Clustering Classifiers for Knowledge Discovery from Physically Distributed Databases

Grigorios Tsoumakas, Lefteris Angelis, Ioannis Vlahavas

Department of Informatics, Aristotle University of Thessaloniki
Thessaloniki 54124, Greece

Abstract

Most distributed classification approaches view data distribution as a technical issue and combine local models aiming at a single global model. This however, is unsuitable for inherently distributed databases, which are often described by more than one classification models that might differ conceptually. In this paper we present an approach for clustering distributed classifiers in order to discover groups of similar classifiers and thus similar databases with respect to a specific classification task. We also show that clustering distributed classifiers as a pre-processing step for classifier combination enhances the achieved predictive performance of the ensemble.

Key words: Multi DBs; Knowledge Discovery; Machine Learning

1 Introduction

Nowadays, physically distributed databases are increasingly being used for knowledge discovery. Advances in network technology and the Internet as well as the growing size of data being stored in today's information systems have contributed to the proliferation of distributed data mining. Globalization, business-to-business commerce and online collaboration between organizations rose lately the need for inter-organizational data mining, which also involves mining physically distributed databases.

It is often unrealistic to collect distributed data for centralized processing. The necessary central storage capacity might not be affordable, or the necessary bandwidth to efficiently transmit the data to a single place might not be available. In addition, there are privacy issues preventing sensitive data (e.g. medical, financial) from being moved around the distributed databases. The most effective approach to deal with these problems is to locally mine each database and combine the resulting models with an appropriate technique.

However, most classifier combination methodologies view data distribution as a technical issue and treat distributed databases as if they were parts of a single database. This has been identified as a very narrow view of distributed data mining [17],[26]. Real-world, inherently distributed databases have an intrinsic data skewness property. The data distributions in different partitions are not identical. For example, data related to a disease from hospitals around the world might have varying distributions due to different nutrition habits, climate and quality of life. The same is true for buying patterns identified in supermarkets at different regions of a country. Another example are Web document classifiers trained from directories of different Web portals.

The classic approach of combining distributed classifiers in an attempt to derive a single global model is unsuitable for cases similar to the above. There might not really exist a single model that describes the distributed data, but two or more *groups* of models. Therefore, one should first explore the dependencies of local models instead of a straightforward integration.

This paper presents an approach for clustering local classification models induced at physically distributed relational databases. The proposed approach groups together classifiers with similar behavior and thus i) facilitates the learning of new concepts that characterize important common features of, and differences between, the respective databases and ii) leads to the creation of a classification model for each cluster that together exhibit more accurate predictions than a single global model. Therefore, it can be used both as a method to discover groups of similar databases with respect to a specific classification application as well as a pre-processing step to enhance predictive performance in distributed classification. Experimental results on real-world data, synthetic data, and data produced by a new technique for splitting a single database into various parts with different context, confirm the effectiveness of the proposed approach.

It is assumed that the distributed databases have the same set of attributes and are syntactically homogeneous. These assumptions are often true when the distributed databases belong to the same organization, for example health care units and hospitals of a regional health network, or local branches of a financial institution.

The rest of this paper is organized as follows. Section 2 provides background knowledge on supervised classification and related work on classifier combination methods. Section 3 presents our approach on clustering classifiers for knowledge discovery from physically distributed databases. Section 4 gives comparative experimental results confirming the effectiveness of our approach and Section 5 discusses its complexity and scalability. Finally, Section 6 concludes, summarizes the advantages of the proposed approach and points to a future research direction.

2 Background

2.1 Supervised Classification

Supervised classification is one of the most common machine learning and data mining tasks [19]. It deals with the problem of identifying interesting regularities between a number of independent variables and a target or dependent categorical variable in a given data set. For example, given a set of training instances $(x_{i1}, x_{i2}, ..., x_{ik}, y_i), i = 1..N$, the task is to compute a classifier, or model, or concept that approximates an unknown function $y = f(x)$ that correctly labels any instance drawn from the same source as the training set.

There exist many ways to represent a classification model and many more algorithms to generate it. Typical classifier learning approaches include concept learning, neural networks, decision trees, rule learning, Bayesian learning and instance-based learning [14]. All of these approaches construct models that share the common ability to classify previously unknown examples of a domain based on examples of the same domain that were used for their training.

The output of a classifier can be i) the label of a class, ii) rankings for all the classes and iii) measures of uncertainty such as belief, confidence, probability, possibility, plausibility or other for each class. Consider for example, a domain for predicting tomorrow's weather with three possible classes: *sunny*, *windy*, *rainy*. The corresponding output for the three types of classifiers could be: i) sunny, ii) 1 - sunny, 2 - windy, 3 - rainy and iii) 0.8 - sunny, 0.5 - windy, 0.1 - rainy. Classifiers that output labels are often called hard classifiers, while those that output measures of uncertainty are called distribution/soft classifiers. Classifiers that output rankings are not so common in the machine learning literature.

2.2 Classifier Combination

The way that multiple classifiers are combined is an important research issue that has been investigated in the past from the communities of statistics, pattern recognition, machine learning and data mining.

When only the label of the predicted class is available, then the simplest combination method that can be used is Majority Voting [9], which does not require a training stage. In this case, the class that receives the most classifier predictions is the final result. Weighted Majority Voting [12], weights the decision of each classifier by its performance on the training data.

When a measure of belief, confidence, certainty or other about the classification is available along with the class label, then a number of different rules for combining these measures have been suggested, like Sum, Min, Max, Prod and Median. An an interesting study of these rules is [10].

An alternative approach to classifier combination involves learning a global classifier from distributed data. Stacked Generalization [28], also known as Stacking in the literature, combines multiple classifiers by learning the way that their output correlates with the true class on an independent set of instances.

The concept of Stacked Generalization was applied to distributed data mining, via the Meta-Learning methodology [3]. Meta-Learning focuses on combining distributed data sets and investigated various schemes for structuring the meta-level training examples. It assumes that there is a single model that could be induced from the distributed databases, and thus could benefit by using our approach as a pre-processing step.

The idea of all-to-all exchange of classifiers in order to avoid moving raw data around the distributed nodes and make use of all available data for validation purposes has been introduced in [22] in an extension of Stacking. The same idea is used in our approach to ensure the availability of as many data as possible for calculating the classifier distance, and at the same time avoiding the costly process of moving raw data around the distributed databases.

An approach that discovers a single comprehensible model out of the distributed information sources can be found in [4]. The main idea in the DAGGER algorithm is to selectively sample each distributed database so as to form a new data set that will be used for inducing the single model. Sampling aims at selecting a minimal spanning example set from every decision region of each model. This is a subset of examples that demonstrated all the values of the attributes of the set of examples within the region. This approach requires moving raw data from each distributed database, which could be costly, but more importantly, it could be not allowed in applications regarding sensitive or private data.

Another approach that aims at the creation of a single model is [5], where direct integration of distributed models is performed. It involves learning decision trees in parallel from disjoint data, converting trees to rules and combining the rules into a single rule set.

An abstract approach that addresses the problem of conceptual differences of distributed databases is Knowledge Discovery from Models [26]. It consists of four steps: 1) build local models, 2) compare local models at a central site to identify interesting differences and similarities 3) Explain differences and similarities through additional analysis steps 4) Act on the insights. An in-

stantiation of this abstract approach based on Bayesian networks is presented in [2]. This approach is specific to Bayesian classifiers, while the proposed one can make use of any type of classifier.

In [13], a different clustering approach is followed in order to deal with the problem of semantic heterogeneity of distributed databases. Clustering of the databases is performed based on the distances of aggregated data that summarize each distributed database. This approach demands the exchange of data, that even aggregated could carry important sensitive information. In addition it does not use the clustering result for guiding a classifier combination method.

Finally, [15] presents an approach on clustering distributed databases, based on association rules. The clustering method used, is an extension of hierarchical agglomerative clustering, that uses a measure of similarity of the association rules at each database.

3 Clustering the Classifiers of Distributed Databases

Clustering distributed classifiers is based on: i) a measure of classifier distance, ii) an efficient algorithm to compute this distance measure for classifiers induced at physically distributed databases and iii) a clustering algorithm that will receive as input the calculated distances and will output the clusters. These issues along with the subject of exploiting the clustering result in order to achieve better classification accuracy are the topics of the following sub-sections.

3.1 Classifier Distance

We here introduce the notion of *classifier distance* as a measure of how different two classification models are and propose its empirical measurement based on the classifiers' predictions on instances with known classes of an *independent* data set. By independent, we mean a data set whose instances were not part of the classifiers' training set. This will ensure unbiased results, as the predictions of classifiers on their training data tend to be optimistic.

If both models are soft classifiers, then some measures that can be used for calculating classifier distance are the Euclidean Distance, the Canberra Distance and the Czekanowski Coefficient[11]. In this case, the distance of two classifiers is defined as the average distance of their output vectors with respect to all instances of the independent data set.

If both models are hard classifiers, then some measures that can be used for calculating classifier (dis)similarity are Yule's Q statistic, the correlation coefficient, the disagreement measure and the double-fault measure [21].

If one model is a soft classifier and the other a hard classifier, then one could transform the output of the soft classifier to a single class label by selecting the label with the maximum certainty value, breaking ties arbitrarily. Then the distance measures for hard classifiers can be used for calculating classifier distance. Another solution is to transform the output of hard classifiers to a vector of certainty values for each class, but this usually requires additional training data. Therefore we suggest the first solution for calculating the distance of mixed type of classifiers.

The proposed empirical evaluation of classifier distance exhibits the following beneficial properties:

- *Independence of the classifier type.* It is able to measure the distance of two classification models, whether they are decision trees, rules, neural networks, Bayesian classifiers, or other. This is useful in applications where different types of learning algorithms might be used at each distributed node.
- *Independence of the classifier opacity.* It is able to measure the distance of two classification models, even if they are black boxes, providing just an output with respect to an input. This is useful in applications where the models are coming from different organizations that might not want to share the details of their local models.

In this paper we focus on the use of the disagreement measure for hard classifiers because i) it is simple and fast to compute ii) it can be computed incrementally, iii) it gives a value that directly expresses the distance of two classifiers that can be used without any transformation for the clustering process, and iv) it can be used for mixed type of classifiers, by transforming the output of soft classifiers to a single class label.

Consider two hard classifiers, C_x and C_y and a database D with M tuples. The disagreement measure is calculated as follows:

$$d_D(C_x, C_y) = \frac{\sum_{i=1}^{M} \delta_{x,y}^{(i)}}{M} \tag{1}$$

where $\delta_{x,y}^{(i)}$ equals 1 if classifiers C_x and C_y have different output on tuple i, and 0 otherwise.

3.2 Distributed Classifier Distance

Accurate calculation of the distance between distributed classifiers is needed, in order to accomplish an effective clustering result. This can be achieved by ensuring the availability of substantial independent instances, as the proposed distance measuring method depends on the output of the classifiers on independent instances. At the same time, the communication cost should be kept at the minimum due to the distributed nature of data.

According to the above, we propose the use of each distributed database for measuring the distance of each pair of classifiers apart from the pairs that contain the local classifier. This way only classifiers get to be exchanged and no raw data at all, and the distance of all pairs of classifiers is calculated based on all independent data.

The whole process of calculating the distance of the distributed classifiers can be broken down into the following steps:

(1) Consider N distributed databases D_i, $i \in \{1..N\}$ and the corresponding classification models C_j, $j \in \{1..N\}$, that were induced at those databases. At first, each database D_i imports all classifiers C_j, $j \in \{1..N\}$, $j \neq i$ from the rest of the distributed databases.
(2) Then, at each database D_i we calculate the distance for all pairs of classifiers apart from the ones that contain the local classifier C_i, according to Equation 1. Thus, we calculate:

$$d_{D_i}(C_x, C_y), \forall (x, y) \in S_i^2 : x < y$$

where $S = \{1, \ldots, N\}$ and $S_i = S - \{i\}$.
(3) The result of the distance calculation for each pair of classifiers C_x, C_y, at each database is broadcasted to every other database. Therefore, at each database there will be $N - 2$ calculated distances for each pair of classifiers [1].
(4) The average of these distances is obtained as the overall distance for each pair of models:

$$d(C_x, C_y) = \frac{1}{N-2} \sum_{i \in S_x \bigcap S_y}^{N} d_{D_i}(C_x, C_y) \tag{2}$$

Algorithm 1, illustrates the above process of calculating the distances of all pairs of classifiers given all distributed databases. The input is an array DB of N databases and an array C of N classifiers. The output is an array $Dist$ with

[1] The distance of each pair of local classifiers is evaluated in all N databases, apart from the two databases that were used for training these two classifiers.

distances for all distinct pairs of classifiers. *Dist* has a size of $\frac{N*(N-1)}{2}$, which corresponds to the lower triangular part of the NxN distance matrix of the classifiers. This encoding is used to save space in storing classifier distances.

Algorithm 1 *Distributed Classifier Distance*
Input
DB: an array of N databases
C: an array of N classifiers
Output
Dist: an array of $N*(N-1)/2$ *distances*
Begin
// Calculate the distance based on each database
For $j \leftarrow 1$ **To** N
$TempDist[j] \leftarrow CalcDist(DB[j], C, j)$;

// Average distances
For $i \leftarrow 1$ **To** $N*(N-1)/2$
begin
$Dist[i] \leftarrow 0$;
For $j \leftarrow 1$ **To** N
$Dist[i] \leftarrow Dist[i] + TempDist[j][i]$;
end
For $i \leftarrow 1$ **To** $N*(N-1)/2$
$Dist[i] \leftarrow Dist[i]/(N-2)$;
End

The algorithm first calculates for each database the distances of all pairs of classifiers. This is achieved using the *CalcDist* function, which is presented in Algorithm 2. All calculated distances are stored in the two-dimensional array $TempDist$. They are then divided by $N-2$ to obtain the final distance vector.

Given a database, function *CalcDist* calculates the distances of all pairs of classifiers apart from those that contain the classifier that was trained on that database. The input is an array D of M instances (a database), an array C of N classifiers and the database's index j. The output is an array $Dist$ with distances for all distinct pairs of classifiers, which has a size of $\frac{N*(N-1)}{2}$.

For each tuple i of the database the algorithm calculates the output O of all classifiers once (apart from the local classifier) and then proceeds by comparing the output for all pairs (C_x, C_y) (apart from the pairs that contain the local classifier). Each time a pair of classifiers disagrees on the output, its distance is increased by one. In the end, the distance is divided by the number of instances to obtain the actual disagreement measure.

Algorithm 2 *Classifier Distance*
Input
D: an array of M instances (a database)
C: an array of N classifiers
j: the index of database D
Output:
Dist: an array of $N * (N-1)/2$ *distances*
Begin
For $i \leftarrow 1$ **To** M
begin
// Calculate the output of classifiers
For $x \leftarrow 1$ **To** N
If $x \neq j$ **Then**
$O[x] \leftarrow C[x](D[i]);$

// Update distances
$index \leftarrow 1;$
For $x \leftarrow 1$ **To** $N-1$
For $y \leftarrow x+1$ **To** N
begin
If $x \neq j$ *and* $y \neq j$ **Then**
If $O[x] \neq O[y]$ **Then**
$Dist[index] \leftarrow Dist[index] + 1;$
$index \leftarrow index + 1;$
end
end

// Normalize distances
For $index \leftarrow 1$ **To** $N * (N-1)/2$
$Dist[index] \leftarrow Dist[index]/M;$
End

3.3 Clustering

Having calculated the pairwise distances of all distributed classifiers, we proceed by clustering them using Hierarchical Agglomerative Clustering [6].

Agglomerative clustering algorithms usually start with assigning each of the data points to a single cluster. Then in each step two clusters are merged, until only one is left. The merging process is based on measures of distance between clusters. There are various strategies for evaluating inter-cluster distances including *single linkage*, *complete linkage*, *Ward's method* and *weighted*

average linkage [8]. The sequence of merging the clusters can be visualized as a tree-shaped graph, which is called a dendrogram. For the automatic selection of a single clustering result from the sequence, a user-specified cutoff value can be provided, that affects when the agglomeration of clusters will stop.

We chose this clustering method because: i) it is not possible to know the number of clusters in advance, ii) it requires the pairwise distances of the items to be clustered, which have already been computed for the distributed classifiers, iii) although the space and time complexity of the method is $O(n^2)$, the number of classifiers will not usually be very large in common applications of distributed data mining and iv) it produces a convenient visualization of the clusters.

3.4 Classification

The descriptive knowledge that the final clustering result conveys about the distributed classifiers, can be used for guiding the combination of the classifiers. Specifically, the classifiers of each cluster can be combined in order to produce a single classifier corresponding to each cluster. The rationale is that this approach leads to superior results both in interpretability and predictive performance.

A single model has poor interpretability because it is the fusion of the different underlying concepts of the distributed databases. For the same reason it will be poor in classifying new examples located at any database. In contrast, the classifiers that correspond to each cluster capture the cluster contexts and their interpretation can provide useful knowledge with respect to each cluster. Moreover, when classifying a new example at a database only the classifiers that belong to the cluster of this database will be combined, leading to increased accuracy.

4 Empirical Evaluation

A series of experiments was used to evaluate both the capability of our approach to detect groups of similar classifiers and the predictive performance of clustered classifiers. The following sections describe these experiments and the data that were used.

4.1 Data Sets

Our approach demands data with natural skewness and variability in context, which are found in real-world distributed databases. However, the availability of such data sets offered as a test-bed for experimenting is limited, an important hindrance to empirical research on distributed knowledge discovery.

This raises the issue of how to consistently simulate the data properties of inherently distributed databases, in order to setup a robust platform for experiments. Clearly, plain splitting of a large database to many smaller parts isn't realistic enough for mining inherently distributed data. A more efficient way to simulate the distribution is to create synthetic data from different concepts or modify existing domains by adding different noise for each context.

The first collection of data sets that we used in our experiments were synthetic. We created them according to the technique described in [20], which is based on a multi-normal pseudo-random data generator. Each data set contained 12 variables $a_1, \ldots, a_{12}$, following the multivariate normal distribution with parameters: mean 0, variance for each one of them 1 and covariances between them either 0, ± 0.2 or ± 0.5. A binary (0, 1) classification variable was also calculated as a function of the variables. Specifically, three data sets were created with covariances 0, ± 0.2 and ± 0.5 and classification rule $a_1 + a_{11} > 0$. Another three data sets were created with covariances 0, ± 0.2 and ± 0.5 and classification rule $a_1 + a_2 + a_{11} + a_{12} > 0$. Finally, three data sets were created with covariances 0, ± 0.2 and ± 0.5 and classification rule $a_1 \times a_2 + a_{12} > 0$. All of the above 9 data sets comprised of 10000 instances each.

The second and third collection of data sets that we used, were created based on a new technique that we propose for controllably creating data sets with different context from a single data set. The main idea is based on the notion of contextual attributes [23], [24], whose values are associated with a different context within a classification domain. A change of value of such an attribute signifies a change of the underlying classification concept. For example, the attribute *season* is a contextual attribute within the domain of predicting weather.

The technique splits a single database into as many parts as the distinct values of a contextual attribute of that database. In each of the resulting parts of the database all tuples will have the same value for the contextual attribute. This attribute is then removed from those parts. This leads to syntactically homogeneous databases with a different classification context. By further splitting each of these parts into as many smaller parts as necessary for an experiment, one can controllably create a varying number of syntactically homogeneous databases with the characteristics of physically distributed data.

We used our technique on the *US Census Bureau* data set from the UCI Machine Learning repository [1], due to its large size and the availability of discrete attributes that we could choose to split. We chose the *sex* of the persons as a contextual attribute, because it is probably not a primary factor that influences the class attribute (gross income), but it can influence the way the rest of the attributes relate with the class attribute. Therefore, it makes a good choice of a contextual attribute. We also used the *vowel* data set, because it was used in the past for studying contextual attributes and we knew that the attribute *sex* of a person has been identified as a contextual attribute.

The last collection of data sets that we used in our experiments were the physically distributed *Heart* data from the UCI Machine Learning repository, which are data about heart-disease collected from 4 different hospitals: 1) Cleveland Clinic Foundation, 2) Hungarian Institute of Cardiology, Budapest, 3) V.A. Medical Center, Long Beach, CA and 4) University Hospital, Zurich, Switzerland.

4.2 Clustering

In order to evaluate whether our approach correctly groups the relevant classifiers together, we used data sets with known clustering. The clusters of synthetic data and data produced by our technique are artificially created and therefore known a priori. A previous study on clustering distributed databases [13] that used the *Heart* data, resulted in grouping together the Hungarian and Cleveland hospitals and considered the two other hospitals as stand-alone clusters.

We used an implementation of the c4.5 algorithm [18] in Java from the WEKA toolkit [27] as the local learning algorithm for every database and calculated the distance based on the disagreement measure. The resulting distance vector then served as input to the *linkage* function of Matlab [7] which hierarchically clustered the classifiers using the *complete linkage* strategy. The dendrograms produced by this process are depicted in Figure 1.

We notice that classifiers from the same clusters are clearly grouped correctly together. Besides the visual interpretation of clustering, the actual clusters produced by the *cluster* function of Matlab match the true clustering of the databases.

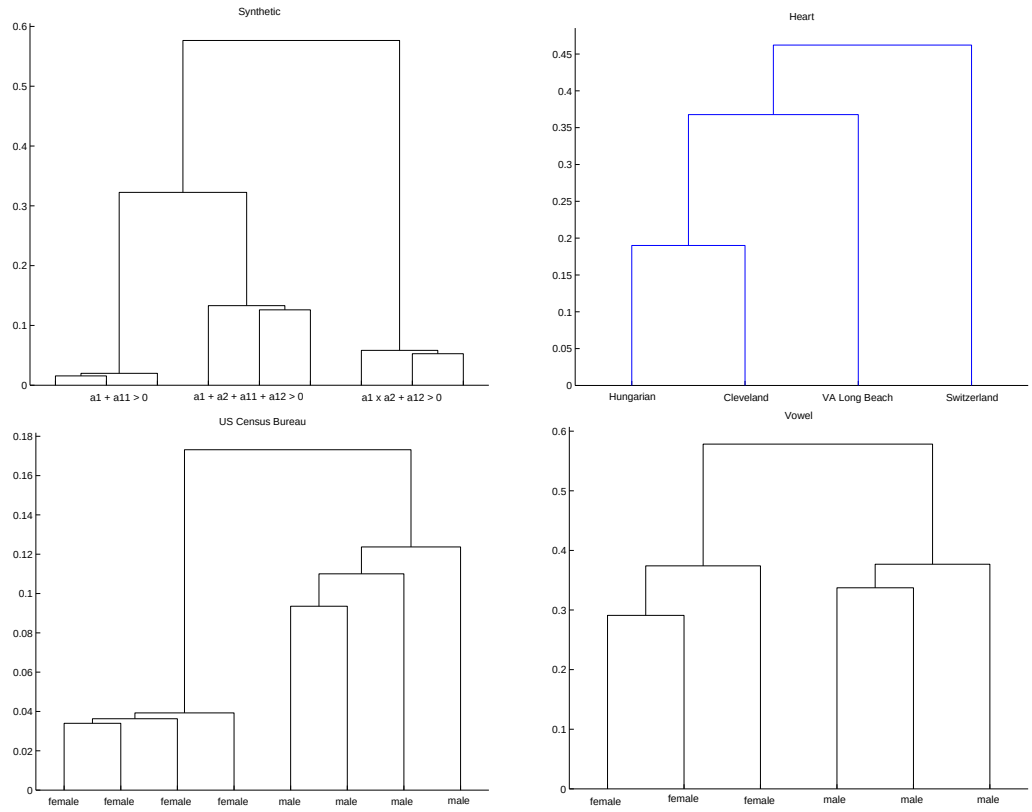

Fig. 1. Dendrograms produced by Hierarchical Agglomerative Clustering

4.3 Classification

The second set of experiments aims at discovering whether clustering classifiers adds to distributed classification approaches as a pre-processing step. Intuitively the answer to this question is positive. If there are more than one groups of classification models that describe the distributed data, then a single classifier will function as an *average model* that will under-perform in data of either cluster.

To empirically confirm this claim, we conducted a comparative study of the predictive performance of i) stand-alone local classifiers, ii) Majority Voting [9], iii) Stacking [3] and iv) clustering local classifiers using our approach and combining the classifiers of each cluster with Majority Voting.

The setup of the experiments was the following:

(1) Randomly split each of the distributed databases into a training set (75%) and a test set (25%).
(2) Train a c4.5 classifier at each one of the distributed nodes using the locally available training set.
(3) Compare the following approaches:

- Evaluate each classifier on each of the corresponding local test data and average the results.
- Evaluate the majority voting of all classifiers on the sum of the test data.
- Split the sum of all test data into 50% of meta-train data and 50% of test data. Train a c4.5 meta-classifier on the first 50% and then evaluate it on the rest of the data.
- Cluster classifiers and for each cluster evaluate the majority voting of the classifiers of that cluster on the sum of the test data of that cluster. Weight the results of each cluster by the number of databases it contains and average over the total number of databases.

Table 1 presents the results that are averages over 10 runs of the above described experiment. The first column indicates the classification method. The next columns show the accuracy result of each method on the four collections of data sets. The best result for each collection of data sets is indicated with bold typeface.

Table 1
Results showing the accuracy percentage for the four different approaches

Approach	Synthetic	US Census	Vowel	Heart
Single Classifiers	94.84	85.53	56.23	54.97
Majority Voting	77.24	86.40	59.84	53.11
Stacking	76.61	85.06	57.50	**58.17**
Clustering + Majority Voting	**96.30**	**86.73**	**64.18**	55.00

The results with the *Synthetic* data sets prove on one hand the bad performance of Majority Voting and Stacking, which are methods that assume there is only one model describing the distributed data. Usually such methods lead to better results than the average accuracy of single classifiers. In this case, where data come from different concepts the failure of such methods is evident. In contrast, the clustering of classifiers manages to increase the average classification accuracy.

The results with the *US Census Bureau* and *Vowel* data sets are comparable, although there is a smaller accuracy increase in comparison to the *Synthetic* data sets. The obvious reason is that the *Synthetic* data sets were artificially created from three clearly different concepts. The concepts in the clusters of the *US Census Bureau* and *Vowel* data sets that were split using the proposed technique are more related to each other. This shows that there is a trade-off between the similarity of the concepts of the underlying clusters and the increase in classification accuracy by using our approach. This is a reasonable conclusion as very similar concepts can be easier described by a single average model.

The only collection of data sets where our approach is not the best, is the *Heart* data. However, the results on these data should be taken with care. When experimenting with a classifier combination technique, it is much easier to draw conclusion with respect to the predictive performance of the ensemble when all of the participating classifiers have the same accuracy. In the *Heart* data, two of the local classifiers had very poor quality due to a lot of missing values within the corresponding data sets. Therefore, the results might be misleading. For example, the meta-classifier of Stacking might have learned to select only the two good classifiers in classifying the data. This might lead to better accuracy results, but there is no practical use of the global classification model.

Table 2 sheds more light into the details of the accuracy results. It shows the clustering result and the average accuracy of the single classifiers induced from the synthetic databases. We notice that the average accuracy of each classifier and each cluster varies, but all classifiers exhibit high accuracy in general. Therefore it is clear that approaches 2 and 3 exhibit bad performance due to their ineffectiveness in detecting the three different classification concepts and not due to the ineffectiveness of the local classifiers. It also shows the average accuracy of the single classifiers per cluster, the accuracy of the best classifier per cluster as well as the accuracy of our approach per cluster. We notice that our approach not only exceeds the average accuracy of the local classifiers but it is better than all of the best local classifiers.

Table 2
Accuracy of the local classifiers

Cluster	Classifier	Accuracy	Average	Best	Clustering
1	1	98.56	98.84	98.99	99.22
	3	98.99			
	6	98.99			
2	2	91.34	90.39	91.34	92.90
	4	90.40			
	5	89.42			
3	7	95.66	95.28	95.71	96.79
	8	94.46			
	9	95.71			

5 Scalability

This section starts the discussion of the scalability of the proposed approach, by an analysis of its computational complexity. It is assumed that each database calculates the disagreement measure of each pair of classifiers in parallel with every other database and thus the focus is on the complexity of the algorithm at a single database, as described by Algorithm 2.

For every tuple of the database the algorithm first calculates the output of $N-1$ classifiers and then the disagreement measure is computed for all combinations of $N-1$ classifiers in pairs. This number is equal to:

$$\binom{N-1}{2} = \frac{(N-1)(N-2)}{2}$$

Therefore, for a database with M tuples the order of time complexity of the algorithm is $O(MN^2)$.

5.1 Scaling Up to Very Large Databases

The proposed approach scales up linearly with respect to the number of tuples in a database, which is very efficient. In addition, the computation of the disagreement measure is incremental, which means that the space complexity of the algorithm with respect to the number of tuples is constant and equal to one tuple.

Still, one might want to use only a sample of the database for the calculation of the disagreement measure, in order to reduce the computational burden, especially in the case of a very large database. This is not a problem for the proposed approach, as Algorithm 2 can be easily extended to provide control over the number of tuples used at each database for distance calculation. This leaves open the question of whether the quality of the obtained distance vector will degrade.

To answer this question, Algorithm 2 was extended and an experiment was setup using the 4 collections of data sets for the calculation of the disagreement measure and varying the number of tuples used at each database. For the collections of the large *Synthetic* and *US Census Bureau* data sets, the number of tuples used for distance calculation varied from 50 to 1000 with a step of 50. Given that there were 9 *Synthetic* databases and 8 *US Census Bureau* databases, the total number of tuples used for distance calculation varied from 450 to 9000 and 400 to 8000 respectively. For the collections of the

smaller *Heart* and *Vowel* data sets, the number of tuples used for distance calculation varied from 10 to 120 with a step of 10. Given that there were 4 *Heart* databases and 6 *Vowel* databases, the total number of tuples used for distance calculation varied from 40 to 480 and from 60 to 720 respectively.

The experiment showed that the clustering result was always correct for all runs with the 4 collections of data sets, even with the minimum number of tuples used for the calculation of the distance of each pair of classifiers. This interesting finding shows that the disagreement measure is a robust distance calculation metric that does not require a lot of data in order to be adequately computed. Therefore, a small sample of each database is sufficient for the process of distance calculation, saving computational time without trading off quality.

We also recorded the time in milliseconds needed to calculate the disagreement measure at a single database with respect to the number of tuples used for the calculation. The plots in Figure 2 verify the linear scaling of the algorithm with respect to the number of tuples.

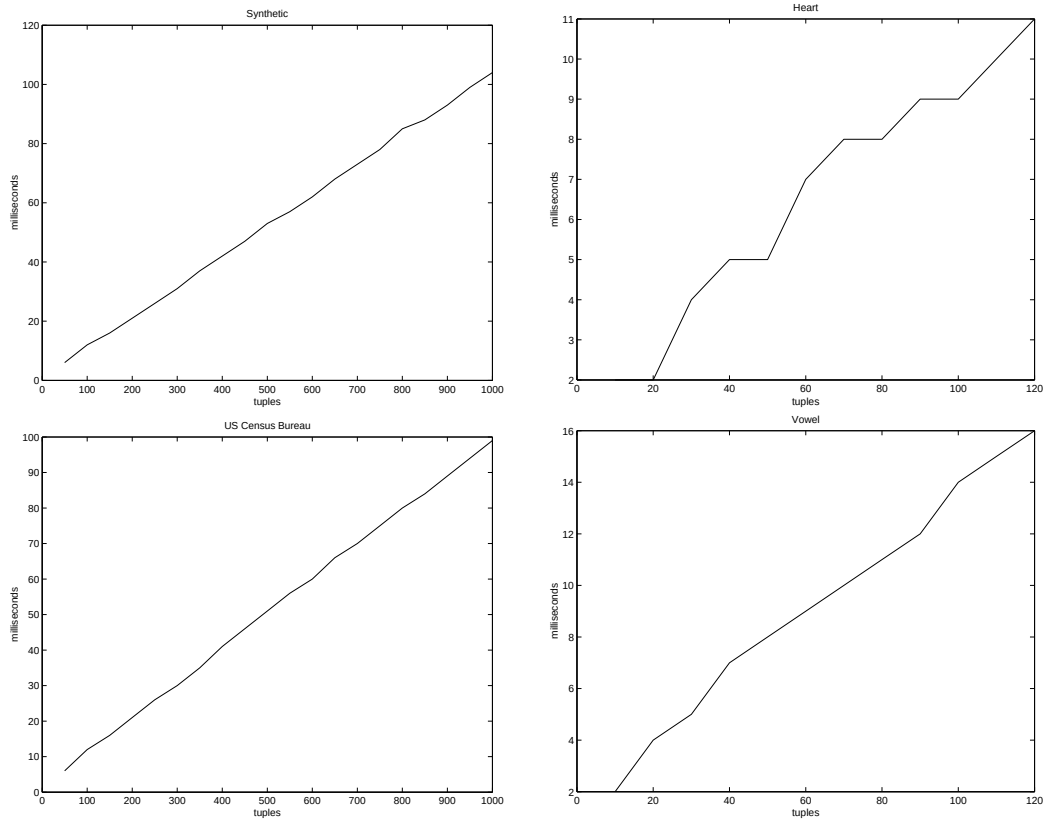

Fig. 2. Scalability with respect to tuples

The complexity of distance calculation is square with respect to the number of distributed databases. This could be a serious threat to the scalability of the proposed approach, despite the fact that the computation of the disagreement measure is simple and fast and that the number of distributed databases in common applications is small. There still could be domains with large numbers of databases, especially in this era of increasingly networked information systems.

It is possible to modify the proposed approach in such a way that the order of time complexity is reduced from square to linear. This can be accomplished by having the distance of each of the $\frac{N*(N-1)}{2}$ pairs of classifiers calculated at K databases instead of N-2, where K is a small constant, for example 1 or 2. The total number of distance calculations is then $\frac{K*N*(N-1)}{2}$. If these calculations are equally assigned to the N databases then each database would perform $\frac{K*(N-1)}{2}$ distance calculations instead of $\frac{(N-1)(N-2)}{2}$, thus reducing the complexity from square to linear.

This process demands an algorithm for equally assigning the $\frac{N*(N-1)}{2}$ pairs of classifiers to K databases, such that all databases calculate the distance of $\frac{K*(N-1)}{2}$ databases on average. For example, if K equals 2, then the distance of each pair of classifiers must be calculated based on 2 databases and the algorithm must ensure that each database calculates the distance of only N-1 pairs.

A greedy algorithm that implements the assignment process was constructed. The algorithm iterates over each pair of classifiers and tries to allocate K databases for it. The allocation satisfies the constraint of avoiding the two databases that were used for training the two classifiers of the pair and further selects the K databases with the minimum allocated pairs so far. The pseudocode is presented in Algorithm 3.

The algorithm uses three main structures: Table *count* of size N is used for storing the number of pairs that allocated each database. The logical table *sel* of size N is used to mark the databases that have been allocated sofar for the current pair. Finally, the 2-dimensional table *usedb* of size $\frac{N*(N-1)}{2} \times K$, is used for holding the indexes of the K databases that are assigned to the $\frac{N*(N-1)}{2}$ pairs. The algorithm starts by iterating each pair and allocating the database with the minimum allocated pairs. In ties, the last member of the table is considered to be the minimum. Once the minimum database is found (min_k), table *usedb* is updated with this allocation, table *count* is updated by increasing the number of allocations for the selected database and table *sel* is updating by setting to true the selected database.

Algorithm 3 *Assigning classifier pairs to databases*
Input
N: the number of classifiers
K: the number of databases for each pair
Output:
usedb: an array of $N * (N-1)/2 \times K$ *with the assignment*
Begin
$pair \leftarrow 0$
For $i \leftarrow 1$ **To** N
$count[i] \leftarrow 0$;
For $i \leftarrow 1$ **To** $N-1$
For $j \leftarrow i+1$ **To** N
begin
For $k \leftarrow 1$ **To** N
$sel[k] \leftarrow false$;
For $k \leftarrow 1$ **To** K
begin
$min \leftarrow (n-1) * (n-2)/2$;
For $l \leftarrow 1$ **To** N
If $l \neq i$ *and* $l \neq j$ *and* $sel[l] = false$ *and* $count[l] <= min$ **Then**
begin
$min_l \leftarrow l$;
$min \leftarrow count[l]$;
end
$usedb[pair][l] \leftarrow min_k$;
$count[min_k] \leftarrow count[min_k] + 1$;
$sel[min_k] \leftarrow true$;
end
$pair \leftarrow pair + 1$;
end
End

An experiment was setup in order to visualize the complexity of the proposed approach with respect to the number of databases, using both the original algorithm as well as the extra algorithm for assigning pairs to databases, for K equal to 1 and 2. Firstly, each of the 9 *Synthetic* databases with 10000 tuples was split into 8 equal parts, resulting into 72 databases with 1250 tuples. Then, distance calculation was performed using 4 to 72 databases. Figure 3 shows a plot of the time in milliseconds needed to calculate the disagreement measure with respect to the number of databases used for the calculation.

The plot verifies the square complexity of the original approach. It further shows that using 1 or 2 databases for distance calculation of each pair reduces this complexity to super-linear as there is also the overhead of the assignment

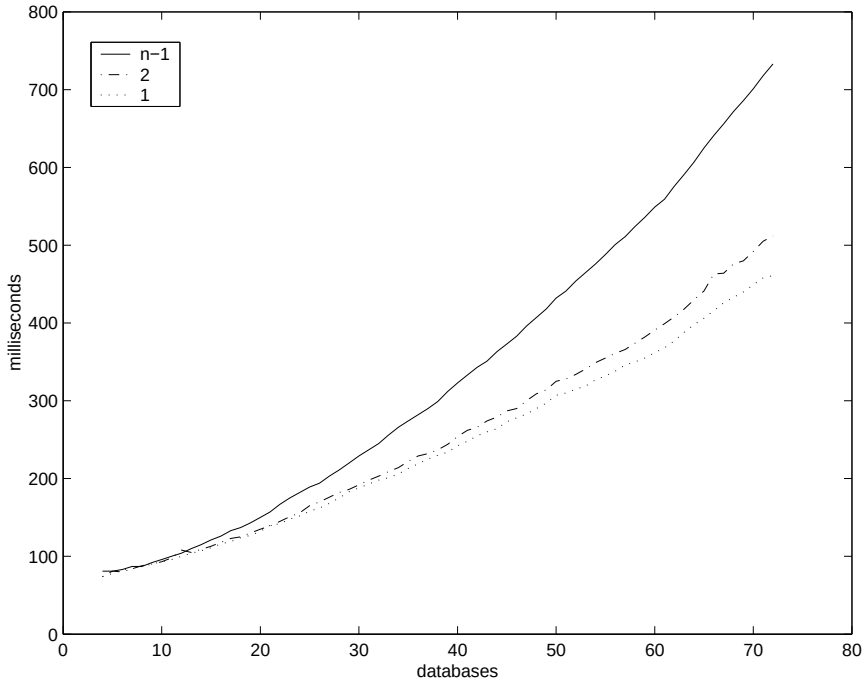

Fig. 3. Scalability with respect to databases

algorithm. It was also noticed that the clustering result remained correct for all numbers of participating databases even when using only one database for distance calculation. This reinforces the conclusion of the previous section that the disagreement measure is a robust metric for classifier distance that does not require a lot of data to be computed.

6 Conclusions and Future Work

This paper has presented a new approach for clustering classifiers induced from physically distributed relational databases. The proposed approach groups together classifiers with similar predictive behavior by measuring the disagreement of their output on a set of independent data.

The clustering of distributed classifiers enables the discovery of interesting similarities and differences between the respective databases. It can be used to detect the different classification concepts underlying several a collection of distributed databases, an important contribution to today's distributed classification approaches.

Furthermore, clustering a set of distributed classifiers increases the predictive performance that can be achieved by combining them. A classifier ensemble technique can be guided by the knowledge obtained from the clustering process in order to only combine classifiers of the same cluster. This way the combina-

tion becomes more effective and the accuracy results superior to conventional approaches that neglect clustering.

Applications of the proposed approach include detecting interesting similarities and differences in physically distributed scientific (medical networks, physics and astronomy data-grids) and business (banks, supermarkets) data and effectively combining classifiers induced from such distributed data sources.

6.1 Other Advantages

The proposed approach doesn't require moving raw data around the distributed nodes, an important constraint in mining inherently distributed data sources. All tuples from all databases get exploited for distance calculation leading to high quality of the distance measure, but no tuples are moved away from the databases. The only network traffic is the classification models and distance measures, which have negligible size.

The scalability of the proposed approach is efficient with respect to the number of tuples and a solution for the square complexity with respect to the number of databases has been discussed. Note also, that common applications of this approach deal with small collections of databases that allow the tractability of distance calculation within acceptable time.

The clustering algorithm is incremental and can handle the addition and deletion of data and complete databases. Firstly, when new data are added (or deleted) at a database, then there is no need to recalculate the distances based on the whole data, but only update them based on the batch of data that was added (or deleted). Secondly, when a classifier gets refined (e.g. due to many additions and deletions of data) at a database, then it is reimported by the rest of the databases and its distance with all other classifiers is recalculated. The distance of only $N-2$ pairs must be recalculated, while the rest are left unchanged. Thirdly, when a new database is added to the collection, it has to import all other classifiers and calculate the distance for all pairs of classifiers from scratch. However, the rest of the databases import the new classifier and calculate only the distance of the new $N-1$ pairs of classifiers. Finally, when a database is removed from the collection then just an update of the distance vector based on the local disagreement vector of that database occurs.

Finally, the proposed approach doesn't require transparent classifiers. This is important for applications involving different organizations that want to hide the details of their local models, but at the same time benefit from each other's knowledge. An application example in this area involves the cooperation of different bank organizations [16] for credit risk assessment. Data privacy requirements are also met by our method as there is no raw data exchange.

6.2 Future Work

An important limitation of the proposed and other approaches that perform learning from distributed data sources is the potential syntactic heterogeneity of the distributed databases. For example, the same attribute could have different names, or the same name but different values at the different distributed databases. This is an active research issue in the areas of cooperative information systems and multi-database systems.

Most solutions to schema integration are based on an architecture that features a common mediator among the distributed systems [25]. The mediator could either be a global schema that characterizes the distributed local schemas (structural approach), or a common ontology that contains all the necessary information for the integration of the local schemas (conceptual approach). As future work we intend to look into the employment of tools that support the conceptual approach, which is more promising and at the same time challenging. Such tools could be used in a pre-processing stage, before the application of the proposed approach.

Acknowledgements

The authors would like to thank the anonymous referees for their valuable comments and suggestions on this work.

References

[1] Catherine L. Blake and Christopher J. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[2] Michael Borth. Learning from multiple bayesian networks for revision and refinement of expert systems. In M. Jarke, J. Koehler, and G. Lakemeyer, editors, *Proceedings of the 25th Annual German Conference on AI*, pages 82–96, Aachen, Germany, September 2002.

[3] Philip Chan and Salvatore Stolfo. Meta-learning for multistrategy and parallel learning. In *Proceedings of the Second International Workshop on Multistrategy Learning*, 1993.

[4] Winton Davies and Pete Edwards. Using instance selection to combine multiple models learned from disjoint subsets. In H. Liu and H. Motoda, editors, *Instance Selection and Construction for Data Mining*. Kluwer Scientific Publishers, February 2001.

[5] Lawrence Hall, Nitesh V. Chawla, and Kevin W. Bowyer. Decision tree learning on very large data sets. In *Proceedings of the IEEE SMC Conference*, pages 2579–2584, San Diego, California, 1998.

[6] David Hand, Heikki Mannila, and Padraic Smyth. *Principles of Data Mining*. MIT Press, 2001.

[7] The MathWorks Inc. *MATLAB Reference Guide*. Academic Press, 1993.

[8] Leonard Kaufmann and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Interscience, 1990.

[9] Fumitaka Kimura and Malayappan Shridhar. Handwritten numerical recognition based on multiple algorithms. *Pattern Recognition*, 24(10):969–983, 1991.

[10] Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–238, March 1998.

[11] Wojtek J. Krzanowski. *Priniciples of Multivariate Analysis: A user's perspective*. Oxford Science Publications, 1993.

[12] Louisa Lam and Ching Y. Shen. Optimal combinations of pattern classifiers. *Pattern Recognition Letters*, 16:945–954, 1995.

[13] Sally McClean, Bryan Scotney, Kieran Greer, and Ronan Pairceir. Conceptual clustering of heterogeneous databases. In *Proceedings of the PKDD 2001 Workshop on Ubiquitous Data Mining for Mobile and Distributed Environments*, pages 46–55, 2001.

[14] Tom M. Mitchell. *Machine Learning*. McGRAW HILL, 1997.

[15] Srinivasan Parthasarathy and Mitsunori Ogihara. Clustering distributed homogeneous datasets. In D.A. Zighed, J. Komorowski, and J. Zytkow, editors, *PKDD 2000, LNAI 1910*, pages 566–574. Springer-Verlag, 2000.

[16] Andreas Prodromidis, Philip Chan, and Salvatore Stolfo. Meta-learning in distributed data mining systems: Issues and approaches. In *Advances in Distributed and Parallel Knowledge Discovery*. MIT Press, 2000.

[17] Foster Provost. Distributed data mining: Scaling up and beyond. In *Advances in Distributed and Parallel Knowledge Discovery*. MIT Press, 2000.

[18] Ross J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, San Mateo, 1993.

[19] Lorenza Saitta. Machine learning: A technological roadmap. Technical report, University of Amsterdam, 2000.

[20] Paul D. Scott and E. Wilkins. Evaluating data mining procedures: techniques for generating artificial data sets. *Information and Software Technology*, 41:579–587, 1999.

[21] Catherine A. Shipp and Ludmila I. Kuncheva. Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion*, 3(2):135–148, 2002.

[22] Grigorios Tsoumakas and Ioannis Vlahavas. Effective stacking of distributed classifiers. In *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 340–344, 2002.

[23] Peter D. Turney. Exploiting context when learning to classify. In *Machine Learning: ECML-93, European Conference on Machine Learning, Proceedings*, volume 667, pages 402–407. Springer-Verlag, 1993.

[24] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

[25] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.

[26] Ruediger Wirth, Michael Borth, and Jochen Hipp. When distribution is part of the semantics: A new problem class for distributed knowledge discovery. In *Proceedings of the PKDD 2001 Workshop on Ubiquitous Data Mining for Mobile and Distributed Environments*, pages 56–64, 2001.

[27] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

[28] David Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.