In Proc. 9th Pacific-Asia Conference (PAKDD 2005), Hanoi, Vietnam, May 18-20, 432-441, 2005.

Approximated Clustering of Distributed High-Dimensional Data

Hans-Peter Kriegel, Peter Kunath, Martin Pfeifle, Matthias Renz
University of Munich, Germany
{kriegel, kunath, pfeifle, renz}@dbs.ifi.lmu.de

Abstract. In many modern application ranges high-dimensional feature vectors are used to model complex real-world objects. Often these objects reside on different local sites. In this paper, we present a general approach for extracting knowledge out of distributed data sets without transmitting all data from the local clients to a server site. In order to keep the transmission cost low, we first determine suitable local feature vector approximations which are sent to the server. Thereby, we approximate each feature vector as precisely as possible with a specified number of bytes. In order to extract knowledge out of these approximations, we introduce a suitable distance function between the feature vector approximations. In a detailed experimental evaluation, we demonstrate the benefits of our new feature vector approximation technique for the important area of distributed clustering. Thereby, we show that the combination of standard clustering algorithms and our feature vector approximation technique outperform specialized approaches for distributed clustering when using high-dimensional feature vectors.

1 Introduction

One of the primary data mining tasks is clustering. Clustering aims at partitioning the data set into distinct groups, called clusters, while maximizing the intra-cluster similarity and minimizing the inter-cluster similarity [8]. Traditionally, the clustering algorithms require full access to the data which is going to be analyzed. All data has to be located at the site where it is processed. Nowadays, large amounts of heterogeneous, complex data reside on different, independently working computers which are connected to each other via local or wide area networks. Examples comprise distributed mobile networks, sensor networks or supermarket chains where check-out scanners, located at different stores, gather data unremittingly. Furthermore, international companies such as DaimlerChrysler have some data which are located in Europe and some data located in the US and Asia. Those companies have various reasons why the data cannot be transmitted to a central site, e.g. limited bandwidth or security aspects.

Many of these real-world distributed data sets consist of objects modeled by high-dimensional feature vectors. For instance, a starting point for applying clustering algorithms to distributed unstructured document collections is to create a vector space model, alternatively known as a bag-of-words model [13], where each document is represented by a high-dimensional feature vector. Other examples for high-dimensional feature vectors representing distributed complex objects can be found in the area of image retrieval [12], and molecular biology [4].

The requirement to extract knowledge from distributed data, without a prior unification of the data, created the rather new research area of Distributed Knowledge Discov-

ery in Databases (DKDD). In this paper, we present a general approach which helps to extract knowledge out of high-dimensional feature vectors spread over several sites. To get specific, we demonstrate the benefits of our approach for distributed density-based clustering. Our approach tries to describe local feature vectors as accurately as possible with a certain number of granted bytes. These approximations are sent to a server site, where the global server clustering is carried out based on a suitable distance function measuring the similarity between the locally determined feature vector approximations.

The remainder of the paper is organized as follows: In Section 2, we present the related work in the area of distributed clustering. In Section 3, we explain how to form the local approximations which are sent to a central server site. Then, in Section 4, a meaningful similarity measure for the feature vector approximation is introduced. In Section 5, we demonstrate the suitability of our feature vector approximation technique and the corresponding distance function. In Section 6, we close this paper with a short summary and a few remarks on future work.

2 Related Work

Distributed Data Mining (DDM) is a dynamically growing area within the broader field of KDD. Generally, many algorithms for distributed data mining are based on algorithms which were originally developed for parallel data mining. In [10], some state-of-the-art research results related to DDM are summarized. Whereas there already exist algorithms for distributed classification and association rules, there is a lack of algorithms for distributed clustering.

In [5] the "collective hierarchical clustering algorithm" for vertically distributed data sets was proposed which applies single link clustering. In contrast to this approach, we concentrate in this paper on horizontally distributed data sets.

In [14] the authors presented a technique for centroid-based hierarchical clustering for high-dimensional, horizontally distributed data sets by merging clustering hierarchies generated locally. Unfortunately, this approach can only be applied for distance-based hierarchical distributed clustering approaches, whereas our aim is to introduce a generally applicable approach.

In [6, 7], density-based distributed clustering algorithms were presented which are based on the density-based partitioning clustering algorithm DBSCAN. The idea of these approaches is to determine suitable local objects representing several other local objects. Based on these representatives a global DBSCAN algorithm is carried out. These approaches are tailor-made for the density-based distributed clustering algorithm DBSCAN.

The goal of this paper is to introduce an approach which is generally applicable to DDM. To get specific, we demonstrate the benefits of our approach for distributed clustering algorithms. In contrast to the above specific distributed clustering approaches, our approach is not susceptible to an increasing number of local clients. It does only depend on the overall allowed transmission cost, i.e. on the number of bytes we are allowed to transmit from the local clients to a server. In order to keep these transmission cost low, we introduce in the following section a suitable client-side approximation technique for describing high-dimensional feature vectors.

3. Client-side Approximation

The hybrid-approximation approach which we propose in this section is quite similar to the idea of the IQ-tree [2] which is an index structure especially suitable for managing high-dimensional feature vectors. First, we divide the data set into a set of partitions represented by minimum bounding rectangles (MBRs) of the points located in the corresponding region in the data space. This kind of data set approximation is further elaborated in Section 3.1. In Section 3.2, we describe each single feature vector by an approximation hierarchy where in each level of the hierarchy K more bits are used to describe the feature vector more accurately.

3.1 Data Set Approximation

The goal of this section is to find a rough description of the complete data set by means of some (flat) directory pages which conservatively approximate the complete data space. The problem of finding these MBRs is related to clustering. We are not mainly interested in the clusters themselves but rather in a partitioning of the data space into rectangular cuboids. Similar, to directory pages in an index structure, these cuboids should be formed as quadratic as possible for efficient query processing [3]. We can achieve such cuboids with only a little variation of the lengths of the edges by applying the *k*-means clustering algorithm [11]. Thereby the data set is approximated by k centroids, and each vector is assigned to its closest centroid. All feature vectors which are assigned to the same centroid form a cluster and are approximated by an MBR of all the vectors of this cluster. As desired, the form of these MBRs tend to be quadratic as the centroid of a cluster tends to be close to the middle of the MBR. Thus, the *k*-means clustering algorithm indirectly also minimizes the average length of the space diagonals of the k MBRs.

3.2 Feature Vector Approximation

After having partitioned the local data space into k clusters represented by MBRs, we express each feature vector v w.r.t. to the lower left corner of its corresponding minimum bounding rectangle $\mathrm{MBR}_{\mathrm{Cluster}}(v)$.

Definition 1 *Feature Vector*
Each feature v_i of a d-dimensional feature vector $v = (v_1,..., v_d)^t \in IR^d$ is represented by a sequence of bytes $<b_{i,1},.., b_{i,m}>$ where each byte consists of w bits. The feature value v_i is calculated by

$$v_i = \sum_{j=1}^{m} val(b_{i,j}) \text{ , where } val(b_{i,j}) = b_{i,j} \cdot 2^{w(m-j)}$$

For clarity, we assume in this paper that each feature of a d-dimensional feature vector is represented by a byte string of length m. We will describe each feature vector by a conservative hierarchy of approximations where in each level we use some more bytes to approximate the feature vector more closely. By traversing the complete approximation hierarchy, we can reconstruct the correct feature vector.

The client first computes a byte ranking of all the bytes $b_{i,j}$ of v. Then the most meaningful bytes are transmitted to the server along with positional information of the bytes. By means of this additional positional information, the server can construct an accurate server side approximation of v.

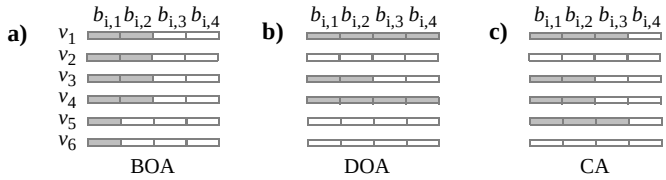

Figure 1. Approximation techniques. (L = 10 Bytes)

Definition 2 *Ranking and Approximation Function*
Let W be the set of all byte sequences of length $m \cdot d$. Let $v = (v_1, ..., v_d)^t \in IR^d$ be a feature vector where each feature v_i is represented by a sequence of bytes $<b_{i,1},.., b_{i,m}>$. Then, we require a byte *ranking function* f_{rank}: $IR^d \rightarrow W$ and a feature vector *approximation function* f_{app}: $W \times \{0 \ldots m \cdot d\} \rightarrow [IR \times IR]^d$ to have the following properties:

- $f_{rank}(v) = < b_1, ..., b_{m \cdot d} >$ where $b_l = b_{\pi(i,j)}$, for a bijective ranking function π_{rank}: $\{1 \ldots d\} \times \{1 \ldots m\} \rightarrow \{1 \ldots m \cdot d\}$
- $f_{app}(f_{rank}(v), 0) = \mathrm{MBR}_{\mathrm{Cluster}}(v)$, $f_{app}(f_{rank}(v), L_1) \subseteq f_{app}(f_{rank}(v), L_2)$ iff $L_1 \geq L_2$, and $f_{app}(f_{rank}(v), m \cdot d)) = v$

After having received a certain number of L bytes the server can compute the approximation area $A = f_{app}(f_{rank}(v), L)$. In the following subsections, we present three approximation techniques of high-dimensional feature vectors, i.e. the *byte-oriented*, the *dimension-oriented*, and the *combined* approximation technique (cf. Figure 1). All three approaches fulfill rather obviously the properties stated in Definition 2. Nevertheless, they differ in the way they actually define the ranking and approximation functions. In the following, we assume that the cluster MBR of a feature vector v $\mathrm{MBR}_{\mathrm{Cluster}}(v) = [MBR\_l_1 \times MBR\_u_1] \times ... \times [MBR\_l_d \times MBR\_u_d]$ has already been transmitted to the server. Furthermore, we assume that v is defined according to Definition 1.

3.2.1 Byte-Oriented Approximation (BOA)

As the first bytes of each feature contain the most significant information, we rank the bytes $b_{i,j}$ by means of the bijective function $\pi: \{1 \ldots d\} \times \{1 \ldots m\} \rightarrow \{1 \ldots m \cdot d\}$ according to their j-positions, i.e. $\pi(i,j) < \pi(i',j')$ iff ($j < j'$) or ($j = j'$ and $i < i'$).

The server computes the approximation area $a = f_{app}(f_{rank}(v), L) = [l_1,u_1] \times ... \times [l_d,u_d]$ as follows:

$$l_i = MBR\_l_i + \sum_{j=1}^{m} \begin{cases} val(b_{i,j}) & \text{, if } (\pi(i,j) \leq L) \\ 0 & \text{, else} \end{cases}$$

$$u_i = min\left(MBR\_u_i, MBR\_l_i + \sum_{j=1}^{m} \begin{cases} val(b_{i,j}) & \text{, if } (\pi(i,j) \leq L) \\ (2^w - 1) \cdot 2^{w(m-j)} & \text{, else} \end{cases} \right)$$

3.2.2 Dimension-Oriented Approximation (DOA)

In the above approach, we considered the first bytes of each dimension to build the approximation. In this approach, we select significant dimensions and then transmit all bytes of the selected features to the server. The dimension oriented approximation approach (cf. Figure 1b) selects $\lceil L/m \rceil$ dimensions i having the highest values v_i. Thus, we

rank the bytes $b_{i,j}$ by means of the bijective function $\pi: \{1 \ldots d\} \times \{1 \ldots m\} \rightarrow \{1 \ldots m \cdot d\}$ as follows: $\pi(i,j) < \pi(i',j')$ iff ($v_i > v_i'$) or ($v_i = v_i'$ and $i < i'$) or ($v_i = v_i'$ and $i = i'$ and $j < j'$).

The big advantage of this technique is that it implies an upper bound for those dimensions which have not been selected for transmission. Thus, we can shrink the approximation area also for those dimensions for which we have not received any bytes. This shrinking is possible due to the ranking agreement between the clients and the server that the value of the dimensions for which we have not received any bytes is equal or smaller to the smallest value for which we have already received some bytes. Let $i' \in \{1,.., d\}$ now be the transmitted dimension with the smallest feature value of all transmitted dimensions.

Then, the server computes the approximation area $a = f_{app}(f_{rank}(v), L) = [l_1,u_1] \times ... \times [l_d,u_d]$ as follows:

$$l_i = MBR\_l_i + \sum_{j=1}^{m} \begin{cases} val(b_{i,j}) & \text{, if } (\pi(i,j) \le L) \\ 0 & \text{, else} \end{cases}$$

$$u_i = min\left(MBR\_u_i, MBR\_l_i + \sum_{j=1}^{m} \begin{cases} val(b_{i,j}) & \text{, if } (\pi(i,j) \le L) \\ val(b_{i',j}) & \text{, if } (\pi(i,j) > L) \wedge (\pi(i',j) \le L) \\ (2^w - 1) \cdot 2^{w(m-j)} & \text{, else} \end{cases} \right)$$

3.2.3 Combined Approximation (CA)

This approach combines the two previous approaches. According to Definition 1, each byte $b_{i,j}$ of v can be assigned to a value $val(b_{i,j}) = b_{i,j} \cdot 2^{w(m-j)}$. Now, we can rank the set of bytes $\{b_{i,j}: i = 1, .., d; j = 1, .., m\}$ according to their value $val(b_{i,j})$, and transmit the L bytes having the highest ranking values. Thus the bijective function $\pi: \{1 \ldots d\} \times \{1 \ldots m\} \rightarrow \{1 \ldots m \cdot d\}$ is defined as follows: $\pi(i,j) < \pi(i',j')$ iff ($val(b_{i,j}) > val(b_{i',j'})$) or ($val(b_{i,j}) = val(b_{i',j'})$ and $i < i'$) or ($val(b_{i,j}) = val(b_{i',j'})$ and $i = i'$ and $j < j'$).

Let now $b_{i',j'}$ be the byte with the L highest $val(b_{i',j'})$. Then, the server computes the approximation area $a = f_{app}(f_{rank}(v), L) = [l_1,u_1] \times ... \times [l_d,u_d]$ as follows:

$$l_i = MBR\_l_i + \sum_{j=1}^{m} \begin{cases} val(b_{i,j}) & \text{, if } (\pi(i,j) \le L) \\ 0 & \text{, else} \end{cases}$$

$$u_i = min(MBR\_u_i, MBR\_l_i + offset), \text{ where}$$

$$offset = \sum_{j=1}^{m} \begin{cases} val(b_{i,j}) & \text{, if } (\pi(i,j) \le L) \\ 0 & \text{, if } (\pi(i,j) > L) \wedge (j < j') \\ val(b_{i',j'}) & \text{, if } (\pi(i,j) > L) \wedge (j = j') \\ (2^w - 1) \cdot 2^{w(m-j)} & \text{, if } (\pi(i,j) > L) \wedge (j > j') \end{cases}$$

The example presented in Figure 2 demonstrates the conservative approximation areas for the three proposed approaches. The figure shows clearly that the combined approach leads to a smaller approximation area than the byte-oriented and the dimension-oriented approach.

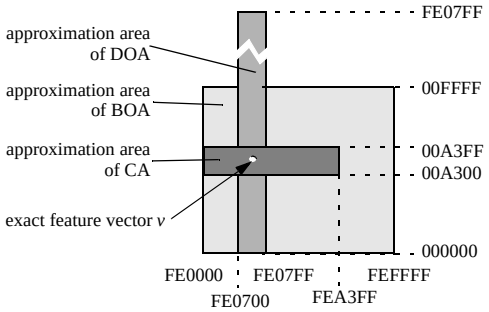

Figure 2. Approximation areas (BOA, DOA, CA).

4 Approximated Clustering based on Fuzzy Distance Functions

In this section, we will show how to compute the similarity between two feature vector approximations. The most straightforward approach is to use the box center to compute the distance between two box approximations. This center oriented box distance approximates the exact distance between the feature vectors rather accurate if the boxes are rather small and do not overlap.

On the other hand, imagine that we have two rather big boxes where the box centers are identical. The center oriented distance would assign a zero distance to the approximated feature vectors, although the exact distance between the feature vectors might be very high. Therefore, it is better to generally use the expectation value of the exact distances between the feature vectors rather than the distances between the box centers. This distance expectation value is based on the distance distribution function P_d: $O \times O \rightarrow (IR_0^+ \rightarrow [0..1])$, which assigns a probability value p to each possible distance τ (cf. Figure 3a). The value p indicates the probability that the exact distance between the feature vectors is smaller than τ. Figure 3b shows how we can compute P_d for two feature vectors based on two arbitrary conservative approximations $A=f_{app}(f_{rank}(v),L)$ and $A'=f_{app}(f_{rank}(v'),L')$. First, we measure those portions of the area A' which are overlapped by a sphere around $x \in A$ with radius τ. Summing up all these values for all $x \in A$ yields the probability $P_d(v,v')(\tau)$ that the distance $d(v, v')$ is smaller than τ. The following lemma describes formally how to compute P_d for two approximated feature vectors.

Lemma 1 *Distance Distribution Function*. Let $A=f_{app}(f_{rank}(v),L)$ and $A'=f_{app}(f_{rank}(v'),L') \in [IR \times IR]^d$ be two arbitrary conservative approximations of the feature vectors $v, v' \in IR^d$. Let $R(x, \tau)$ denote a sphere around the feature vector $x \in IR^d$ with radius $\tau \in IR$. Then the *distance distribution function* P_d: $IR^d \times IR^d \rightarrow (IR_0^+ \rightarrow [0..1])$ based on the approximations A and A' can be computed as follows.

$$P_d(v, v')(\tau) = \frac{\int_A |A' \cap R(x, \tau)|\, dx}{|A| \cdot |A'|}$$

As already mentioned clustering algorithms can only handle unique distance values. In order to put clustering methods into practice, we extract an aggregated value which we call distance expectation value. The distance expectation value E_d: $O \times O \rightarrow IR_0^+$

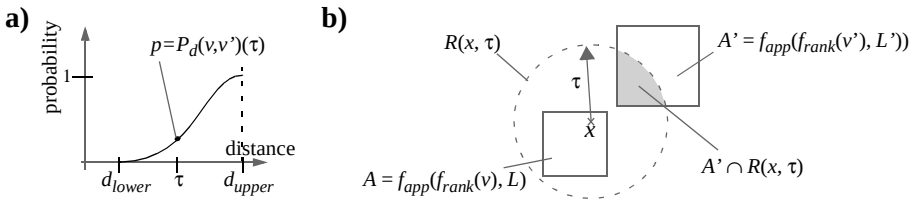

Figure 3. Computation of the distance distribution function P_d.
a) distance distribution function **b)** computation of the probability $P_d(v,v')(\tau)$

represents the similarity between two box approximations in the best possible way by one single value $E_d(v, v') = \int_{-\infty}^{\infty} ((P'_d(v, v')(x) / dx) \cdot x) dx$, where $P'_d(v, v')$ denotes the derivation of $P_d(v, v')$.

Practically, we can compute this distance expectation value between two approximated feature vectors by means of monte-carlo sampling. Thereby, we create randomly feature vectors located in the boxes and compute the average distance of these randomly created feature vectors to each other, Obviously, the higher the sample rate the more accurate is the computation of the distance expectation value. Note, that the center oriented distance can be regarded as a distance expectation value where only one sample pair, i.e. the box centers, is used.

5 Experiments

In this section, we evaluate the performance of our approach with a special emphasis on the overall transmission cost. The tests are based on an artificial dataset ART and two real world data sets PLANE and PDB which were distributed to two clients:

ART dataset. The artificial dataset ART consists of 1000 feature vectors, equally distributed in a 30-dimensional vector space.

PLANE dataset. The PLANE dataset consists of 1000 high-resolution 3D CAD objects provided by our industrial partner, an American airplane manufacturer. Each object is represented by a 42-dimensional feature vector which is derived from the cover sequence model as described in [9].

PDB dataset. This 3D protein structure dataset is derived from the Brookhaven Protein Data Bank (PDB). The 1000 objects are represented by 3D shape histograms [1] resulting in a 120-dimensional feature vector per object.

5.1 Quality of the Feature Vector Approximation Techniques

In a first experiment, we examined the quality of the three approximation techniques *BOA*, *DOA* and *CA* (cf. Section 3). For each feature vector, we transmitted once *L* bytes (measured in percent of all bytes of a feature vector) to the server which then constructs the approximations based on the transmitted data. Figure 4 depicts how the approximation error depends on the transmission cost. The error is measured by the average length of the diagonal of the approximation areas.

Figure 4a shows that the average approximation error rapidly decreases for the *BOA* approach as well as for the *CA* approach. For high values of *L*, the *DOA* approach per-

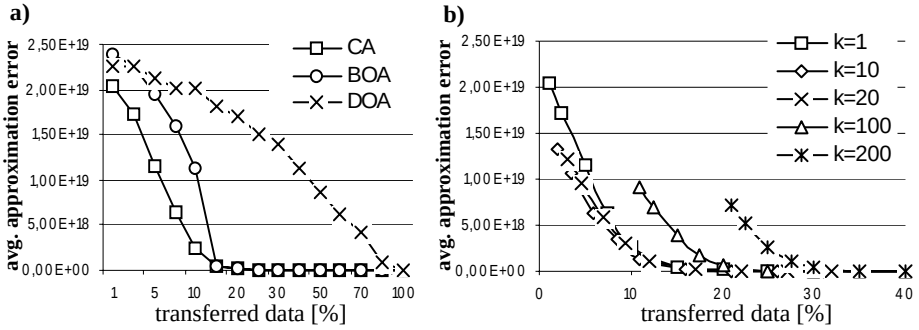

Figure 4. Average approximation error (ART dataset)
a) varying approximation techniques ($k = 1$) **b)** varying k parameter (*CA* approach)

forms worst. Only for very small values of L it outperforms the *BOA* approach. However, our *CA* approach yields to the best results, especially for low transmission cost.

Furthermore, we examined the *CA* approach for a varying parameter k used for the k-means based pre-clustering of the client sites. Figure 4b shows that if we initially transmit only the pre-clustering information of the feature vectors, i.e. the dataset approximations (cf. Section 3.1), the approximation quality increases slowly with increasing k. Obviously, an increasing k parameter yields higher transfer cost. In contrast to the dataset approximation approach, the quality increases more rapidly when we increase the amount of transmitted data of the feature vector approximations (cf. Section 3.2). Figure 4b shows that we achieve the best trade-off between accuracy and transfer overhead when we set $k = 10$, especially for low transfer cost.

5.2 Distance Measures

In this section, we investigate the accuracy of the two distance measures, *mid*(A, A') and *exp*(A, A'). The distance function *mid*(A, A') denotes the distance between the center points of the approximations A and A' and the distance function *exp*(A, A') denotes the expected distance of the feature vectors approximated by A and A' (cf. Section 4). For the computation of the expected distance *exp*(A, A'), we used monte-carlo sampling with a sample rate s. For measuring the quality we summed up the quadratic distance error of the examined distance measures *exp*(A, A') and *mid*(A, A') with respect to the exact distance of the feature vectors. Figure 5 depicts the average quadratic distance error of all feature vector approximations.

In the first experiment, we observed the behavior of *exp*(A, A') for a varying sample rate s. Figure 5a shows that the distance function *exp*(A, A') reflects the exact distance between the feature vectors much more accurately than the distance function *mid*(A, A'), already for a sample rate $s > 2$. Figure 5b shows that the difference between the two distance measures *exp*(A, A') and *mid*(A, A') increases with decreasing transfer cost. Therefore it is especially important to use the *exp*(A, A') distance measure when only small transfer cost are allowed.

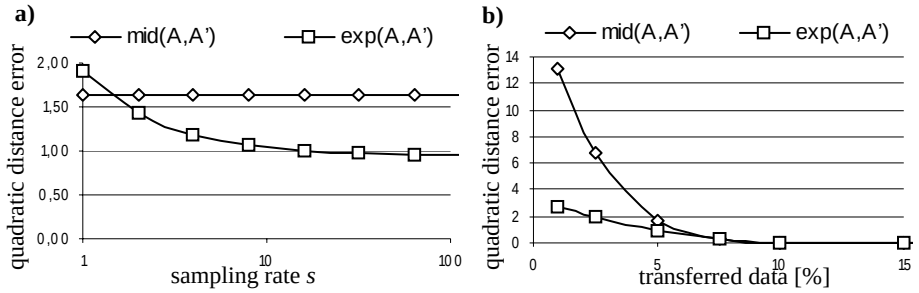

Figure 5. Evaluation of the Distance Measures (ART dataset)
a) varying sampling rates *s* (5% transferred) **b)** varying approximation accuracy (*s*=50)

5.3 Density-Based Clustering

In a last experiment, we compared a standard DBSCAN run based on the *exp*(*A*, *A'*) measure to the distributed clustering approach introduced in [7]. We measured the quality of the approximated clustering result by the quality criterion used in [7]. Figure 6 shows clearly that for a certain amount of transferred information our approach performs much better, i.e. our approach yields higher quality values than the approach presented in [7]. Note that the approach of [7] was especially designed for achieving high-quality distributed clusterings based on little transmitted information. We would like to point out that this experiment shows that our approximation technique for high-dimensional feature vectors can beneficially be used as basic operation for distributed data mining algorithms.

6 Conclusion

In this paper, we presented a novel technique for approximating high-dimensional distributed feature vectors. In order to generate suitable approximations, we enhanced the idea of state-of-the-art index structures for high-dimensional data which approximate each single feature vector by a certain number of granted bytes. Based on this technique we can limit the transmission cost considerably while only allowing a small decrease of

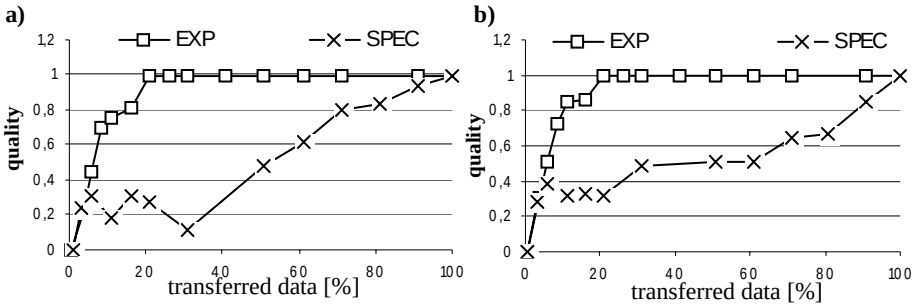

Figure 6. Clustering quality (s = 10, k = 10)
a) PLANE dataset **b)** PDB dataset

the quality. We demonstrated the benefits of our technique for the important area of distributed clustering.

In our future work, we will show that also other distributed data mining algorithms benefit from our high-dimensional feature vector approximation technique.

References

1. Ankerst M., Kastenmüller G., Kriegel H.-P., Seidl T.: *3D Shape Histograms for Similarity Search and Classification in Spatial Databases*. Proc. 6th Int. Symposium on Large Spatial Databases (SSD'99), Hong Kong, China, in: Lecture Notes in Computer Science, Vol. 1651, Springer, 1999, pp. 207-226.
2. Berchtold S., Böhm C., Jagadish H. V., Kriegel H.-P., Sander J.: *Independent Quantization: An Index Compression Technique for High Dimensional Data Spaces*. ICDE'00, 2000.
3. Beckmann N., Kriegel H.-P., Schneider R., Seeger B.: *The R*-tree: An Efficient and Robust Access Method for Points and Rectangles*. Proc. ACM SIGMOD'90, 1990.
4. Golub, T., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M. L., Downing, J., Caligiuri, M., Bloomeld, C., Lander, E.: *Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring*. Science, 286, pp. 531-537.
5. Johnson E., Kargupta H.: *Hierarchical Clustering From Distributed, Heterogeneous Data*. In Zaki M. and Ho C., editors, Large-Scale Parallel KDD Systems. Lecture Notes in Computer Science, colum 1759, pp. 221-244. Springer-Verlag, 1999
6. Januzaj E., Kriegel H.-P., Pfeifle M.: *DBDC: Density Based Distributed Clustering*. Proc. 9th Int. Conf. on Extending Database Technology (EDBT 2004), Heraklion, Greece, 2004, pp. 88-105.
7. Januzaj E., Kriegel H.-P., Pfeifle M.: *Scalable Density-Based Distributed Clustering*. Proc. 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), Pisa, Italy, 2004.
8. Jain A. K., Murty M. N., Flynn P. J.: *Data Clustering: A Review.* ACM Computing Surveys, Vol. 31, No. 3, Sep. 1999, pp. 265-323.
9. Kriegel H.-P., Brecheisen S., Kröger P., Pfeifle M., Schubert M.: *Using Sets of Feature Vectors for Similarity Search on Voxelized CAD Objects*. Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'03), San Diego, CA, 2003, pp. 587-598.
10. Kargupta H., Chan P. (editors) : *Advances in Distributed and Parallel Knowledge Discovery.* AAAI/MIT Press, 2000.
11. McQueen J.: *Some Methods for Classification and Analysis of Multivariate Observation*. Proc. 5th Berkeley Symp. on Math. Statist. and Prob., Vol. 1, 1965
12. Shawney H., Hafner J.: *Efficient Color Histogram Indexing.* Proc. Int. Conf. on Image Processing, 1994, pp. 66-70.
13. Salton G., McGill M. J.: *Introduction to Modern Retrieval.* McGraw-Hill Book Company, 1983.
14. Samatova N.F., Ostrouchov G., Geist A., Melechko A.V.: *RACHET: An Efficient Cover-Based Merging of Clustering Hierarchies from Distributed Datasets*. Distributed and Parallel Databases 11(2): pp. 157-180; Mar 2002.