In Proc. 5th IEEE International Conference on Data Mining (ICDM), Houston, TX, 2005

Effective and Efficient Distributed Model-based Clustering

Hans-Peter Kriegel, Peer Kröger, Alexey Pryakhin, Matthias Schubert
Institute for Computer Science, University of Munich, Germany
{kriegel,kroegerp,pryakhin,schubert}@dbs.ifi.lmu.de

Abstract

In many companies data is distributed among several sites, i.e. each site generates its own data and manages its own data repository. Analyzing and mining these distributed sources requires distributed data mining techniques to find global patterns representing the complete information. The transmission of the entire local data set is often unacceptable because of performance considerations, privacy and security aspects, and bandwidth constraints. Traditional data mining algorithms, demanding access to complete data, are not appropriate for distributed applications. Thus, there is a need for distributed data mining algorithms in order to analyze and discover new knowledge in distributed environments. One of the most important data mining tasks is clustering which aims at detecting groups of similar data objects. In this paper, we propose a distributed model-based clustering algorithm that uses EM for detecting local models in terms of mixtures of Gaussian distributions. We propose an efficient and effective algorithm for deriving and merging these local Gaussian distributions to generate a meaningful global model. In a broad experimental evaluation we show that our framework is scalable in a highly distributed environment.

1 Introduction

Many companies gather terabytes of data containing significant, hidden information that needs to be uncovered using data mining techniques. Nowadays, more and more companies typically are decentrally organized and spatially distributed. Each site generates its own data and manages its own data repository. The challenge, however, is to access current knowledge from anywhere at any time with a delay of no more than a few minutes. Analyzing and mining these distributed sources requires distributed data mining techniques because the wanted patterns should be mined on a global view of these distributed sources. However, the traditional approach for a data mining environment is a centralized warehouse-based architecture where the relevant data is regularly uploaded into the warehouse for centralized data mining. Obviously, this centralized approach causes high response times and may be limited by network and security constraints. It does usually not exploit the full capacity of all distributed resources since the data mining step resides on the central sever while the computational power of the clients is idle. In addition, in terms of privacy issues, it may be inappropriate to send the entire data over a public network to a global server. Thus, the traditional, centralized data mining model is not suitable for most distributed environments. Thus, a scalable and privacy preserving solution for distributed data mining applications requires the distributed processing of data [10]. A good distributed data mining framework performs data mining operations based on the type and the availability of the distributed resources. As suggested in [10], a distributed data mining solution consists of the following steps. First, a data mining algorithm is locally applied to each of the k sites separately and independently. The results are k local sets of patterns called *local models*. Second, the local models are transferred to a central server. The central server combines the local models to generate a *global model*. Third, the global model may optionally be sent back to local sites.

The data mining technique we focus in this paper is clustering which aims at partitioning the data objects into distinct groups (clusters) while maximizing the intra-cluster similarity and minimizing the inter-cluster similarity. Many clustering algorithms for the centralized approach have been proposed so far using different clustering notions, e.g. distribution-(or model-)based, center-based, or density-based (cf. [7] for an overview). In general, all those methods are applicable for a distributed solution as far as they produce a local model in Step 1 of the distributed data mining process that is as compact as possible but provides as much information as needed for building a global model in Step 2. Unfortunately, many traditional clustering algorithms produce a clustering that cannot be easily described by a simple prototype. For example, density-based clustering [3] detects clusters of arbitrary shape. However, describing a cluster having a complex shape might become quite expensive possibly causing large transfer rates. Thus,

a local model should describe each cluster by a "suitable" prototype. Obviously, such prototyping should also meet privacy constraints. We argue, that the EM clustering algorithm provides exactly such prototypes. EM describes the data set by a set of Gaussian distributions consisting of the cluster center (mean) and covariance matrix. The latter describes the density of points around the center of the cluster. If certain constraints are met, privacy is preserved because the exact values of the data objects cannot be retrieved from the distribution.

We propose a novel distributed clustering algorithm called DMBC (Distributed Model-Based Clustering). The local models are acquired using EM clustering. Since the necessary number of clusters on each site might be strongly varying, DMBC automatically determines a suitable number of local clusters based on three privacy and performance constraints. The constraints control the maximum transfer volume that is allowed from an individual site and assure that each local data object is described as good as possible and prohibit the transfer of clusters that could leas to a violation of privacy aspects. To combine the local clusters at the central server, the aggregation step of DMBC can employ two variants of parametrization to either derive a global clustering offering k clusters or an arbitrary set of clusters that are considerably different from each other. In both cases, DMBC derives a meaningful global mixture model of Gaussian in efficient time. Our broad experimental evaluation shows that DMBC is a scalable solution for clustering in a distributed environment that achieves comparative results compared to a centralized EM-based approach.

The rest of the paper is organized as follows. Section 2 discusses related work on distributed clustering and introduces the main concepts of EM clustering. In Section 3 we describe our novel DMBC algorithm in more detail. Section 4 provides an extensive experimental evaluation of the performance and the accuracy of DMBC. Section 5 concludes the paper.

2 Related Work

2.1 Distributed and Parallel Clustering

In the following, we will review recent work on parallel and distributed clustering. Parallel clustering is related to the problem of distributed clustering because the data objects are also distributed over several clients where a local clustering is performed. The local clusterings are merged to produce the final model. However, parallel clustering methods can control the assignment of data objects to each site. Thus, the merge step is usually less complex and implying different problems than the merge step of distributed data mining approaches. However, several recent approaches for distributed data mining are adoptions of parallel clustering algorithms.

Parallel versions of k-means, k-Harmonic-Means, EM [2, 5], and DBSCAN [12] are all not applicable within distributed environments because all methods rely on a centralized view of the data during or before the clustering is computed.

In [11] a parallel algorithm is proposed for clustering web documents distributed randomly over several sites. Any clustering algorithm can be used to generate local clusters. The entire local clusters are sent back to the server rather than compact prototypes. Clusters are merged if they share a given number of documents which is determined by deriving maximum-sized itemsets from the documents. Obviously, since all local documents are transferred to the server, this approach does not consider any privacy issues.

In [8] a distributed version of DBSCAN [3] is presented. The local clusters are represented by special objects that have the best representative power. This representative power is based on two quality measures that take the density-based clustering concepts into account. For each representative, a covering radius and a covering number is aggregated for the global merge step. The performance of the proposed method is heavily dependent on the number of representatives. If it is chosen too small, the accuracy significantly decreases. Otherwise, the runtime increases due to high transfer cost. In addition, since real data objects are sent to the global server, this approach does also not consider any privacy issues.

In [9] a single-link hierarchical clustering algorithm for vertically distributed data is proposed. However, our new approach DMBC is focused on horizontally distributed data.

2.2 Model-based Clustering

Clustering algorithms can also be used to obtain a compact representation of a data set. An efficient and effective way to represent a local subset of a distributed data set is to use a mixture of different distribution functions. The most prominent algorithm that tries to describe the data by multiple distribution functions is the EM algorithm [1]. In the following, we describe a variant of this algorithm which is used by our DMBC algorithm.

Let $\mathcal{D}$ be a set of d-dimensional points, i.e. $\mathcal{D} \subseteq \mathbb{R}^d$. The general idea of the EM algorithm is to describe the data by a mixture model M of k Gaussian distributions, where k is the only input parameter. Instead of assigning each object to a cluster as it is the case for k-means-based clustering algorithms, EM assigns each object to a cluster according to a weight representing the probability of membership.

Each cluster $C \in M$ is a tuple $C = (\mu_C, \Sigma_C)$, where μ_C is the mean value of all points in C and Σ_C is the $d \times d$

covariance matrix of all points in C. To compute the probability distributions, we need the following concepts.

The probability density of a point $\vec{x} \in \mathcal{D}$ within a Gaussian density distribution $C = (\mu_C, \Sigma_C)$ is computed in the following way:

$$N_{\mu_C,\Sigma_C}(\vec{x}) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_C|}} e^{-\frac{1}{2}(\vec{x}-\mu_C)^{\mathbf{T}}(\Sigma_C)^{-1}(\vec{x}-\mu_C)}.$$

The combined density for k clusters can then be computed by:

$$P(\vec{x}) = \sum_{i=1}^{k} w_{C_i} \cdot N_{\mu_{C_i},\Sigma_{C_i}}(\vec{x}),$$

where w_{C_i} is the fraction of points that belongs to cluster $C_i = (\mu_{C_i}, \Sigma_{C_i})$, i.e. w_{C_i} is the weight of C_i.

Then, the probability that a point $\vec{x} \in \mathcal{D}$ belongs to a cluster C can be computed by the rule of Bayes:

$$P(C|\vec{x}) = w_C \frac{N_{\mu_C,\Sigma_C}(\vec{x})}{P(\vec{x})}.$$

The log-likelihood of a mixture model $M = (C_1, \ldots, C_k)$ of k Gaussian distributions which describes how good the model approximates the actual data set can be computed by:

$$E(M) = \sum_{\vec{x} \in \mathcal{D}} \log\left(P(\vec{x})\right).$$

The higher the value of $E(M)$, the more likely it is that the data set $\mathcal{D}$ corresponds to the mixture M. Thus, the aim of the EM algorithm is to optimize the parameters of M in a way that $E(M)$ is maximized. For that purpose, the algorithm proceeds in four steps:

1. Initialization Since the clusters, i.e. Gaussian distributions $C_1, \ldots, C_k$, are unknown at the beginning, a set of k initial clusters are generated randomly. For that purpose, each point $\vec{x} \in \mathcal{D}$ is randomly assigned to one cluster C_i. An initial model is produced by computing μ_C and Σ_C for each cluster $C \in M$.

2. Expectation Based on the current model, the parameters μ_C and Σ_C can be computed for each cluster $C \in M$ and the log-likelihood $E(M)$ of this mixture model M is obtained.

3. Maximization In this step, $E(M)$ is improved via a recomputation of the parameters for each of the k clusters. Given a mixture model M, the parameters μ_C, Σ_C, and w_C of each cluster $C \in M$ are recomputed. The resulting mixture M' has an equal or higher log-likelihood than M, i.e. $E(M) \leq E(M')$. For improving the mixture, the parameters are recomputed as follows:

$$w_C = \frac{1}{|\mathcal{D}|} \sum_{\vec{x} \in \mathcal{D}} P(C|\vec{x}),$$

$$\mu_C = \frac{\sum_{\vec{x} \in \mathcal{D}} \vec{x} \cdot P(C|\vec{x})}{\sum_{\vec{x} \in \mathcal{D}} P(C|\vec{x})},$$

$$\Sigma_C = \frac{\sum_{\vec{x} \in \mathcal{D}} P(C|\vec{x})(\vec{x} - \mu_C)(\vec{x} - \mu_C)^{\mathbf{T}}}{\sum_{\vec{x} \in \mathcal{D}} P(C|\vec{x})}.$$

4. Iteration Step 2 and 3 are iterated until the log-likelihood of the improved mixture model M' differs from the log-likelihood of the previous mixture M by a smaller value than a user specified threshold ε, i.e. until $|E(M) - E(M')| < \varepsilon$.

The result of the EM algorithm is a set of k d-dimensional Gaussian distributions, each represented by the mean value μ and the covariance matrix Σ and a weight w. The assignment of a point $\vec{x} \in \mathcal{D}$ to a cluster C is given by the probability $P(C|\vec{x})$. We thus can compute how likely a point is assigned to each of the k clusters.

The log-likelihood of the result of the EM algorithm is usually dependent on the initial mixture model, i.e. on the model assumed in step 1, and on the number of clusters k. In [4] a method for producing a good initial mixture is presented which is based on multiple sampling. It is empirically shown that using this method, the EM algorithm achieves accurate clustering results.

3 Distributed Model-based Clustering

In the following, we will refer to the clustering generated by the centralized approach as *centralized clustering* and call the clusters that are part of the global clustering *centralized clusters*.

3.1 Problem Analysis

As discussed above, the centralized solution has several drawbacks which led to the distributed approach where the data is clustered locally at each site. Afterwards only the information about the local clusters are transfered to a central server. The server can now reconstruct the global clustering which should be as similar to the centralized clustering as possible, by combining the local clusters. For this recombination, we can distinguish the following cases:

Case 1: All objects of a global cluster are found on a single local site. In this case, the global cluster can be spotted easily at the local site and should be added to the global clustering on the central site.

Case 2: The objects of a global cluster are spread over several sites. In this case, we have to distinguish:
Case 2(a): All objects are rather well described by some local cluster. In this subcase, a good clustering algorithm should discover which local clusters belong to the same global cluster and merge them at the central site.
Case 2(b): Some of the objects of a global cluster are locally considered as noise or as members of local clusters that are built from object predominantly belonging to other global clusters. These objects contribute to the wrong global cluster or noise.
Case 2(c): A cluster is distributed over several sites and none of them contains enough data objects for deriving a local cluster. In this case, the objects of the global cluster are considered as noise or parts of other clusters at each client site. Only by combining the local objects the global cluster would become visible.

Because of the last two subcases, it is quite difficult, if not impossible, to develop an efficient distributed clustering algorithm that transmits local clusters and exactly rebuilds the centralized clustering for all cases. Since some of the centralized clusters may only be discovered if the noise objects of several sites are combined, the clustering algorithm would have to transmit all objects that are not well described by any local cluster as well as the clusters. However, transmitting all these objects has two major drawbacks. First, privacy preservation gets almost impossible by transmitting single data objects. Second, with large amounts of noise, it becomes necessary to transmit large amounts of data as well and thus, the advantages of distributed clustering might get lost. However, since there is no other solution for this dilemma, a good distributed clustering algorithm should at least offer the possibility to adjust to the users preferences on privacy, performance and the degree of how good the derived distributed clustering corresponds to the centralized clustering. In the following, we will refer to the degree of how good a distributed clustering corresponds to the centralized clustering as the *agreement* of both clusterings. Note, that a good distributed clustering algorithm cannot guarantee an agreement of 100% in all scenarios as discussed above.

In the following, we describe a method, called Distributed Model-based Clustering (DMBC) that is based on the EM clustering of local sites. Instead of transmitting the complete local data set, we only transmit a number of local Gaussians and their weights to the central site. Since a Gaussian distribution is represented only by a mean vector and a covariance matrix, the amount of transferred information is much smaller. Therefore, the needed bandwidth is much smaller. Furthermore, the Gaussians derived by EM are always built according to all underlying data objects and drawing detailed conclusions about individual data objects is not possible in almost all settings. Since we will additionally control the remaining cases, the privacy of local data objects is preserved. Thus, our method avoids the problem of building a global clustering and still derives a mixture of Gaussian distributions achieving a high agreement with the centralized clustering.

The DMBC algorithm proceeds in 4 steps:
Step 1: The local data on each client site is clustered using EM. Thus, the data at each client site is now represented by a small but descriptive set of Gaussian distributions and a distribution of weights over these Gaussians. The number of clusters for the EM algorithm is optimized automatically with respect to the parametrization controlling the privacy level and the transfer volume.
Step 2: The local Gaussians and weights are transmitted to the central site.
Step 3: Similar local Gaussians are joined to find a compact global distribution. Thus, each cluster in a global EM clustering is only represented by a single Gaussian.
Step 4: The calculated global clustering can optionally be transmitted back to the client sites.

3.2 Computation of Local Models

To cluster the data at each client site, we employ the EM algorithm as described in section 2.2. The most important aspect of this step is the question how to choose the parameter k, i.e. the number of Gaussians that is used to describe the local data distribution. Since the data distribution can strongly vary between each site, simply selecting a global value for k as the expected number of global clusters might be rather inappropriate. Therefore, our algorithm automatically determines a particular value k_i for each site S_i. Let us note that k_i not only influences the transfer volume, but also the privacy and exactness of the clustering as well. The larger k_i, the higher is the probability that a Gaussian is strongly influenced by only a few data sets. In this case, the privacy can be seriously jeopardized because it might be easy to approximate the instances that are represented by this Gaussian. On the other hand, a large number of k_i usually increases the tendency that all local data objects are well represented by a local Gaussian. To conclude, a very high value for k_i will increase the agreement between our derived clustering and the global clustering, but it will also increase the transferred data volume.

To find a clustering containing an appropriate number of clusters, we first of all introduce the parameter k_{max} describing the maximum number of Gaussians for each local site. k_{max} limits the maximum transfer from a local site to the central site in step 2 and thus can be derived from the available bandwidth. To measure the degree that all data objects are well represented by the given clustering, we introduce the function $cover(C_{i,j})$.

Definition 1 (Cover) *Let $M = C_1, \ldots, C_k$ be a mixture*

```
localEM(Database 𝒟, Integer k_max)
  maxcover = 0;
  bestClustering = ∅;
  for k := 1 to k_max do
    M := EM(𝒟, k);
    if cover(M) = |𝒟| then
      return M;
    end if
    if cover(M) > maxcover then
      maxcover = cover;
      bestClustering = M;
    end if
  end for
  return bestClustering;
```

Figure 1. Algorithm for local clustering.

of Gaussians describing the density distribution within $\mathcal{D}$. Furthermore, let $t \in [0, \ldots, 1]$ be a probability threshold. Then, the cover *of the model M, denoted by $Cov(M)$, is defined as follows:*

$$Cov(M) = |\{ \vec{x} \,|\vec{x} \in \mathcal{D} \cap \exists\, C_i \in M \; : P(C_i|\vec{x}) \geq t\}|$$

Intuitively, the cover is the number of data objects that provide at least a probability of t for some Gaussian in the clustering. Let us note that $Cov(M)$ is related but different to the log-likelihood $E(M)$ that is optimized by the EM algorithm.

The pseudo code of the algorithm localEM to derive a local clustering is depicted in Figure 1. The algorithm chooses the smallest clustering M that achieves a maximum cover by successively increasing k and testing the cover of the resulting clustering.

At last, we have to control the level of privacy, we need to ensure. Thus, we have to measure how far it is possible to draw conclusions about individual data sets from the found clustering C. Therefore, we define the so-called privacy score (PScore):

Definition 2 (Privacy Score) *Let $C_i \in M$ be a cluster that is described by a d-dimensional Gaussian determined by the variance vector μ_i and a covariance matrix Σ_i. Then, the* privacy-score*, denoted by $PScore(C_i)$, is defined as follows:*

$$PScore(C_i) = \sum_{j=1}^{d} \Sigma_{j,j}$$

The idea of the $PScore(C_i)$ is quite simple. Only if the variance in each dimension is very small, it is possible to draw conclusions about the underlying feature vectors. Let us note that the local cluster description of a cluster determined by the EM algorithm is always built using the complete data set. As a result, it is impossible to derive detailed information about single feature vectors even for clusters having small weights if the variance values are large enough for at least a single dimension. Thus, we define a privacy threshold τ_p that is the lower limit for the $PScore(C_i)$ of a cluster C_i that is allowed to be transferred. If a cluster C_i has a smaller privacy-score, i.e. $PScore(C_i) < \tau_p$, we do not transmit the cluster because it would be possible to conclude that there is at least one feature vector stored on the local site that strongly resembles the transferred mean value.

To conclude, at each site we determine the smallest EM clustering providing a maximum cover and afterwards transfer all clusters that do not violate the predefined level of privacy.

3.3 Computation of the Global Model

The purpose of this step is to combine the locally derived clusters to a distributed clustering describing the complete data distribution in a best possible way. The difficulty in this step is to find out which of the clusters are likely to describe the same global cluster. To find out which of the given local clusters should be joined, first of all we need a measure that describes the likelihood of two local Gaussians C_1 and C_2 to model the same global cluster. Simply, using the distance between mean vectors is not applicable here because the significance of this distance strongly decreases with increasing variance values. Therefore, we define a new measure that considers the dependency between variance and mean value, called mutual support.

Definition 3 (Mutual Support) *Let C_1, C_2 be two Gaussian determined by a mean vector μ_i and a covariance matrix Σ_i. Then the* mutual support *of C_1, C_2 is given by:*

$$MS(C_1, C_2) = \int_{-\infty}^{+\infty} N_{\mu_1,\Sigma_1}(\vec{x}) \cdot N_{\mu_2,\Sigma_2}(\vec{x}) d\vec{x}$$

Let us note that $\vec{x}$ is a d-dimensional feature vector and thus $MS(\vec{x})$ is defined using the integral over all d dimensions. The mutual support has several characteristics that makes it well suited for measuring the similarity between two Gaussians. The larger the variance values become, the less steep are the probability density functions of the Gaussians and the less important is the distance between the mean values. Comparing a low variance distribution with a high variance Gaussian, will display a small mutual support. In the comparably small range where a low-variance distribution displays strong density the high-variance distribution provides only moderate density and in the large area where the high-variance distribution displays still moderate density, the density of the low-variance distribution decreases the product very strongly. Thus, the mutual support of two

```
globalMerge(SetOfLocalClusters C, Integer k)
  for each pair (C_i, C_j) ∈ C do
    compute MS(C_i, C_j);
  end for
  sort the pairs w.r.t. descending mutual support;
  mark the first |C| − k pairs of clusters;
  build the transitive closure over the pairs having some
    common clusters and unite them into a common
    global cluster;
```

Figure 2. Algorithm for global clustering.

Gaussians specified by very similar mean values but quite different covariance matrices is also rather small.

After finding a method to compare two local Gaussians, we now start to determine which of the local clusters should be merged. To determine a distributed clustering from a set of local clusters C with a number of k global clusters, we can now proceed as described in Figure 2.

Another alternative for deriving a joined distributed clustering is to specify a threshold parameter τ and join all clusters displaying a mutual support of at least τ. In this case, all pairs of clusters (C_i, C_j) are marked if $MS(C_i, C_j) \geq \tau$. Again we first of all, find the marked pairs of clusters that are connected by common clusters and afterwards merge all clusters in this connected set. Therefore, both approaches are independent of the order the clusters are merged. After determining which clusters have to be joined, we still need to derive a common Gaussian from a connected set of local clusters. Therefore, we derive a new mean μ_C for a set of Gaussian clusters $C = \{C_1, \ldots, C_m\}$ that are specified by μ_i and Σ_i in the following way.

$$\mu_C = \frac{\sum_{k=1}^{m}(w_{C_k} \cdot \lambda(C_k) \cdot \mu_k)}{\sum_{k=1}^{m}(w_{C_k} \cdot \lambda(C_k))}.$$

Here, $\lambda(C_k) = Cov(M_l)$ where cluster C_k has its origin on site l, i.e. $\lambda(C_k)$ denotes the cover of site l. The entries of the covariance matrix for the ith line and the jth column are calculated as following:

$$\Sigma_C^{i,j} = \frac{\int_{-\infty}^{+\infty}(\sum_{k=1}^{m} w_{C_k}\lambda(C_k)N_{\mu_k,\Sigma_k}(\vec{x}) \cdot (\vec{x}^i - \mu_C^i)(\vec{x}^j - \mu_C^j))d\vec{x}}{\sum_{k=1}^{m}(\int_{-\infty}^{+\infty} w_{C_k}\lambda(C_k)N_{\mu_k,\Sigma_k}(\vec{x}))d\vec{x})}$$

Let us note that we again need to employ a multiple integral to calculate the new covariance matrix because we do not have the actual data distribution at each site. Therefore, we assume that the local density given by each local clustering is a well enough description of this distribution. To consider the number of data objects that are stored at each site, we additionally weight the influence of each distribution with the cover we transmitted from this site.

The weight of C can be determined as

$$w_C = \frac{\sum_{C_i \in C} w_{C_i} \cdot \lambda(C_i)}{\sum_{C_i \in C} \lambda(C_i)}.$$

3.4 Scaling to High Dimensional Data Sets

If we apply DMBC as proposed in the previous subsection on higher dimensional data sets ($d > 2$), we face the problem that, in order to compute both the mutual support as well as the covariance matrix of a merged cluster, we have to evaluate multiple integrals.

Thus, in order to be scalable for higher dimensional data sets, we propose a variant of DMBC that uses variances instead of covariances for cluster representation. In particular, we assume the attributes to be indepent of each other and represent a cluster C by its mean vector μ_C and its d-dimensional variance vector σ_C. The i-th value of σ_C, denoted by σ_C^i, indicates the variance of the Gaussian along attribute i.

As a consequence, the resulting Gaussians form ellipsoid-shaped clusters that are constrainted to be axis-parallel. We will see later in the experimental evaluation, that this simplification does not cause a significant loss of quality. However, the benefits of this modification are the following. First, we are able to solve the integral of the mutual support analytically. Second, to compute the variance vector of a merged cluster, we need to solve only one integral rather than multiple integrals. Third, the transfer cost for each local cluster are reduced from $O(d^2)$ for the covariance matrix to $O(d)$ for the variance vector.

In fact, the mutual support of a pair of clusters $C = \{C_1, C_2\}$ can be computed as

$$MS(C_1, C_2) = \prod_{i=1}^{d} \int_{-\infty}^{+\infty} N_{\mu_1^i,\sigma_1^i}(\vec{x}) \cdot N_{\mu_2^i,\sigma_2^i}(\vec{x})d\vec{x}$$

The following lemma enables us to solve this integral over d-dimensions analytically.

Lemma 1 *Let $C_1 = (\mu_{C_1}, \sigma_{C_1})$ and $C_2 = (\mu_{C_2}, \sigma_{C_2})$ be local clusters. Then*

$$MS(C_1, C_2) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi} \cdot (\sigma_1^i + \sigma_2^i)} \cdot \exp - \frac{(\mu_1^i - \mu_2^i)^2}{2 \cdot (\sigma_1^i - \sigma_2^i)}$$

Proof. *Let $N_{\mu_1^i,\sigma_1^i}(\vec{x}) \cdot N_{\mu_2^i,\sigma_2^i} = \vartheta_i \cdot N_{\mu,\sigma}$. If we replace μ and σ by:*

$$\mu = \frac{\mu_1^i \cdot \sigma_2^i + \mu_2^i \cdot \sigma_1^i}{\sigma_2^i + \sigma_1^i} \quad \textit{and} \quad \sigma = \frac{\sigma_2^i \cdot \sigma_1^i}{\sigma_2^i + \sigma_1^i}$$

and apply the logarithm to the equation, it follows that

$$\vartheta_i = \frac{1}{\sqrt{2\pi} \cdot (\sigma_1^i + \sigma_2^i)} \cdot \exp - \frac{(\mu_1^i - \mu_2^i)^2}{2 \cdot (\sigma_1^i - \sigma_2^i)}$$

Thus, we obtain

$$MS(C_1, C_2) = \prod_{i=1}^{d} \int_{-\infty}^{+\infty} N_{\mu_1^i, \sigma_1^i}(\vec{x}) \cdot N_{\mu_2^i, \sigma_2^i}(\vec{x}) d\vec{x} =$$

$$\prod_{i=1}^{d} \vartheta_i \cdot \int_{-\infty}^{+\infty} N_{\mu^i, \sigma^i}(\vec{x}) d\vec{x} = \prod_{i=1}^{d} \vartheta_i \cdot 1$$

□

The j-th component of the variance vector of the global cluster C which evolved from the merge of m clusters C_i is given as:

$$\sigma_C^j = \sqrt{\frac{\int_{-\infty}^{+\infty} (\sum_{i=1}^{m} w_{C_i} \lambda(C_i) N_{\mu_i^j, \sigma_i^j}(\vec{x^j})) \cdot (\vec{x^j} - \mu_C^j)^2 d\vec{x^j}}{\sum_{i=1}^{m} (\int_{-\infty}^{+\infty} w_{C_i} \lambda(C_i) N_{\mu_i^j, \sigma_i^j}(\vec{x^j})) d\vec{x^j}}}$$

Note that for component σ_C^j we compute only a 1-dimensional integral over dimension j.

4 Experimental Evaluation

We implemented our versions of DMBC in Java and run several tests on a workstation featuring two 1.8 GHz Opteron processors and 8 GByte main memory.

The test bed consists of one artificial 2-dimensional data set (denoted as DS1) and two real word data sets (denoted as DS2 and DS3). The latter two are derived from 68,040 images of the corel image feature collection of the UCI KDD archive[1]. DS2 contains 9-dimensional color moments of images in HSV color space (mean, standard deviation and skewness). DS3 comprises a description of the corel images based on co-occurrence textures with 16 dimensions.

The experimal results of DMBC on the synthetic DS1 (Figure 3) demonstrates that our algorithm is capable to handle cases 1, 2(a)-(c) described in Section 3.1: DMBC finds the global cluster "A" the objects of which are existent on only one single client, i.e. client 2 (Case 1). DMBC finds the global clusters "B", "D", and "E" the objects of which are rather well described by local clusters on all sites (Case 2(a)). DMBC finds the global cluster "C" the objects of which are distrubuted over all sites such that none of the sites exhibit a local cluster (Case 2(c)). Several objects of clusters "D" and "E" are prototypes for Case 2(b) because they are members of local clusters that are built from objects predominantly belonging to another global cluster.

In order to demonstrate the robustness of our clustering algorithm w.r.t. the number of clusters on each client site, we performed distributed clustering with a predetermined number of clusters. We measured the agreement between the distributed clustering and the results of the centralized EM algorithm using the Rand Index [6], also known as Rand Statistics.

[1] http://kdd.ics.uci.edu/

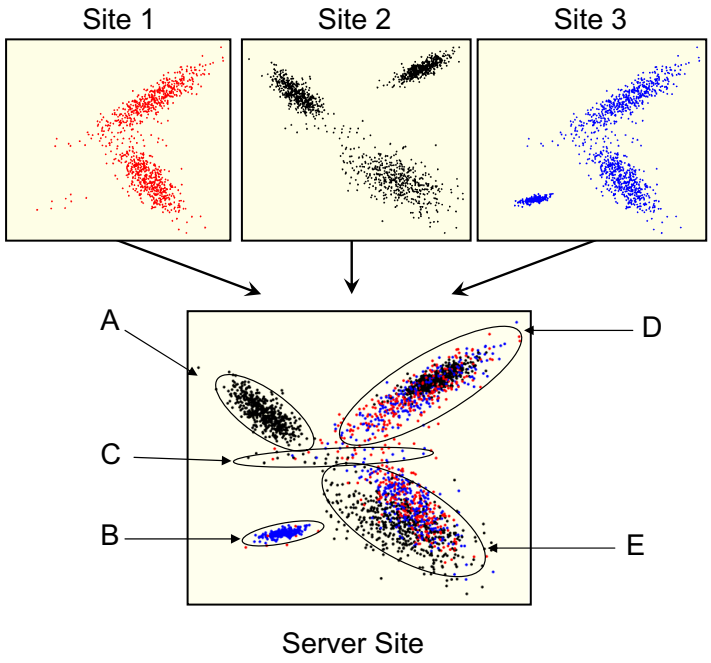

Figure 3. Results of DMBC on DS1.

On the 2-dimensional synthetic data set (DS1) shown in Figure 3 DMBC achieved a Rand index of approximately 99.9%, indicating a high agreement of our method with the centralized clustering. For data DS2 and DS3 we used the variant of DMBC based on variances (cf. Section 3.4). As shown in Figure 4, DMBC achieves high Rand Index values, i.e. our distributed approach produces a high level of agreement with the results of centralized clustering algorithms on all numbers of clusters. This also indicates that the variant proposed in Section 3.4 using variances instead of covariances does produce accurate results, too.

We evaluated the scalability of the proposed algorithm w.r.t. the number of clusters on each client site. The results are depicted in Figure 4(a). As it can be observed, in all settings, the Rand Index is near the optimal value. Thus, the agreement between the centralized clustering and the global distributed clustering is very high.

In addition, we investigated the scalability of the proposed algorithm w.r.t. the number of client sites. Figure 4(b) presents the agreement using the Rand Index between results calculated by our approach and the centralized clustering algorithm. The number of client sites involved in the distributed clustering was varying from 2 to 10. The high value of the Rand Index in all experiment evaluations shows that our algorithm is scalable w.r.t. the number of client sites and delivers results that do not differ from that of the global

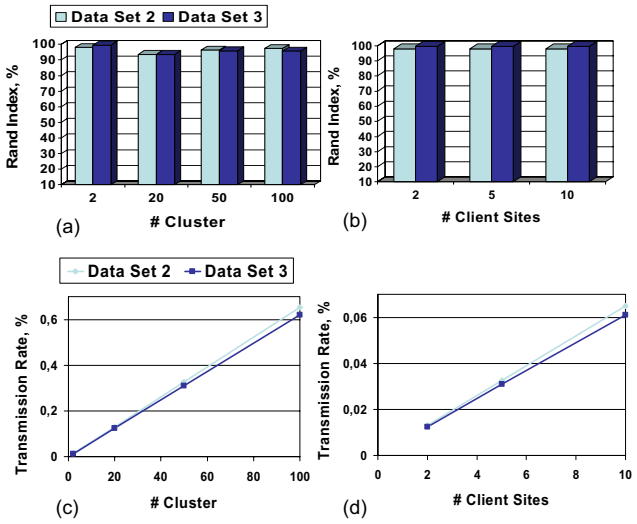

Figure 4. Results of DMBC on DS2 and DS3.

acting algorithm.

We also investigated the transfer cost w.r.t. the number of clusters on each client site and w.r.t. the number of client sites. The results are depicted in Figure 4(c) and 4(d). As transfer cost we measured the ratio of the number of bytes that are transfered using DMBC and of the number of bytes that are transfered using the centralized approach. As it can be seen, the transfer cost is in general very low. Even for a very large number of clusters, DMBC needs less than 1% of the bytes transfered by the centrailized approach. In addition, we can observe, that the transfer cost increases only linearly w.r.t. the number of local clusters and w.r.t. the number of client sites. Compared to other existing distributed clustering approaches, e.g. the density-based distributed approach in [8], where the local transfer cost is at least 15% of the local data in order to achieve a high agreement, our DMBC reduces the transfer cost dramatically.

Last, we investigated the robustness of DMBC w.r.t. the probability threshold t which affects the cover of the local models. As the results (not shown due to space limitations) suggest, DMBC is rather robust w.r.t. a broad range of values for t. In fact, we observed a Rand Index over 98% when varying the values of t from 0.05 up to 0.45.

To sum up, our experiments demonstrated the robustness, the efficiency, and the applicability of both of our proposed variants for distributed model-based clustering.

5 Conclusion

In this paper, we proposed a distributed version for EM clustering called distributed model-based clustering (DMBC). Our method applies the EM algorithm at the local sites generating a model containing a set of Gaussian distributions. Each Gaussian is represented by its mean and its covariance matrix or — for higher dimensions the variance vector. We also proposed a merge step of the local Gaussians that can handle covariances as well as variances. Compared to recent approaches for pure distributed clustering, DMBC enables respecting an arbitrary level of privacy and dramatically reduces the transfer costs. Our experimental evaluation demonstrates the robustness, the efficiency, and the applicability of both of our proposed variants for distributed clustering.

For future work we will examine the use of other distribution functions instead of Gaussian for clustering.

References

[1] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of the Royal Statistical Society, Series B*, 39(1):1–31, 1977.

[2] I. S. Dhillon and D. S. Modha. "A Data-Clustering Algorithm On Distributed Memory Multiprocessors". In *Proc. KDD-WS on High Performance Data Mining*, 1999.

[3] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In *Proc. Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 291–316, 1996.

[4] U. Fayyad, C. Reina, and P. Bradley. "Initialization of Iterative Refinement Clustering Algorithms". In *Proc. Int. Conf. on Knowledge Discovery in Databases (KDD)*, 1998.

[5] G. Forman and B. Zhang. "Distributed Data Clustering can be Efficient and Exact". *SIGKDD Explorations*, 2, 2000.

[6] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. "On Clustering Validation Techniques". *Journal of Intelligent Information Systems*, 2/3(17):107–145, 2001.

[7] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Academic Press, 2001.

[8] E. Januzaj, H.-P. Kriegel, and M. Pfeifle. "Scalable Density-Based Distributed Clustering". In *Proc. Europ. Conf. PKDD*, 2004.

[9] E. Johnson and H. Kargupta. "Hierarchical Clustering from Distributed, Heterogeneous Data". In M. Zaki and C. Ho, editors, *Large-Scale Parallel KDD Systems*, volume 1759 of *Lecture Notes in Computer Science (LNCS)*. Springer Verlag, 1999.

[10] B.-H. Park and H. Kargupta. "Distributed Data Mining: Algorithms, Systems, and Applications". In N. Ye, editor, *The Handbook of Data Mining*. Lawrence Erlbaum Associates Publishers, 2003.

[11] M. Sayal and P. Scheuermann. "A Distributed Algorithm for Web-Based Access Patterns". In *Proc. KDD-WS on Distributed and Parallel Knowledge Discovery*, 2000.

[12] X. Xu, J. Jäger, and H.-P. Kriegel. "A Fast Parallel Clustering Algorithm for Large Spatial Databases". *Data Mining and Knowledge Discovery, an International Journal*, 3(3):263–290, 2003.