of Toronto, 1993.

[15] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, New York, 1984.

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

[17] A. Strehl and J. Ghosh. A scalable approach to balanced, high-dimensional clustering of market-baskets. In *Proc. HiPC*, pages 525–536. LNCS Vol. 1970, Springer, 2000.

[18] A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining partitionings. *JMLR*, 3(3):583–617, 2002.

[19] J. Vaidya and C. Clifton. Privacy-perserving k-means clustering over vertically patitioned data. In *KDD*, pages 206–215, 2003.

[20] K. Yamanishi. Distributed cooperative Bayesian learning strategies. *Information and Computation*, 150:22–56, 1998.

[21] S. Zhong and J. Ghosh. A unified framework for model-based clustering and its applications to clustering time sequences. Technical report, ECE Dept., The University of Texas at Austin, 2002.