GOCE DELCEV UNIVERSITY - STIP
FACULTY OF COMPUTER SCIENCE

ISSN 2545-4803 on line

BALKAN JOURNAL OF APPLIED MATHEMATICS AND INFORMATICS (BJAMI)

YEAR 2020

VOLUME III, Number 1

GOCE DELCEV UNIVERSITY - STIP, REPUBLIC OF NORTH MACEDONIA
FACULTY OF COMPUTER SCIENCE

ISSN 2545-4803 on line

BALKAN JOURNAL OF APPLIED MATHEMATICS AND INFORMATICS

BALKAN JOURNAL
OF APPLIED MATHEMATICS AND INFORMATICS

(BJAMI)

YEAR 2020 VOLUME III, Number 1

AIMS AND SCOPE:
BJAMI publishes original research articles in the areas of applied mathematics and informatics.

Topics:
1. Computer science;
2. Computer and software engineering;
3. Information technology;
4. Computer security;
5. Electrical engineering;
6. Telecommunication;
7. Mathematics and its applications;
8. Articles of interdisciplinary of computer and information sciences with education, economics, environmental, health, and engineering.

Managing editor
Biljana Zlatanovska Ph.D.

Editor in chief
Zoran Zdravev Ph.D.

Lectoure
Snezana Kirova

Technical editor
Slave Dimitrov

Address of the editorial office
Goce Delcev University – Stip
Faculty of philology
Krste Misirkov 10-A
PO box 201, 2000 Štip,
Republic of North Macedonia

BALKAN JOURNAL
OF APPLIED MATHEMATICS AND INFORMATICS (BJAMI), Vol 3

ISSN 2545-4803 on line
Vol. 3, No. 1, Year 2020

EDITORIAL BOARD

CONTENT

UDC: 004.658:519.254]:616.44

COMPARISON OF CLUSTERING ALGORITHMS FOR THYROID DATABASE

ANASTASIJA SAMARDZISKA AND CVETA MARTINOVSKA BANDE
Computer Science Faculty, University Goce Delcev, Shtip
anastasija.102036@student.ugd.edu.mk, cveta.martinovska@ugd.edu.mk

Abstract. The main idea of this paper is to propose a methodology for analyzing, visualizing and clustering data of patients with different symptoms from a thyroid database. In previous work, the thyroid data were analyzed using the WITT algorithm. This clustering method properly formed the clusters of a control group and hypothyroid patients but failed to cluster the hyperthyroid patients. In this paper we analyzed the data using several algorithms: K-means, hierarchical clustering, EM algorithm, DBSCAN and Cobweb algorithm. The main idea is to determine the degree of matching between the clusters produced and the class labels in order to determine which algorithms give better results. Classification-oriented measures are used to validate the clustering results. We propose several preprocessing steps to overcome the problems with the large amount of noise and unbalanced classes in the given data set.

1. Introduction

The goal of clustering is to divide a set of objects into groups based on similarity [1]. There are different approaches to dividing objects into clusters and different types of clusters [2] [3].

In this paper, we perform experiments with 5 specific clustering techniques that are representatives of broad categories of algorithms and illustrate different concepts: K-means, hierarchical clustering, EM, DBSCAN and Cobweb algorithm. These conventional cluster algorithms lack descriptions of the created clusters while the WITT algorithm is a representative of conceptual clustering techniques.

The WITT algorithm is described by Hanson and Bauer [4] and corrected by Talmon and Braspenning [5]. The clusters created are accompanied by the correlational structure of the attribute values of the cases in those clusters.

The correlational structure can be used to describe the clusters. Additionally, this structure provides a means to determine the extent to which a new case belongs to each of the already defined clusters.

The goal of this paper is to propose a methodology for clustering data from a thyroid database [6]. Cluster analysis is an important step in exploratory data analysis. To overcome the problem with weak classification of the thyroid data set we performed experiments with several algorithms implemented in the well-known data mining software Weka [7]. Because the default values of the parameters in some of the algorithms did not provide satisfactory results, the data were preprocessed and the parameter values were experimentally determined to provide comparable results.

The organization of the paper is as follows. First, we describe the thyroid database. Next, we present the results obtained using the WITT algorithm and the experiments performed with several algorithms implemented in WEKA. The paper ends with a short discussion of the results and conclusions.

2. Thyroid data set

The thyroid database (Fig.1) contains data about the following categories of patients: 49 thyrotoxic cases labeled as thyrod, 21 hypothyroid cases and 129 cases of control group labeled normal [6].

```
 1,THYROT,264286,211628,294709,203571,468345,240972,502034,263768,223237,220848
 2,THYROT,257143,246512,246173,239286,472632,237500,508565,288406,198660,235247
 3,THYROT,271429,213953,299391,203571,463260,245139,497270,275362,219635,213869
 4,THYROT,371429,269767,324920,395238,713360,516667,765899,189855,435910,391078
 5,THYROT,271429,226744,282502,276190,593071,313889,636724,171014,353635,364929
 6,THYROT,292857,323256,213801,429762,647308,370139,695931,247101,264068,360071
 7,THYROT,364286,274419,313279,305952,542822,386111,583565,241304,336372,313030
 8,THYROT,185714,162791,269223,166667,498504,166667,494111,247101,167450,181316
 9,THYROT,407143,280233,342867,413095,717747,570139,771039,247101,367121,312147
10,THYROT,278571,313953,209394,425000,659123,358333,708244,171014,362948,505300
```

Figure 1. *Part of the thyroid database*

Not all 199 cases are used in all of the algorithms because some attribute values are missing. Some of the algorithms tolerate missing data and noise in the data. For each case, 15 attributes are stored in the database, but just the first 10 are used because their correlational structure is representative for the categories. These 15 attributes are:

1.T3 - thyroid hormone triiodothyronine
2.T4 - thyroid hormon thyroxine
3.T3/T4
4.FT4I - free T4 index
5.T3UA - T3 uptake calculated from T4 and FT4I
6.FT3I - free T3 index
7.T3UB - calculated T3 uptake from T3 and FT3I for comparison
8.TBG - thyroid binding globulin
9.T3/TBG
10.T4/TBG
11.TSH - thyroid stimulating hormone
12.CK1 - casein kinase 1
13.CK2 - casein kinase 2
14.LDH1 - lactate dehydrogenase 1
15.LDH2 - lactate dehydrogenase 2

3. WITT algorithm

Conceptual clustering algorithms describe clusters using the features of the members in a cluster. The main idea of the WITT algorithm is to find clusters with members that have highly correlated feature sets, which can represent the clusters. Additionally, this structure provides a means to determine the extent to which a new case belongs to each of the already defined clusters.

The WITT algorithm follows the way that people use to construct the categories, which means that categories are formed as a contrast between one another. Based on that idea Hanson and Bauer propose a measure for determining the cohesion between the existing clusters and a new cluster c, which is given by the following formula:

$$Cc = \frac{Wc}{Oc}, \tag{1}$$

where Wc represents the within-cluster cohesion and Oc represents the average cohesion between c and all other clusters. In the WITT algorithm, the measure Cc is used to assess whether some case can be added to an existing cluster or a new cluster has to be formed.

Cohesion measures Wc and Oc are defined using the contingency table. The contingency table Fij is a two - dimensional matrix in which each element contains the number of cases with the same value l for an attribute i, and the same value m for an attribute j. The within-cluster cohesion Wc is defined as the average variance in the co-occurrences of all possible attribute-value pairs for a given cluster:

$$Wc = \frac{\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} D(i,j)}{\frac{K*(K-1)}{2}} \tag{2}$$

where D(i,j) is the co-occurrence distribution obtained from the contingency table Fij, and K is the number of attributes. D(i,j) is computed as:

$$D(i,j) = \frac{\sum_{l=1}^{L} \sum_{m=1}^{M} Fij(l,m) * ln\, Fij(l,m)}{\sum_{l=1}^{L} \sum_{m=1}^{M} Fij(l,m) * ln(\sum_{l=1}^{L} \sum_{m=1}^{M} Fij(l,m))} \tag{3}$$

with L and M being the numbers of distinct values of attributes i and j respectively. L and M are the numbers of rows and columns of the contingency table Fij. The values of D(i,j) is 1 when the cases in the cluster have the same value for each attribute, and 0 when the cases are evenly distributed in the contingency table.

Talmon et al. [4] [5] proposed several changes to the measures that are used to compute the within-cluster cohesion Wc and the outer-cluster cohesion Oc.

The co-occurrence distribution becomes greater instead of remaining the same when the number of cases is doubled. As the number of cases goes to infinity D tends to 1, which means that Wc also tends to 1. The new D(i,j) measure is obtained using a different formula for the entropy of the contingency table

$$E(i,j) = -\sum_{l=1}^{L} \sum_{m=1}^{M} FRij(l,m) * ln\, F\, Rij(l,m) \tag{4}$$

where $FR_{ij} = \frac{F_{ij}}{N}$ is the fraction of cases instead of Fij. The entropy goes from 0 when all the cases are located in the same field of the table to lnT when all the cases are evenly distributed, where T=L*M is the number of entries in the table.

So D is computed after normalizing E with lnT and subtracting it from 1:

$$D(i,j) = 1 - \frac{\ln N - \frac{\sum_{l=1}^{L}\sum_{m=1}^{M} Fij(l,m)*\ln Fij(l,m)}{N}}{\ln T} \tag{5}$$

The outer-cluster cohesion is computed according to the following formula:

$$Oc = \frac{\sum_{k=1,k\neq c}^{P} Bck}{P-1} \tag{6}$$

where P is the number of clusters. Bck is defined as the difference in the within-cluster cohesion for the maximally similar and the disjunct union of the clusters divided by the difference in the within-cluster cohesion for the maximally similar and the real union of the clusters:

$$B_{ck} = \frac{W^*{}_{c\cup k_{max}} - W^*{}_{c\cup k_{min}}}{W^*{}_{c\cup k_{max}} - W^*{}_{c\cup k}} \tag{7}$$

According to the formula (7), the value of Bck varies between 1 and infinity. When the clusters are maximally disjunct, the value of Bck is 1, otherwise it is greater than 1. The union of two clusters has a maximal value when as many as possible non-zero elements of the contingency tables of the first cluster can be matched with non-zero elements of the contingency tables of the second cluster. While, the minimal value for the union of two clusters is obtained when as many as possible non-zero elements of the contingency tables of the first cluster can be matched with zero elements of the contingency tables of the second cluster.

In the WITT algorithm, the measure Cc given with (1) is used to determine whether the case has to be added to an existing cluster. Instead of using the within-cluster cohesion of a cluster with one case added, Talmon et al. propose the following formula:

$$C_c = \frac{\frac{W_{c+1} - W_{c+1min}}{W_c - W_{c+1min}}}{O_c} \tag{8}$$

The numerator of the formula (8) is computed as a quotient of the difference between the within-cluster cohesion of a cluster with one case added and the corresponding minimally achievable within-cluster cohesion divided by the difference between the within-cluster cohesion of the cluster without the case and the corresponding minimally achievable within-cluster cohesion.

Minimal within-cluster cohesion is obtained when the case that is added to a cluster is completely different from the cluster, so $W_{c+1\min}$ can be computed by:

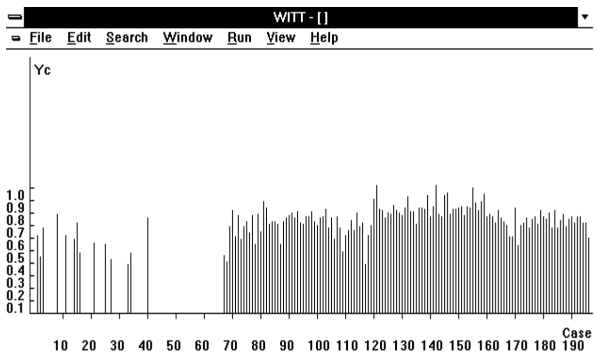
$$W_{c+1min} = \frac{N_c W_c + 1}{N_c + 1} - \frac{(N_c+1)\ln(N_c+1) - N_c ln N_c}{(N_c+1)} * \frac{\sum \frac{1}{lnT}}{N_T} \tag{9}$$

where Nc is the number of cases in a cluster C, and N_T is the number of contingency tables.

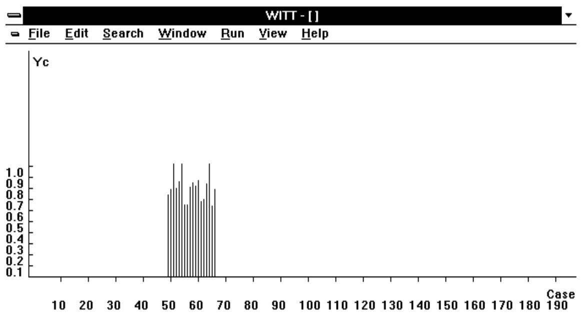
The WITT algorithm with corrected measures together with the pre-clustering and refinement algorithms are implemented as described in [8]. First, the initial clusters are defined, using a pre-clustering algorithm, from the representative cases based on the previous domain knowledge. The refinement algorithm adds cases to the initial clusters when those cases are similar enough to one of these clusters. When the refinement algorithm fails, the pre-clustering algorithm tries to make new clusters using cases not yet assigned to the existing clusters. When this process fails, the existing clusters may be merged.

The algorithm does not always cluster all the cases. When the remaining cases cannot be clustered and no existing clusters can be merged, the algorithm stops.

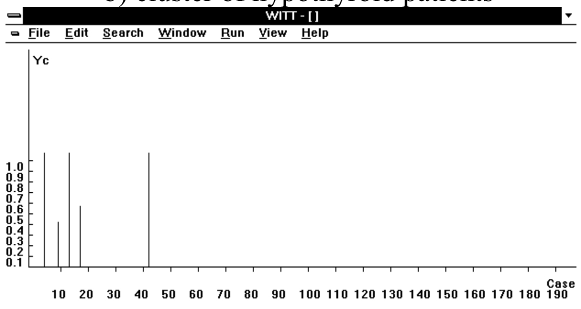
The plots representing the results of the clustering process for the thyroid database using the WITT algorithm are shown in Fig. 2 [8].



a) cluster of control group



b) cluster of hypothyroid patients



c) cluster of hyperthyroid patients

Figure 2. *Results of the clustering with the WITT algorithm*

Almost all of the cases with normal values of the attributes are in the cluster of the control group. Also, the patients from the category hypothyroid are in the cluster of hypothyroid patients, but some of the thyroid patients are in the cluster of the control group, and just five thyroid patients are in the cluster of thyroids.

Using the algorithm for creating initial clusters, three clusters are formed: the initial cluster for the control group is formed by cases 121 and 142, for the category of hypothyroid patients from 54 and 64 and for the thyroid category from cases 13 and 42. One of the reasons why thyroid patients are not well clustered might be that the cases in the initial thyroid cluster are not good representatives of this cluster. Several alternative ways for creating initial clusters are considered, because it is obvious that the clustering process is very much dependent on them. For example, making the distance between the clusters smaller, which can result in finding the clusters that are not very far away from each other, but maybe they are in the multidimensional space with a greater density of cases. Visually, the ideal initial clusters have to be parts of the multidimensional space where points are 'densely' packed, but have to be far away from each other. If this requirement is not satisfied, as for hyperthyroid patients that are not well correlated so there is no region with great density, then the cases from that category are not clustered.

No alternative ways for creating the initial clusters gave better results for the thyroid cluster.

4. Comparison methodology

In this section, we describe the methodology for comparing several clustering algorithms: K-means, hierarchical clustering, EM, DBSCAN and Cobweb. The algorithms are applied to a labeled data set. The actual class labels are given in the attribute *dijagnoza*. We will assume that the proper clusters are the clusters matching the class labels. The purpose of the comparison is to determine the degree to which clusters produced by a clustering algorithm correspond to the actual class labels in order to determine which algorithms produce better quality.

The measures' precision, recall and F-measure are often used to evaluate the performance of the classification models. In classification, we measure the correspondence between the predicted class labels and actual class labels, but nothing essentially changes if we use clusters instead of predicted classes, so the same measures can be used to evaluate clustering.

For each cluster, we first calculate the distribution of objects in classes, i. e. for the cluster i we calculate p_{ij}, the probability that a member of cluster i belongs to class j as $p_{ij}=m_{ij}/m_i$, where m_i is the number of objects in cluster i and m_{ij} is the number of objects of class j in cluster i.

Precision: The fraction of the cluster that consists of objects of a specified class. Precision of cluster i with respect to class j is $\mathrm{precision}(i, j) = p_{ij}$.

Recall: The extent to which a cluster contains all objects of a specified class. Recall of cluster i with respect to class j is $\mathrm{recall}(i, j) = \frac{m_{ij}}{m_j}$, where m_j is the number of objects in class j.

F-measure: A combination of precision and recall that measures the extent to which the cluster contains only objects of a particular class and all objects of that class. The F-measure of cluster i with respect to class j is $F(i,j) = \frac{2\times \text{precision}(i,j)}{\text{precision}(i,j)+\text{recall}(i,j)}$.

To perform the comparison, we have to add the attribute “cluster” to the data set. The value of this attribute for each object in the data set is the cluster to which this object is assigned according to the used clustering algorithm. This can be achieved by applying the *Add Cluster* filter to the original data set for each of the algorithms.

The meta-classifier *Classification Via Clustering* has to be installed using the Weka packet manager. If we chose the option Use Training Set, we will get the same Confusion matrix as if we had applied the appropriate clustering algorithm to the original data set from the Cluster tab, but additionally we get the values of the selected comparison measures.

4.1 K-means algorithm

K-means is one of the widely used prototype based clustering techniques [9]. This algorithm assumes that the prototype of the cluster is the cluster centroid, which is actually the mean of the group of points in n-dimensional space.

At the beginning, we choose K initial centroids, where K is a parameter defined by the user and represents the number of clusters we want to create. Every point is assigned to the nearest centroid. The group of points assigned to the centroid represents the cluster. The centroid of each cluster is then updated based on the points assigned to that cluster. This procedure is repeated until there are no points that change the cluster to which they have been assigned, or equivalently, the position of the centroids is unchanged.

In order to assign a point to a centroid, we need a measurement that defines „proximity“. In Weka several such functions are implemented: *Chebyshev Distance, Euclidean Distance, Filtered Distance, Manhattan Distance* and *Minkowski Distance*. There are combinations of functions for calculating the distance between points and centroids for which this algorithm always converges. Usually, convergence happens after a few algorithm iterations.

4.2 Agglomerative Hierarchical Clustering

With the partitional clustering algorithms, we specify the initial number of groups, and the objects are iteratively rearranged in groups until we achieve convergence. Contrary to that approach, the hierarchical clustering algorithms merge or divide the existing groups, and create a hierarchical structure that represents the order in which the groups are merged or divided.

Hierarchical Clusterer in WEKA implements the agglomerative method. The hierarchy is built using merging. The objects initially belong to a list of single element sets S1, S2,…,Sn. Then the distance is calculated to find pairs of sets {Si, Sj} that are closest to each other and have to be merged. After they are merged, sets Si and Sj are removed from the list of sets and replaced by $S_i \cup S_j$ [9]. This process repeats iteratively as long as all of the objects are in the same group.

In WEKA several types of links to determine the distance between objects are implemented: SINGLE, COMPLETE, AVERAGE, EMEAN, CENTORID, WARD, ADJ_COMPLETE and NEIGHBOR_JOINING. The advantage of this type of algorithms is that they can be applied to any type of attributes. In addition, we need to specify the required number of clusters in order to get the required level of granularity.

4.3 EM

In practice, each cluster can be represented mathematically using the distribution of the probabilities of the attributes. The overall data is a *mixture* of these distributions, where every individual distribution is usually called *component distribution*.

EM (Expectation Maximization) is a popular iterative algorithm that can be used to find the parameters of the distribution. It can be seen as an extension of the K-means algorithm. Instead of assigning each object to the most similar cluster, EM assigns each object to a cluster based on a weight that represents the probability of membership. In other words, strict boundaries between the clusters do not exist. According to this, the new average values are calculated using weighted measurements. EM begins with an initial assessment or "guess" of the parameters of the probability model (that together are referred to as a *vector of parameters*).

This algorithm iteratively evaluates the objects based on the distribution created by the parameter's vector. The re-evaluated objects are then used to update the parameter values. Each object is assigned a probability that it will have a determined set of attribute values provided that the object is a member of a given cluster. The algorithm iteratively finds the distribution parameters that maximize the measure of the model quality, called *log likelihood*. The EM algorithm is simple and easy to implement and converges quickly in practice.

Although EM is guaranteed to converge to a maximum, this is a *local* maximum and it is not necessarily the same with the global maximum. To improve the chance of finding the global maximum the process should be repeated several times, with different initial assumptions for the parameter values [10].

4.4 DBSCAN

DBSCAN (Density-based spatial clustering of applications with noise) is a clustering algorithm based on density that produces a partitional clustering where the number of clusters is automatically determined by the algorithm. Density-based clustering locates regions of high density that are separated by regions of low density. Objects in low-density regions are classified as noise and are discarded, and this means that DBSCAN does not produce a complete clustering.

DBSCAN is a simple and effective algorithm that illustrates several significant concepts that are important to any density-based clustering approach. There are several approaches to defining the term density. In the center-based approach, the density of a point in a data set is determined by the number of points in a specific radius Eps for that point. This method is simple to implement, but the density of any point is determined by the specified radius. For example, if the radius is large enough, all points will have an equal density of m, which is the number of points in the data set. Similarly, if the radius is small enough, all points will have a density of 1.

Using this method we can classify a point as (1) inside a dense region (core point), (2) on the edge of a dense region (border point) or (3) inside a region of low density (noise point). Using these definitions of core, border and noise points, DBSCAN algorithm can be informally described in the following way: Any two core points that are close enough – at an Eps distance of each other – are assigned to the same cluster. Similarly, any border point that is close enough to a core point is assigned to the same cluster as that core point. Situations can arise and have to be resolved when a border point is close to the core point of different clusters. Noise points are discarded.

4.5 COBWEB

COBWEB is a popular and simple method of incremental conceptual clustering. COBWEB is a clustering technique that was developed by the researchers in the machine-learning field in the 1980s. This is a method of incremental clustering that consists of two systems: Cobweb (for nominal attributes), and Classit (for numerical attributes). The COBWEB algorithm produces a clustering dendrogram that is called a classification tree and describes each cluster with a probability description.

COBWEB uses a heuristic evaluation measurement called *category utility* in order to guide the construction of the tree. A new class can be created on the fly, which is one of the major differences between COBWEB and K-means. COBWEB allows merge and split of classes based on the category utility, which allows it to perform a two-way search.

The COBWEB algorithm has several limitations. First, it is based on the assumption that the distributions of probabilities of different attributes are statistically independent from one another. This assumption is not always accurate as there is often a correlation between the attributes. Also, the probability distributions of clusters make the updating and saving of the clusters an expensive operation [10][11].

In WEKA COBWEB implements both the Cobweb algorithm for nominal attributes and the *Classit* algorithm for numerical attributes. The order and priority of the merge and split operations are different form the original Cobweb and *Classit* algorithms. This implementation always compares four different ways of treating a new instance and chooses the best one: adding the instance to the best existing node, creating a new node, merging two good nodes and adding the instance to the merged node, and dividing the best node and adding the instance to one of the parts. The parameters of this algorithm are *acuity* and *cutoff*.

5. Experimental results and discussion

Table 1 presents the comparison of the results for the applied algorithms. The best results are obtained using DBSCAN with 99% of correctly classified instances. Analyzing the process of clustering with DBSCAN, we realized that more than the half of the hyperthyroid cases (27 out of 49) were treated as noise and discarded from the final results. The results were comparable to those obtained with the WITT algorithm where the hyperthyroid cluster of data is not well formed. Using the application that implements the WITT algorithm, 16.09% of the cases are left unclassified and 7.03% are incorrectly classified.

Table 1. *Comparison of the clustering algorithms*

Method	Correctly classified instances	Incorrectly classified instances	F-measure
SimpleKMeans	70.92%	29.08%	0.77
HierarchicalClusterer	81.63%	18.37%	0.83
EM	59.18%	40.82%	0.65
DBSCAN	99.49%	0.51%	/
COBWEB	69.39%	28.06%	0.76
WITT	76.88%	23.12%	0.69

In order to be able to use the DBSCAN algorithm in WEKA, we first installed this algorithm through the Weka Packet Manager available in the "Tools" menu. Performing the algorithm with default parameters did not produce acceptable results. Useful results were obtained after normalization of the numerical attributes for the given data set and after setting the suitable values for the *epsilon* and *min Points* parameters that were determined experimentally. The WEKA implementation of this algorithm does not allow classification of new instances and therefore it cannot be used with the *Classification Via Clustering* meta-classifier.

The next promising result is obtained with *Hierarchical Clusterer* for which the F measure is 0.83. For this algorithm it is experimentally determined that better results are achieved for complete link and *Chebyshev Distance* as a distance function.

As Table 2 shows using *Hierarchical Clusterer,* there is an overlapping between the cluster of the control group and hypothyroid patients, but the cluster of hyperthyroid patients is well formed.

Table 2. *Confusion Matrix for Hierarchical Clusterer*

a	b	c	<-- classified as
48	0	0	a = THYROT
0	12	8	b = HYPOTHYROID
3	25	100	c = NORMAL

When running the EM algorithm, by setting the *numClusters* parameter to -1, the algorithm will determine the number of clusters. In order to achieve better results, the number of clusters was explicitly set to match the given number of classes. Additionally, in order to improve the result, the numerical attributes were normalized and discretized in 10 groups, but despite the adjustments made, no satisfactory results were obtained. As Table 3 shows, two of the clusters (NORMAL and HYPOTHYROID) are not properly defined.

Table 3. *Confusion Matrix for EM*

a	b	c	<-- classified as
39	0	9	a = THYROT
18	1	1	b = HYPOTHYROID
0	53	75	c = NORMAL

Since the default values of COBWEB parameters did not give satisfactory results, data were normalized and *acuity* and *cutoff* values were experimentally determined which gave comparable results. From Table 4 it is evident that there is an overlapping between the cluster of the control group and hypothyroid patients, but here the cluster of hypothyroid patients is well formed.

Table 4. *Confusion Matrix for COBWEB*

a	b	c	<-- classified as
36	3	4	a = THYROT
0	20	0	b = HYPOTHYROID
1	47	80	c = NORMAL

Table 5 shows the confusion matrix for Simple K-means algorithm. There is an overlapping between all three clusters. Although the F measure is 0.76, the results are not acceptable.

Table 5. *Confusion Matrix for Simple K-means*

a	b	c	<-- classified as
35	13	0	a = THYROT
0	20	0	b = HYPOTHYROID
0	44	84	c = NORMAL

6. Conclusion

Several clustering algorithms: K-means, hierarchical clustering, EM, DBSCAN and Cobweb algorithm are applied to thyroid database in order to find the most appropriate clustering method. The best results were obtained using DBSCAN but there are cases that are left unclassified from the cluster of hyperthyroid patients. This result is comparable with the results obtained with the WITT algorithm. None of the other algorithms produces better results.

The main problem with the WITT algorithm and DBSCAN is poor clustering of the hyperthyroid patients. One reason that thyroid patients are not well clustered is that they are not well correlated, which can be seen from the histograms of the attributes. To overcome this problem with the WITT algorithm, different coding of cases is considered. For example, the square root is taken from the intervals that represent the distinct values for the attributes, so that the first intervals are smaller and the last ones bigger. Another alternative that is examined is taking the same number of cases in each interval, and the last alternative was leaving the intervals with a great number of cases the same and widening the intervals with a smaller number of cases to have the 'necessary ' number of cases. None of these methods gave some better results, so this problem has not been solved.

For future work, we plan to make experiments with different subsets of attributes in order to discover the attributes that lead to acceptable clustering of the thyroid database.

References

[1] *Larose, D.* (2006) Data Mining, Methods and models. John Wiley

[2] *Tan, P., Steinbach, V. and Kumar, V.* (2006) Introduction to Data Mining, Pearson

[3] *Witten, I. and Frank, E.* (2005) Data Mining, Practical Machine Learning Tools and Techniques. Morgan Kaufmann

[4] *Hanson, H. J. and Bauer, M.* (1989) Conceptual Clustering, Categorization and Polymorphy, Machine Learning, 3, pp. 343-372.

[5] *Talmon, J.L., Braspenning, P.J. and Fonteijn, H.* (1993) An Analysis of the WITT algorithm, Machine Learning, 11, pp. 91-104.

[6] *Cekovska, S. (2004).* Disorders of the thyroid gland. M.Sc. thesis. Faculty of Pharmacy, UKIM. (in Macedonian language)

[7] *WEKA* – Data Mining with Open Source Machine Learning Software in Java, https://www.cs.waikato.ac.nz/ml/weka/

[8] *Martinovska C. (2009)* Conceptual Clustering of Data from Thyroid Database, ETAI, Ohrid

[9] *Frank, E., Hall, M. and Witten, I.* (2016) The Weka Workbench: Online Appendix for Data Mining: Practical Machine Learning Tools and Techinques, Morgan Kaufmann, Fourth Edition

[10] *Sharma, N., Bajpai, A. and Litoriya, Mr. R. (2012) Comparison of the various clustering algorithms of WEKA tools, IJETAE, vol. 2 (5)*

[11] *Chaudhari, B. and Parikh, M. (2012) A Comparative Study of Clustering Algorithms Using WEKA Tools, IJAIEM, vol. 1 (2)*