

# Scalable clustering on the data grid

Patrick Wendel  
pjw4@doc.ic.ac.uk

Moustafa Ghanem  
mmg@doc.ic.ac.uk  
Discovery Net  
Imperial College, London

Yike Guo  
yg@doc.ic.ac.uk

## Abstract

With the development of e-Science service-oriented infrastructures based on the Grid and service computing, many data grids are being created and are becoming a new potential source of information available to scientists and data analysts. However mining distributed data sets still remains a challenge. Following the assumption that data may not be easily transferred from one site to another, or gathered at a single location (for performance, confidentiality or security reasons), we present a framework for distributed clustering where the data set is partitioned between several sites and the output is a mixture of Gaussian models. The data providers generate locally a clustering model using different classic clustering techniques (for the moment K-Means, EM) and return it to one central site using a standard PMML representation for the model. The central site uses these models as starting observations for EM iterations to estimate the final model parameters. Although known to be slower than other simpler techniques, EM is here applied to a relatively small set of observations and provides a probabilistic framework for the model combination. An initial version of the framework has been implemented and deployed on the Discovery Net infrastructure that provides support for data, resource management and workflow composition. We present empirical results that show the advantages of this approach and show that the final model stays accurate while allowing the mining of very large distributed data sets.

## 1 Introduction

With the development of e-Science service-oriented infrastructures based on the Grid and service computing, many data grids are being created and are becoming a new potential source of information available to scientists and data analysts. Hence, it is becoming a common scenario for an analytical process to require data spread across multiple organisations,

in different cities or countries. Each data source may have different bandwidth, security or restriction policies, thus making the analysis more difficult.

On the other hand, the analysis of larger and larger data sets is not only a consequence of the increasing amount of information stored in databases and data warehouses, but also a requirement for improving the quality of the analysis [12]. However most of the techniques developed for data analysis were not initially designed to cope with these amounts of data. Neither did they take into account data location and accessibility.

One of those techniques is data clustering, an unsupervised technique to find homogeneous groups of instances within a data set. It has been successfully applied to sociological data in order to find patterns of behaviour [2], to image processing [15] and more recently to genomic data such as gene expression data [14], where partitional clustering is used to find groups of genes with similar activities over a set of experiments.

Clustering large data sets [3] is an inherently difficult problem. Solutions like sampling the data are not always accurate enough for discovering representative patterns. In the case of distributed data sources, the problem becomes even more complex, as the data may not or should not be moved between sources because of restrictive policies, or at least because it is very expensive to do so. Thus, we dismiss the possibility of applying either a sample-based approach or gathering all the data first. Another issue further arising from having distributed data sources, is the *skewness* of the data distribution. It is probable that different parts of the data contain different clusters, and if this is the case, then a local model may not represent very well the full data set.

In this paper, we aim to provide a framework for distributed clustering. The basic approach is to cluster distributed data locally and then combine clusters generated separately, on different distant sites, on different sets of observations by using a model combiner.

An outline of the framework described in Figure 1 is as follows:

1. Local clustering sites generate clusters and associated statistics, based on local data sets only.
2. These models are returned to a combiner site using a standard representation.
3. Model weights are normalised
4. The combiner applies a learning process to generate a global clustering model.

The learning approach adopted by the combiner is based on a model parameters estimation technique (EM clustering), taking into account all the statistics available as input.

Using a probabilistic approach also opens new possibilities to extend the framework. Examples of extensions include the estimation of the number of clusters contained in the data set using the Bayesian Information Criterion [9], and an iterative refinement process using the global model as prior knowledge for estimating new local models. These extensions are not discussed in this article which concentrate on the framework for combining the models generated locally.

We have deployed this framework using Discovery Net[1], an e-Science platform which provides data management and resource composition.

We evaluate this framework against synthetic data sets to show that it consistently outputs models of better quality than the average quality of local models. We also show that under certain conditions relating to the skewness of the data partitioning, the quality of the combined model becomes significantly better.

## 2 Model-based Clustering

Clustering techniques are unsupervised techniques used to find groups of points within a data set. We focus on model-based clustering where a cluster is defined by the parameters of the distribution that best fit it. Clusters are usually supposed to follow a Gaussian distribution and the problem of estimating clusters in a data set can be reduced to a statistical parameter estimation problem for a mixture of Gaussian models. Therefore, the general statistical EM (Expectation-Maximisation) algorithm[7], which finds the parameters that locally maximise the likelihood of the distribution, can be applied naturally for computing the cluster density functions. In this case, each data point does not simply belong to one and only one partition but has a probability of belonging to each Gaussian component. This is why the technique is often referred to as a *soft assignment* technique.

## 3 Meta-learning and cluster combination

Learning from models, also called meta-learning [5], is a possible strategy for scaling up learning processes, as the data set can be partitioned or sampled to generate several partial models that can be used to build a global model. Meta-learning for distributed data has been successfully applied, for instance, to classification algorithms [10]. We are aiming to apply this strategy to clustering models. In essence, the combination process consists of stacking the density estimations provided by the input clusters[13] before starting the final parameters estimation.

### 3.1 Goal

The main advantages of the framework are:

- **Distributed mining:** Data is not moved from site to site.
- **Openness:** Clustering algorithms on each site can be different and exchanged with any clustering program, compliant with a standard representation, for instance PMML[6].
- **Incremental clustering:** If local clustering processes can return results at any time then the combiner can use this information to return a model of the data set processed so far.

### 3.2 Designing a combiner

#### 3.2.1 Local cluster representation

In the process of combining the models, each input cluster is represented by its weight, its mean and its variance along each dimension (diagonal of the covariance matrix of this cluster).

#### 3.2.2 Process

We describe here the process for building a model.

1. Request clustering models to be built for each partition where the data is situated.
2. Generate local models by using any algorithm whose output would comply with the cluster representation described before. In our experiments, the local models are generated by using K-Means with random initial points. The result provides an initial parameter estimation for EM clustering, and the final result is an estimation of the cluster weights, centroids and covariance matrices. For local clustering we estimate the diagonal of the covariance matrix for each cluster.

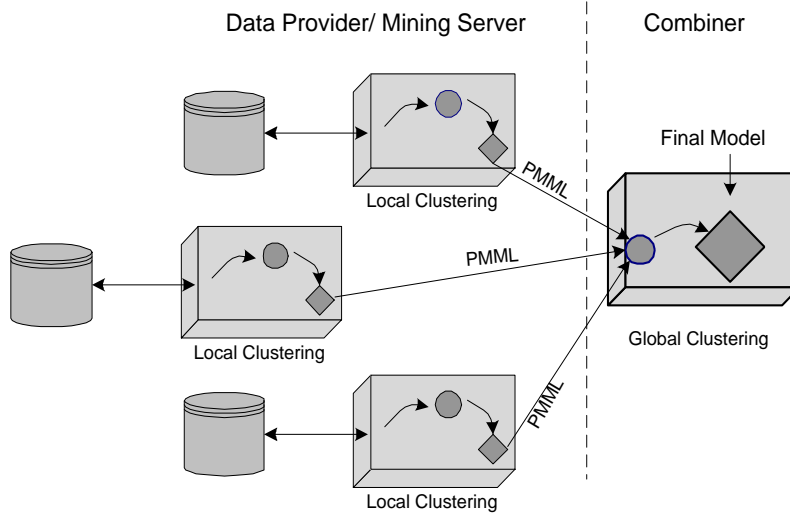


Figure 1: Framework for distributed clustering

3. Send local models to the combiner site waiting for all the processes to be finished.
4. Re-weight models: taking into account what fraction of the complete data set the model represents, each cluster is assigned a new weight such that the total weight of all the input clusters is 1. If the data set is partitioned into  $p$  sets, each generating  $k$  clusters  $C_k^p$  of weight  $w_k^p$ , then  $\sum_{i=1}^p \sum_{j=1}^k w_j^i = 1$

At this stage, the combiner site has all the input needed to start building the global model. We can now describe in detail the EM iteration.

### 3.2.3 EM iteration for clusters

We adopt the following notations:

- The  $d$ -dimensional Gaussian distribution of mean  $\mu \in \mathbf{R}^d$  and diagonal covariance matrix  $\Sigma = \sum_{i=0}^d \sigma_i^2 \delta_{i,i}$  is denoted by  $\mathcal{N}(\mu, \Sigma)$ .
- $f_X$  is the density function of a random variable  $X$ .
- The density function of the Gaussian distribution  $\mathcal{N}(\mu, \Sigma)$  is denoted by  $f_{\mathcal{N}(\mu, \Sigma)}$ .
- The input  $C$  is the set of observations and  $|C| = I$ . Each observation  $C_i$  is a triplet of sufficient statistics representing a group of points.  $C_i = \{w_i, \mu_i, \Sigma_i\}$ , where  $w_i$  is the weight of the cluster,  $\mu_i$  is its mean and finally  $\Sigma_i$  is the covariance matrix. Therefore each cluster is only represented by its variance along each attribute.

- The output model at iteration  $l$   $M^l$  is a mixture model composed of  $K$  Gaussian models.  $M^l = \sum_{k=1}^K \alpha_k \cdot M_k^l$  where  $M_k^l \sim \mathcal{N}(\mu_k^l, \Sigma_k^l)$

Each iteration of the EM algorithm is decomposed into a likelihood estimation step (E-step) and a parameters re-estimation step (M-step). The algorithm stops when the likelihood no longer increases. We present now a modified version of the original EM clustering algorithm, in order to accommodate the particular type of input (clusters) we are dealing with.

**Likelihood estimation (E-step)** The first step of the algorithm consists in calculating the probability of each model knowing the observations:

$$P(M_k^l | C_i) = w_i \cdot P(M_k^l | \mu_i, \Sigma_i) \quad (1)$$

Then using the Bayes formula:

$$P(M_k^l | \mu_i, \Sigma_i) = \frac{P(M_k^l) \cdot P(\mu_i, \Sigma_i | M_k^l)}{P(\mu_i, \Sigma_i)} \quad (2)$$

We can first estimate  $P(\mu_i, \Sigma_i | M_k^l)$  without considering the covariance matrix  $\Sigma_i$ :

$$P(\mu | M_k^l) = f_{M_k^l}(\mu) \quad (3)$$

However, instead of using the centroid as the only representative of the local cluster, we can use its estimated distribution. First, we can estimate the density function representing an observation  $f_{C_i} = f_{\mathcal{N}_{\mu_i, \Sigma_i}}$ . Then, we can define  $X_i$  a random variable that represents the observation  $C_i$ . The variable  $Y_i = f_{M_k^l}(X_i)$

represents the probability of  $X_i$  belonging to cluster  $M_k^l$ , and finally we can use the expectation of  $Y_i$  as an estimation of the probability of the observation  $C_i$  belonging to cluster  $M_k^l$ . In our case, we only store information about the variance of each attribute within a cluster, thus simplifying the formula as follows:

$$\begin{aligned} P(\mu_i, \Sigma_i | M_k^l) &= E[f_{\mathcal{N}_{\mu_k^l, \Sigma_k^l}}(X_i)] \\ &= f_{\mathcal{N}_{0, \Sigma_i + \Sigma_k^l}}(\mu_i - \mu_k^l) \end{aligned} \quad (4)$$

Using that, the probability takes into account the distribution of the observation  $C_i$  as well as the distribution  $M^l$  at iteration  $l$ .

### Parameters update (M-step)

- The weight of each cluster is the sum of the membership of all the observation. There is no need to divide by the number of points, as the weight of each point is taken into account in the probability calculation:

$$\alpha_k^{l+1} = \sum_{C_i} P(M_k^l | C_i) \quad (5)$$

- Estimate the new mean, knowing the membership of all the points (as in the standard version):

$$\mu_k^{l+1} = \frac{\sum_{C_i} P(M_k^l | C_i) \cdot \mu_i}{\sum_{C_i} P(M_k^l | C_i)} \quad (6)$$

- Compute the new covariance matrix. Instead of estimating the covariance directly from the centroid coordinates of the input clusters, we use their estimated covariance:

$$\Sigma_k^{l+1} = \frac{\sum_{C_i} P(M_k^l | C_i) \cdot \Sigma_i}{\sum_{C_i} P(M_k^l | C_i)} \quad (7)$$

This is also how the covariance is estimated in SEM[4], when updating the model against the summarization structure.

### 3.2.4 Initialisation

The EM algorithm being quite sensitive to initial parameters, we have to make sure it starts with meaningful values. The basic idea is to form some preliminary clusters based on a partition of the set of input clusters, into  $K$  groups. This is achieved by using K-Means on the set of input cluster centroids. We choose not to use any modified version of K-Means that would take into account the weight of each input, in order to make sure we start with well-separated means. After that, the estimation of the

initial weight of a cluster is the sum of the weight of the clusters of the partition. The covariance is also the weighted average covariance for each partition found by K-Means.

Assuming K-Means partitioned the input set into  $k$ , each part containing  $n_j$  input clusters,  $j \in 1, \dots, k$ ,  $C_{i,j} = \{w_{i,j}, \mu_{i,j}, \Sigma_{i,j}\}$ ,  $i \in 1, \dots, n_j$ , then the estimated weight of the output cluster  $j$  is  $\sum_{i=1}^{n_j} w_{i,j}$  and its estimated covariance is  $\sum_{i=1}^{n_j} w_{i,j} \cdot \Sigma_{i,j}$ .

## 4 Deployment

In this section, we describe how we use the Discovery Net infrastructure to deploy the framework.

### 4.1 Discovery Net

Discovery Net[1] is an infrastructure for building Grid-based knowledge discovery applications. It enables the creation of high-level, re-usable and distributed application workflows that use a variety of common types of distributed resources. It is built on top of standard protocols and standard infrastructures such as Globus but also defines its own protocols such as the Discovery Process Mark-up Language (DPML) for data-flow management. In the context of this application, it is used to coordinate the execution at each clustering site and the gathering of the resulting models to be merged. It also provides the table management required for handling the type of relational data structure used as input for data clustering.

### 4.2 Distributed Clustering

As shown in Figure 4.2, we use the Discovery Net client to compose, trigger and monitor the execution of the distributed mining process.

## 5 Evaluation

### 5.1 Methodology

The first evaluation is carried out on the quality of the models. For that, we use the likelihood as a quality measure. We compare both the likelihood of the model against the entire data set and the likelihood of the model against the generating model. The second evaluation focuses on measuring performance and scalability.

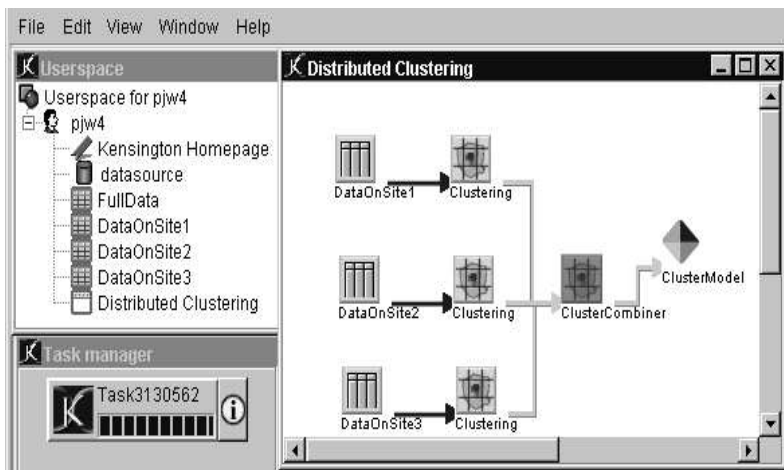


Figure 2: Discovery Net client. Result of distributed clustering

### 5.1.1 Data Skewness

The way the data is partitioned is very likely to influence our quality assessment. If the data is fairly randomly distributed, then the model generated from any sample will be of high quality, particularly with very large data sets. The combiner will keep the accuracy of these models, but in any case cannot output a significantly better model, as this is a worst case configuration.

Conversely if the data sets have completely different densities then the combiner will certainly outperform the locally estimated models.

In order to test this, we introduce a ‘*randomness*’ factor in the data set generator we use for quality evaluation. If this factor is 0 then the generator just concatenates values simulated for each Gaussian component one after the other. A factor of 1 will generate a randomly distributed data set i.e. the choice of which component a point will be simulated from, is random.

### 5.1.2 Quality evaluation

For each experiment we show the average log-likelihood (AvgLL1) of the input models and the combined model log-likelihood (CLL1) against the complete data set. We also show the same measure against the generating model (AvgLL2 and CLL2)

Table 1 shows how the framework scales in terms of data size. We use a data set with 10 Gaussian components with 5 dimensions, a randomness ratio of 0.2, 5 groups and a number of points  $\in \{1000, 10000, 100000, 1000000\}$ . Clearly the combined model stays accurate even with large data sets.

Table 2 shows how the combined model becomes

Ratio	AvgLL1	CLL1	AvgLL2	CLL2
1.0	-40983	-40970	-73	-73
0.8	-35460	-35010	-68	-67
0.6	-31357	-29681	-59	-55
0.4	-44996	-38160	-90	-71
0.2	-46453	-31175	-86	-52
0.0	-110602	-18127	-141	-21

Table 2: *Randomness* ratio effect

more accurate when the *randomness* ratio decreases. The data set has 5 dimensions, 5 Gaussian components for a total of 10000 points and is partitioned into 5 sets. The best case is obviously when the ratio is null, meaning that each partition has completely different underlying distribution and cannot provide an accurate model of the whole data set whereas the combined model can easily return a more accurate model by taking into account the different density distributions.

In Table 3, the parameter is the number of clusters. The data set contains 10000 points of dimension 5 and a randomness ratio of 0.2. It is also partitioned into 5 data sets of equal size. The increasing number of clusters does not prevent the combined model from outperforming the local models.

We then test the scalability of the framework in terms of dimensionality. Again the other fixed parameters are 10000 points partitioned in 5 data sets, a ratio of 0.2 and 5 clusters. The results are shown in Table 4. Although we did not test for very high dimensional data set, the results are encouraging, as there is no degradation of the likelihood of the combined model.

NbPts	AvgLL1	CLL1	AvgLL2	CLL2
1000	-4331	-2558	-81	-43
10000	-42306	-32623	-81	-63
100000	-478452	-333851	-87	-56
1000000	-4577461	-2649639	-84	-50

Table 1: Increasing data size

K	AvgLL1	CLL1	AvgLL2	CLL2
2	9559	24091	5	8
3	8109	21165	5	10
5	-26312	8620	-26	8
10	-33520	-7824	-94	-24
15	-48041	-33174	-173	-113

Table 3: Increasing number of clusters

D	AvgLL1	CLL1	AvgLL2	CLL2
2	-21970	-15154	-25	-17
4	-20458	-2652	-35	-3
5	-30272	6656	-36	6
10	-73376	28561	-73	30
15	-165102	26884	-116	-8

Table 4: Increasing dimensionality

Finally the variable is the number of partitions. The number of points equals to 10000 with 5 clusters, 0.2 ratio, and 5 dimensions. Table 5 contains the results.

### 5.1.3 Performance

To measure performance, we use a homogeneous cluster with a maximum of 5 workstations (Pentium III 700 Mhz). The data set is first partitioned and sent to the 5 machines. In our experiment, one of the

D	AvgLL1	CLL1	AvgLL2	CLL2
2	-4926	-1683	-14	-5
3	-22122	-7255	-72	-49
4	-35659	-11827	-78	-26
5	-32320	-6000	-66	-15
6	-44027	-17068	-93	-25
7	-50808	-20183	-125	-40
8	-54917	-20759	-120	-46
9	-59847	-16208	-135	-22
10	-66745	-19331	-150	-35

Table 5: Number of partitions

machines also acted as the combiner site. We have measured the total time in seconds to generate the global model. In Figure 5.1.3 and Figure 5.1.3,  $k$  represents the number of clusters and  $d$  the dimension of the data set. As expected the framework scales well in terms of execution time, as standalone clustering processes are carried out concurrently on fragments of the data.

## 6 Related work

A parallel version of K-Means for distributed memory architecture[8] using MPI allows the application of very large data sets while conserving the initial exact K-means algorithm. While related to the method we propose, solutions based on MPI usually requires a tight integration between the processors such as on distributed memory multiprocessors machines. However there are Grid-enabled versions of MPIs that could be used here instead. In [11], the author presents an incremental version of the EM algorithm (online EM) based on a gradient descent method. SEM[4] reuses the summarization framework defined for ScaleKM but the model estimation uses the EM method, resulting in a fast and incremental algorithm. These methods are other ways to scale up the algorithm we are using and although they cannot be applied under the same constraints of distributed data, they do solve the scalability issue for one data source.

## 7 Conclusion-Discussion

We presented and evaluated a framework for combining models based on a parameter estimation technique (EM). The result is promising as the combined model can consistently outperform the average likelihood of the models generated on each part of the data set, thus allowing the mining of distributed data sets without data transfer, or high bandwidth requirements between the collaborating data grids. Models have to be sent from each data source to the combiner site but the size of these models allows, for instance, the utilisation of secure transmission techniques in a

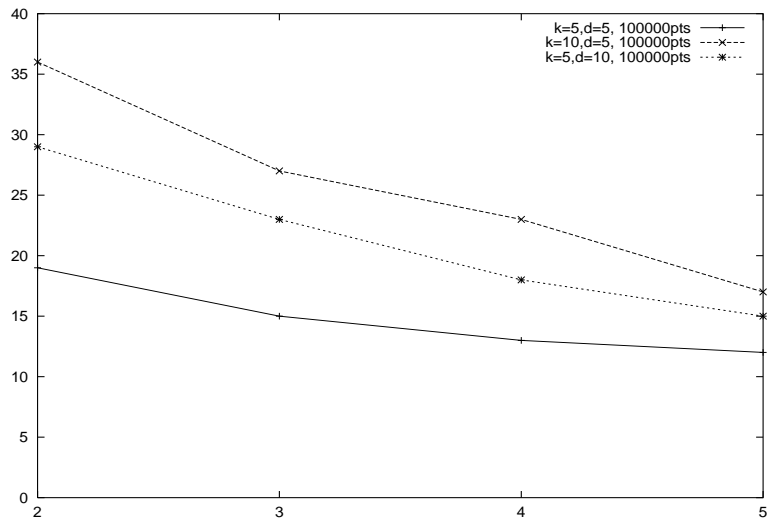


Figure 3: Execution time / Number of data partitions

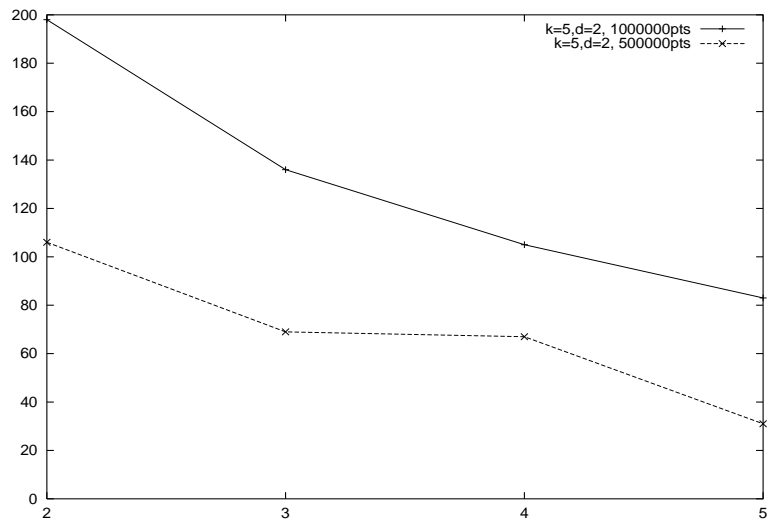


Figure 4: Execution time / Number of data partitions

reasonable time. The use of a standard cluster representation has been a choice for the future openness and reusability of the framework.

We restricted our evaluation to the case where the number of clusters is known and therefore the problem of choosing the right  $k$  for the local site is avoided, however the framework has theoretically no constraints on the number of clusters generated on each site. It only assumes that it can start with the defined compressed representation of the distributed data set. This is why we are working now on extending the framework in order to use other data summarization techniques on each site.

The Discovery Net infrastructure has provided us the service required to build this application to distributed clustering, in particular through its data management and workflow composition services.

In an e-Science environment where more and more data grids are created, more and more data is generated and functionalities are provided through services using open standards, this approach also shows that it is possible to compose complex solutions based on these building blocks.

## 8 Acknowledgements

This research is part of the Discovery Net, an EPSRC e-Science pilot project.

## References

- [1] S. AlSairafi, F.-S. Emmanouil, M. Ghanem, N. Giannadakis, Y. Guo, D. Kalaitzopoulos, M. Osmond, A. Rowe, J. Syed, and P. Wendel. The design of discovery net: Towards open grid services for knowledge discovery. *International Journal of High Performance Computing Applications*, 17, Aug. 2003.
- [2] E. Bijnen. *Cluster analysis*. Tilburg University Press, 1973.
- [3] P. Bradley, U. Fayyad, and C. Reina. Scaling Clustering Algorithms to Large Databases. In *Fourth International Conf. on Knowledge Discovery and Data Mining*. AAAI Press, Aug. 1998.
- [4] P. Bradley, U. Fayyad, and C. Reina. Scaling EM (expectation maximization) clustering to large databases. Technical Report MSR-TR-98-35, Microsoft Research, Redmond, WA, Nov. 1998.
- [5] P. K. Chan and S. J. Stolfo. Experiments on Multi-Strategy Learning by Meta-Learning. In B. Bhargava, T. Finin, and Y. Yesha, editors, *2nd International Conference on Information and Knowledge Management*, pages 314–323, New York, NY, USA, Nov. 1993. ACM Press.
- [6] The Data Mining Group. [<http://www.dmg.org>].
- [7] A. Dempster, N. Laird, and D. Rubin. Maximum Likelihood from Incomplete Data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.
- [8] I. Dhillon and D. Modha. A Data Clustering Algorithm on Distributed Memory Multiprocessors. In *Workshop on Large-Scale Parallel KDD Systems*, 1999.
- [9] C. Fraley and A. E. Raftery. How many clusters? Which clustering method? Answers via model-based cluster analysis. *Computer Journal*, 41(8):578–588, 1998.
- [10] Y. Guo and J. Sutiwaraphun. Distributed classification with knowledge probing. In H. Kargupta and P. Chan, editors, *Advances in distributed and parallel knowledge discovery*, chapter 4. AAAI Press, 2000.
- [11] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998.
- [12] F. Provost. Distributed data mining: scaling up and beyond. In H. Kargupta and P. Chan, editors, *Advances in distributed and parallel knowledge discovery*, chapter 1. AAAI Press, 2000.
- [13] P. Smyth and D. Wolpert. Stacked Density Estimation. In *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.
- [14] J. Vilo, A. Brazma, I. Jonassen, A. Robinson, and E. Ukkonen. Mining for Putative Regulatory Elements in the Yeast Genome using Gene Expression Data. In *Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 383–394, 2000.
- [15] T. Zhang. *Data Clustering for Very Large Datasets Plus Applications*. PhD thesis, Computer Sciences Dept. at Univ. of Wisconsin-Madison, 1996.