A Framework for Unsupervised Learning With Multiple Criteria

Bill Andreopoulos

Aijun An

Xiaogang Wang

Technical Report CS-2005-10

May 2005

Department of Computer Science and Engineering

4700 Keele Street North York, Ontario M3J 1P3 Canada

A Framework for Unsupervised Learning with Multiple Criteria

Bill Andreopoulos[1], Aijun An[1], and Xiaogang Wang[2]

[1]Department of Computer Science and Engineering, York University, 4700 Keele Street, Toronto, M3J1P3, Ontario, Canada, {billa,aan}@cs.yorku.ca
[2]Department of Mathematics and Statistics, York University, 4700 Keele Street, Toronto, M3J1P3, Ontario, Canada, stevenw@mathstat.yorku.ca

Abstract. Unsupervised learning algorithms that have been presented in the literature so far, rarely incorporate more than one criterion for learning. Moreover, the results from learning based on old criteria are not incorporated into the process of learning based on new criteria.

This paper presents a framework for unsupervised learning called *Doctris* "Double Criteria Triple Step". Learning algorithms based on Doctris use two different criteria at two levels of learning. The first level forms a prior for the second level, simulating a Bayesian process. We refer to this process as *multi-criteria unsupervised* learning. We justify Doctris with the Bayesian theory of classification. We present a metric for evaluating the results. We discuss extending Doctris to use more than two criteria. We describe the BILCOM clustering algorithm as an example of Doctris. Experimental results from clustering mixed data types with BILCOM indicate that it partitions mixed data accurately.

1 Introduction

Unsupervised learning, also called clustering, aims to partition a set of objects into classes without learning from a priori classifications of objects. Objects with similar characteristics are assigned to the same class and different classes contain objects with dissimilar characteristics. Clustering algorithms presented in the literature are often not based on a theoretically justified framework [3-6]. We define the *Doctris* "Double Criteria Triple Step" framework for a family of algorithms. Doctris algorithms classify objects at two levels, using a different criterion at each level.

An object *o* has *m* attributes $\{o_1,\ldots,o_m\}$ and each attribute o_i, $i=1..m$, has a value taken from a set of possible values $S=\{s_1,\ldots,s_y\}$. *S* is referred to as a domain and its values are of a specific data type, such as categorical or numerical data types. *S* might have an ordering defined for its values, as would be true for a domain of numerical data type. Alternatively, the values of *S* might have no ordering defined, as would be true for a domain of categorical data type. The values of *S* might be continuous or discrete. A domain of categorical data type is SEX with the discrete values M or F. A domain of numerical data type is GPA with continuous values in the range 0-4.

A *criterion* is a set of objects' attributes with values taken from a specific domain. Classification is based on the criteria and the objects' values. In the case of classification at two levels, the two criteria used have values taken from different domains. All objects to be classified, initially possess two different criteria *A* and *B*. Selected objects are classified at the first level using criterion *A* and the other objects are classified at the second level using criterion *B*. The criteria are likely to contain knowledge about the correct classifications of objects. The results of the first level are used as a prior for classifying objects at the second level, simulating a Bayesian process. We refer to this process as *multi-criteria unsupervised* learning.

The Doctris framework can be extended to more than two levels, implying that new levels of classified objects are added to the results of previous classifications in an ongoing learning process. A new level is created when a new set of objects is introduced to the classification process, having attribute values of a new criterion *Z*.

Doctris algorithms have the following general characteristics:

a. Many criteria are used in the process, one criterion at each level of classification.
b. Objects are classified considering the result of previous levels of classified objects that used different criteria.
c. Rules guide the process on the basis of Bayesian theory of classification to improve the results.
d. Classes can be refined and changed as new criteria are introduced at new levels.

This paper is organized as follows. Section 2 gives an overview of previous related work. Section 3 presents the Doctris framework for two criteria, its Bayesian rationale and a metric for evaluating the quality of the results. Section 4 proposes an extension of the Doctris framework for more than two criteria and levels. Section 5 describes a Doctris algorithm, called *BILCOM*, for "Bi-Level Clustering of Mixed categorical and numerical data types". Finally, Section 6 concludes the paper.

2 Related Work

A few algorithms have been proposed for clustering mixed categorical and numerical data types. *AutoClass* can cluster categorical as well as numerical data [10]. It does not require the user to specify the number of clusters beforehand. AutoClass uses a Bayesian method for determining the optimal classes. AutoClass takes a prior distribution of each attribute in each cluster, symbolizing the prior beliefs of the user. It changes the classifications of items in clusters and changes the means and variances of the distributions, until the means and variances stabilize.

k-Modes is a clustering algorithm that deals with categorical data [8,9]. The k-Modes clustering algorithm requires the user to specify the number of clusters to be produced and the algorithm builds and refines the specified number of clusters. Each cluster has a mode associated with it. Assuming that the objects in the data set are described by m categorical attributes, the mode of a cluster is a vector $Q=\{q_1, q_2, \ldots, q_m\}$ where q_i is the most frequent value for the ith attribute in the cluster of objects. A similarity metric is needed to choose the closest cluster to an object by computing the similarity between the cluster's mode and the object. Let $X=\{x_1, x_2, \ldots, x_m\}$ be an object, where x_i is the value for the ith attribute. The similarity between X and Q is defined as:

$$\text{similarity}(X, Q) = \sum_{i=1}^{m} \sigma(x_i, q_i) \qquad \text{where } \sigma(x_i, q_i) = \begin{cases} 1 & (x_i = q_i); \\ 0 & (x_i \neq q_i). \end{cases}$$

An extension of k-Modes called *k-Prototypes* was proposed in [7] to deal with mixed numerical and categorical data. K-Prototypes also adopts an iterative approach to clustering that continues until objects stop changing clusters.

ROCK is a hierarchical clustering algorithm for categorical data [14]. ROCK assumes a similarity measure between tuples and defines a link between two tuples whose similarity exceeds a threshold w. Initially, each tuple is assigned to a separate cluster and then clusters are merged repeatedly according to the closeness between clusters. The closeness between clusters is defined as the sum of the number of "links" between all pairs of tuples, where the number of "links" represents the number of common neighbors between two clusters.

Supervised learning and *Support Vector Machines* classify objects based on prior knowledge. Supervised learning draws a boundary separating classes, based on a training data set of labeled objects. Future unlabelled objects are classified on one side of the boundary. References are Burges' tutorial [11] and Vladimir Vapnik's book [12].

3 The Doctris Learning Framework

The *Doctris "Double Criteria Triple Step"* framework defines a family of classification algorithms. Doctris assumes that two different criteria exist and will be used at different levels of the classification process, such that the result of the first level provides a prior for the second level. The criteria are sets of objects' attributes with values taken from different domains, that could be of different data types such as numerical, categorical or boolean.

1. Two criteria A and B exist. For example, similarity between objects could be computed on the basis of numerical and categorical data type values.
2. The input is a set of N objects $D = \{D_1 \ldots D_N\}$.
3. The output is a set of X classes $C = \{C_1 \ldots C_X\}$ that are learnt from D, such that $X \leq N$.
4. The process of learning C from D consists of 3 steps:

First level

Step 1. On the basis of criterion A, form a number of X classes $C = \{C_1 \ldots C_X\}$, that are most likely to have as few misclassified objects as possible. Only M of the objects in D are classified, where $X \leq M \leq N$. Using more objects than M would increase the likelihood of misclassified objects in C.

Second level

Step 2. On the basis of criterion B, form a number of M subclasses $d = \{d_1 \ldots d_M\}$, by matching each object in the set $u=\{u_1 \ldots u_{N-M}\}$ of N-M objects not classified in Step 1, to the closest one of the M objects classified in Step 1. Each subclass d_i, $i=1 \ldots M$, consists of one of the M objects classified in Step 1 and zero or more of the objects in u not classified in Step 1.

Step 3. On the basis of both criteria A and B, selectively merge the M subclasses $d = \{d_1 \ldots d_M\}$ to one another, until a set of X classes $C = \{C_1 \ldots C_X\}$ emerges.

Fig. 1. The Doctris framework for multi-criteria unsupervised learning.

Figure 1 presents the general steps that define the Doctris framework. As shown, classification is performed on the basis of two criteria *A* and *B*. The first level consists of Step 1 that is based on criterion *A*. The second level consists of Steps 2 and 3 that are based on criteria *A* and *B*. When learning in Step 1 on the basis of criterion *A*, the likelihood of misclassification will increase to an unacceptable level after a number of *M* objects have been classified. A clustering algorithm for which this assumption holds is the MULIC clustering algorithm [2]. After *M* objects have been classified in Step 1, the unclassified *N-M* objects are classified in Steps 2 and 3 based on both of the criteria *A* and *B*. In Step 2, each one of the remaining *N-M* objects is matched to the closest of the *M* objects based on criterion *B*, thus resulting in *M* subclasses. In Step 3, the subclasses resulting from Step 2 are selectively merged to one another, based on both criteria *A* and *B*, until *X* classes emerge from the process. This sequence of steps simulates a Bayesian process described in Section 3.1, where the result of Step 1 is the prior for Steps 2 and 3.

The general steps that define the Doctris framework serve the ultimate purpose of enhancing the classification process to produce more accurate results. The process will be significantly improved if certain rules are satisfied. In the descriptions that follow $lmcAandB[M_A]$ is the "likelihood of misclassification of objects based on criteria *A* and *B*, after *M* objects have been classified based on criterion *A*". $lmcA[M_A]$ is the "likelihood of misclassification of objects based on criterion *A*, after *M* objects have been classified based on criterion *A*".

In order for a Doctris algorithm to produce more accurate results than Step 1 of the framework would produce alone based on criterion *A* only, the following inequality needs to be satisfied:

$$lmcAandB[M_A] \times \frac{(N-M)+(M-X)}{N} < lmcA[M_A] \times \frac{N-M}{N}$$
$$\Leftrightarrow lmcAandB[M_A] \times (N-X) < lmcA[M_A] \times (N-M)$$
$$\Leftrightarrow lmcAandB[M_A] < lmcA[M_A] \times \frac{(N-M)}{(N-X)} \qquad \textbf{(1)}$$

Inequality (1) states that from the total number of objects $(N\text{-}M)+(M\text{-}X)=N\text{-}X$ whose classification may change in Steps 2 and 3 based on criteria *A* and *B*, fewer objects should be likely to be misclassified, than if using only criterion *A* in Step 1 to classify the remaining *N-M* objects. *(N-M)* is the number of objects that are matched to a subclass in Step 2, while *(M-X)* is the number of merges between subclasses that may occur in Step 3. The products estimate the number of objects that are likely to be misclassified, based on the likelihoods of misclassification.

As *M* increases, the term $lmcAandB[M_A]$ on the left-hand side of (1) remains stable because we are assuming both criteria *A* and *B* will be used and any objects misclassified in Step 1 will be given a second chance to be classified correctly during Steps 2 and 3. We would like to estimate the value of *M* such that the left-hand side of (1) is lower than the right-hand side of (1) and the difference between the left-hand side and the right-hand side is maximized. We would like to use the value of *M* such that the following ratio is maximized:

$$\frac{lmcA[M_A] \times (N-M) / (N-X)}{lmcAandB[M_A]}$$

As *M* increases the term $lmcA[M_A]$ increases, at constant or varying rates for different values of *M*. However, as *M* increases the ratio *(N-M)/(N-X)* decreases at a constant rate since *N* and *X* are constants; since $X \le M \le N$, this ratio is in the range 0.0 to 1.0. A maximal product of the terms on the right-hand side of (1) can be estimated fairly easily. For example, 0.9×0.1 and 0.1×0.9 return 0.09, but 0.5×0.5 returns 0.25. A conservative approach would be to classify few objects in Step 1 such that *(N-M)/(N-X)* is high while $lmcA[M_A]$ remains relatively low. A bolder approach would be to classify many objects in Step 1 such that *(N-M)/(N-X)* is low while $lmcA[M_A]$ increases as much as possible.

Figures 2 and 3 show the graphs for two cases of the products of $lmcA[M_A]$ and *(N-M)/(N-X)*, for $X=10$ and $N=100$. *M* is represented by the horizontal axis. The term *(N-M)/(N-X)* has a constant decrease rate. In the first case, $lmcA[M_A]$ increases at a rapid rate with *M*. The maximal product value of the two terms is at $M=35$. In the second case, $lmcA[M_A]$ increases at a lower rate with *M*. The maximal product value of the two terms is at $M=25$.

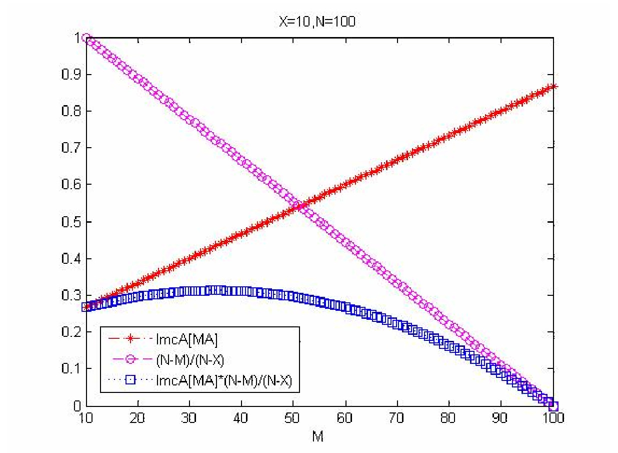

Fig. 2. This graph shows how *lmcA[M_A]* increases at a *high* rate, while *(N-M)/(N-X)* decreases at a constant rate. Number of clusters *X* is 10 and number of objects *N* is 100. *M* ranges between *X* and *N* as represented by the horizontal axis. The product value is maximized at an *M* value of 35.

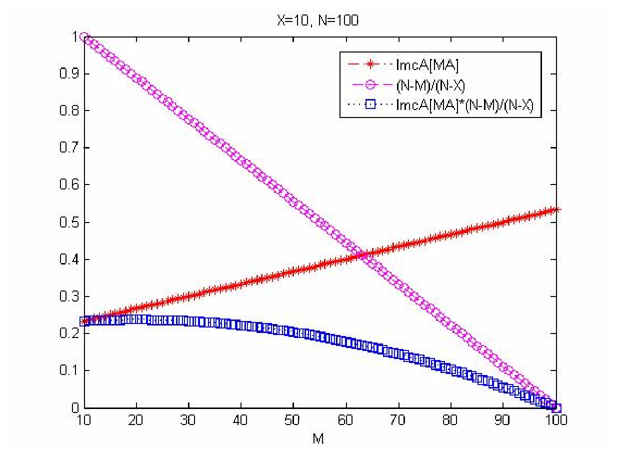

Fig. 3. This graph shows how *lmcA[M_A]* increases at a *lower* rate, while *(N-M)/(N-X)* decreases at a constant rate. Number of clusters *X* is 10 and number of objects *N* is 100. *M* ranges between *X* and *N* as represented by the horizontal axis. The product value is maximized at an *M* value of 25.

3.1 Bayesian Rationale for the Doctris Framework

We justify Doctris with the Bayesian theory of classification. The idea of Bayesian classification is to find for each data object, the classification *H* for which the following Bayesian equation gives the maximum result:

$$\pi(H \mid E) = \frac{\pi(E \mid H)\pi(H)}{\pi(E)} \qquad \textbf{(2)}$$

π(H) is the prior probability of hypothesis *H*. *π(E)* is the prior probability that some evidence *E* is observed on the data object x. *π(E|H)* is the likelihood of *E* given *H*. *π(H|E)* is the posterior probability of *H* given *E*. We seek the hypothesis *H* such that the numerator *π(E|H)π(H)* of the Bayesian equation is maximized. In other words, we seek the classification *H* of each object for which this numerator returns the maximum value. This gives the most probable classification *H* for an object.

Linking our framework to Bayesian theory of classification, the prior *E* for an object is the classification of the object in a class in Step 1, based on criterion *A*. The hypothesis *H* for an object is the classification of the object in a class in Steps 2 and 3, based on criteria *A* and *B*. For example, the criterion *A* could be attributes of a categorical domain that represent our knowledge about the object or our observations before an experiment takes place. The criterion *B* could be attributes of a numerical domain that represent the results of an experiment.

In this framework, *π(E)* is a constant for each object that depends on the likelihood that the classification *E* of an object in a class in Step 1 is correct. This likelihood increases with the strength of the similarity of the object to the class to which it is assigned in Step 1 according to criterion *A*. Since *π(E)* is a constant for each object and it does not affect the choice between classifications *H* we do not use *π(E)* in our calculations. In traditional Bayesian clustering algorithms, such as AutoClass, the Evidence *E* used in the Bayesian equation is given by the categorical or numerical attribute values of each object. In such traditional Bayesian clustering algorithms, the denominator *π(E)* is often not considered in the process of finding the best classification. The term *π(H)* is often not considered either, leaving the most important term to be *π(E|H)*.

In the following descriptions the term $similarity_{AB}(o,classStepx_o)$ is the similarity of object *o* to the class in which *o* is classified in Step *x*, according to the criteria *A* and *B*. This similarity can be computed using various algorithms, such as ROCK's similarity metric for numerical or categorical attribute value types [14].

Linking the Doctris framework to the Bayesian theory of classification, *π(H)* is the likelihood that the classification of an object in a class in Steps 2 and 3 is correct – meaning that *H* is likely to be true. *π(H)* increases with the strength of the similarity of the object to the subclass to which it is assigned in Step 2 according to criterion

B. Since in Step 3 the subclasses are selectively merged to one another to form classes, to find *π(H)* we are interested both in the similarity of the object to its Step 2 subclass according to criterion *B*, as well as the similarity of that subclass to the subclass to which it gets merged in Step 3 according to criteria *A* and *B*. Thus, we simulate maximizing *π(H)* by seeking the classification *H* for an object *o* such that the following term is maximized:

$$\max(\pi(\mathrm{H})) =$$
$$\max(similarity_{AB}(o, classStep3_o)) =$$
$$\max(similarity_{\mathrm{B}}(o, subclassStep2_o) \times similarity_{A\mathrm{B}}(subclassStep2_o, classStep3_o)) \qquad \textbf{(3)}$$

Maximizing *π(E|H)* assists our understanding of (3) above. Linking the Doctris framework to the Bayesian theory of classification, *π(E|H)* is the likelihood that the classification of an object in a class in Step 1 is correct, meaning that *E* is likely to be true, given that the object is classified in Steps 2 and 3 in the class represented by *H*. *π(E|H)* increases with the strength of the similarity of the object to the class to which it was assigned in Step 1 according to criterion *A*, as represented by *E* and *π(E)*. *π(E|H)* also increases if the Step 1 class for the object, represented by *E,* is the same one as the class to which the object is assigned in Steps 2 and 3 based on criteria *A* and *B*, represented by *H*. It is obvious that maximizing *π(E|H)* affects (3) above, in the sense that the $classStep3_o$ that previously yielded the highest similarity for *o* according to criteria *A* and *B* might not necessarily be the best choice. Instead, *o* might need to be assigned back to the class in which *o* was classified in Step 1, if the similarity of *o* to this class according to criteria *A* and *B* suggests this is a better choice. In our final decision on which class to assign *o* to, the similarity according to criteria *A* and *B* needs to be computed between *o* and the classes to which it was assigned in Steps 1 and 3, since both criteria are likely to contain knowledge about the correct classification of *o*. Thus, we simulate maximizing *π(E|H)* by selecting between the Step 1 or Step 3 classification *H* for an object *o*:

$$\max(\pi(E \mid H)) =$$
$$\max(similarity_{AB}(o, classFinal_o)) =$$
$$\max(\max(\pi(H)), similarity_{AB}(o, classStep1_o)) =$$
$$\max(\max(similarity_{AB}(o, classStep3_o)), similarity_{AB}(o, classStep1_o)) \qquad \textbf{(4)}$$

Objects with low similarity to the closest Step 1 class according to criterion *A* are not classified in Step 1, thus ignoring for these objects the term $similarity_{AB}(o,classStep1_o)$. Such an object *o* will produce a higher value for $max(similarity_{AB}(o,classStep3_o))$ than $similarity_{AB}(o,classStep1_o)$ from (4), since Steps 2 and 3 will classify *o* based on both criteria *A* and *B*. Step 2 will assign *o* to the closest subclass based on criterion *B* and Step 3 will merge this subclass to the closest class based on criteria *A* and *B*, thus simulating maximizing *π(H)* and $similarity_{AB}(o,classStep3_o)$. Furthermore, not classifying these objects in Step 1 reduces the total computation time.

The above similarity terms are defined below, where $metric_Z(x,y)$ represents a metric based on criterion *Z* for estimating the similarity between objects *x* and *y*, returning a value in the range 0.0 to 1.0 for low and high similarity between *x* and *y* respectively. For example, this metric could be the Euclidean distance for numerical attributes, or the modes-based similarity for categorical attributes as defined by Huang et al. [8,9]:

$\mathbf{similarity_B(o,subclassStep2_o)}$ can be estimated using $metric_B(o,y)$ where *y* is a representative object for $subclassStep2_o$.

$\mathbf{similarity_{AB}(o,classStep1_o)}$ can be estimated using $\alpha\times metric_A(o,y)+\beta\times metric_B(o,y)$, where *y* is a representative object for $classStep1_o$ and *α* and *β* are weights in the range 0.0 to 1.0, such that $\alpha+\beta=1.0$.

$\mathbf{similarity_{AB}(subclassStep2_o,classStep3_o)}$ can be estimated using:

$$\frac{\sum_{x \in subclassStep2_o, y \in classStep3_o} \alpha \times metric_A(x, y) + \beta \times metric_B(x, y)}{size(subclassStep2_o) \times size(classStep3_o)}$$

A Doctris algorithm takes into consideration all of these probabilities. The purpose is to classify each object in the class represented by *H* that maximizes the probability for the numerator *π(E|H)π(H)* of the Bayesian equation.

3.2 An Evaluation Metric

The similarity metric of the Doctris framework can be adopted as a metric for evaluating the quality of the results. This would involve using *similarity$_{AB}$(o,classFinal$_o$)* from (4) to calculate the average similarity of all objects *o* in the data set *D* to their respective classes. This evaluation metric can be described as follows:

$$Quality = \sum_{o \in D} similarity_{AB}(o, classFinal_o) \big/ size(D) \qquad \textbf{(5)}$$

4 Extension of the Doctris Framework

The Doctris framework can be extended to more than two levels and criteria. New objects with a new criterion *Z* may be presented to the classification process, after object classification has been done using criteria *A…Y*. Criterion *Z's* attribute values might be of the same or different domains as the previous criteria *A…Y*. For example, *Z* might be numerical while the previous criteria were categorical. In either case, criterion *Z* is presented to the classification process at a different time point from criteria *A…Y*. The new objects possess the previous criteria *A...Y* as well as the new criterion *Z*. *M* is the number of objects that were previously classified using criteria *A…Y* and *N* is the total number of objects including the new objects with criterion *Z*. Inequality (6) holds for the general case:

$$lmcAandB...YandZ[M_{A...Y}] \times \frac{(N-M)+(M-X)}{N} < lmcAand...andY[M_{A...Y}] \times \frac{N-M}{N} \qquad \textbf{(6)}$$

The new objects are added to one of the Step 2 subclasses based on the new criterion *Z*, as shown in Step 2 of the Doctris framework. Then Step 3 is repeated based on all criteria *A...Z*, so that the subclasses from Step 2 are refined. For Step 1 there exist two options:

a. Step 1 may not be repeated each time new objects are presented, in which case the Step 1 results remain stable throughout the classification process based on the initial criterion *A*. Thus, the Step 1 results can serve as a constant basis for the future classification process.

b. Step 1 may be repeated based on the criteria *A…Y*, when new objects are presented with a new criterion *Z*. Thus, the basis of the classification process could change when new objects are presented. However, this is time consuming and inefficient for classification.

After a series of objects with new criteria have been presented to the algorithm, the size of the subclasses formed in Step 2 will increase beyond an acceptable level. In this case, option *b* described above should be executed, to decrease the size of the Step 2 subclasses and increase the accuracy of the results.

5 BILCOM Clustering as an Instance of the Doctris Framework

The *BILCOM "Bi-Level Clustering of Mixed categorical and numerical data types*" algorithm performs clustering at two levels, where the first level clustering acts as a prior for the second level, thus simulating a Bayesian process, as described in Section 3.1. Similarity based on categorical attribute values (CAs) is emphasized at the first level and similarity based on numerical attribute values (NAs) at the second level. The first level result is given as input to the second level and the second level result is the output of the BILCOM process. Figure 4 shows an example, where the first level and second level involve four clusters. The second level clusters consist of subclusters. Object *A* is assigned to different first level and second level clusters, because the numerical similarity at the second level is stronger than the categorical similarity at the first level. Thus, the following relationship holds for object *A*:

$$\begin{aligned} & categorical\_similarity(A, cluster2) + \\ & numerical\_similarity(A, cluster2) > \\ & categorical\_similarity(A, cluster3) + \\ & numerical\_similarity(A, cluster3) \end{aligned}$$

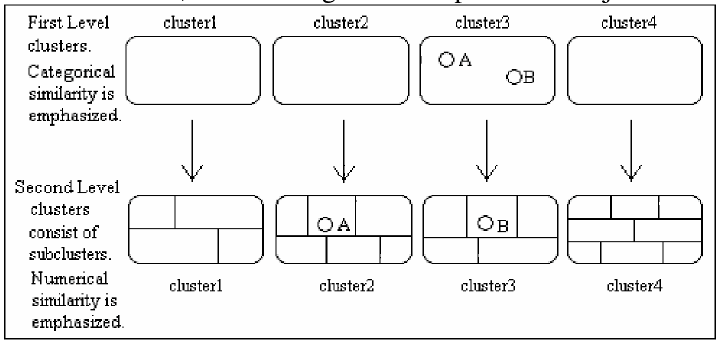
First Level clusters. Categorical similarity is emphasized.
cluster1
cluster2
cluster3
cluster4
A
B
Second Level clusters consist of subclusters. Numerical similarity is emphasized.
A
B
cluster1
cluster2
cluster3
cluster4

Fig. 4. Overview of the BILCOM clustering process.

On the other hand, object *B* is assigned to the same clusters in the first and second levels, because both categorical and numerical similarities support this classification. Thus, BILCOM considers both categorical and numerical similarities of an object to the clusters to which it may be assigned.

Different types of data are used at the first and second levels. The numerical data represent experimental results involving the objects. For example, the numerical data used at the second level might look as follows: BILIRUBIN:0.39; ALBUMIN:2.1; PROTIME:10. The categorical data represent what was observed to be true about the objects, before the experiment. For example, the categorical data used at the first level might be existing knowledge on objects looking as follows: Class:LIVE; SEX:male; STEROID:yes; FATIGUE:no; ANOREXIA:no. BILCOM clustering has the following characteristics:

a. The coherence of each cluster is maximized, considering both numerical and categorical similarity of the objects.
b. Only the objects with highest categorical similarity to a cluster form the basis for clustering at the first level.
c. The results of the first level clustering, which is the prior for the process, do not exert an overly strong effect on the second level, so that the second level clustering can escape a poor prior.
d. The number of clusters to be formed does not need to be specified by the user beforehand.

5.1 Design of BILCOM

This section describes the first level and second level algorithms that form the BILCOM process, shown in Figure 4. The first level is the MULIC[1] categorical clustering algorithm [2] and it clusters only a subset of the data set objects. The reason MULIC was chosen for the first level is that it creates multiple layers for each cluster and objects in top layers are more likely to be clustered correctly than objects in bottom layers. Thus, it is easy to select the objects in the top layers of MULIC clusters, as the most reliable classifications for the first level of BILCOM.

5.1.1 First level clustering

At the first level, clustering is performed using MULIC [2], an extension of the k-Modes clustering algorithm for categorical data sets that was discussed in Section 2 [8,9]. The k-Modes clustering algorithm requires the user to specify the number of clusters to be produced and the algorithm builds and refines the clusters. Each cluster has a mode associated with it. Assuming that the objects in the data set are described by *m* categorical attributes, the mode of a cluster is a vector $Q=\{q_1, q_2, \ldots, q_m\}$ where q_i is the most frequent value for the *i*th attribute in the given cluster.

MULIC makes substantial changes to the k-Modes algorithm. The purpose of the MULIC clustering algorithm is to maximize the similarity between the object and the mode of the cluster in which the object is inserted:

$$similarity(o_i, \text{mode}_i) \qquad \textbf{(7)}$$

where o_i is the *i*th object in the data set and mode_i is the mode of the *i*th object's cluster. The similarity metric is the k-Modes similarity, described in Section 2, which returns the number of identical CAs between an object and a mode. Maximizing (7) ensures that all objects are as similar to their clusters' modes as possible when they are clustered.

The MULIC algorithm has the following characteristics. First, the number of clusters is not specified by the user. Clusters are formed, removed or merged during the clustering process, as the need arises. Second, a cluster *c* must contain at least two objects, otherwise, it is not a cluster. Third, it must be possible for all objects to be inserted in clusters by the end of the clustering process.

Figure 5 shows the main part of the MULIC clustering algorithm. The algorithm starts by reading all objects from the input file and storing them in *S*. The first object is inserted in a new cluster, the object becomes the mode of the cluster and the object is removed from *S*. Then, it continues iterating over all objects that have not been assigned to clusters yet, to find the closest cluster. In all iterations, the closest cluster for each unclassified object is the cluster with the highest similarity between the cluster's mode and the object, as computed by the similarity metric [2,8].

The variable φ is maintained to indicate how strong the similarity has to be between an object and the closest cluster's mode for the object to be inserted in the cluster – initially φ equals 0, meaning that the similarity has to be very strong between an object and the closest cluster's mode. If the number of different CAs between the object and the closest cluster's mode is greater than φ then the object is inserted in a new cluster on its own, else, the object is inserted in the closest cluster and the mode is updated.

[1] http://www.cs.yorku.ca/~billa/MULIC/

```
Input: (1) a set $S$ of objects;
Parameters: (1) $\delta\varphi$ : the increment for $\varphi$;
            (2) threshold for $\varphi$ : the maximum number
                of values that can differ between an
                object and the mode of its cluster;
Default parameter values: (1) $\delta\varphi = 1$;
                          (2) threshold = the number of
                              categorical attributes $m$;
Output: a set of clusters;
Method:
  1. Insert the first object into a new cluster, use the
     object as the mode of the cluster, and remove the
     object from $S$;
  2. Initialize $\varphi$ to 0;
  3. Loop through the following until S is empty or $\varphi$ is
     greater than the specified threshold
     a. For each object $o$ in $S$
        i.   Find $o$'s closest cluster $c$ by using the
             similarity metric to compare $o$ with the
             modes of all existing cluster(s);
        ii.  If the number of different values between $o$
             and $c$'s mode is larger than $\varphi$, insert o into a
             new cluster
        iii. Otherwise, insert $o$ into $c$ and update $c$'s
             mode;
        iv.  Remove object $o$ from $S$;
     b. For each cluster $c$, if there is only one object in
        $c$, remove $c$ and put the object back in $S$;
     c. If in this iteration no objects were inserted in a
        cluster with size > 1, increment $\varphi$ by $\delta\varphi$.
```

Fig. 5. The MULIC clustering algorithm.

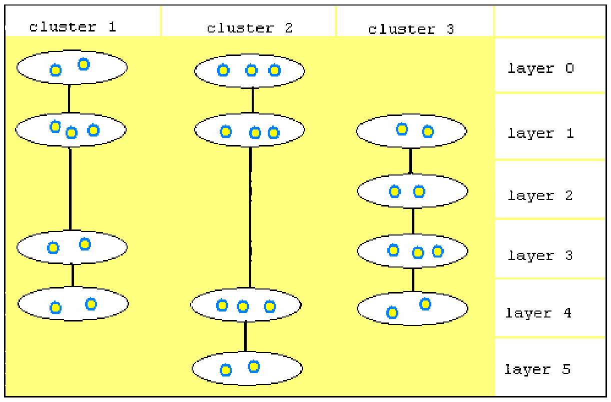

Fig. 6. MULIC results. Each cluster consists of one or more different layers representing different similarities of the objects attached to the cluster.

At the end of each iteration, all objects classified in clusters of size one have their clusters removed so that the objects will be re-clustered at the next iteration. This ensures that the clusters that persist through the process are only those containing at least 2 objects for which the required similarity can be found. Objects belonging to clusters with size greater than one are removed from the set of unclassified objects S, so those objects will not be re-clustered.

At the end of each iteration, if no objects have been inserted in clusters of size greater than one, then the variable φ is incremented by $\delta\varphi$. Thus, at the next iteration the criterion for inserting objects in clusters will be more flexible. The iterative process stops when all objects are classified in clusters of size greater than one, or φ exceeds a user-specified *threshold*. If the threshold equals its default value of the number of attributes m, the process stops when all objects are assigned to clusters of size greater than one.

The MULIC algorithm can eventually classify all objects in clusters, even if the closest cluster to an object is not that similar, because φ can continue increasing until all objects are classified. Even in the extreme case, where an object o with m attributes has only zero or one value similar to the mode of the closest cluster, it can still be classified when $\varphi = m$ *or* $m-1$ respectively.

Figure 6 illustrates what the results of MULIC look like. Each cluster consists of many different "layers" of objects. The layer of an object represents how strong the object's similarity was to the mode of the cluster when the object was assigned to the cluster. The cluster's layer in which an object is inserted depends on the value of φ. Lower layers have a lower coherence - meaning a lower average similarity between all pairs of objects in the layer - and correspond to higher values of φ. MULIC starts by inserting as many objects as possible in high layers – such as layer 0 or 1 - and then moves to lower layers, creating them as φ increases.

If an unclassified object has equal similarity to the modes of the two or more closest clusters, then the algorithm tries to resolve this 'tie' by comparing the object to the mode of the top layer of each of these clusters – the top layer of a cluster may be layer 0 or 1 or 2 and so on. Each cluster's top layer's mode was stored by MULIC when the cluster was created, so it does not need to be recomputed. If the object has equal similarity to the modes of the top layer of all of its closest clusters, the object is assigned to the cluster with the highest bottom layer. If all clusters have the same bottom layer then the object is assigned to the first cluster, since there is insufficient data for selecting the best cluster.

The runtime complexity of MULIC is *O(mN)*, where *m* represents the number of categorical attributes in an object and *N* represents the number of objects in the data set [2].

The question remains of how to select the data objects to be used at the first level. The first level objects are those whose comparison to the mode of the closest cluster by the similarity metric yields a result that is greater than or equal to a value *minimum_mode_similarity*. The rest of the objects are used at the second level. For this purpose, the user can specify a threshold for φ - a value of *m-minimum_mode_similarity*, where *m* is the number of categorical attributes in an object. When φ passes this value any remaining objects are considered at the second level instead. The reason only the objects whose similarity to the closest mode is greater than *minimum_mode_similarity* are clustered at the first level is because the objects that yield a low similarity to the closest mode are more likely to be inserted in a wrong cluster, as we showed in [1,2]. Thus, the objects whose classification in clusters based on categorical similarity is not reliable enough are clustered in the second level instead, where the numerical similarity of objects to clusters is more influential. We discussed what the value of *minimum_mode_similarity* should be in [1].

5.1.2 Second level clustering

The first level result is the input to the second level. The second level clusters all of the data set objects, including the objects clustered at the first level. The second level uses numerical data type similarity and the first level result as a prior. The second level clustering consists of 5 steps, whose rationale is to simulate maximizing the numerator of the Bayesian equation, as discussed in Section 3.1. The second level result is the output of the BILCOM process.

Step 1. One object in each first level cluster is set as a *seed*, while all the rest of the objects in the cluster are set as *centers*. The seed is an object that is at the top layer of the cluster – ideally in layer 0. The reason we choose for seed a top layer object is that the most influential objects at the second level should be those that have the minimum average distance to all other objects in the first level cluster. The MULIC paper [2] showed that objects at the top layer have a smaller average distance to all other cluster objects than lower level objects do.

If the top layer of a cluster is layer 0 then we have no difficulty in choosing the seed since all objects have the same CAs. If the top layer of a cluster is not layer 0 and it contains more than one object, then we choose the seed by comparing all top layer objects to the cluster's mode to find the closest object. If this does not resolve the ambiguity then we compare all top layer objects to the cluster's top layer mode – which was stored by MULIC when the cluster was created - to find the closest object. If all top layer objects have the same similarities to modes then we assign the seed to be the first top layer object, since there is insufficient information for choosing the best seed.

Step 2. Each *seed* and *center* is inserted in a new *second level subcluster*. The output of this step is a set of *subclusters*, referred to as *seed-containing* or *center-containing* subclusters, whose number equals the number of objects clustered at the first level.

Step 3. Each object that did not participate at the first level is inserted into the second level subcluster containing the most numerically similar *seed* or *center*. Numerical similarity for Steps 3-5 is determined by the *Pearson correlation coefficient* or the *Shrinkage-based similarity metric* introduced by Cherepinsky et al [15].

Step 4. Each center-containing subcluster is merged with its most numerically similar seed-containing subcluster. The most numerically similar seed-containing subcluster is found using our version of the ROCK goodness measure [14] that is evaluated between the center-containing subcluster in question and all seed-containing subclusters:

$$G(Ci, Cj) = \frac{link[Ci, Cj]}{size(Ci) \times size(Cj)}$$

where $link[C_i,C_j]$ stores the number of cross links between clusters C_i and C_j, by evaluating $\Sigma(p_q \in C_i,\ p_r \in C_j)\ link(p_q,p_r)$. $link(p_q,p_r)$ is a boolean value specifying whether a link exists between points p_q and p_r. It is computed by determining if the two points' numerical similarity is higher than a threshold *minimum_numerical_similarity* and if yes setting a link between them. The rationale for using a variation of ROCK's goodness measure for this step is that the link-based approach of ROCK adopts a global approach to the clustering problem, by capturing the global knowledge of neighboring data points between clusters. It has been shown to be more robust than methods that adopt a local approach to clustering, like hierarchical clustering [14].

Step 5. The loop below refines the step 4 subcluster merges. All variables take real values in the range 0.0-1.0.

```
repeat {
    for (each center-containing subcluster)
        if(numerical_similarity_of_center_subcluster_to_1st_level_seed_cluster ×
categorical_similarity_of_center_to_seed_of_1st_level_cluster >
numerical_similarity_of_center_subcluster_to_its_numerically_similar_2nd_level_cluster ×
categorical_similarity_of_center_to_seed_of_its_numerically_similar_2nd_level_cluster)
             merge center_subcluster to seed_subcluster from 1st_level;
} until (no center_subclusters change);
```

The variable:

`categorical_similarity_of_center_to_seed_of_1st_level_cluster`

represents the *categorical* similarity of the center *c* of a subcluster *C* to the seed *s*, such that *c* and *s* were in the same first level cluster.

The variable:

`categorical_similarity_of_center_to_seed_of_its_numerically_similar_2nd_level_cluster`

represents the *categorical* similarity of the center *c* of a subcluster *C* to the seed of C's most numerically similar seed-containing subcluster *N* determined in step 4. The categorical similarity is computed as follows:

$$\text{similarity(center, seed)} = \frac{\sum_{i=1}^{m} \sigma(\text{center}_i, \text{seed}_i)}{m}$$

where $\sigma(center_i, seed_i) = 1$ *if* $center_i = seed_i$ *, 0 otherwise.*

The variables:

`numerical_similarity_of_center_subcluster_to_1st_level_seed_cluster`
`numerical_similarity_of_center_subcluster_to_its_numerically_similar_2nd_level_cluster`

represent the *numerical* similarity of a subcluster *C* containing center *c* to the cluster containing seed *s*, such that *c* and *s* were in the same first level cluster, and to the cluster containing C's most numerically similar seed-containing subcluster *N* determined in step 4, respectively. These similarities include the subclusters that were merged to the clusters in previous iterations of the loop.

According to this loop, a subcluster *C* containing *center c* is attracted to the subcluster *S* containing *seed s*, such that *c* and *s* were in the same first level cluster. The attraction is stronger if there is high categorical similarity between *c* and *s* and lower if there is low categorical similarity between *c* and *s*. The subclusters *C* and *S* get merged if both the categorical similarity between *c* and *s* and numerical similarity between *C* and *S* are high enough. If *c* is not categorically similar enough to *s*, then, *C* should be likely to remain merged with its most numerically similar seed-containing subcluster *N* determined in step 4. Figure 7 shows steps 4 and 5.

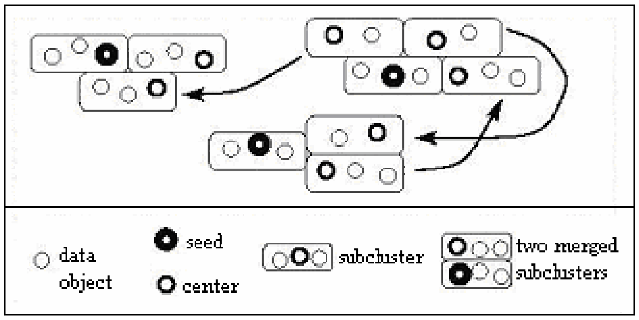

Fig. 7. Steps 4 and 5 of the second level of BILCOM clustering.

We have done tests to show that BILCOM is able to *escape a poor prior* – for instance, if a center *c* was inserted in a first level cluster with weak similarity to the cluster mode, or if the similarity to the mode was erroneously high enough, or if *c* had wrong CAs with low confidence to be correct. The categorical similarity between the center *c* and the seed *s*, such that *c* and *s* were in the same first level cluster, is likely to return a low value when the prior is poor. In this case, the subcluster *C* containing center *c* will be likely to remain merged with its most numerically similar seed-containing subcluster *N* determined in step 4, instead of the subcluster *S* containing the seed *s*. Thus, the prior can be escaped and the data can be clustered correctly. In this case, *C* will not be merged to *S*, unless their numerical similarity is very high.

On the other hand, if the subcluster *C* containing center *c* is merged to the subcluster *S* containing the seed *s*, such that *c* and *s* were in the same first level cluster, then *C* must be numerically similar enough to *S*. This way we ensure that if a subcluster *C* is merged to the subcluster *S* that is suggested by the results of the first level clustering, the numerical similarity between *C* and *S* is high enough to support the merging.

The reason why the inequality comparison in step 5 considers the seeds of clusters, instead of the cluster modes, is that by considering similarity to seeds we are effectively giving the objects a *second chance* to reorganize and to escape their first level clustering, if the first level clustering was weak. Since the first level clustering was based on comparisons to modes that often yield wrong results and, therefore, objects may be attached to wrong clusters, the comparison in step 5 allows the similarities to be reconsidered. We showed in [2] that objects in the top layers 0 and 1 such as seeds have a higher average similarity to all other cluster objects than do lower layer objects.

5.2 Experimental Evaluation of BILCOM

We evaluated BILCOM's accuracy on data sets of objects with mixed categorical and numerical attribute values (CAs and NAs) and showed that it is better than clustering the objects using either NAs or CAs only, or clustering mixed CAs and NAs with previous algorithms. Our detailed experiments, including yeast data set experiments, are described in [1]. Because of space limitations, here we discuss just the results for the *hepatitis* and *thyroid disease* data sets from the UCI Irvine Machine Learning Repository [13]. The hepatitis data set has 155 objects representing patients and each object has 13 CAs and 6 NAs. The thyroid disease data set has 3163 objects representing patients and each object has 12 CAs and 7 NAs. Furthermore, we know the class values of the objects of these data sets.

Our misclassification rate measure is the *classes to clusters evaluation* that is used by the clustering algorithms of the WEKA package. In this mode we first ignore the class attribute and generate the clusters. Then during the test phase we assign classes to the clusters, based on the majority value of the class attribute within each cluster. Then we compute the classification error, based on this assignment. Tables 1 and 2 show the clustering algorithms and their corresponding misclassification rates, for the hepatitis and thyroid disease data sets, respectively.

Table 1. Clustering algorithms and misclassification rates for the *hepatitis* data set

AutoClass for 2 clusters taking as input CAs and NAs	32/155 = 20.64%
k-Modes for 2 clusters taking as input CAs	32/155
k-Means for 2 clusters taking as input NAs	39/155 = 25.1613%
k-Means for 2 clusters taking as input CAs and NAs	41/155 = 26.4516%
MULIC taking as input CAs	20/155 = 12.9%
BILCOM taking as input CAs and NAs	15/155 = 9.67%

Table 2. Clustering algorithms and misclassification rates for *thyroid disease* data set

AutoClass for 2 clusters taking as input CAs and NAs	151 / 3163 = 4.77%
k-Modes for 2 clusters taking as input CAs	151 / 3163
k-Means for 2 clusters taking as input NAs	1093 / 3163 = 34.5558%
k-Means for 2 clusters taking as input CAs and NAs	1203 / 3163 = 38.0335%
MULIC taking as input CAs	145 / 3163 = 4.58%
BILCOM taking as input CAs and NAs	140 / 3163 = 4.42%

The results show that BILCOM taking as input both the CAs and NAs of the objects of the data sets produces a slightly lower misclassification rate than MULIC taking as input only the CAs. BILCOM and MULIC produce lower misclassifications than other algorithms taking as input both CAs and NAs, such as AutoClass and k-Means. They also have lower misclassification rates than k-Means taking as input NAs and k-Modes taking as input CAs. For k-Means and k-Modes the number of clusters is a parameter specified by the user. We specified a number of 2 clusters. However, the misclassification rate did not decrease as we increased the number of clusters to 4, 6 and 10.

6 Conclusion and Future Work

We have presented the Doctris learning framework and we have described the rationale of the Doctris framework on the basis of the Bayesian classification theory. We have defined a metric for evaluating the classification results. We have proposed extending Doctris to more than two criteria and levels. We have described an example of Doctris, the BILCOM clustering algorithm for mixed categorical and numerical data types.

Doctris algorithms are unsupervised learning, since they do not assign an object to a class based on a boundary between classes learnt from a sample set of labeled objects, as done in supervised learning. On the other hand, it could be argued that Doctris algorithms possess elements of supervised learning since different criteria are used in a bottom-up fashion. Lower-level criteria that may represent knowledge on the classification of objects create an underlying prior for higher-level classification. We refer to this novel process as *multi-criteria unsupervised* learning. Multi-criteria unsupervised learning has the potential to be an accurate and competent representation of human cognition, since the mind's classification of objects may change significantly as experience accumulates and as new instances become absorbed by the mind's conceptual framework. Multi-criteria unsupervised learning can potentially absorb new information as it is presented and incorporate it in the previous learning results.

This novel work opens new research directions in the fields of machine learning and empirical Bayesian theory. In the future we will show that Doctris algorithms can model a non-linear separator between two classes A and B, which is extremely desirable for classification. When new objects are presented and classified using new criteria, the values of the older criteria in the classes A and B could eventually have a non-linear separator between A and B.

In the future we will be formalizing the extension of the Doctris framework to more than two criteria and levels. We will present a detailed algorithm for the case of more than two levels that will be used to classify new objects with new criteria, as they are presented to the algorithm.

We will show that as the number of criteria presented to a Doctris algorithm increases, the misclassification rate ideally approaches zero. The condition for the misclassification rate to approach zero is that when new objects with a new criterion Z are presented, the likelihood of misclassification is lower than if using only the previous criteria $A \ldots Y$ for learning. Computational preprocessing of the data to decrease the likelihood of misclassification when new objects with a new criterion are presented can be performed if the new criterion has values taken from the same domain as previous criteria. For example, for domains of a numerical data type, a wavelet transformation could amplify the values in a newly presented object that occur in a previously formed class more frequently than other values, thus, increasing the effect of the values that should be more influential in determining the proper class. We will develop preprocessing methods that can be applied to decrease the likelihood of misclassification for newly presented objects.

References

[1] B. Andreopoulos, A. An and X. Wang. (2005) BILCOM: Bi-level Clustering of Mixed Categorical and Numerical Biological Data. Technical Report # CS-2005-01. Department of Computer Science and Engineering, York University.

[2] B. Andreopoulos, A. An and X. Wang. (2004) MULIC: Multi-Layer Increasing Coherence Clustering of Categorical Data Sets. Technical Report # CS-2004-07. Department of Computer Science and Engineering, York University.

[3] Fasulo D. (1999) An Analysis of Recent Work on Clustering Algorithms, Technical Report # 01-03-02, Department of Computer Science & Engineering, University of Washington.

[4] Goebel, M. & Gruenwald, Le (1999). A survey of data mining and knowledge discovery software tools. ACM SIGKDD Explorations 1, 20-33.

[5] Grambeier J., Rudolph A. (2002) Techniques of Cluster Algorithms in Data Mining. Data Mining and Knowledge Discovery 6: 303-360.

[6] Hartigan, J. A. (1975) Clustering algorithms. (John Wiley and Sons, New York, 1975).

[7] Huang, Z. (1997) Clustering Large Data Sets with Mixed Numeric and Categorical Values. Knowledge discovery and data mining: techniques and applications. World Scientific.

[8] Huang Z. (1998) Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. Data Mining and Knowledge Discovery 2(3): 283-304.

[9] Z. Huang, M.K.Ng. (1999). A fuzzy k-modes algorithm for clustering categorical data. IEEE Transaction on Fuzzy Systems, 1999, 7(4): 446-452.

[10] Stutz J. and Cheeseman P. (1995) Bayesian Classification (AutoClass): Theory and results. Advances in Knowledge Discovery and Data Mining, 153-180, Menlo Park, CA, AAAI Press.

[11] Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery. Volume 2, Issue 2. Pages: 121–167. (June 1998).

[12] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[13] C.J.Mertz, P.Merphy. "UCI Repository of Machine Learning Databases", 1998, http://www.ics.uci.edu/~mlearn.

[14] Guha S., Rastogi R., Shim K. (2000). ROCK: A Robust Clustering Algorithm for Categorical Attributes. Information Systems 25(5): 345-366.

[15] Cherepinsky V., Feng J., Rejali M. and Mishra B. (2003) Shrinkage-Based Similarity Metric for Cluster Analysis of Microarray Data. PNAS 100(17): 9668-9673.