ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΔΙΔΑΚΤΙΚΗΣ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΨΗΦΙΑΚΕΣ ΕΠΙΚΟΙΝΩΝΙΕΣ ΚΑΙ ΔΙΚΤΥΑ

ΣΧΕΔΙΑΣΗ ΣΥΣΤΗΜΑΤΩΝ ΓΙΑ COGNITIVE ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ

ΜΠΑΝΤΟΥΝΑ ΑΙΜΙΛΙΑ

ΜΕ/07065

ΠΕΙΡΑΙΑΣ, 2010

UNIVERSITY OF PIRAEUS

DEPARTMENT OF DIGITAL SYSTEMS

MASTER IN TECHNOLOGY EDUCATION AND DIGITAL SYSTEMS

DIGITAL COMMUNICATIONS AND NETWORKS

DESIGNING SYSTEMS FOR COGNITIVE RADIO SYSTEMS

BANTOUNA AIMILIA

ME/07065

PIRAEUS, 2010

ABSTRACT

Cognitive radio systems (CRSs) have been proposed as a promising technology for intelligently handling the rapid evolution of wireless communications, especially in terms of managing and allocating the scarce, radio spectrum in the highly varying and disparate modern environments. A typical CRS implements a so called "cognition cycle", during which it senses its environment, evaluates a set of candidate radio configurations to operate with and finally decides and adjusts its operating parameters expecting to move the radio toward an optimized operational state. Such a process is often proved to be rather arduous and time consuming. Accordingly, learning mechanisms that are capable of exploiting measurements sensed from the environment, gathered experience and stored knowledge can be judged as rather beneficial in terms of speeding up the whole cognition process. Framed within this statement, this paper introduces and evaluates a mechanism which is based on a well-known unsupervised learning technique, called Self-Organizing maps (SOM), and is used for assisting a CRS to predict the bit-rate that can be obtained, when it senses specific input data from its environment, such as Received Signal Strength Identification (RSSI), number of input/output packets etc. Results show that the proposed method can provide predictions which are correct up to a percent of 78.9%

Keywords: Cognitive Radio Systems (CRS), Cognition Cycle, Learning, Self-Organizing Maps (SOMs)

CONTENTS

FIGURE LIST

TABLE LIST

CHAPTER 1: INTRODUCTION

Nowadays communications are getting more and more wireless and each one needs its different piece of electromagnetic radio spectrum. However, this specific natural source is limited. Moreover, its current static assignment often leads to its underutilization. As a result the deployment of a technology which will have the ability to exploit the underutilized frequency bands is needed.

One way out of this problem is the development and utilization of cognitive radio systems [7] [8]. Cognitive systems have the ability to adjust their function according to the external, environmental stimuli, the demands of the users/applications and their past experience. Based on this ability, future cognitive systems will be able to change their parameters (carrier frequency, radio access technology, transmit power, modulation type etc), observe the results and decide which is the best combination of those parameters in order to get into a better operational state. So, in terms of flexible spectrum management concept, use of cognitive systems will allow the use of a spectrum band in different radio access technologies (RATs) [2], [3].

A typical cognitive operation consists of three cooperative phases shown in Figure 1 [2] [8]. During, the first phase, known as “radio scene analysis”, the system collects measurements from its environment (e.g. conditions related to interference) and explores different configurations. In the 2nd phase, “channel estimation and predictive modeling”, the output of the 1st phase is used for discovering the capabilities of each candidate configuration, wherein past experience of the system may also be used. Finally, in the last phase, known as “configuration selection”, the system decides for the best configuration and accordingly adjusts its operation parameters.

Further analysis of a cognitive system reveals the existence of a cognitive engine (Figure 2), a software package which is the one that provides to cognitive radio systems their capabilities [3]. Cognitive engines support software defined radios

by changing their parameters, observing them and taking measurements and actions which are defined by the desired operational state. On the other hand, software defined radios "learn the lesson" and store it in the knowledge base of the cognitive engine. Another part of the cognitive engine, reasoning, decides whether the actions are possible, given an environmental state, or not. This is proved to be a very arduous and time-consuming task, which can be relaxed by using learning mechanisms.

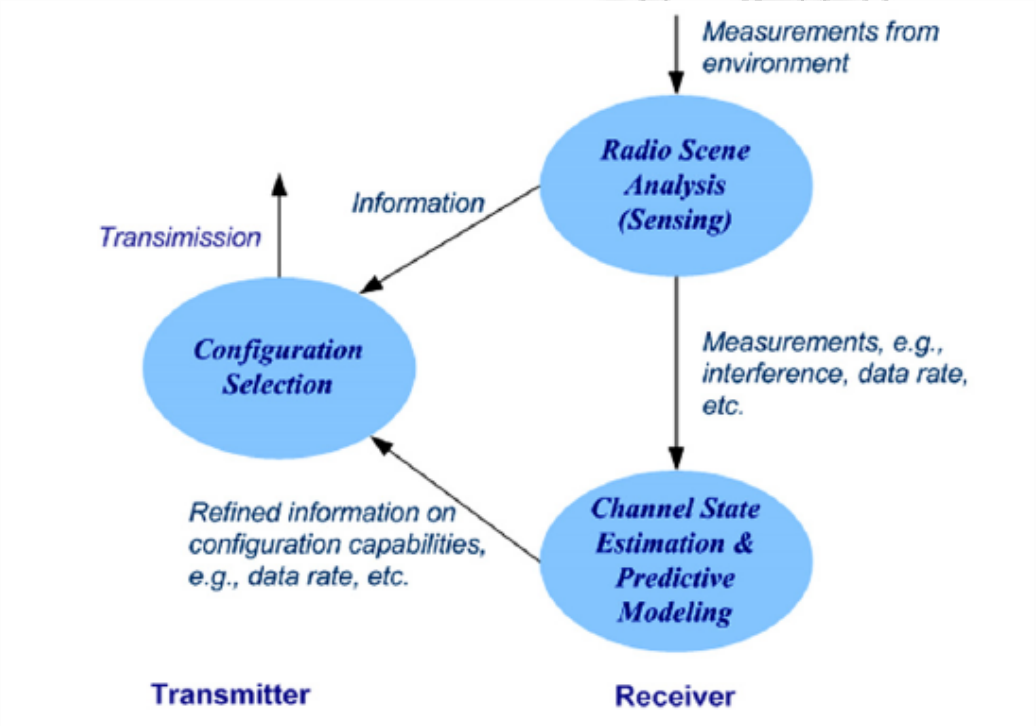


Figure 1. Simplified representation of cognitive radio cycle [2]

Supervised learning through neural networks-based schemes have been used recently in [1] [3]. Bayesian networks have been also used in [2]. Our proposal is based on an unsupervised neural network technique, called Self-Organizing Maps (SOMs). SOM is a technique for representation and classification of multidimensional data into 2D maps. These maps consist of rectangular or hexagonal cells on a regular grid and according to the technique; each data sample correlates with one cell/neuron of the map in order to be closest to those who are most like it. In this term, the created map represents the similarity of the data and their classification. SOMs are very popular in data mining problems such as

identification of illicit drugs [13], chemical analysis [14], document collections [16], speech recognition [17], identification of a cancer cell gene [18], hematopoietic differentiation [19] and more. In our case we examined the possibility of connecting parameters that can be achieved under a configuration in question of a CRS, such as noise, received signal strength Indication (RSSI), errors (input and output), packets (received and sent) and Bytes (received and sent) with an anticipated QoS metric, namely the bitrate that can be achieved under the same configuration.

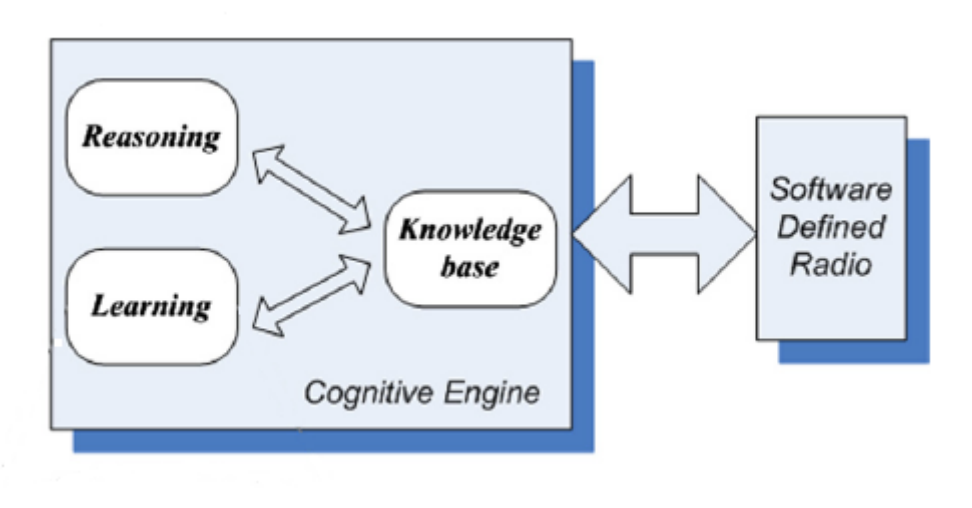


Figure 2. Representation of cognition cycle

Moreover, in order to validate the technique, we have setup and executed a program by using MATLAB SOM toolbox. The developed SOMs are trained with measurements that have taken place in a real working environment within our University premises. The method exhibits a satisfactory capability of predicting the achieved bitrate when facing both known and unknown exemplars (combinations of achieved parameters given a configuration).

In order to do so, matlab data files, which consist of different parameters that are observed as a result of the configuration in question of a CRS, including bitrate, were created. Parameters, except for bitrate, were used for the training of a SOM map and the respective data samples were depicted on it. Bitrate was used as a label of the cell which was correlated to the respective data sampple. It is worth

mentioning that more than one combination of parameters may be correlated to one cell.

Having trained the SOM map, we were able to proceed with the prediction of the bitrate of known or unknown data samples. The prediction was based on a similar idea to the training. First of all, each data sample was correlated to a cell by using the same parameters which were used during the training. After the correlation, the prediction of the bitrate was able to take part with respect to the label/bitrate of the closest cells. Finally, conclusions were reached by comparing the predicted bitrate of the data sample with its actual respective measured bitrate.

The rest of the thesis is structured as follows: a review of SOM technique is presented in chapter 2 while a short analysis of our proposal is made in chapter 3. A short analysis of the created algorithm is made in chapter 4. Chapter 5 presents the results of our test cases, including a comparison of different versions of our program (chapter 5.1), the choice of the variables of the input data samples (chapter 5.2), different cases which include different number of data samples (chapter 5.3) and different evaluating scenarios with different parameters of SOM technique (chapter 5.4), are presented. Finally, in chapter 6, our conclusions are presented.

CHAPTER 2: SELF-ORGANIZING MAPS (SOMs)

SOM, introduced by Teuvo Kohonen, is a type of neural network-based scheme whose training algorithms are unsupervised. An overview of its theoretical base may be found in [9]. It is a technique for representation and classification of multidimensional data into 2D maps. These maps consist of cells, whose shape is rectangular or hexagonal, on a regular grid. According to the technique, each data sample correlates with one cell/neuron of the map, called Best Matching Unit (BMU). The BMU and a neighborhood around it stretch towards the inserted data sample (Figure 3). This process is called SOM training and results in an ordered SOM map where similar neurons are close. In this term, the created map represents the similarity of the data and their classification.

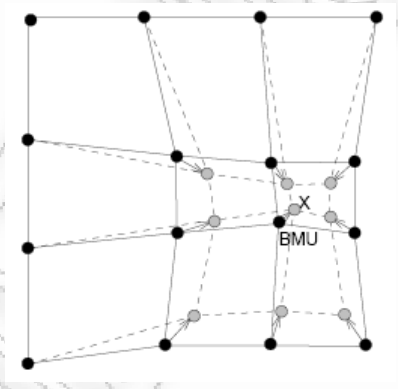


Figure 3. The inserted data sample x affects its BMU and its neighborhood. The solid and dash-doted lines correspond to the situation before and after the input of the data sample [10]

This attribute of SOM has made it very popular in data mining problems such as identification of illicit drugs [13], chemical analysis [14], document collections [16], speech recognition [17], identification of a cancer cell gene [18], hematopoietic differentiation [19] and more. In our project we attempted to predict the bitrate of a configuration according to other parameters that are observed as a result of the specific configuration.

Two different training algorithms are used for SOM training: the sequential and the batch training algorithm. In the first case each data sample is inserted in the

training process one by one, thus affecting its own best matching unit (BMU) and a neighborhood around it. In case 2, all data samples are inserted together in the process and eventually affect their BMUs and their neighbors at once.

In autumn of 1997 a toolbox for SOM technique, called SOM TOOLBOX, was created while in spring of 2000 its 2nd version was created and released, executable in MATLAB version 5 or in newer versions. This toolbox enables the user to manage the data, initialize and train the maps and finally represent and analyze them. During the initialization and the training phases of the map, the user is able to change many different parameters of the algorithm and select between different combinations of the topology (local lattice structure, size of the map, global map shape) and the training parameters. Figure 4 and Figure 5 depict different local lattice structures and different global map shapes, respectively.

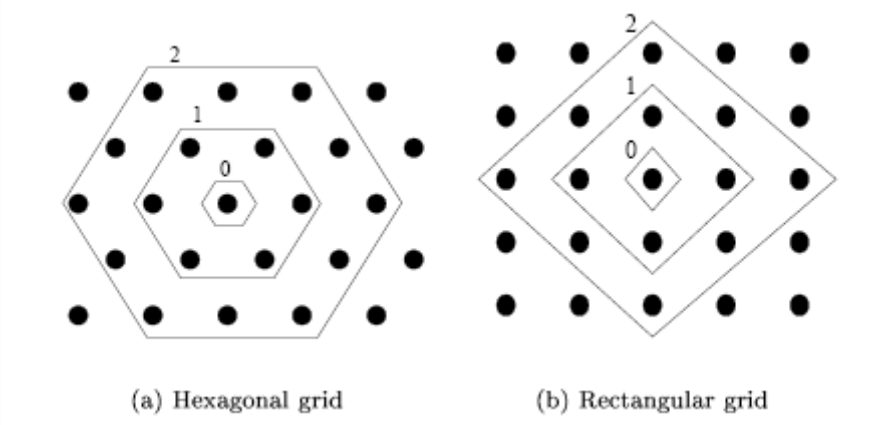


Figure 4. Representation of local lattice structures: (a) hexagonal lattice and (b) rectangular lattice [10]

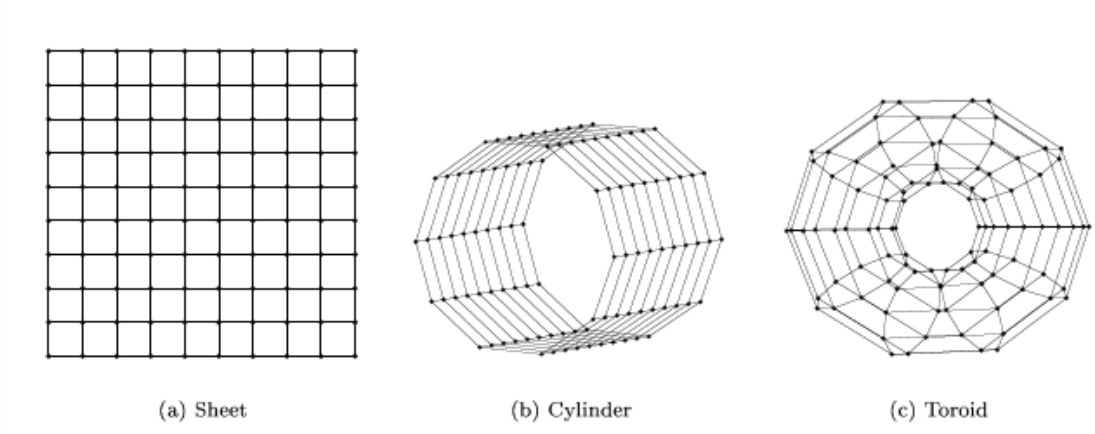


Figure 5. Representation of different global map shapes: (a) Sheet (default value), (b) Cylinder and (c) Toroid [10]

2.1 Sequential Training Algorithm

2.1.1 Theoretical Base of Sequential Training Algorithm

As mentioned above, this training algorithm is iterative as each data sample x is inserted into the process by itself. In our project, data sample x is the combination of the observed parameters of a configuration, except for the bitrate of the configuration. The process is described below: First of all, a random data sample x is selected and then its distance from each neuron m_i is calculated. The neuron which is closest to the selected data sample is called best matching unit (BMU) of the data sample, denoted here by c, while m_c is the cell which is finally associated with the data sample/combination of the parameters of the configuration:

$$\| x(t) - m_c(t) \| \leq \| x(t) - m_i(t) \|, \quad (1)$$

where ||.|| is the method, according to which the distance is calculated. The method which is most usually used is the Euclidean Distance.

Having correlated the data sample with its BMU, the weights of the BMU's vector and those of its neighbors change in order to become more alike with the weights

of the vector of the data sample. The equation that is used for this process is the next one:

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)], \quad (2)$$

where *c* denotes the neuron, *m* is the weight which will be updated, *x(t)* is the data sample and *t* is the index of the regression step. The function h_{ci} is known as neighborhood function and in case of being Gaussian is calculated by the following equation:

$$h_{c(x),i} = a(t)\exp\left(-\frac{\|r_i - r_c\|^2}{2\sigma^2(t)}\right) \quad (3)$$

and describes the area around the BMU which will be affected by the calculations. The factor *α(t)* is the learning-rate factor while *σ(t)* corresponds to the width of the neighbourhood function and they both decrease monotonically with the regression steps.

Finally it's worth mentioning that the training phase usually consists of two phases. The first one is the rough phase where the initial learning rate and the neighborhood radius are relatively large. On the other hand, during the 2nd phase, which is called fine-tuning phase, the initial learning rate and the neighborhood radius are small from the beginning. As a result, the map is approximately shaped in the first phase while in the second one it is fine-tuned.

2.1.2 The Sequential Training Algorithm in the SOM toolbox

In case of using matlab toolbox, the calculation is slightly different due to the fact that there may be some missing values of the variables of a data sample or the selected "mask" may dictate something different. Considering the above facts, the specific calculation transforms into the next equation:

$$|| x - m ||^2 = \sum_{k \in K} w_k (x_k - m_k)^2, \quad (4)$$

where k denotes the set of known (not missing) variables of sample vector x, x_k and m_k are k-th components of the sample and weight vectors and w_k is the k-th mask value.

Moreover, equation (2) transforms into the next equation:

$$m_i(t+1) = m_i(t) + a(t)\,h_{ci}(t)[x(t) - m_i(t)] \quad (5)$$

where neighborhood function h_{ci} and learning rate factor $\alpha(t)$ may be a number of different functions.

There are three different functions which can be used as learning rate:

- Linear function: $a(t) = a_0(1 - t/T)$ (6)
- Power function: $a(t) = a_0(0.005/a_0)^{t/T}$ and (7)
- Inv function: $a(t) = a_0/(1+100t/T)$, (8)

where T is a constant variable, called training length and a_0 is also a constant variable, known as initial learning rate. Figure 6 shows the diagrams of these functions:

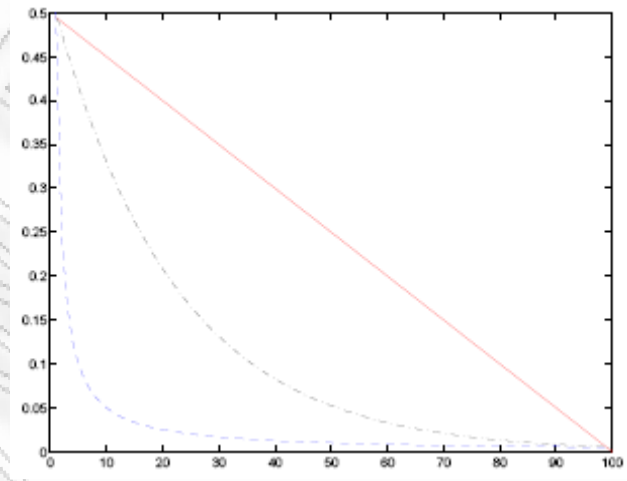


Figure 6. Learning Rate Functions: (a)'Linear' (solid line), (b) 'Power' (dot-dashed line) and (c) 'Inv' (dashed line) [10]

In contrast with learning rate, neighborhood kernel does not depend on time. Moreover, it does not depend on the distance between the neuron m_c and the BMU c. The four functions that may be used in this case are the following:

- Bubble: $h_{ci}(t) = l(\sigma_t - d_{ci})$ (9)
- Gaussian: $h_{ci}(t) = e^{-d_{ci}^2/2\sigma_t^2}$ (10)
- Cutgauss: $h_{ci}(t) = e^{-d_{ci}^2/2\sigma_t^2} l(\sigma_t - d_{ci})$ and (11)
- Ep: $h_{ci}(t) = \max\{0, 1-(\sigma_t - d_{ci})^2\}$, (12)

Where σ_t is the radius of the neighborhood, $d_{ci} = \| r_c - r_i \|$ is the distance between the cells m_c and m_i and $l(x)$ is the step of the function: $l(x) = 0$ if $x < 0$ and $l(x) = 1$ if $x \geq 0$. Figure 7 depicts diagrams of the functions when neighbourhood radius is σ_t=2:

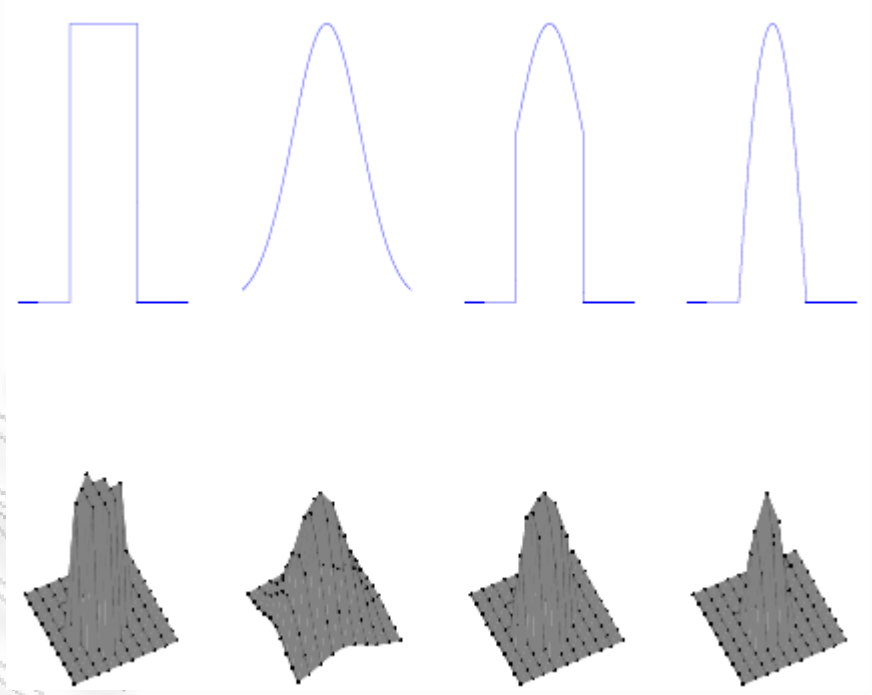


Figure 7. Representation of Neighborhood functions. From left to right: (a) Bubble, (b) Gaussian, (c) Cutgauss and (d) Ep [10]

2.2 Batch Training Algorithm

2.2.1 Theoretical Base of Batch Training Algorithm

As well as in the case of sequential training algorithm, this algorithm is also iterative. On the contrary to the previous training algorithm, in this one all data samples are inserted simultaneously. Thus, it is quicker and does not demand the specification of the learning rate.

All data samples are inserted in the process before any adjustments. In each training step the new weight of each cell *m* is given by the following relation:

$$m_i(t+1) = \frac{\sum_{j=1}^{n} h_{ic}(t) x_j}{\sum_{j=1}^{n} h_{ic}(t)}. \qquad (13)$$

where $c = \arg\min_k \{|| x_j - m_k ||\}$ denotes the BMU cell of data sample x_j and $h_{ic}(t)$ is the neighbourhood function of BMU.

Alternatively, the sum of the vectors of each Voronoi region of the map can be calculated first according to the following function:

$$s_i(t) = \sum_{j=1}^{n_{Vi}} x_j , \qquad (14)$$

where n_{Vi} is the number of data samples of cell *i*, and then the weights of the vectors can be calculated by the equation:

$$m_i(t+1) = \frac{\sum_{j=1}^{m} h_{ij}(t) s_j(t)}{\sum_{j=1}^{m} n_{v_i} h_{ij}(t)}, \qquad (15)$$

where *m* is the number of the neurons of the map.

Finally, it is worth mentioning that, like in the case of sequential training algorithm, in this algorithm as well, the process is divided in two phases: the rough and the fine-tuning phase.

2.2.2 The Batch Training Algorithm in the SOM toolbox

In the SOM toolbox, the batch training algorithm is executed using the alternative with the Voronoi regions while, like in the case of sequential training algorithm, the variables of the data samples which have missing values are ignored and data sample x_j is the combination of the observed parameters of a configuration, except for bitrate. Finally, neighbourhood functions that can be used are the same with those of the sequential training algorithm while there is no need of defining a learning rate factor.

CHAPTER 3: CONTRIBUTION OF THE SOM TECHNIQUE TO THE PREDICTION OF THE BITRATE

As mentioned in chapter 1, we examined the possibility of connecting parameters that are observed as a result of the configuration in question of CRS, such as noise, received signal strength Indication (RSSI), number of errors (input and output), number of packets (received and sent) and number of Bytes (received and sent) with one QoS metric, the bitrate in order to predict it. This is done by using the unsupervised training technique of SOM. However, as also mentioned in chapter 2, SOM is a technique for representation and classification of multidimensional data into 2D maps. So how could it be useful in our case?

To begin with, measurements that have taken place in a real working environment within the premises of our University were used to create different data files. Each file comprised a different test case including different combinations of the parameters. The last column of each data file was the measured value of the bitrate which was related to the combination and was taking part only for separating the data samples. So, in each data file there was a number of columns and each column was a different parameter of the data sample. Moreover, each row of the data file corresponds to one different data sample x (see Figure 8).

In the sequence the created data file and SOM toolbox v.2 of MATLAB were used to train the SOM. However, before training the SOM map, it was essential to decide if the data would be normalized or not and according to what it would be normalized. We examined three different scenarios for the normalization and run many different test cases around them. The first scenario referred to data that had not been normalized; the second one referred to data that had been normalized to [0,1] and the last scenario referred to data whose variance had been normalized to [0,1]. So, after having normalized or not the data according to the scenario, we used them to train a SOM map. For facilitating the analysis, SOM toolbox offers

the ability to use labels in order to distinguish the data samples. In our case, each label corresponds to the measured value of the bitrate of the data sample. However, the fact that more than one data samples may have the same cell of the map as BMU (m_c) leads to the fact that each cell of the map may have more than one labels appearing more than once. At this point, it is worth mentioning that SOM toolbox offers enough different ways for labeling the map. In our case we used three of them ending up with three different versions of our program, respectively. The first way is to put on each cell only the most frequently appearing label, the second one is to put all labels in descending order with respect to the appearance frequency while the third one is to put all labels in descending order with respect to their appearance frequency as before, but also followed by the number of appearances.

```
 1  5
 2  #n RSSI IPKTS   OPKTS   IBYTES  OBYTES
 3      -35 926     1750    32630   880650  54
 4      -32 908     1680    31932   845424  54
 5      -32 888     1680    31272   845424  54
 6      -35 888     1679    31272   845468  54
 7      -36 890     1683    31338   845598  54
 8      -36 960     1818    33812   915702  54
 9      -36 890     1682    31338   845598  54
10      -36 928     1766    32756   887366  54
11      -34 920     1731    32328   873760  48
12      -33 926     1751    32586   880650  54
13      -33 924     1750    32564   880650  54
14      -32 894     1684    31494   845688  54
```

Figure 8. Matlab Data File: the number of the first line refers to the number of the parameters of the configuration, here equal to 5 (RSSI, Input PacKeTS, Output PacKeTS, Input BYTES, Output BYTES), and the last column refers to the bitrate which was used as label. Each line is a data sample and each column is a different parameter of the configuration.

At this point, the output of our program was a labeled SOM map (Figure 19, Figure 20 and Figure 21) and our program was able to represent a new data sample on the map but could not predict the bitrate of a data sample. In order to train our program how to predict the bitrate of a data sample we transformed our visualization into mathematical functions. Normally, a data sample has the same

bitrate with the bitrate of the closest cells. However, the closest cells may have more than one different values of bitrate. So, we need to find the bitrate that represents the most cells which are close to the BMU of the data sample.

In other words, cells which have the same bitrate comprise a cluster. In order to define the bitrate we needed to find to which cluster the cell of the data sample belonged. Trying to transform this thought to mathematical function the need of a center of each cluster revealed. The center of a cluster may be calculated by the equations

$$x = \sum_{i}^{n} \frac{w_i x_i}{n} \quad (16) \qquad \text{and} \qquad y = \sum_{i}^{n} \frac{w_i y_i}{n} \quad (17),$$

where n is the number of cells which belong to the cluster, x_i and y_i are the co-ordinates of the cell i and w_i is the weight according to which the cell i participates to the calculation. In the first two versions (VOTE, ADD1) w_i is always equal to 1 while in the last version (FREQ) w_i may be calculated by the following function:

$$w_i = \frac{k}{r} \quad (18),$$

where k is the number of instances of the specific bitrate in the cell i and r is the sum of the instances of all bitrates of the cell.

At this point, it was easy to predict the bitrate of a data sample. All that was needed to be done was to find its BMU and then, find the center of the cluster that was closest to the BMU. The bitrate of the data sample was the one that represented the cluster. For calculating the distance we used the Euclidean Distance.

Finally, for evaluating our process and reaching conclusions our program is able to compare the predicted values of the bitrate of each data sample with its real measurement. These comparisons are also expressed in percent. It is worth mentioning that the data samples whose bitrates were predicted were inserted with data files similar to those which were used for the training phase.

CHAPTER 4: THE ALGORITHM AND THE SOM_AUTOLABEL FUNCTION

Our program depends on the 2nd version of SOM toolbox, so in order to be functional MATLAB 5 (or newer versions) is required. As mentioned in chapter 3, three different versions of our program have been created and differ to the way of representation and labeling while each version may be combined with three different scenarios of normalization. The difference of representation arises from the difference of the value of a parameter of a function called som_autolabel. In the first version, the parameter of som_autolabel function is set to "VOTE", in the second one it is set to "ADD1" while in the third version it is set to "FREQ". So, in order to distinguish the three versions of our program, each version was named after the parameter of the som_autolabel function which is used. As a result, the first version is the "VOTE" version, the second one is the "ADD1" version and the last one is the "FREQ" version. Finally, the VOTE, ADD1 and FREQ versions were compared to each other. All three of them have two basic phases: the training and the evaluation one. Each phase may consist of one or more m-files.

The first phase is executed using the m-file training.m or training1.m, depending on the version used, which consist of 8 functions of SOM toolbox. The first two of them are the functions som_read_data and som_normalize. These two are responsible for reading the data file and normalizing the data samples. In cases of the scenario of data without normalization the 2nd function did not exist. The next function is the function som_gui. This specific function is a very basic one as it is connected to a very useful user interface. Using this GUI (Figure 9) one may control the initialization and the training part of SOM.

In order to train a SOM, we first need to load the file which will include the data according to which the map will be trained. The data needs to be saved in a dat-file in which each row is one data sample and each column is one variable or component of the data sample. As soon as the reading of the data file and the

normalization process (when used) have finished the loading of the file becomes available by following the next steps:

- Press “load” button
- The “load data” GUI appears (Figure 10), select “Ws” and then write “sd” (sd is the name of the matlab variable, according to our program, which is related with the data file after the normalization of the data samples or not)
- After having selected the data file, press the “load” button and the data will load automatically.

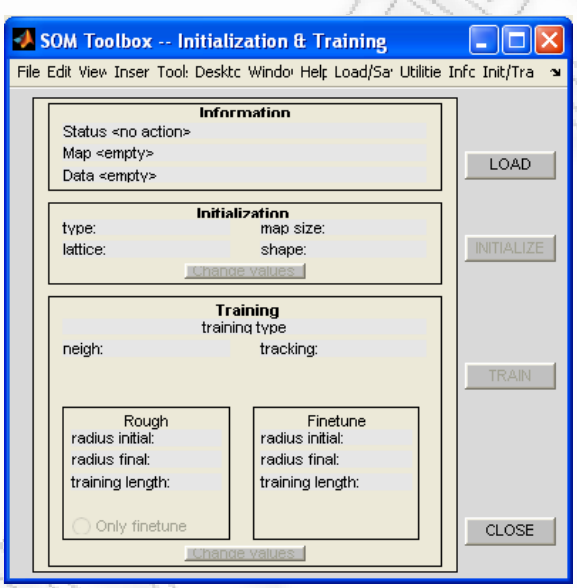


Figure 9. Initialization and Training SOM GUI as introduced in the SOM toolbox (version 2)

When the data loads, at the SOM GUI the buttons “initialize” and “change values” (in the initialization part) become available (Figure 11). Also, in the initialization part, the “type”, the “map size”, the “lattice” and the “shape” have been selected automatically. The “type” refers to the way that the data samples are used for the initialization and may be “linear” or “random”. The “map size” refers to the number of the output neurons of the SOM. As mentioned in [20], “the number of output neurons in a SOM can be selected using the heuristic rule suggested by Vesanto et al. (2000) [21], and applied in Park et al. (2006) [22] in a study of diatom communities: the optimal number of map units is close to 5*sqrt(n), where *n* is the number of training samples (sample vectors). In this case, the two largest eigenvalues of the training data are first calculated, then the ratio between side

lengths of the map grid is set to the ratio between the two maximum eigenvalues. The actual side lengths are finally set so that their product is close to the number of map units determined according to Vesanto et al.'s rule." The "lattice" refers to the shape that each neuron has on the SOM and may be "hexagonal" or "rectangular", as depicted in Figure 4.

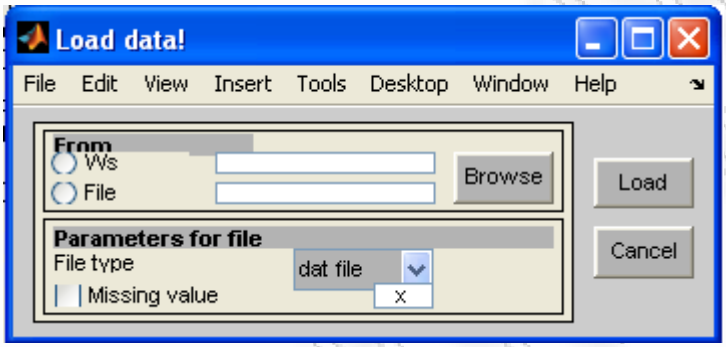


Figure 10. Selection of the data file which will be loaded and used for the initialization and the training.

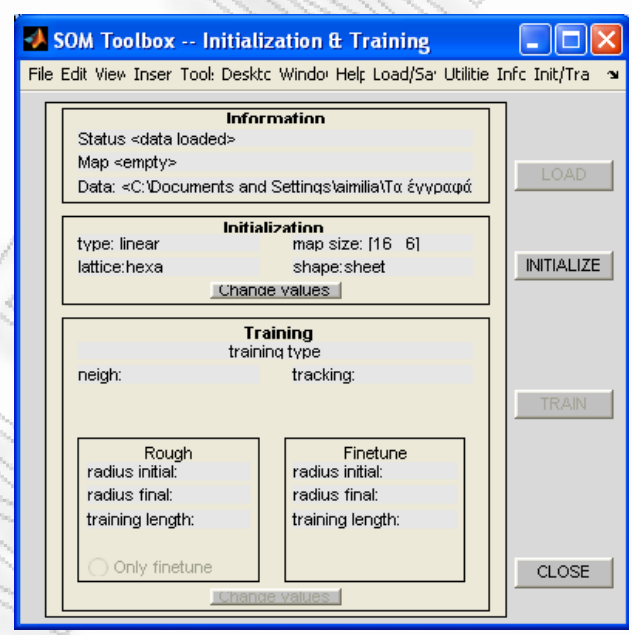


Figure 11. Initialization phase using the SOM GUI

Finally, the "shape" refers to the global map shape and may be "sheet", "cylinder" or "toroid", as depicted in Figure 5. Their values are set by default but in case one decides to change them, he may use the "change values" button and the "change initialization parameters" GUI appears (Figure 12). As soon as the new values are

selected, the "OK" button sets them in the SOM GUI. Finally, the "initialize" button (Figure 11) initializes the SOM according to the given data and the given preferences.

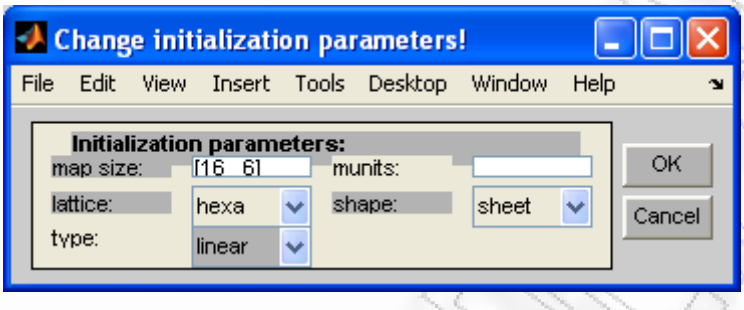


Figure 12. Change initialization parameters GUI

The next part of the SOM GUI is the training one. As soon as the initialization is completed the "train" and the "change values" buttons (in the training section) become available (Figure 13). In this part the user may choose the training algorithm as well as its parameters.

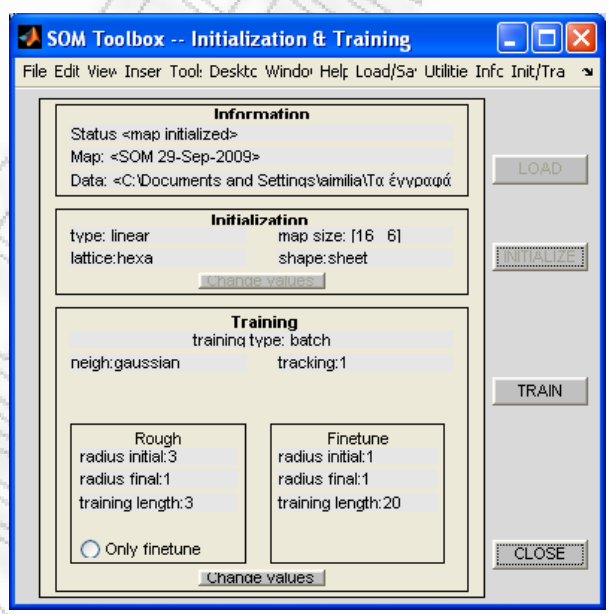


Figure 13. Training phase using the SOM GUI

Pressing the "change values" button will first trigger a new GUI on which one may choose between the "batch" and the "sequential" training algorithms (Figure 14).

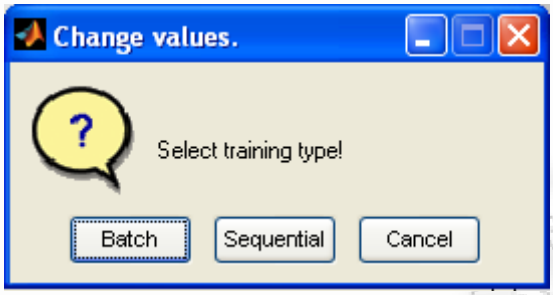


Figure 14. Choice of the training algorithm

In case of batch training algorithm the following GUI appears (Figure 15). The "mask" defines which of the variables will take part in the training of the SOM and which will not. In order to specify them each variable should be chose from the "set" and its value should be set to 1, in case of participation, or 0, in the opposite case. The mask is the string of zeros and ones with respect to the value of each variable. Another parameter that can be set through this GUI is the "tracking". Its possible level values are 0, 1 by default, 2 and 3. If the chosen level is 0, then in matlab command window the estimated time can be seen while in case of 1 the time is tracked in matlab command window. In case of 2 and 3, apart from the track of time in the command window, there are also available diagrams. If level 2 is the chosen the diagram concerns only the quantization error, while case 3 includes also a diagram of the two first components of the data sample. The last parameters that can be set through this GUI are the parameters of the batch training algorithm which are the neighborhood function ("neigh."), the initial and the final radius and the training length of both rough and fine-tuning phases (see chapter 2).

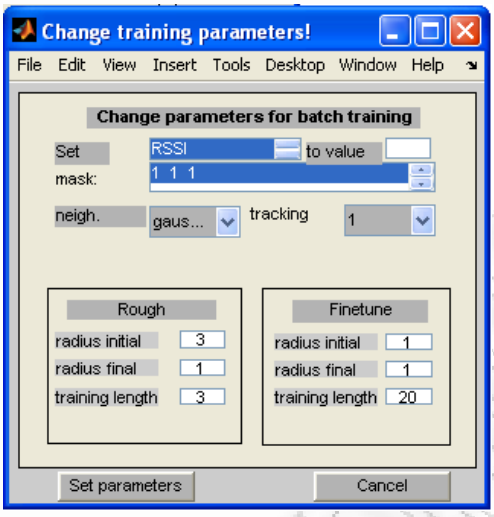


Figure 15. Change training parameters in case of batch training algorithm

In case of the sequential training algorithm the GUI of Figure 16 is the one that arises. Here, apart from the above parameters that can be changed there are also some more. These parameters exist only in this training algorithm and they concern the length type, the learning function, the order according to which the data will be used in the sequential training algorithm and the alpha initial of both rough and fine-tuning phases (see chapter 2).

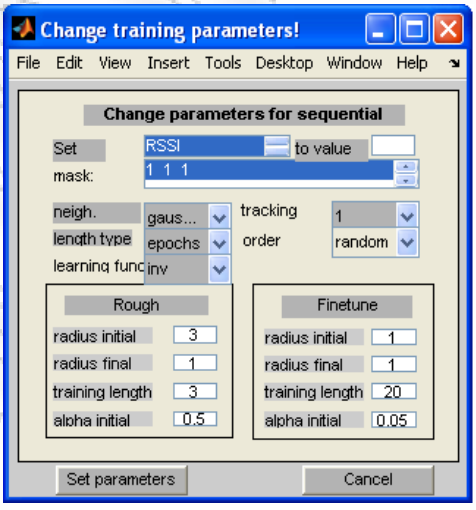


Figure 16. Change training parameters in case of sequential algorithm

The final step of the initial GUI is to close it. As soon as the "close" button is pressed, the GUI of Figure 17 appears, waiting for the user to choose whether he wants to save the created map or not. In our program it is necessary to save the

map as the file will be needed right after. The file will be saved as .cod file (Figure 18).

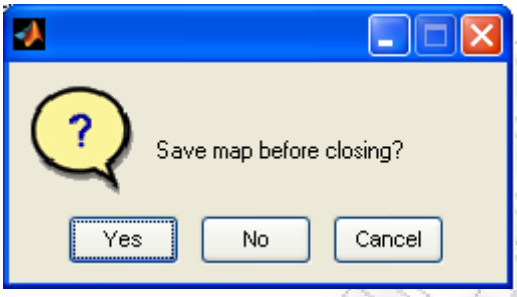


Figure 17. Save the trained SOM

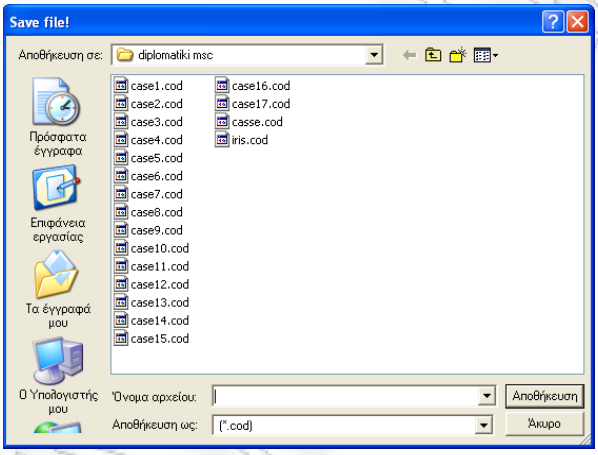


Figure 18. The cod-file will include the trained SOM

The next three functions of SOM TOOLBOX which are used in the training phase concern the labeling of the created SOM. These functions are som_read_cod and som_autolabel. Som_read_cod is responsible for the reading part of the cod file, which has been created and saved in the previous step.

Som_autolabel is a very important function in our program as it associates the som map structure created by the som_read_cod function with the data structure created by the som_read_data in order to add labels in the map and distinguish the data samples on it. As a result, except the som map structure and the data structure, one more parameter is needed. This parameter describes the way that the labels will be added. As mentioned above, this specific parameter is also the "source" of the three different versions of our program. In the first version, the

parameter is set to ‘vote’. This means that the only label which is added in the cell is the one which appears more frequently (Figure 19).

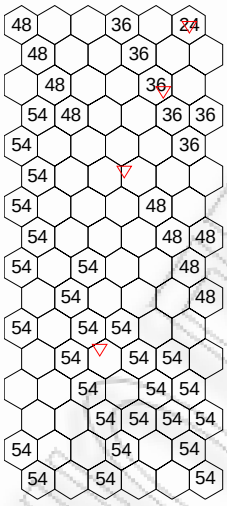


Figure 19. Example of som map using the VOTE version

In the second version the parameter is set to ‘add1’ which means that all labels of each cell are added. An example of this kind of labeling is shown in Figure 20. It’s worth mentioning that the labels of each cell are ordered according to their frequency even if their frequency is not shown.

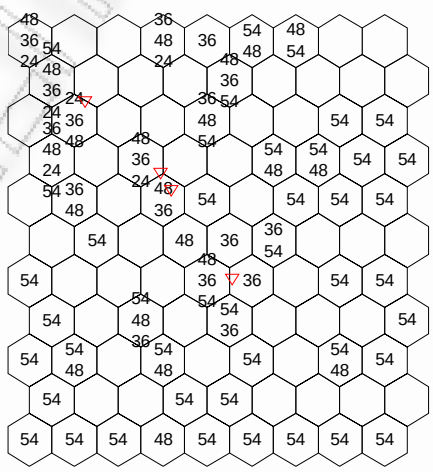
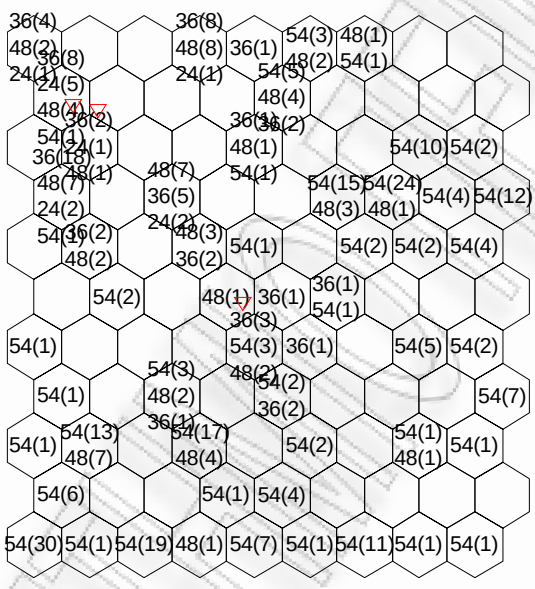


Figure 20. Example of som map using the ADD1 version

In the last version of our program the parameter is set to ‘freq’. The only difference between ‘add1’ and ‘freq’ lies in the fact that in the second case, apart from the labels, the number of instances of each label is mentioned in brackets next to the label. An example of the last choice used in our program is given at Figure 21.



case8.cod

Figure 21. Example of som map using the FREQ version

Finally, the three last functions are used for depicting the created SOM including the labels which were added before. These functions are som_grid, som_show and som_show_add. The first one creates a structure which consists of all the parameters accompanied by the labels and the coordinates of each cell. The coordinates will be useful later in the data evaluation part. The second function, using the parameter empty, draws an empty map on which the labels by the last function (som_show_add) are added.

At this point, the first phase of the program has been completed so the next phase to analyse is the data evaluation one. This phase differs from version to version. The VOTE and ADD1 versions evaluate the data in a similar way but the FREQ one is quite different.

In all versions the data evaluation is based on the calculation of the centre of each cluster (group of cells with the same label) and, after having found the BMU (best matching unit) of each data sample, which needs evaluation, the selection of the cluster and the label of the data sample according to the distance between the BMU of it and the centres of the clusters. As mentioned before, the difference of the three versions (VOTE, ADD1, FREQ) arises from the different labelling methods and lies in the way that they calculate the centres of the clusters.

The VOTE version calculates the centres by simply calculating the average of the coordinations. As mentioned above, each cell has only one label, so the calculation is simple. In the ADD1 version the calculation of the centres of the clusters is also the average of the coordinations of the cell with the same label but in this case each cell may have more than one label. However, each label is used with the exact same way without variations. On the other hand, the FREQ version depends on a weighted average of the coordinations of the cells with the same label. This means that each cell with the same label takes part in the calculation of the cluster centre according to its frequency in each cell which equals to the number of instances of this label in this cell divided by the sum of the number of instances of all labels in the same cell. So, each coordinate of the cell before its participation in the calculation, is multiplied with its frequency.

Finally, in all versions, the predicted values of bitrate are compared to the measured ones. The result of the comparison is also expressed in percent in the command window of matlab. Moreover, a diagram of the predicted bitrate, a diagram of measured bitrate and a comparison of the above are plotted in a new figure.

CHAPTER 5: TEST CASES AND RESULTS

A number of test cases that correspond to variations of input parameters of the proposed method have been set up in order to reach useful conclusions. In particular, the focus is placed on exploring the following: a) which the best choice between the three versions of the method (VOTE, ADD1 and FREQ) is, b) what variables of our data samples are going to be used, c) how many data samples are needed for the training phase and d) what the training algorithm and the values of its parameters should be. For evaluation and comparison reasons, the lower the percent of the wrong prediction is, the better the choice. As a result, the metric used was the number of data samples whose bitrate was wrongly predicted (expressed in percent). The different test cases are presented and compared to each other below for all three scenarios of normalization.

5.1 Comparison of the labelling versions

Having analyzed the three versions, we needed to compare them in order to use the best one according to their results.

5.1.1 Scenario of no normalization of the data samples

As mentioned above, the VOTE version uses only the label that appears more frequently when calculating the centres of the clusters. In this case, it is possible that a label doesn't appear in the created SOM even if it has been used as label in a data sample. This happens because different data samples with the same or different labels may have the same BMU but according to the som_autolabel(sm, sd, 'vote') function, and as mentioned in chapter 4, only the most frequently appeared label takes part into the calculations. As a result, less frequently appeared labels may not appear in the created SOM.

This causes the elimination of one or more clusters as there is no centre of them. In addition to the above, the program terminates a little after the calculation of the centres as according to its programming it needs all four centres. Finally, even if the program wouldn't stop, the data, which were used for evaluation and belonged to the eliminated cluster, would be correlated with a wrong cluster/label.

Trying to find a solution to the existing deficiency of VOTE version, ADD1 version was created. In ADD1 version, all possible labels of each cell participate equally. So, each label that was used in the data file as label of a data sample, even if it appeared only one time in a cell, was depicted at the created SOM. As a result, in ADD1 version, a label of a cell participated in the calculations of the centres of the clusters independently of its instances.

For comparison reasons between the first two versions, we executed both versions using the case8.data file in both training and evaluating phases. In order to do so, we executed version12.m file and chose between the two versions right after. In case of VOTE version, the centre of the cluster with label/bitrate equal to 24 was (NaN, NaN) which meant that this cluster ceased to exist. Finally, the program terminated mentioning an error as there was no centre calculated for the cluster with bitrate equal to 24. On the other hand, in case of ADD1 version, there was one centre for each cluster (24, 36, 48 and 54). Moreover, the evaluation completed with no matlab error and the percentage of wrong predictions was found 47.8%.

For further comparison, we repeated the experiment using another data file. The chosen file was case4.data. In this experiment both versions ended successfully in results. In case of VOTE version the percentage of wrong predictions was 41.9% while in case of ADD1 version it was 43.2%.

In conclusion, ADD1 version solved the problem of VOTE version but, in cases where VOTE version worked properly, ADD1 version had highest percentage of wrong predictions.

The above conclusion led us in a new version, the FREQ version. In this version, all types of labels of a cell appear followed by the number of instances that they are correlated to the cell. So, contrarily to ADD1 version, they participate in the calculation of the centres of the clusters unequally. Each cell that is correlated with the label takes part in the calculation of the centre of the cluster with the specific label according to the frequency of this label in this cell. As a result, the centres are a weighted average of this frequency (see chapter 4).

Having created this version all that was left to be done was its comparison with the VOTE and ADD1 versions. In order to do so, we executed FREQ version using the case4.data file in both training and evaluating phase. Executing version3.m file revealed a percentage of wrong predictions equal to 35.7%. It is worth mentioning that in all three versions the training parameters were the same. The results of FREQ version were all better (with smaller percentage of wrong bitrate) and thus FREQ version is the one that was used in the rest of the experiments.

5.1.2 Scenario of normalization of the data samples to 1

The logical process was similar to the one used in the scenario of no normalization of the data samples. The results were similar as well even if the data files were not all the same.

The first data file that was used was case1.data. As well as in scenario of no normalization of the data samples, in case of VOTE version, the centre of the cluster with label/bitrate equal to 24 was (NaN, NaN) which meant that this cluster ceased to exist and the program terminated mentioning an error as there was no centre calculated for the cluster with bitrate equal to 24. On the other hand, in case of ADD1 version, there was one centre for each cluster (24, 36, 48 and 54) and the evaluation completed with no matlab error resulting in a percentage of the wrong predictions equal to 47.4%.

Once again, we repeated the experiment using another data file. The chosen file was, again, case4.data. In this experiment both versions ended successfully in results. In case of VOTE version the percentage of wrong predictions was 37.8% while in case of ADD1 version it was 39.2%.

As a result, similarly to the above scenario, ADD1 version solved the problem of VOTE version but, in cases where VOTE version worked properly, ADD1 version had highest percentage of wrong predictions. So, FREQ version was tested as well.

In order to compare it with the VOTE and ADD1 versions, it was executed using the case4.data file in both training and evaluating phase. Executing version3.m file revealed a lower percentage of wrong predictions than the first two versions and equal to 25.9%. Since the results of FREQ version were again all better (with smaller percentage of wrong bitrate) FREQ version is the one that was used in the rest of the experiments, as well as in the 1st scenario.

5.1.3 Scenario of normalization of the variance of the data samples to 1

The logical process was similar to the one used in the first two scenarios, the data files which were used were the same with those of the 2nd scenario (case1.data, case4.data) while the results were similar but not always the same.

In the first test case, using case1.data for the comparison between the VOTE and the ADD1 versions, the results were exactly the same with those of the 2nd scenario. In case of the VOTE version, there was no centre for the cluster of bitrate equal to 24 and the program terminated mentioning matlab error while in case of the ADD1 version there was a centre for each cluster and the percentage of wrong predictions was found 47.4%.

In the second test case, using case4.data for the comparison between all three versions the results were the following:

- In case of VOTE version: 37.3%
- In case of ADD1 version: 39.7%
- In case of FREQ version: 28.1%

So, according to the above results, the main conclusions were the same with those of the first two scenarios. First of all, ADD1 version had solved the problem of the VOTE version but in cases where the VOTE version worked properly, its results were worse than those of the VOTE version leading to the need of the FREQ version. Finally, the FREQ version, ended in better results than those of both VOTE and ADD1 versions and was selected to be the one that was used for the rest of the experiments.

5.2 Selection of the variables of a data sample

The next step of our research concerns the variables of the data samples that suit better for predicting the bitrate. In order to do so we created many different cases which use the FREQ version of our program and the same training variables. Each test case uses a different data file. The difference between them lied in the number and the type of the variables of the data samples.

At the created cases there are nine variables of a data sample that are used in different combinations, namely: noise, RSSI (Received Signal Strength Identifier), number of input and output packets, number of input and output errors, number of input and output bytes and bitrate. The combinations of the variables for each case appear on the following table (Table 1).

So, using these cases with the FREQ version of our program, batch training algorithm, Gaussian neighborhood function and both rough and fine-tuning phases we received the results which appear on Table 2, Table 3 and Table 4. For the rough phase, the initial radius was 3 while the final one was 1 and the training

length was 3. On the other hand, the same parameters for the fine-tuning phase were 1, 1 and 20, respectively.

Table 1. Data files and their containing variables

Data file	RSSI	Noise	Input Packets	Output Packets	Input Errors	Output Errors	Input Bytes	Output Bytes	Bitrate
1	✓	✓							
2	✓	✓							✓
3	✓	✓							✓
4	✓		✓		✓		✓		
5	✓		✓		✓		✓		✓
6	✓	✓							
7	✓		✓	✓	✓		✓		
8	✓		✓	✓					
9	✓	✓	✓	✓	✓	✓	✓	✓	
10	✓		✓	✓			✓	✓	
11	✓						✓	✓	
12	✓		✓	✓	✓	✓			
14	✓		✓	✓					✓
15			✓	✓			✓	✓	
16	✓		✓						

5.2.1 Scenario of no normalization of the data samples

Before reaching conclusions from Table 2, we needed to compare same cases to each other. As it can be seen from Table 1, case1 and case6 data files contain the same variables. However, they are different. Data samples of case1 are ordered according to the bitrate while these of case6 are not. The same difference appears between case3 and case2 data files. Finally, in all cases, bitrate is used as label of the data samples but there are four data files (case2, case3, case5 and case14) which also use it as a variable of the data samples. This last choice is actually

wrong because it is not proper to use as input the variable which we are going to predict and is going to be our output. We only used these files in order to reach some conclusions.

Having the above observations in mind, we may state our conclusions. First of all, comparing case1 with case6 and case2 with case3, it is clear that the result does not depend on the fact that data samples are or are not ordered according to the bitrate. The results in both cases are the same.

Table 2. Scenario of no normalization of the data samples: The result of each case

Case	Percentage of wrong predictions
1	36.7%
2	29.4%
3	29.4%
4	35.7%
5	43.5%
6	36.7%
7	38.1%
8	28.6%
9	54.1%
10	54.1%
11	31.9%
12	67.3%
14	68.6%
15	54.1%
16	43.5%

Our second conclusion refers to the existence or not of bitrate as variable of the data sample. Comparing case 1 with case 2, case 4 with case 5 and case 8 with case 14, whose only difference is the existence of bitrate as variable in cases 2, 5 and 14, it is also clear that this existence does not influence the results always in the same way. In case 2 the result is reduced while in cases 5 and 14 it is

increased. Moreover, the case with the lowest percentage of wrong predictions is not one of these whose data file contains the bitrate as a variable.

The case with the lowest percentage, equal to 28.6%, is case 8 whose variables are the number of input and output packets and RSSI. Its variables are the ones that are used in the rest of our research.

5.2.2 Scenario of normalization of the data samples to 1

As in the first scenario, before reaching conclusions from Table 3, we needed to compare same cases to each other. Having the same observations in mind as in the 1st scenario, we may state our conclusions. First of all, comparing case1 with case6 and case2 with case3, similarly to the 1st scenario, it is clear that the result does not depend on the fact that data samples are or are not ordered according to the bitrate. The results in both cases are the same.

Our second conclusion refers again to the existence or not of bitrate as variable of the data sample. Contrarily to the 1st scenario, comparing case 1 with case 2, case 4 with case 5 and case 8 with case 14, whose only difference is the existence of bitrate as variable in cases 2, 5 and 14, it is clear that this existence influence the results always in the same way. In all cases the result is reduced. However, as mentioned before, data files which use bitrate as a variable of data samples cannot be used due to the fact that bitrate is supposed to be unknown in order to be predicted.

Finally, the case with the lowest percentage which does not use the bitrate as variable of the data samples, equal to 24.6%, was case 7 whose variables are the number of input and output packets, the number of input errors, the number of input Bytes and RSSI. Its variables are the ones that were used in the rest of our research.

Table 3. Scenario of normalization of the data samples to 1: The result of each case

Case	Percentage of wrong predictions
1	36.7%
2	14.3%
3	14.3%
4	25.9%
5	6.8%
6	36.7%
7	24.6%
8	50.8%
9	25.1%
10	25.1%
11	27.0%
12	48.6%
14	11.1%
15	51.6%
16	56.2%

5.2.3 Scenario of normalization of the variance of the data samples to 1

As well as in the first two scenarios, before reaching conclusions from Table 4, we needed to compare same cases to each other. In this scenario, the first two conclusions, referring to the ordered data samples according to the bitrate and the existence of the bitrate as parameter of the data samples, are the same with the 2nd scenario: the result does not depend on the fact that data samples are or are not ordered according to the bitrate, they are the same, and the existence of the bitrate as parameter of the data samples always reduces the results but the bitrate cannot be used as parameter of the data samples as it is the parameter in question.

Table 4. Scenario of normalization of the variance of the data samples to 1: The result of each case

Case	Percentage of wrong predictions
1	36.7%
2	16.7%
3	16.7%
4	28.1%
5	20.8%
6	36.7%
7	25.4%
8	25.4%
9	27.0%
10	25.7%
11	28.6%
12	60.0%
14	22.2%
15	43.5%
16	53.2%

Finally, the cases, which do not use bitrate as variable of the data samples, with the lowest percentage, equal to 25.4%, are case 7, whose variables are the number of input and output packets, the number of input errors, the number of input Bytes and RSSI, and case 8, whose variables are the number of input and output packets and RSSI. However, case 7 needs a better computer processor than case 8 because of the fact that it has more variables. So, between these two choices, the best one is case 8. Its variables are the ones that were used in the rest of our research.

5.3 Selection of the number of data samples

Having selected the variables of a data sample, we needed to decide the number of data samples to participate in the training process. In order to do so we created

cases which included the variables in which we resulted from the analysis in paragraph 5.2 for each scenario but different number of data samples.

For taking results we used once more the same parameters as before (FREQ version of the program, batch training algorithm, Gaussian neighborhood function, Rough phase: initial radius equal to 3, final one equal to 1 and training length equal to 3, Fine-Tuning phase: initial radius equal to1, final one equal to 1 and training length equal to 20).

5.3.1 Scenario of no normalization of the data samples

In this scenario the created cases were 7, include the variables RSSI, number of input and output packets and appear on the following table:

Table 5. Scenario of no normalization of the data samples: Cases with different number of data samples

case	number of data samples
8	370
13	617
17	300
19	844
20	748
21	718
22	668

Their results are depicted on Table 6:

Table 6. Scenario of no normalization of the data samples: Percentage of wrong predictions for each case

case	percentage of wrong predictions
8	28.6%
13	26.4%
17	38.0%
19	39.8%
20	38.8%
21	34.4%
22	37.4%

According to the above results, the number of data samples affects the results of our predictions but not always in the same direction. The diagram from these results is depicted in Figure 22:

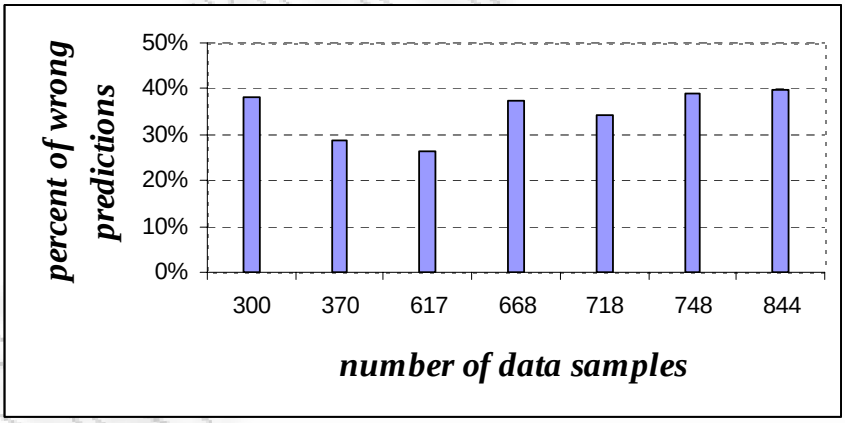


Figure 22. Scenario of no normalization of the data samples: Diagram of the percentage of wrong predictions according to the number of the used data samples

Finally, the minimum result is 26.4% and appears when the number of data samples is 617. Increasing the number of data samples up to 617 decreases the percentage of wrong predictions but increasing them more than that value appears to deteriorate the results. So, in case of no normalization of the data, the teat case which was used for our next tests is the 13th one.

5.3.2 Scenario of normalization of the data samples to 1

In this scenario the created cases were 9, include the variables RSSI, input and output packets, input errors and input Bytes and appear on the following table:

Table 7. Scenario of normalization of the data samples to 1: Cases with different number of data samples

case	number of data samples
7	370
23	617
24	844
25	623
26	704
27	644
28	534
29	434
30	324

The results are depicted onTable 8.

Table 8. Scenario of normalization of the data samples to 1: Percentage of wrong predictions for each case

case	Percentage of wrong predictions
7	24.6%
23	24.0%
24	25.9%
25	26.8%
26	26.4%
27	27.3%
28	27.0%
29	23.5%
30	25.3%

According to the above results, and similarly to the 1st scenario, the number of data samples affects the results of our predictions but not always in the same direction. The diagram from these results is depicted in Figure 23:

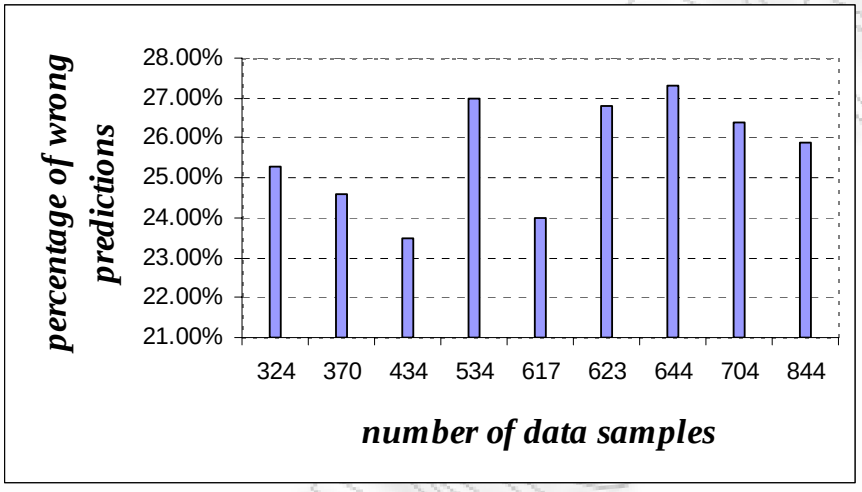


Figure 23. Scenario of normalization of the data samples to 1: Diagram of the percentage of wrong predictions according to the number of the used data samples

Analyzing the above diagram, it is easy to discern that the results fluctuate between the values 23.5 and 27.3. The diagram has two local minimum points ((434, 23.5), (617, 24.0)) and two local maximum points ((534, 27.0), (644, 27.3)). Finally, the minimum result is 23.5% and appears when the number of data samples is 434. So, the case which was used for our next tests during the scenario of normalization of the data samples to 1 is the 29th one.

5.3.3 Scenario of normalization of the variance of the data samples to 1

In this scenario the created cases were 7, include the variables RSSI, input and output packets and appear on the following table:

Table 9. Scenario of normalization of the variance of the data samples to 1: Cases with different number of data samples

case	number of data samples
8	370
13	617
17	300
19	844
20	748
21	718
22	668

The results are depicted on Table 10.

Table 10. Scenario of normalization of the variance of the data samples to 1: Percentage of wrong predictions for each case

case	Percentage of wrong predictions
8	25.4%
13	24.5%
17	30.0%
19	25.9%
20	27.3%
21	25.8%
22	23.1%

Once again, according to the above results, the number of data samples affects the results of our predictions but not always in the same direction. The diagram from these results is depicted in Figure 24:

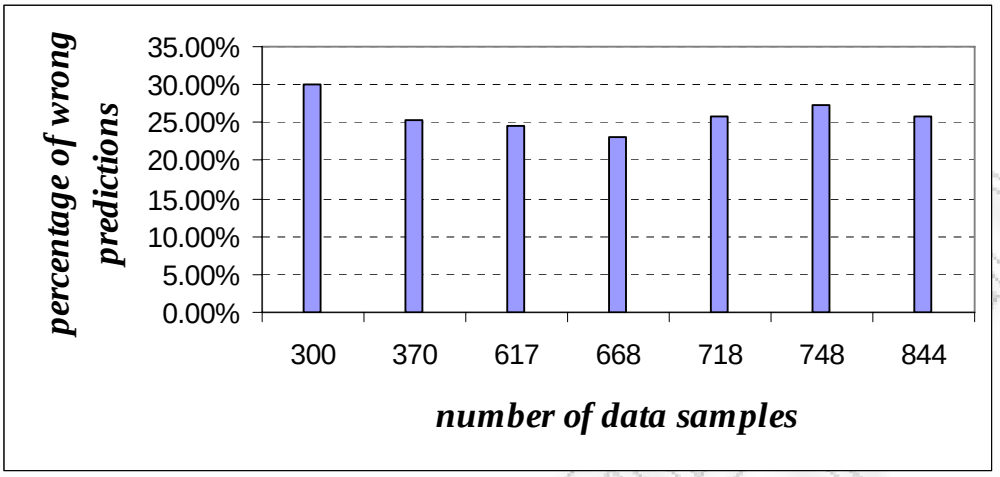


Figure 24. Scenario of normalization of the variance of the data samples to 1: Diagram of the percentage of wrong predictions according to the number of the used data samples

Finally, the minimum result is 23.1% and appears when the number of data samples is 668. Increasing the number of data samples up to 668 decreases the percentage of wrong predictions but increasing them more than that value appears to deteriorate the results. So, the case which was used for our next tests during this scenario of normalization is the 22nd one.

5.4 Selection of the training algorithm and its parameters

Our next concern was to decide between the two training algorithms and finding the most suitable values for their parameters. In order to make such a decision we firstly defined the most suitable values for each training algorithm separately and then we compared them to each other.

5.4.1 Scenario of no normalization of the data samples

In order to decide which are the most suitable values for the parameters of the batch training algorithm we tried different test cases changing only one parameter at a time. The parameters were tested randomly. These test cases appear in the following table:

Table 11. Scenario of no normalization of the data samples: Test cases using batch training algorithm in order to decide the most suitable values per each parameter

	Neighborhood Function	Rough phase			Fine-tuning phase			Percentage of wrong predictions
		Initial radius	Final radius	Training length	Initial radius	Final radius	Training length	
1	Gaussian	3	1	1	1	1	20	27.6%
2	Gaussian	3	1	2	1	1	20	28.5%
3	Gaussian	3	1	3	1	1	20	26.4%
4	Gaussian	3	1	4	1	1	20	26.4%
5	Gaussian	3	1	5	1	1	20	27.1%
6	Gaussian	3	1	6	1	1	20	26.3%
7	Gaussian	3	1	7	1	1	20	27.1%
8	Gaussian	3	1	6	1	1	60	26.7%
9	Gaussian	3	1	6	1	1	50	26.1%
10	Gaussian	3	1	6	1	1	45	27.1%
11	Gaussian	3	1	6	1	1	49	25.8%
12	Gaussian	3	1	6	1	1	48	25.8%
13	Gaussian	3	1	6	1	1	47	27.2%
14	Gaussian	4	1	6	1	1	48	26.3%
15	Gaussian	5	1	6	1	1	48	25.6%
16	Gaussian	6	1	6	1	1	48	26.4%
17	Gaussian	4	1	6	1	1	47	26.7%
18	Gaussian	5	1	6	1	1	47	26.9%
19	Gaussian	6	1	6	1	1	47	26.6%
20	Gaussian	7	1	6	1	1	47	26.3%
21	Gaussian	8	1	6	1	1	47	26.4%
22	Gaussian	9	1	6	1	1	47	26.9%
23	Gaussian	5	2	6	2	1	48	26.4%
24	Gaussian	5	3	6	3	1	48	26.3%
25	Gaussian	5	4	6	4	1	48	27.9%
26	Gaussian	5	5	6	5	1	48	29.0%
27	Gaussian	7	2	6	2	1	47	27.2%

28	Gaussian	7	3	6	3	1	47	29.0%
29	Gaussian	7	4	6	4	1	47	29.8%
30	Gaussian	7	5	6	5	1	47	26.7%
31	Gaussian	7	6	6	6	1	47	26.9%
32	Gaussian	7	7	6	7	1	47	27.2%
33	Gaussian	5	1	6	1	0	48	27.2%
34	Gaussian	7	1	6	1	0	47	28.2%
35	Cutgauss	5	1	6	1	1	48	31.9%
36	ep	5	1	6	1	1	48	27.6%
37	Bubble	5	1	6	1	1	48	25.8%

Comparing the results it is obvious that the best choice in the case of batch training algorithm, when data samples are not normalized, is the 15th one with the following values of the parameters:

- Neighborhood: Gaussian
- Initial radius for the rough phase: 5
- Final radius for the rough phase: 1
- Training length for the rough phase: 6
- Initial radius for the fine-tuning phase: 1
- Final radius for the fine-tuning phase: 1
- Training length for the fine-tuning phase: 48

Figure 25 depicts a) the predicted values of the bitrate, b) the real measured values of the bitrate and c) a comparison among the two above.

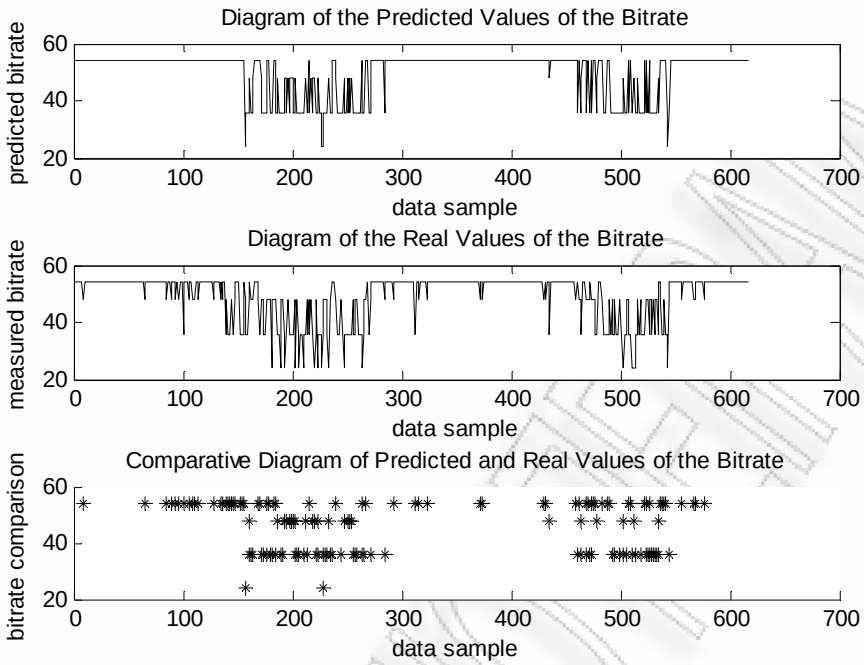


Figure 25. Batch training algorithm during the scenario of no normalization of data samples: Diagram of predicted values of bitrate, Diagram of their real values and Comparative Diagram of the above. The symbol * depicts the data samples which have different predicted and real values.

For the sequential training algorithm, following the same technique with the case of the batch training algorithm, we created different test cases in this situation as well. Although the technique is the same it's worth mentioning an important difference: in sequential training algorithm the samples do not enter the training phase at the same time. As a result, the order that they enter the system causes different results. In order to avoid such a situation we select the entrance of the samples to be ordered according the data file.

Test cases which examine the possible values of the initial and the final radius, the training length and the initial alpha during both rough and fine-tuning phases are shown on Table 13. During these cases the neighborhood function, the length type and the learning function are constant and equal to Gaussian, epochs and inv, respectively. According to these test cases, the best choice is the 13^{th} set of values. Keeping these parameters constant, the final parameters which are to be selected are the neighborhood function, the length type and the learning function. In order

to do so, we created the next seven cases where the first set parameters remained constant and we tested the rest of them. The results referring to these tests are listed below (Table 12).

Finally, according to both tables (Table 12 and Table 13) the best set of values for the sequential training algorithm when the data samples are not normalized is the following:

- ✓ Neighborhood function: Gaussian
- ✓ Length type: epochs
- ✓ Learning function: inv
- ✓ Initial radius for the rough phase: 3
- ✓ Final radius for the rough phase: 1
- ✓ Training length for the rough phase: 4
- ✓ Initial alpha for the rough phase: 0.5
- ✓ Initial radius for the finetuning phase: 1
- ✓ Final radius for the finetuning phase: 1
- ✓ Training length for the finetuning phase: 21
- ✓ Initial alpha for the finetuning phase: 0.05

Table 12. Scenario of no normalization of the data samples: Test cases with different sets of values of the neighborhood function, the length type and the learning function while the parameters of the rough and fine-tuning phases remain constant

Neighborhood function	Length type	Learning function	Percentage of wrong predictions
Cutgauss	Epochs	inv	30.5%
Ep	Epochs	inv	29.2%
Bubble	Epochs	inv	26.7%
Gaussian	Samples	inv	26.4%
Gaussian	Epochs	linear	25.8%
Gaussian	Epochs	power	25.6%

Table 13. Scenario of no normalization of the data samples: Test cases with different values of the initial and the final radius, the training length and the initial alpha during both phases while the neighborhood function is Gaussian, the length type is epochs and the learning function is inv.

	Rough phase				Finetuning phase				Percentage of wrong predictions
	Radius initial	Radius final	Training length	Alpha initial	Radius initial	Radius final	Training length	alpha initial	
1	3	1	2	0.5	1	1	20	0.05	33.1%
2	3	1	3	0.5	1	1	20	0.05	28.4%
3	3	1	4	0.5	1	1	20	0.05	26.1%
4	3	1	5	0.5	1	1	20	0.05	28.8%
5	3	1	6	0.5	1	1	20	0.05	27.9%
6	3	1	7	0.5	1	1	20	0.05	28.4%
7	3	1	4	0.5	1	1	40	0.05	28.0%
8	3	1	4	0.5	1	1	35	0.05	28.7%
9	3	1	4	0.5	1	1	30	0.05	28.4%
10	3	1	4	0.5	1	1	25	0.05	28.8%
11	3	1	4	0.5	1	1	45	0.05	28.2%
12	3	1	4	0.5	1	1	19	0.05	27.7%
13	3	1	4	0.5	1	1	21	0.05	24.6%
14	3	1	4	0.5	1	1	22	0.05	25.6%
15	4	1	4	0.5	1	1	21	0.05	28.2%
16	2	1	4	0.5	1	1	21	0.05	34.2%
17	5	1	4	0.5	1	1	21	0.05	28.7%
18	6	1	4	0.5	1	1	21	0.05	28.4%
19	7	1	4	0.5	1	1	21	0.05	28.8%
20	3	2	4	0.5	2	1	21	0.05	27.6%
21	3	3	4	0.5	3	1	21	0.05	30.1%
22	3	1	4	0.5	1	0	21	0.05	29.2%
23	3	1	4	1	1	0	21	0.05	28.5%
24	3	1	4	0.51	1	0	21	0.05	31.9%
25	3	1	4	0.49	1	0	21	0.05	28.2%
26	3	1	4	0.5	1	1	21	0.10	28.0%

27	3	1	4	0.5	1	1	21	0.051	27.7%
28	3	1	4	0.5	1	1	21	0.049	25.0%

As previously, Figure 26 depicts a) the predicted values of the bitrate , b) the real measured values of the bitrate and c) a comparison among the two above..

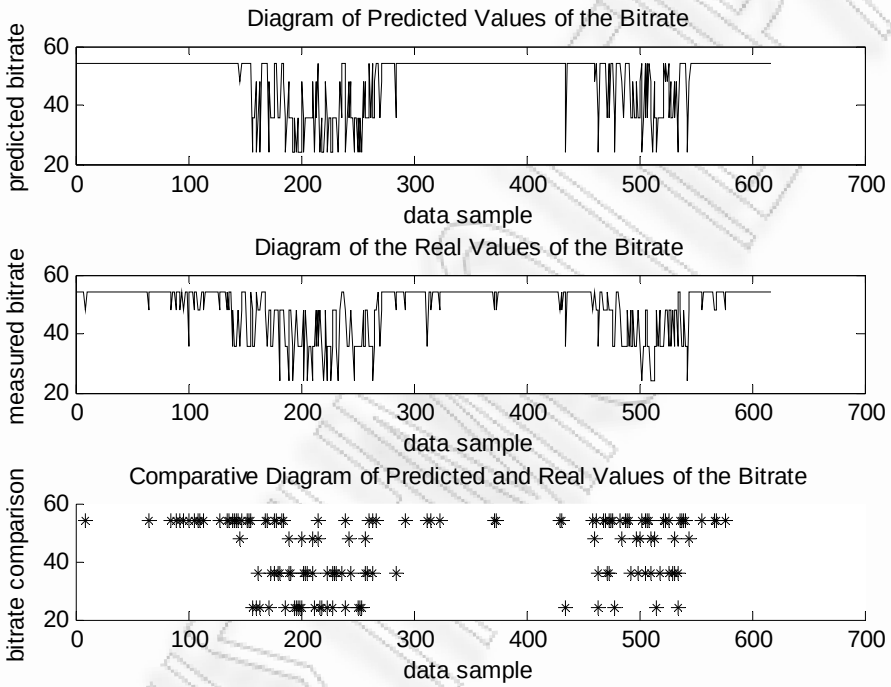


Figure 26. Sequential training algorithm during the scenario of no normalization of the data samples: Diagram of predicted values of bitrate, Diagram of their real values and Comparative Diagram of the above. The symbol * depicts the data samples which have different predicted and real values.

The comparison of the set of the values of the batch training algorithm with the one of the sequential training algorithm reveals that the first result, equal to 25.6%, is a little higher than the second one, equal to 24.6%. As the result refers to wrong predictions, the first impression is that the best choice is the sequential training algorithm. In addition we have measured the time that is needed to complete the training phase of the SOM, which is sometimes crucial. According to the program, batch training algorithm requires about 3 to 4 seconds to complete the training phase, while sequential one requires about the double time (7-8

seconds). As a result, and because of the fact that the difference between the two results is rather small, the choice between the two algorithms is subjective and depends on the existence or not of the requirement of a quick training.

5.4.2 Scenario of normalization of the data samples to 1

As in the first scenario of normalization, in order to decide which are the most suitable values for the parameters of the batch training algorithm we tried different test cases changing only one parameter at a time. The parameters were, once more, tested randomly. These test cases appear in the following table (Table 14).

Comparing the results it is obvious that the best result is 22.6% and appears in two cases, the 17th and the 29th one. We randomly selected case 17 where the parameters are the following:

- ✓ Neighborhood function: Gaussian
- ✓ Initial radius for the rough phase: 3
- ✓ Final radius for the rough phase: 1
- ✓ Training length for the rough phase: 1
- ✓ Initial radius for the fine-tuning phase: 1
- ✓ Final radius for the fine-tuning phase: 1
- ✓ Training length for the fine-tuning phase: 9

Table 14. Scenario of normalization of the data samples to 1: Test cases using batch training algorithm in order to decide the most suitable values for each parameter

	Neighborhood Function	Rough phase			Finetuning phase			Percentage of wrong predictions
		Initial radius	Final radius	Training length	Initial radius	Final radius	Training length	
1	Gaussian	3	1	1	1	1	20	22.8%
2	Gaussian	3	1	2	1	1	20	22.8%
3	Gaussian	3	1	3	1	1	20	23.5%
4	Gaussian	3	1	4	1	1	20	23.7%

5	Gaussian	3	1	5	1	1	20	23.5%
6	Gaussian	3	1	6	1	1	20	23.7%
7	Gaussian	3	1	7	1	1	20	23.7%
8	Gaussian	3	1	8	1	1	20	23.7%
9	Gaussian	3	1	9	1	1	20	23.7%
10	Gaussian	3	1	1	1	1	31	22.8%
11	Gaussian	3	1	1	1	1	30	22.8%
12	Gaussian	3	1	1	1	1	29	22.8%
13	Gaussian	3	1	1	1	1	13	22.8%
14	Gaussian	3	1	1	1	1	12	22.8%
15	Gaussian	3	1	1	1	1	11	22.8%
16	Gaussian	3	1	1	1	1	10	22.8%
17	Gaussian	3	1	1	1	1	9	22.6%
18	Gaussian	3	1	1	1	1	8	23.7%
19	Gaussian	3	1	1	1	1	5	23.3%
20	Gaussian	1	1	1	1	1	9	23.0%
21	Gaussian	2	1	1	1	1	9	22.8%
22	Gaussian	4	1	1	1	1	9	23.5%
23	Gaussian	5	1	1	1	1	9	23.0%
24	Gaussian	6	1	1	1	1	9	23.3%
25	Gaussian	3	0	1	2	0	9	23.0%
26	Gaussian	3	2	1	2	1	9	23.0%
27	Gaussian	3	3	1	3	1	9	23.5%
28	Gaussian	3	1	1	1	0	9	23.3%
29	Cutgauss	3	1	1	1	1	9	22.6%
30	Ep	3	1	1	1	1	9	26.5%
31	Bubble	3	1	1	1	1	9	23.5%

A diagram of predicted values of bitrate, a diagram of their real values and a comparative one of the above diagrams is depicted on Figure 27.

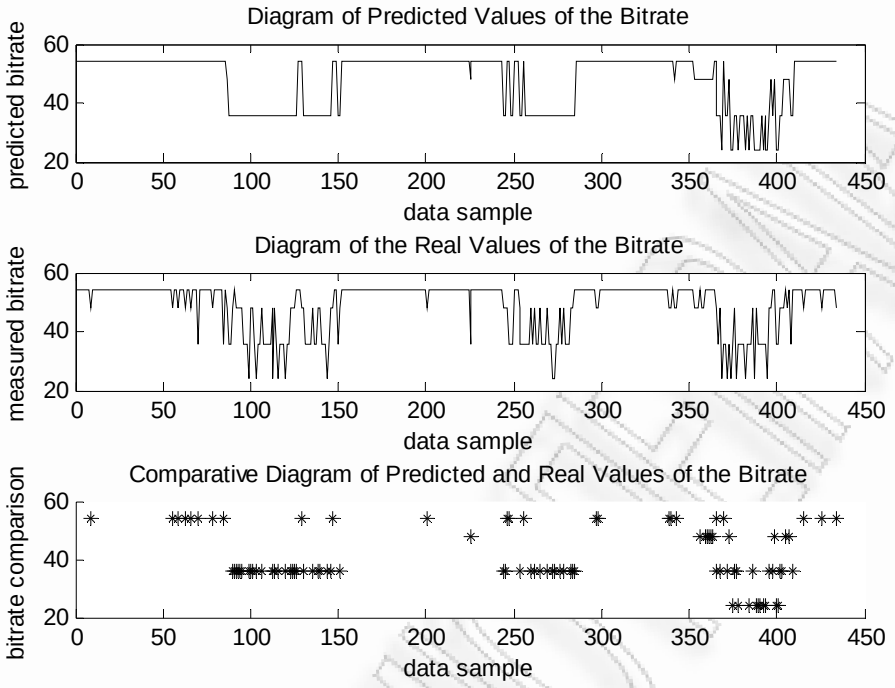


Figure 27. Batch training algorithm during scenario of normalization of the data samples to 1: Diagram of predicted values of bitrate, Diagram of their real values and Comparative Diagram of the above. The symbol * depicts the data samples which have different predicted and real values.

Once more, for the sequential training algorithm, we created different test cases in this situation as well and selected the ordered entrance of the data samples with respect to their order in the data file. Test cases which examine the possible values of the initial and the final radius, the training length and the initial alpha during both phases are shown on Table 16. As well as in the 1st scenario of normalization, during these cases the neighborhood function, the length type and the learning function are constant and equal to Gaussian, epochs and inv, respectively. According to these test cases, there are quite enough combinations whose result is equal to 22.6% and between which we can choose. Between these combinations, we randomly picked the 44th set of values. Keeping these parameters constant, the final parameters which are to be selected are the neighborhood function, the length type and the learning function. In order to do so, we created the next seven cases where the first set parameters remained constant and we tested the rest of them. The results referring to these tests are

listed below (Table 15**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**).

Finally, according to both tables (Table 16 and Table 15**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**) the best set of values for the sequential training algorithm is the following:

- ✓ Neighborhood function: Gaussian
- ✓ Length type: epochs
- ✓ Learning function: inv
- ✓ Initial radius for the rough phase: 4
- ✓ Final radius for the rough phase: 1
- ✓ Training length for the rough phase: 5
- ✓ Initial alpha for the rough phase: 0.5
- ✓ Initial radius for the finetuning phase: 1
- ✓ Final radius for the finetuning phase: 1
- ✓ Training length for the finetuning phase: 21
- ✓ Initial alpha for the finetuning phase: 0.055

Table 15. Scenario of normalization of the data samples to 1: Test cases with different sets of values of the neighborhood function, the length type and the learning function while the parameters of the rough and fine-tuning phases remain constant

Neighborhood function	Length type	Learning function	Percentage of wrong predictions
Cutgauss	Epochs	Inv	24.0%
Ep	Epochs	Inv	23.7%
Bubble	Epochs	Inv	24.2%
Gauss	Samples	Inv	25.8%
Gauss	Epochs	Linear	28.1%
Gauss	Epochs	Power	26.5%

Table 16. Scenario of normalization of the data samples to 1: Test cases with different values of the initial and the final radius, the training length and the initial alpha during both phases while the neighborhood function is Gaussian, the length type is epochs and the learning function is inv.

	Rough phase				Finetuning phase				Percentage of wrong predictions
	Radius initial	Radius final	Training length	Alpha initial	Radius initial	Radius final	Training length	alpha initial	
1	3	1	1	0.5	1	1	20	0.05	30.0%
2	3	1	2	0.5	1	1	20	0.05	27.0%
3	3	1	3	0.5	1	1	20	0.05	24.9%
4	3	1	4	0.5	1	1	20	0.05	24.4%
5	3	1	5	0.5	1	1	20	0.05	23.5%
6	3	1	6	0.5	1	1	20	0.05	24.2%
7	3	1	7	0.5	1	1	20	0.05	23.5%
8	3	1	8	0.5	1	1	20	0.05	24.2%
9	3	1	9	0.5	1	1	20	0.05	23.7%
10	3	1	10	0.5	1	1	20	0.05	24.4%
11	3	1	5	0.5	1	1	31	0.05	23.3%
12	3	1	5	0.5	1	1	30	0.05	23.0%
13	3	1	5	0.5	1	1	29	0.05	23.0%
14	3	1	5	0.5	1	1	28	0.05	23.0%
15	3	1	5	0.5	1	1	27	0.05	23.3%
16	3	1	5	0.5	1	1	25	0.05	23.3%
17	3	1	5	0.5	1	1	22	0.05	23.3%
18	3	1	5	0.5	1	1	21	0.05	23.0%
19	3	1	5	0.5	1	1	19	0.05	24.0%
20	3	1	5	0.5	1	1	10	0.05	24.4%
21	1	1	5	0.5	1	1	21	0.05	24.7%
22	2	1	5	0.5	1	1	21	0.05	24.7%
23	4	1	5	0.5	1	1	21	0.05	22.8%
24	5	1	5	0.5	1	1	21	0.05	25.6%
25	6	1	5	0.5	1	1	21	0.05	25.8%
26	4	2	5	0.5	2	1	21	0.05	24.7%

27	4	3	5	0.5	3	1	21	0.05	23.7%
28	4	4	5	0.5	4	1	21	0.05	24.0%
29	4	1	5	0.5	1	0	21	0.05	23.3%
30	4	1	5	1	1	1	21	0.05	24.4%
31	4	1	5	0.55	1	1	21	0.05	24.7%
32	4	1	5	0.51	1	1	21	0.05	23.3%
33	4	1	5	0.49	1	1	21	0.05	22.8%
34	4	1	5	0.48	1	1	21	0.05	22.8%
35	4	1	5	0.47	1	1	21	0.05	22.8%
36	4	1	5	0.46	1	1	21	0.05	22.8%
37	4	1	5	0.45	1	1	21	0.05	22.8%
38	4	1	5	0.44	1	1	21	0.05	23.3%
39	4	1	5	0.40	1	1	21	0.05	23.5%
40	4	1	5	0.5	1	1	21	0.10	23.3%
41	4	1	5	0.5	1	1	21	0.060	23.3%
42	4	1	5	0.5	1	1	21	0.057	23.3%
43	4	1	5	0.5	1	1	21	0.056	22.6%
44	4	1	5	0.5	1	1	21	0.055	22.6%
45	4	1	5	0.5	1	1	21	0.054	22.6%
46	4	1	5	0.5	1	1	21	0.053	22.6%
47	4	1	5	0.5	1	1	21	0.052	22.8%
48	4	1	5	0.5	1	1	21	0.051	22.8%
49	4	1	5	0.5	1	1	21	0.049	22.8%
50	4	1	5	0.5	1	1	21	0.048	22.8%
51	4	1	5	0.5	1	1	21	0.047	22.6%
52	4	1	5	0.5	1	1	21	0.046	23.0%
53	4	1	5	0.5	1	1	21	0.045	23.0%
54	4	1	5	0.5	1	1	21	0.040	23.5%

A diagram of predicted values of bitrate, a diagram of their real values and a comparative one of the above diagrams is depicted on Figure 28.

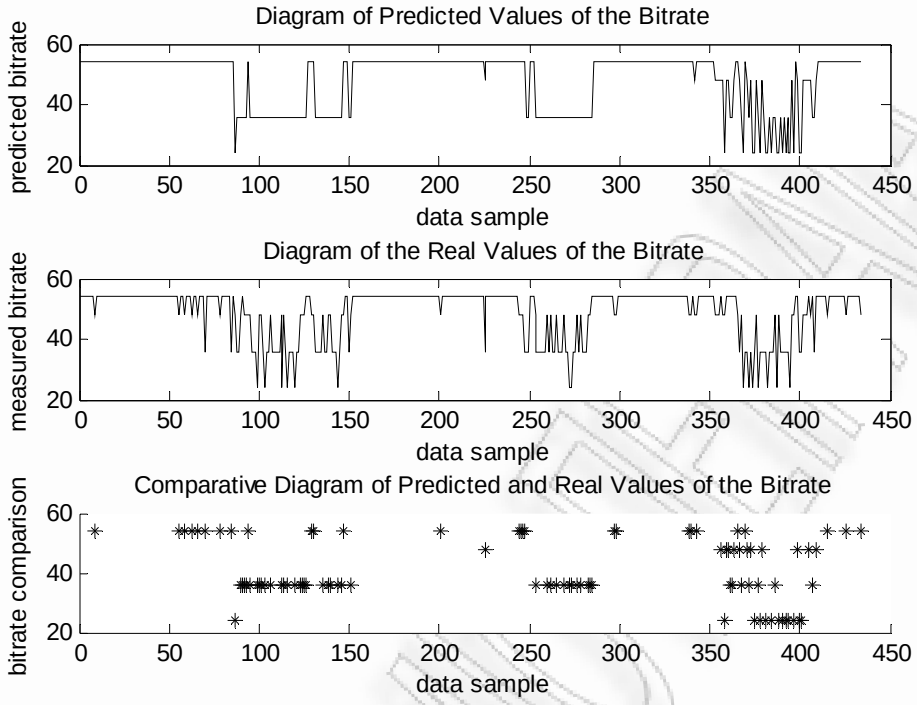


Figure 28. Sequential training algorithm during the scenario of normalization the data samples to 1: Diagram of predicted values of bitrate, Diagram of their real values and Comparative Diagram of the above. The symbol * depicts the data samples which have different predicted and real values.

Finally, the comparison of the set of the values of the batch training algorithm with the one of the sequential training algorithm reveals that both results are equal to 22.6%. However, as mentioned before, the time that is needed to complete the training phase of the SOM is sometimes crucial and batch training algorithm is quicker. According to the program, batch training algorithm requires less than one 1 second to complete the training phase while sequential one requires about four 4 to 5 seconds. As a result, the best choice between the two algorithms in this scenario of normalization is the batch training algorithm.

5.4.3 Scenario of normalization of the variance of the data samples to 1

Following the same process as in the first two scenarios of normalization, for the batch training algorithm we tried the test cases of the Table 17. Comparing the

results it is obvious that the best choice in this scenario in the case of batch training algorithm is the 29th one where

- ✓ Neighborhood function: Gaussian
- ✓ Initial radius for the rough phase: 4
- ✓ Final radius for the rough phase: 1
- ✓ Training length for the rough phase: 2
- ✓ Initial radius for the fine-tuning phase: 1
- ✓ Final radius for the fine-tuning phase: 1
- ✓ Training length for the fine-tuning phase: 10

Table 17. Scenario of normalization of the variance of the data samples to 1: Test cases using batch training algorithm in order to decide the most suitable values for each parameter

	Neighborhood Function	Rough phase			Finetuning phase			Percentage of wrong predictions
		Initial radius	Final radius	Training length	Initial radius	Final radius	Training length	
1	Gaussian	3	1	1	1	1	20	23.2%
2	Gaussian	3	1	2	1	1	20	23.1%
3	Gaussian	3	1	3	1	1	20	23.4%
4	Gaussian	3	1	4	1	1	20	24.9%
5	Gaussian	3	1	5	1	1	20	23.8%
6	Gaussian	3	1	6	1	1	20	25.3%
7	Gaussian	3	1	7	1	1	20	23.2%
8	Gaussian	3	1	8	1	1	20	25.4%
9	Gaussian	3	1	9	1	1	20	25.4%
10	Gaussian	3	1	2	1	1	60	24.3%
11	Gaussian	3	1	2	1	1	50	24.3%
12	Gaussian	3	1	2	1	1	40	24.3%
13	Gaussian	3	1	2	1	1	30	24.3%
14	Gaussian	3	1	2	1	1	17	23.1%
15	Gaussian	3	1	2	1	1	16	22.5%
16	Gaussian	3	1	2	1	1	15	21.9%
17	Gaussian	3	1	2	1	1	14	23.1%

18	Gaussian	3	1	2	1	1	13	22.5%
19	Gaussian	3	1	2	1	1	12	22.0%
20	Gaussian	3	1	2	1	1	11	22.3%
21	Gaussian	3	1	2	1	1	10	21.6%
22	Gaussian	3	1	2	1	1	9	23.7%
23	Gaussian	3	1	2	1	1	8	22.8%
24	Gaussian	3	1	2	1	1	7	23.1%
25	Gaussian	3	1	2	1	1	6	24.0%
26	Gaussian	3	1	2	1	1	5	24.6%
27	Gaussian	3	1	2	1	1	4	24.1%
28	Gaussian	2	1	2	1	1	10	22.3%
29	Gaussian	4	1	2	1	1	10	21.1%
30	Gaussian	5	1	2	1	1	10	22.8%
31	Gaussian	6	1	2	1	1	10	23.7%
32	Gaussian	4	2	2	2	1	10	24.4%
33	Gaussian	4	3	2	3	1	10	25.7%
34	Gaussian	4	4	2	4	1	10	26.5%
35	Gaussian	4	1	2	1	0	10	24.9%
36	Cutgauss	4	1	2	1	1	10	27.1%
37	Ep	4	1	2	1	1	10	24.7%
38	Bubble	4	1	2	1	1	10	26.8%

As previously, Figure 29 depicts a) the predicted values of the bitrate , b) the real measured values of the bitrate and c) a comparison among the two above.

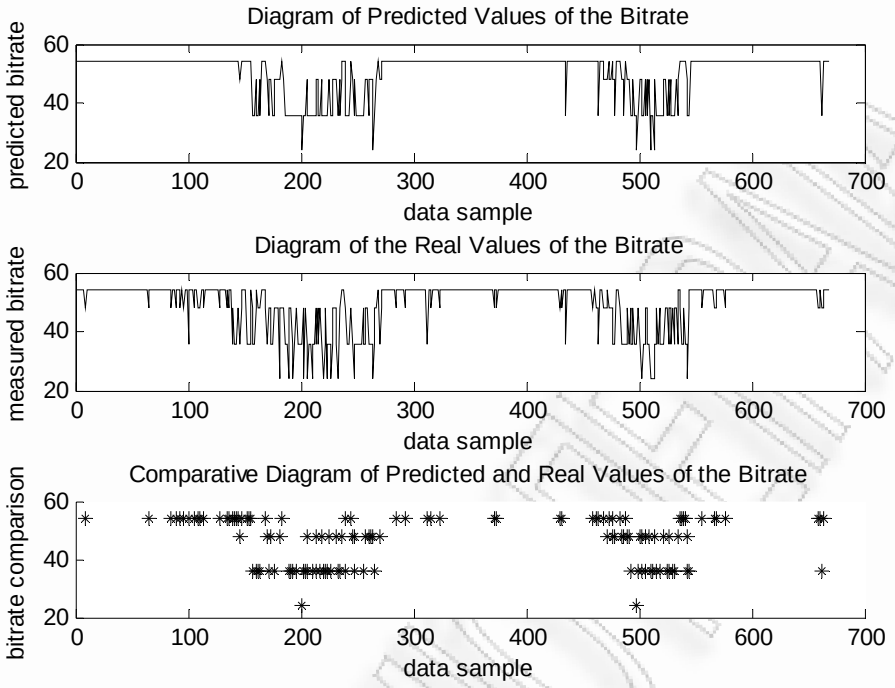


Figure 29. Batch training algorithm during the scenario of normalization of the variance of the data samples to 1: Diagram of predicted bitrate, Diagram of their real values and Comparative Diagram of the above. The symbol * depicts the data samples which have different predicted and real values.

Accordingly, in the sequential training algorithm we created different test cases and, as in the previous scenarios of normalization, we selected the ordered entrance of the samples. Test cases which examine the possible values of the initial and the final radius, the training length and the initial alpha during both phases are shown on Table 18. During these cases the neighborhood function, the length type and the learning function are once more constant and equal to Gaussian, epochs and inv, respectively. According to these test cases, there are quite enough combinations whose result is equal to 23.1% and between which we can choose. Between these combinations, we randomly picked the 8^{th} set of values. Keeping these parameters constant, the final parameters which are to be selected are the neighborhood function, the length type and the learning function. In order to do so, we created the next seven cases where the first set parameters remained constant and we tested the rest of them. The results referring to these tests are listed in Table 19.

Table 18. Scenario of normalization of the variance of the data samples to 1: Test cases with different values of the initial and the final radius, the training length and the initial alpha during both phases while the neighborhood function is Gaussian, the length type is epochs and the learning function is inv.

	Rough phase				Finetuning phase				Percentage
	Radius initial	Radius final	Training length	Alpha initial	Radius initial	Radius final	Training length	alpha initial	of wrong predictions
1	3	1	1	0.5	1	1	20	0.05	25.3%
2	3	1	2	0.5	1	1	20	0.05	25.6%
3	3	1	3	0.5	1	1	20	0.05	24.6%
4	3	1	4	0.5	1	1	20	0.05	25.4%
5	3	1	5	0.5	1	1	20	0.05	24.4%
6	3	1	6	0.5	1	1	20	0.05	24.6%
7	3	1	7	0.5	1	1	20	0.05	23.5%
8	3	1	8	0.5	1	1	20	0.05	23.1%
9	3	1	9	0.5	1	1	20	0.05	24.9%
10	3	1	10	0.5	1	1	20	0.05	24.7%
11	3	1	11	0.5	1	1	20	0.05	24.0%
12	3	1	12	0.5	1	1	20	0.05	24.1%
13	3	1	8	0.5	1	1	30	0.05	23.7%
14	3	1	8	0.5	1	1	25	0.05	23.7%
15	3	1	8	0.5	1	1	21	0.05	24.4%
16	3	1	8	0.5	1	1	19	0.05	24.9%
17	3	1	8	0.5	1	1	15	0.05	23.4%
18	3	1	8	0.5	1	1	10	0.05	23.8%
19	3	1	8	0.5	1	1	5	0.05	25.4%
20	1	1	8	0.5	1	1	20	0.05	23.7%
21	2	1	8	0.5	1	1	20	0.05	23.4%
22	4	1	8	0.5	1	1	20	0.05	24.9%
23	5	1	8	0.5	1	1	20	0.05	23.5%
24	6	1	8	0.5	1	1	20	0.05	25.0%
25	3	2	8	0.5	2	1	20	0.05	23.1%

26	3	3	8	0.5	3	1	20	0.05	27.1%
27	3	1	8	0.5	1	1	20	0.05	24.9%
28	3	1	8	1	1	1	20	0.05	24.0%
29	3	1	8	0.55	1	1	20	0.05	24.7%
30	3	1	8	0.54	1	1	20	0.05	23.2%
31	3	1	8	0.53	1	1	20	0.05	23.1%
32	3	1	8	0.52	1	1	20	0.05	23.1%
33	3	1	8	0.51	1	1	20	0.05	23.1%
34	3	1	8	0.49	1	1	20	0.05	24.9%
35	3	1	8	0.5	1	1	20	0.10	23.5%
36	3	1	8	0.5	1	1	20	0.056	23.7%
37	3	1	8	0.5	1	1	20	0.055	23.1%
38	3	1	8	0.5	1	1	20	0.054	23.1%
39	3	1	8	0.5	1	1	20	0.053	23.1%
40	3	1	8	0.5	1	1	20	0.052	23.1%
41	3	1	8	0.5	1	1	20	0.051	23.1%
42	3	1	8	0.5	1	1	20	0.049	24.9%

Table 19. Scenario of normalization of the variance of the data samples to 1: Test cases with different sets of values of the neighborhood function, the length type and the learning function while the parameters of the rough and fine-tuning phases remain constant

Neighborhood function	Length type	Learning function	Percentage of wrong predictions
Cutgauss	Epochs	inv	24.7%
Ep	Epochs	inv	22.5%
Bubble	Epochs	inv	26.5%
Ep	Samples	inv	26.2%
Ep	Epochs	linear	25.0%
Ep	Epochs	power	23.2%

Finally, according to both tables (Table 18 and Table 19) the best set of values for the sequential training algorithm during this scenario of normalization is the following:

- ✓ Neighborhood function: Ep
- ✓ Length type: epochs
- ✓ Learning function: inv
- ✓ Initial radius for the rough phase: 3
- ✓ Final radius for the rough phase: 1
- ✓ Training length for the rough phase: 8
- ✓ Initial alpha for the rough phase: 0.5
- ✓ Initial radius for the finetuning phase: 1
- ✓ Final radius for the finetuning phase: 1
- ✓ Training length for the finetuning phase: 20
- ✓ Initial alpha for the finetuning phase: 0.05

A diagram of predicted bitrate, a diagram of their real values and a comparison of the above diagrams are depicted on Figure 30.

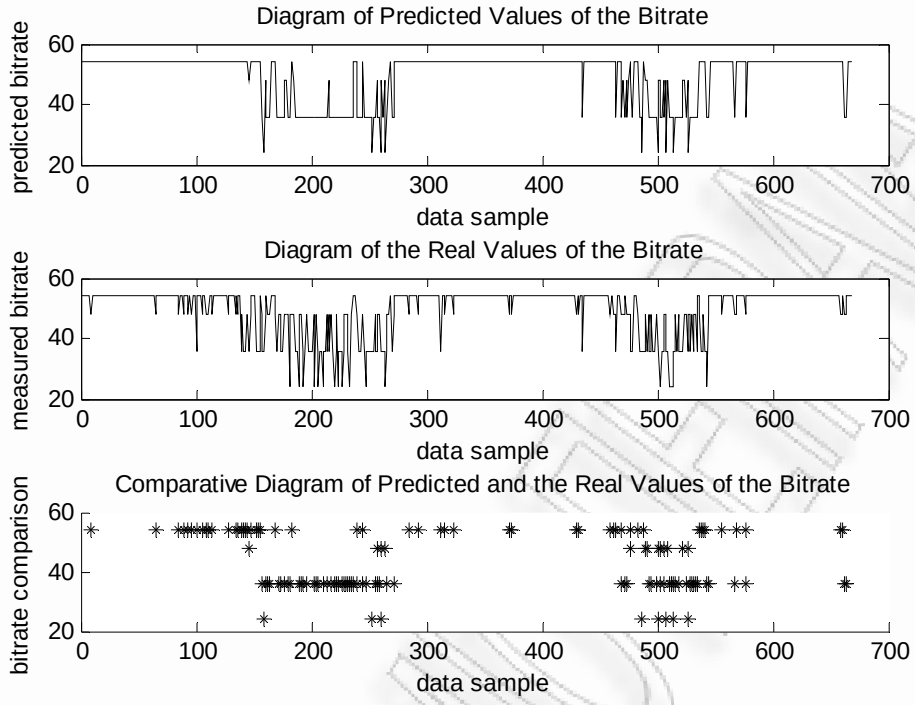


Figure 30. Sequential training algorithm during the scenario of normalization of the variance of the data samples to 1: Diagram of predicted bitrate, Diagram of their real values and Comparative Diagram of above. The symbol * depicts the data samples which have different predicted and real values.

Finally, the comparison of the set of the values of the batch training algorithm with the one of the sequential training algorithm in this scenario of normalization reveals that the first result, equal to 21.1%, is a little less than the second one, equal to 22.5%. Moreover, as well as in the first two scenarios of normalization, batch training algorithm proved to be quicker during the training phase as it requires less than 1 second while sequential one requires about 7 to 8 seconds. As a result, batch training algorithm is once more our best choice.

CHAPTER 6: CONCLUSIONS

Rapid evolution of wireless communications demands the use of systems capable of intelligently adapting to the highly varying and disparate modern environments. In these terms, Cognitive Radio Systems have been a very promising technology but the cognition process, which they utilize in order to monitor, evaluate and select a radio configuration to operate with, is often time-consuming, thus leading to the necessity of a learning technique for speeding it up. In this paper we used an unsupervised leaning technique, Self-Organizing Map, in order to train a CRS to predict the bitrate that can be achieved under a combination of output parameters of a configuration and based on its past experience. Going through numerous test cases we achieved to predict correctly the bitrate at 79.9% of the tested data samples. Such a method is expected to assist CRS to choose among the different candidate configurations by taking into account the predictions of the bitrate that can be achieved.

BIBLIOGRAPHY

Articles

1. K. Tsagkaris, A. Katidiotis, P. Demestichas, Neural Network-based Learning schemes for Cognitive Radio systems, Computer Communications, Vol. 31, Issue 14, pp. 3394-3404, September 2008
2. P. Demestichas, A. Katidiotis, K. Tsagkaris, E. Adamopoulou, K. Demestichas, Enhancing Channel Estimation in Cognitive Radio Systems by means of Bayesian Networks, Wireless Personal Communications, Vol. 49, Issue 1, pp 87-105, April 2009
3. A. Katidiotis, K. Tsagkaris, P. Demestichas, Performance Evaluation of Artificial Neural Network-Based Learning Schemes for Cognitive Radio Systems, Computers & Electrical Engineering, Elsevier, in Press
4. Panagiotis Demestichas, George Dimitrakopoulos, John Strassner, Didier Bourse, Introducing Reconfigurability and Cognitive Networks Concepts in the Wireless World, IEEE Vehicular Technology Magazine, pp. 32-39, June 2006
5. Panagiotis Demestichas, Vera Stavroulaki, Dragan Boscovic, Al Lee, John Strassner, m@ANGEL: Autonomic Management Platform for Seamless Cognitive Connectivity to the Mobile Internet, IEEE Communications Magazine, June 2006
6. Teuvo Kohonen, An Introduction to Neural Computing, Neural Networks, Vol. 1, pp. 3-16, 1998
7. J. Mitola III, G. Maguire Jr., Cognitive radio: making software radios more personal, IEEE Personal Commun., Vol. 6, No 4 , pp13 -18, August 1999
8. S. Haykin, Cognitive radio: brain-empowered wireless communications, IEEE Journal on Selected Areas In Communications, Vol. 23, No. 2, pp. 201-220, Feb. 2005

9. Teuvo Kohonen, The Self-Organizing map, Elsevier, Neurocomputing 21 (1998) 1-6
10. J. Vesanto, J. Himberg, E. Alhoniemi, J. Parhankangas, SOM Toolbox for Matlab 5, April 2000
11. E. Alhoniemi, J. Himberg, J. Hollmén, S. Laine, P. Lehtimäki, K. Raivio, T. Similä, O. Simula, M. Sirola, M. Sulkava, J. Tikka, J. Vesanto, SOM in data mining, Chapter 14, pp. 171-178
12. Teuvo Kohonen, Self-Organizing neural projections, Elsevier, Neural Networks 19 (2006) 723-733
13. M. Liang, J. Shen, G. Wang, Identification of illicit drugs by using SOM neural networks, IOP Publishing, Journal Of Physics D: Applied Physics 41 (2008)
14. H. Tokutaka, K. Yoshihara, K. Fujimura, K. Obu-Cann, K. Iwamoto, Application of self-organizing maps to chemical analysis, Elsevier, Applied Surface Science 144-145 (1999) 59-63
15. Jari Kangas, Teuvo Kohonen, Developments and applications of the self-organizing map and related algorithms, Elsevier, Mathematics and Computers in Simulation 41 (1996) 3-12
16. Samuel Kaski, Timo Hankela, Krista Lagus, Teuvo Kohonen, WEBSOM – Self-organizing maps of document collections, Elsevier, Neurocomputing 21 (1998) 101-117
17. Teuvo Kohonen, Panu Somervuo, Self-Organizing Maps of Symbol, Strings with Appplication to Speech Recognition
18. Yasaburo Matsuura, Tuoya, Masaharu Seno, Heizo Tokutaka, Masaaki Ohkita, The Identification of a Cancer Cell Gene by using SOM
19. P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander, T. R. Golub, Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation, Proc. Natl. Acad. Sci. USA, Vol. 96, pp. 2907-2912, March 1999

20. R. Céréghino, Y. – S. Park, Review of the Self-Organizing Map (SOM) approach in water resources: Commentary, Elsevier, Environmental Modelling & Software 24 (2009) 945-947
21. Vesanto J., Himberg J., Alhoniemi E., Parhankangas J., 2000. SOM Toolbox for Matlab 5. Technical Report A57. Neural Networks Research Centre, Helsinki University of Technology, Helsinki, Finland.
22. Park Y.-S., Tison J., Lek S., Giraudel J.-L., Coste M., Delmas F., 2006. Application of a self-organizing map to select representative species in multivariate analysis: a case study determining diatom distribution patterns across France. Ecological Informatics 1, 247-257.

Websites

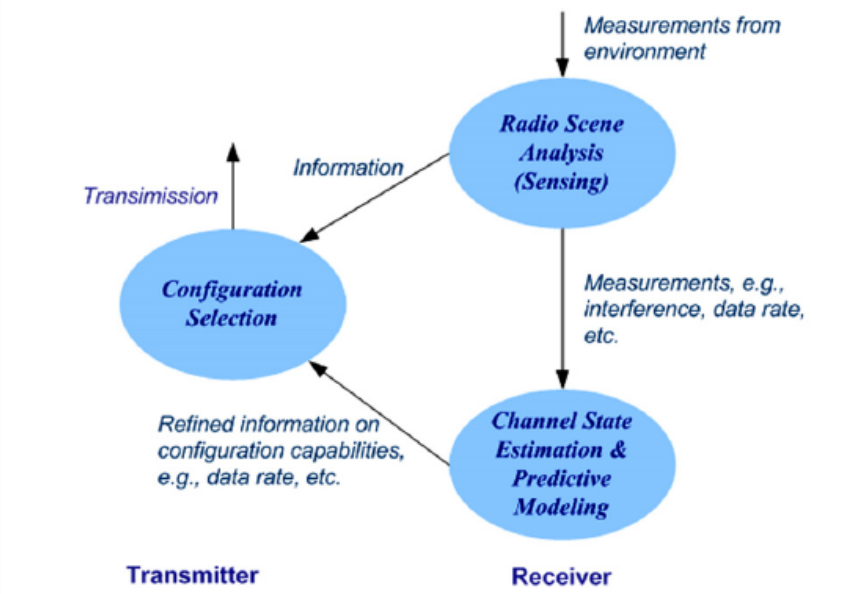
23. http://www.cis.hut.fi/projects/ide/publications/html/mastersJV97
24. http://www.cis.hut.fi/projects/somtoolbox/documentation
25. http://www.cis.hut.fi/projects/somtoolbox/download

ANNEX A: SUMMARY IN GREEK LANGUAGE

ΚΕΦΑΛΑΙΟ 1: Εισαγωγή

Την σήμερον ημέρα, όλο και περισσότερες επικοινωνίες γίνονται ασύρματες απαιτώντας η καθεμία την δική της ζώνη ασύρματων συχνοτήτων ηλεκτρομαγνητικού πεδίου. Παρ' όλ' αυτά η εν λόγω φυσική πηγή είναι περιορισμένη. Επιπλέον, η στατική ανάθεση συχνοτήτων που εφαρμόζεται συχνά οδηγεί στην μη εκτενή αξιοποίηση του ηλεκτρομαγνητικού φάσματος με αποτέλεσμα να καθίσταται αναγκαία η δημιουργία μιας τεχνολογίας που θα έχει την δυνατότητα να εξερευνά τις συχνότητες με μικρή αξιοποίηση.

Την απάντηση σε αυτή την ανάγκη έρχονται να καλύψουν τα «γνωστικά ασύρματα συστήματα» (Cognitive Radio Systems) [7][8]. Τα συστήματα cognitive έχουν την δυνατότητα να προσαρμόζουν την λειτουργία τους σύμφωνα με τις εξωτερικές περιβαλλοντικές συνθήκες, τις απαιτήσεις των χρηστών/εφαρμογών και την προηγούμενη εμπειρία τους. Βάσει αυτής τους της λειτουργίας, τα μελλοντικά αυτά συστήματα θα έχουν την δυνατότητα να αλλάζουν τις παραμέτρους τους (συχνότητα φέροντος διαύλου, τεχνολογία ασύρματης πρόσβασης (radio access technology - RAT), ισχύς εκπομπής, τύπος διαμόρφωσης), να παρακολουθούν τα αποτελέσματα και να αποφασίζουν για τον βέλτιστο συνδυασμό αυτών των παραμέτρων με σκοπό να φτάσουν στην βέλτιστη λειτουργικής τους κατάσταση. Ως εκ τούτου, με βάση την λογική της ευμετάβλητης διαχείρισης του φάσματος συχνοτήτων, η χρήση των «γνωστικών συστημάτων» θα καταστήσει δυνατή την χρήση μίας ζώνης συχνοτήτων σε περαιτέρω της μίας τεχνολογίας ασύρματης πρόσβασης (RATs) [2], [3].

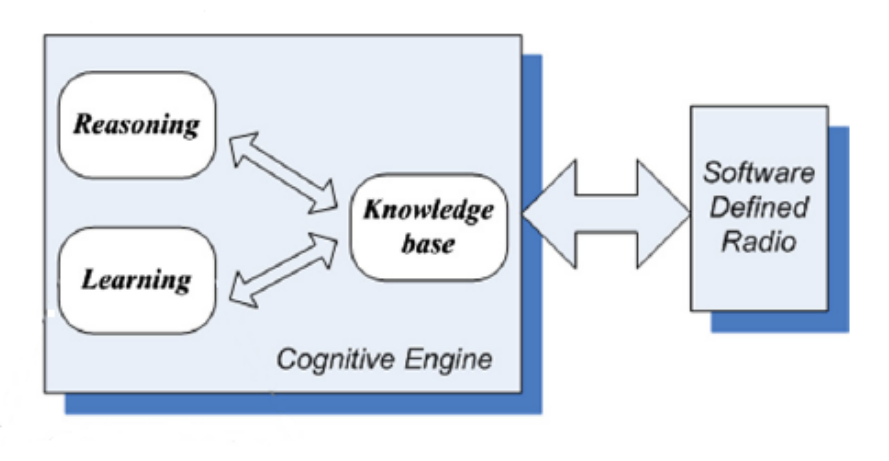


Εικόνα 1. Απλοποιημένη Αναπαράσταση του κύκλου λειτουργίας ενός Γνωστικού Ασύρματου Συστήματος (Cognitive Radio Cycle) [2]

Η λειτουργία ενός cognitive συστήματος αποτελείται από τις τρείς φάσεις που απεικονίζονται στην Εικόνα 1 [2][8]. Κατά την διάρκεια της πρώτης φάσης, φάση ανάλυσης ραδιοσυχνοτήτων (radio scene analysis), το σύστημα λαμβάνει μετρήσεις από το περιβάλλον του (π.χ τις πιθανές παρεμβολές) και δοκιμάζει την εφαρμογή διαφόρων συνδυασμών παραμέτρων. Στην δεύτερη φάση, εκτίμηση του καναλιού και μοντέλο πρόβλεψης (channel estimation and predictive modeling), χρησιμοποιείται το αποτέλεσμα της $1^{ης}$ φάσης για την φάση της εξερεύνησης και η παλαιότερη «εμπειρία» του συστήματος. Τέλος, στην τελευταία φάση, φάση επιλογής παραμέτρων (configuration selection), το σύστημα αποφασίζει τον βέλτιστο συνδυασμό των παραμέτρων του και στέλνει τις επιλεγμένες, από τις προηγούμενες δύο φάσεις του κύκλου λειτουργίας του συστήματος, τιμές των παραμέτρων.

Η περαιτέρω ανάλυση ενός «γνωστικού συστήματος» αποκαλύπτει την ύπαρξη ενός λειτουργικού πακέτου, γνωστό ως «γνωστική» μηχανή (cognitive engine – Εικόνα 2), το οποίο ευθύνεται για τις δυνατότητές του [3]. Συγκεκριμένα, οι cognitive μηχανές υποστηρίζουν τα λειτουργικά καθορισμένα ασύρματα

συστήματα αλλάζοντας τις τιμές των παραμέτρων τους, παρατηρώντας τα αποτελέσματα αυτών των αλλαγών και λαμβάνοντας μετρήσεις και ενέργειες που καθορίζονται από την επιθυμητή κατάσταση λειτουργίας του συστήματος. Αντιστρόφως, τα λειτουργικά καθορισμένα ασύρματα συστήματα εκπαιδεύονται και αποθηκεύουν τα αποτελέσματα στην βάση δεδομένων (knowledge base) των cognitive μηχανών. Ένα άλλο τμήμα της «γνωστικής» μηχανής, γνωστό ως «λογική» (reasoning), αποφασίζει για την καταλληλότητα ή όχι των ενεργειών δεδομένης μιας περιβαλλοντικής κατάστασης. Τέλος, καθώς η τελευταία διαδικασία μπορεί να είναι επίπονη και να απαιτεί πολύ χρόνο, είναι απαραίτητη και μια διαδικασία εκπαίδευσης.



Εικόνα 2. Αναπαράσταση ενός κύκλου εργασιών ενός cognitive συστήματος

Στο παρελθόν έχουν γίνει διάφορες μελέτες για την εύρεση μια διαδικασίας εκπαίδευσης όπως σχήματα βασισμένα σε νευρωνικά [1][3] και σε Bayesian δίκτυα [2]. Στην δική μας περίπτωση, πρόκειται για μια ανεξέλεγκτη τεχνική νευρωνικών δικτύων γνωστή ως Χάρτες Αυτό-Οργάνωσης (Self-Organizing Maps - SOMs). Οι Χάρτες Αυτό-Οργάνωσης είναι μια τεχνική απεικόνισης και κατηγοριοποίησης πολυδιάστατων δεδομένων σε δισδιάστατους χάρτες. Αυτοί οι χάρτες αποτελούνται από τετραγωνικές ή εξαγωνικές κυψέλες οι οποίες βρίσκονται πάνω σε πλέγμα κανονικού συστήματος ενώ κάθε δείγμα δεδομένων αντιστοιχίζεται σε μία κυψέλη/νευρώνα του χάρτη ώστε να είναι όσο το δυνατόν πιο κοντά στα υπόλοιπα συγγενικά δείγματα. Υπό αυτήν την έννοια, ο χάρτης που δημιουργείται αναπαριστά την ομοιότητα των δειγμάτων δεδομένων και την

κατηγοριοποίησή τους. Λόγω, λοιπόν, του συγκεκριμένου χαρακτηριστικού της τεχνικής, η τεχνική είναι πολύ δημοφιλής σε προβλήματα «ανακάλυψης» δεδομένων (data mining) όπως η αναγνώριση παράνομων ναρκωτικών ουσιών [13], χημικής ανάλυσης [14], συλλογής δοκιμίων [16], αναγνώρισης φωνής [17], αναγνώριση καρκινικού κυττάρου [18] και διαφοροποίηση αιματικών κυττάρων [19]. Στην δική μας περίπτωση εξετάστηκε η πιθανότητα σύνδεσης παραμέτρων που παρατηρούνται ως αποτέλεσμα μιας διαμόρφωσης, όπως είναι ο θόρυβος, ο δείκτης έντασης ληφθέντος σήματος (received signal strength indication – RSSI), εισερχόμενα και εξερχόμενα λάθη, πακέτα και Bytes, με ένα μέτρο ποιότητας του σήματος (QoS), όπως είναι το bitrate που μπορεί να επιτευχθεί υπό τον υπό αμφισβήτηση συνδυασμό των παραμέτρων.

Για τον σκοπό αυτό και την αξιολόγηση της εν λόγω τεχνικής έχει δημιουργηθεί και εκτελεστεί ένα πρόγραμμα χρησιμοποιώντας την εργαλειοθήκη Νευρωνικών Δικτύων του MATLAB. Οι χάρτες SOM που αναπτύχθηκαν εκπαιδεύτηκαν χρησιμοποιώντας μετρήσεις που έλαβαν χώρα σε πραγματικό περιβάλλον εντός των ορίων του Πανεπιστημίου Πειραιώς ενώ το πρόγραμμα έχει την δυνατότητα της πρόβλεψης του bitrate τόσο γνωστών όσο και άγνωστων δειγμάτων δεδομένων. Οι προβλέψεις τελικά συγκρίνονται με τις πραγματικές μετρήσεις ώστε να προκύψουν συμπεράσματα.

Για να καταστούν δυνατά τα παραπάνω ήταν απαραίτητη η δημιουργία αρχείων δεδομένων matlab που αποτελούνται από διαφορετικές παραμέτρους που παρατηρούνται ως αποτέλεσμα της διαμόρφωσης, συμπεριλαμβανομένου και του bitrate, και οι οποίες χρησιμοποιήθηκαν για την εκπαίδευση του χάρτη SOM ενώ τα δείγματα των δεδομένων απεικονίστηκαν πάνω στον δημιουργημένο χάρτη με αναγνωριστική ετικέτα την αντίστοιχη τιμή του bitrate. Σε αυτό το σημείο είναι άξιο λόγου να αναφέρουμε ότι σε κάθε κυψέλη του χάρτη μπορεί να αντιστοιχίζονται περαιτέρω του ενός δείγματος δεδομένων (data sample).

Στην συνέχεια και έχοντας εκπαιδεύσει τον χάρτη, ήταν δυνατή και η εκτίμηση του bitrate τόσο γνωστών όσο και άγνωστων δειγμάτων δεδομένων. Η εν λόγω

εκτίμηση του bitrate ακολουθεί την ίδια λογική με την εκπαίδευση του χάρτη: κατ' αρχήν, κάθε δείγμα δεδομένων αντιστοιχήθηκε με μία κυψέλη χρησιμοποιώντας τις ίδιες παραμέτρους με αυτές που χρησιμοποιήθηκαν κατά την διάρκεια της εκπαίδευσης του χάρτη και στην συνέχεια έγινε η εκτίμηση του bitrate ανάλογα με τις ετικέτες των πλησιέστερων κελιών.

Τελικά, τα συμπεράσματα προέκυψαν συγκρίνοντας τις τιμές του bitrate των δειγμάτων δεδομένων που προέκυψαν από το πρόγραμμα με τις πραγματικές μετρήσεις του bitrate των ίδιων δειγμάτων δεδομένων.

Το υπόλοιπο της εργασίας ακολουθεί την έξης δομή: στο κεφάλαιο 2 γίνεται μία ανασκόπηση της τεχνικής SOM ενώ μία σύντομη ανάλυση της τεχνικής που ακολουθήθηκε παρουσιάζεται στο κεφάλαιο 3. Ακολουθεί μία σύντομη περιγραφή του αλγορίθμου που δημιουργήθηκε στο κεφάλαιο 4 και η παρουσίαση των αποτελεσμάτων στο κεφάλαιο 5. Τα αποτελέσματα περιλαμβάνουν την σύγκριση διαφορετικών εκδοχών του προγράμματος (5.1) και την επιλογή των μεταβλητών παραμετροποίησης (5.2), του αριθμού των δειγμάτων δεδομένων (5.3) και των παραμέτρων εκπαίδευσης του χάρτη (5.4). Τέλος, στο κεφάλαιο 6, παρουσιάζονται τα συμπεράσματά μας.

ΚΕΦΑΛΑΙΟ 2: Τεχνική SOM (Self-Organizing Maps)

Η τεχνική χαρτών που αυτό-διοργανώνονται αναπτύχθηκε από τον Teuvo Kohonen [9], αποτελεί έναν σχήμα βασισμένο στα νευρωνικά δίκτυα και ανήκει σε εκείνο το είδος των τεχνικών εκπαίδευσης που το αναμενόμενο αποτέλεσμα δεν ελέγχεται από τον χρήστη. Μία σύντομη παρουσίαση της θεωρητικής βάσης της τεχνικής υπάρχει στο [9]. Είναι τεχνική αναπαράστασης και κατηγοριοποίησης πολυδιάστατων δεδομένων σε δισδιάστατους χάρτες. Οι χάρτες αυτοί αποτελούνται από τετραγωνικές ή εξαγωνικές κυψέλες πάνω σε κανονικό πλέγμα. Σύμφωνα με την μέθοδο, κάθε δείγμα δεδομένων αντιστοιχίζεται με μία κυψέλη έτσι ώστε στο τέλος παρεμφερή δείγματα

δεδομένων να αντιστοιχούν σε κοντινές κυψέλες. Υπό αυτήν την έννοια, ο παραγμένος χάρτης απεικονίζει την ομοιότητα των δεδομένων και τα κατηγοριοποιεί.

Επιπλέον, η τεχνική αυτό-διοργανωμένων χαρτών χρησιμοποιεί δύο αλγορίθμους εκπαίδευσης: τον αλγόριθμο σειριακής εκπαίδευσης (sequential training algorithm) και τον αλγόριθμο ομαδικής εκπαίδευσης (batch training algorithm). Στην πρώτη περίπτωση, κάθε δείγμα δεδομένων εισάγεται στην διαδικασία εκπαίδευσης ξεχωριστά από τα υπόλοιπα και διαμορφώνει την κυψέλη στην οποία αντιστοιχεί (BMU – Best Matching Unit) και τις γειτονικές της. Αντίθετα, στην δεύτερη περίπτωση τα δείγματα δεδομένων εισάγονται μαζικά στην διαδικασία εκπαίδευσης και επηρεάζουν τις κυψέλες στην οποίες έχουν αντιστοιχηθεί (BMUs) καθώς και τις γειτονικές τους παράλληλα.

Τέλος, το φθινόπωρο του 1997 δημιουργήθηκε για την εν λόγω τεχνική η πρώτη εργαλειοθήκη του MATLAB, γνωστή ως SOM TOOLBOX, ενώ την άνοιξη του 2000 αναβαθμίστηκε με την δεύτερη έκδοσή της, συμβατή με την $5^{η}$ έκδοση του MATLAB ή νεότερες. Με αυτήν την εργαλειοθήκη ο χρήστης του MATLAB έχει την δυνατότητα να διαχειριστεί τα δεδομένα του, να αρχικοποιήσει και να εκπαιδεύσει τον χάρτη του και εν τέλει να τον απεικονίσει. Παράλληλα, κατά την φάση της αρχικοποίησης και της εκπαίδευσης του χάρτη, μπορεί κανείς να επιλέξει ανάμεσα σε πληθώρα συνδυασμών των παραμέτρων της τεχνικής ξεκινώντας από την τοπολογία του χάρτη (τοπική δομή πλέγματος, μέγεθος χάρτη και συνολικό σχήμα χάρτη) μέχρι τις παραμέτρους εκπαίδευσης.

ΚΕΦΑΛΑΙΟ 3: Η Χρήση της Τεχνική SOM στην Εκτίμηση του Bitrate

Όπως αναφέρθηκε και στο κεφάλαιο 1, σκοπός μας είναι να εξεταστεί η πιθανότητα σύνδεσης παραμέτρων που παρατηρούνται ως αποτέλεσμα μιας παραμετροποίησης, όπως είναι ο θόρυβος, ο δείκτης ισχύος του σήματος και τα

εισερχόμενα και εξερχόμενα πακέτα, Bytes και λάθη με μια μετρική της ποιότητας του σήματος όπως είναι το bitrate με σκοπό την εκτίμηση της τιμής του. Επιπροσθέτως, σκοπός μας είναι η χρήση της μη ελεγχόμενης τεχνικής εκπαίδευσης SOM. Παρ' όλ' αυτά, όπως επίσης αναφέρθηκε στο κεφάλαιο 1, η τεχνική αυτή είναι τεχνική απεικόνισης και κατηγοριοποίησης πολυδιάστατων δειγμάτων δεδομένων σε δισδιάστατους χάρτες. Πώς λοιπόν είναι χρήσιμη στην εκτίμηση της τιμής του bitrate;

Αρχικά, οι μετρήσεις που συλλέχθηκαν μέσα από πραγματικό περιβάλλον λειτουργίας δικτύου εντός του Πανεπιστημίου Πειραιώς, χρησιμοποιήθηκαν για την δημιουργία διαφορετικών αρχείων δεδομένων του matlab. Κάθε αρχείο αποτελεί μία διαφορετική περίπτωση ελέγχου με διαφορετικούς συνδυασμούς των παραμέτρων. Η τελευταία στήλη του δείγματος δεδομένων είναι πάντα η μετρημένη αντίστοιχη τιμή του bitrate και χρησιμοποιείται ως ετικέτα του δείγματος δεδομένων. Έτσι, σε κάθε αρχείο δεδομένων υπάρχει ένας αριθμός στηλών, καθεμία εκ των οποίων είναι μία διαφορετική παράμετρος, ενώ κάθε γραμμή του αρχείου είναι ένα διαφορετικό δείγμα δεδομένων x. (Εικόνα 3)

Έχοντας λοιπόν ολοκληρώσει την οργάνωσή των δεδομένων σε τέτοια αρχεία ήταν δυνατή η χρήση του στην τεχνική εκπαίδευσης SOM. Πριν την χρήση τους ωστόσο είναι απαραίτητη η λήψη απόφασης ως προς την κανονικοποίηση ή όχι των δεδομένων. Χρησιμοποιήθηκαν τρία σενάρια: στο πρώτο τα δεδομένα εισήχθησαν αυτούσια χωρίς να κανονικοποιηθούν, στο δεύτερο σενάριο τα δεδομένα κανονικοποιήθηκαν ώστε να έχουν τιμές μεταξύ του μηδενός και του ένα ενώ στο τελευταίο, κανονικοποιήθηκαν ώστε η απόκλισή τους να είναι μεταξύ μηδενός και ένα. Το αποτέλεσμα της παραπάνω διαδικασίας είναι αυτό που χρησιμοποιήθηκε εν τέλει για την εκπαίδευση του χάρτη SOM. Συνεχίζοντας την επεξεργασία του χάρτη, ήταν απαραίτητη η χρήση ετικετών ώστε να διαχωρίζονται μεταξύ τους τα δείγματα δεδομένων επί του χάρτη. Όπως, λοιπόν, αναφέρθηκε και πρωτύτερα, στον ρόλο της ετικέτας των δειγμάτων δεδομένων χρησιμοποιήθηκαν οι μετρημένες τιμές του bitrate. Ωστόσο, το γεγονός ότι παραπάνω από ένα δείγμα δεδομένων μπορεί να αντιστοιχηθεί στην ίδια κυψέλη

του χάρτη, οδηγεί αναπόφευκτα στην αντιστοίχιση μια κυψέλης με παραπάνω της μίας ετικέτας που μπορεί να εμφανίζεται μία ή περισσότερες φορές. Σε αυτό το σημείο είναι απαραίτητο να σημειωθεί πως μπορούν να χρησιμοποιηθούν παραπάνω από μία μέθοδοι απεικόνισης των ετικετών του χάρτη. Η πρώτη μέθοδος (ας την ονομάσουμε VOTE) είναι η χρήση μόνο εκείνης της ετικέτας που εμφανίζεται συχνότερα, η δεύτερη (ας την ονομάσουμε ADD1) είναι η εμφάνιση όλων των πιθανών ετικετών, με φθίνουσα σειρά ανάλογα με τον αριθμό εμφάνισης τους, ενώ στην τελευταία μέθοδο (ας την ονομάσουμε FREQ) δίπλα από κάθε ετικέτα εμφανίζεται και ο αριθμός εμφάνισής της στην συγκεκριμένη κυψέλη.

Σε αυτό το σημείο της προσέγγισής μας το αποτέλεσμα είναι η εμφάνιση ενός χάρτη SOM με ετικέτες πάνω στον οποίο μπορεί μεν να αντιστοιχηθεί ένα νέο δείγμα δεδομένων αλλά δεν μπορεί να εκτιμηθεί το bitrate αυτού. Ως εκ τούτου, στην συνέχεια ήταν απαραίτητη η μαθηματικοποίηση της εικόνας που προέκυπτε. Βάσει τον συλλογισμών μας, το bitrate του εισαχθέντος προς εξέταση δείγματος δεδομένων θα πρέπει να είναι ίσο με αυτό των δειγμάτων δεδομένων των πλησιέστερων κυψελών. Ωστόσο, οι πλησιέστερες κυψέλες μπορεί να περιέχουν δείγματα δεδομένων που αντιστοιχούν σε bitrate με τιμές περισσότερες της μία. Έτσι, ήταν απαραίτητη η εύρεση του bitrate που αντιστοιχεί στα περισσότερα δείγματα δεδομένων των πλησιέστερων κυψελών.

1	5					
2	#n	RSSI	IPKTS	OPKTS	IBYTES	OBYTES
3		-35	926	1750	32630	880650 54
4		-32	908	1680	31932	845424 54
5		-32	888	1680	31272	845424 54
6		-35	888	1679	31272	845468 54
7		-36	890	1683	31338	845598 54
8		-36	960	1818	33812	915702 54
9		-36	890	1682	31338	845598 54
10		-36	928	1766	32756	887366 54
11		-34	920	1731	32328	873760 48
12		-33	926	1751	32586	880650 54
13		-33	924	1750	32564	880650 54
14		-32	894	1684	31494	845688 54

Εικόνα 3. Αρχείο Δεδομένων Matlab: ο αριθμός της πρώτης γραμμής αναφέρεται στον αριθμό των μεταβλητών της παραμετροποίησης που συμμετέχουν σε αυτήν την υπό εξέταση περίπτωση, σε αυτό το αρχείο είναι 5 μεταβλητές παραμετροποίησης που συμμετέχουν (Δείκτης έντασης του ληφθέντος σήματος, εισερχόμενα και εξερχόμενα πακέτα και Bytes), και η τελευταία στήλη

αναφέρεται στον μετρημένο bitrate που χρησιμοποιείται ως ετικέτα. Κάθε γραμμή αποτελεί ένα δείγμα δεδομένων και κάθε στήλη μια διαφορετική μεταβλητή παραμετροποίησης.

Με άλλα λόγια, κυψέλες που περιέχουν δείγματα δεδομένων με ίδιο bitrate αποτελούν μία ομάδα (cluster). Επομένως, για να καθοριστεί το bitrate ενός δείγματος δεδομένων είναι απαραίτητο να αντιστοιχηθεί με ένα cluster. Κάθε cluster έχει ένα κέντρο που καθορίζεται από τις εξής εξισώσεις:

$$x = \sum_{i}^{n} \frac{w_i x_i}{n} \quad (1) \qquad \text{and} \qquad y = \sum_{i}^{n} \frac{w_i y_i}{n} \quad (2),$$

Όπου n είναι ο αριθμός των κυψελών που ανήκουν στο συγκεκριμένο cluster, x_i και y_i είναι οι συντεταγμένες του κελιού i και w_i είναι το βάρος με το οποίο το κελί i συμμετέχει στον υπολογισμό των συντεταγμένων του κέντρου του cluster. Στα πρώτα δύο σενάρια το βάρος w_i είναι πάντοτε ίσο με την μονάδα ενώ στο τελευταίο σενάριο κανονικοποίησης το βάρος w_i υπολογίζεται από την σχέση 3:

$$w_i = \frac{k}{r} \qquad (3),$$

Όπου k είναι οι φορές εμφάνισης του συγκεκριμένου bitrate στο κελί i και r είναι ο αριθμός των δειγμάτων δεδομένων της κυψέλης.

Κατόπιν και αυτής της διαδικασίας είναι πλέον εύκολη η εκτίμηση του bitrate ενός δείγματος δεδομένων αρκεί να αντιστοιχηθεί με μία κυψέλη του χάρτη και εν συνεχεία να βρεθεί το κέντρο εκείνου του cluster που είναι πλησιέστερο στην επιλεγμένη κυψέλη. Το bitrate του δείγματος δεδομένων είναι εκείνο που αντιστοιχεί στο επιλεγμένο cluster. Για τον υπολογισμό της απόστασης του κέντρου του cluster και της κυψέλης χρησιμοποιήθηκε η Ευκλείδεια απόσταση.

Τέλος, για την αξιολόγηση της διαδικασίας που ακολουθήθηκε και την εξαγωγή συμπερασμάτων, το πρόγραμμα παρέχει την δυνατότητα της σύγκρισης των τιμών του bitrate που προέκυψαν με τις πραγματικές μετρήσεις του. Αυτή η σύγκριση αποδίδεται και σε ποσοστό ενώ είναι απαραίτητο να σημειωθεί ότι τα προς μελέτη δείγματα δεδομένων εισήχθησαν στο πρόγραμμα αφού δέχτηκαν την

ίδια προεργασία με αυτά που αρχικά χρησιμοποιήθηκαν για την εκπαίδευση του χάρτη SOM.

ΚΕΦΑΛΑΙΟ 4: Αποτελέσματα

Για την εξαγωγή συμπερασμάτων ήταν απαραίτητη η λήψη αποτελεσμάτων ως προς όλες τις πιθανές παραμέτρους του προγράμματος. Έτσι, ήταν απαραίτητη η επιλογή της κατάλληλης έκδοσης του προγράμματος (VOTE, ADD1 και FREQ), των κατάλληλων μεταβλητών του δείγματος δεδομένων, του κατάλληλου αριθμού δειγμάτων δεδομένων και των κατάλληλων τιμών των παραμέτρων της τεχνικής SOM. Όσο χαμηλότερο το ποσοστό εσφαλμένων εκτιμήσεων του bitrate τόσο καλύτερη η εν λόγω επιλογή. Ως εκ τούτου, δημιουργήθηκαν διαφορετικά σενάρια που ελέγχουν τις παραπάνω ερωτήσεις και συγκρίθηκαν βάσει του αποτελέσματος του ποσοστού εσφαλμένων εκτιμήσεων του bitrate. Οι συγκεκριμένες περιπτώσεις ελέγχου παρουσιάζονται παρακάτω για όλα τα σενάρια κανονικοποίησης.

4.1 Σύγκριση των VOTE, ADD1 και FREQ Εκδόσεων του Προγράμματος

Έχοντας αναλύσει τις τρείς εκδόσεις του προγράμματός μας ήταν απαραίτητη η μεταξύ τους σύγκριση ώστε να χρησιμοποιηθεί η καλύτερη εξ αυτών βάσει των αποτελεσμάτων τους.

Όπως σημειώθηκε και παραπάνω, η έκδοση VOTE χρησιμοποιεί για τον υπολογισμό των κέντρων των cluster μόνο την ετικέτα που εμφανίζεται συχνότερα στην κυψέλη. Σε αυτήν την περίπτωση είναι πιθανό μία ετικέτα να μην εμφανίζεται στον δημιουργημένο χάρτη SOM αν και χρησιμοποιήθηκε σε κάποιο δείγμα δεδομένων. Το συγκεκριμένο γεγονός οφείλεται στο γεγονός ότι διαφορετικά δείγματα δεδομένων μπορεί να αντιστοιχηθούν με την ίδια κυψέλη ακόμα και αν έχουν διαφορετική ετικέτα. Έτσι, όπως αναφέρθηκε και στο κεφάλαιο 3, μόνο η ετικέτα με την μεγαλύτερη συχνότητα συμμετέχει στον

υπολογισμό των κέντρων των cluster. Ως εκ τούτου, ετικέτες με μικρότερη συχνότητα ενδέχεται να μην εμφανιστούν στον χάρτη SOM.

Το παραπάνω οδηγεί στην εξάλειψη ενός ή περισσοτέρων cluster καθώς δεν προκύπτει κάποιο κέντρο για αυτά. Επιπροσθέτως, το πρόγραμμα τερματίζει λίγο μετά τον υπολογισμό των κέντρων καθώς ένα από αυτά δεν υπάρχει. Τέλος, ακόμα και αν υποθέσουμε ότι το πρόγραμμα δεν θα τερμάτιζε, τα δείγματα δεδομένων που θα ανήκαν στο cluster που εξαλείφθηκε θα συσχετιζόντουσαν με λάθος cluster και επομένως και με λάθος bitrate.

Στην προσπάθειά μας να επιλύσουμε το παραπάνω σφάλμα του προγράμματος δημιουργήθηκε η εκδοχή ADD1 του προγράμματος στο οποίο συμμετέχουν όλες οι ετικέτες εξίσου για τον υπολογισμό των κέντρων των cluster. Έτσι, κάθε ετικέτα που χρησιμοποιείται στο αρχείο δεδομένων εμφανίζεται στον χάρτη που δημιουργείται ακόμα και αν στο κελί εμφανίζεται μόλις μία φορά. Ως εκ τούτου, στην έκδοση ADD1, μία ετικέτα συμμετέχει στον υπολογισμό των κέντρων των cluster ανεξάρτητα από τις φορές που εμφανίζεται στην κυψέλη.

Για την σύγκριση των παραπάνω εκδόσεων εκτελέσαμε τις ίδιες περιπτώσεις ελέγχου για την κάθε έκδοση χρησιμοποιώντας τα ίδια αρχεία δεδομένων τόσο για την φάση της εκπαίδευσης όσο και για την φάση της αξιολόγησης. Τα αποτελέσματα μας οδήγησαν στο συμπέρασμα ότι η ADD1 έκδοση του προγράμματος έλυσε το σφάλμα της VOTE έκδοσης αλλά στις περιπτώσεις εκείνες όπου η έκδοση VOTE απέδιδε αποτελέσματα, τα αποτελέσματα της ήταν καλύτερα από εκείνα της ADD1 έκδοσης του προγράμματος.

Το παραπάνω συμπέρασμα οδήγησε στην έκδοση FREQ του προγράμματος. Σε αυτήν την περίπτωση, απεικονίζονται όλες οι ετικέτες που έχουν αντιστοιχηθεί με την κυψέλη συνοδευόμενες από τον αριθμό που αντιστοιχήθηκαν με την κυψέλη. Έτσι, αντίθετα με την έκδοση ADD1, οι ετικέτες συμμετέχουν στον υπολογισμό των κέντρων των cluster ανάλογα με την συχνότητα της ετικέτας στην κυψέλη (κεφάλαιο 3).

Τελικά, για την σύγκριση και της τελευταίας έκδοσης με τις προηγούμενες δύο, εκτελέσαμε και πάλι ίδια αρχεία δεδομένων τόσο στην φάση εκπαίδευσης όσο και στην φάση αξιολόγησης. Είναι άξιο λόγου να αναφερθεί πως και στις τρεις περιπτώσεις χρησιμοποιήθηκαν οι ίδιες παράμετροι εκπαίδευσης. Από τα αποτελέσματα προέκυψε ότι η καλύτερη έκδοση και για τα τρία σενάρια κανονικοποίησης είναι η FREQ και έτσι είναι αυτή που χρησιμοποιείται και στην συνέχεια των πειραμάτων μας.

4.2 Επιλογή των Μεταβλητών που θα Χρησιμοποιηθούν στα Δείγματα Δεδομένων

Το επόμενο στάδιο της έρευνάς μας αφορούσε στην επιλογή των μεταβλητών που χρησιμοποιήθηκαν στα δείγματα δεδομένων. Για τον σκοπό αυτό χρησιμοποιήθηκαν διαφορετικές περιπτώσεις ελέγχου με την έκδοση FREQ του προγράμματος και τις ίδιες μεταβλητές εκπαίδευσης ενώ τα αρχεία δεδομένων ήταν ίδια τόσο για την φάση της εκπαίδευσης όσο και για την φάση της αξιολόγησης. Η διαφορά των αρχείων δεδομένων έγκειται στον αριθμό και στον τύπο των μεταβλητών των δειγμάτων δεδομένων.

Οι μεταβλητές των δειγμάτων δεδομένων που χρησιμοποιήθηκαν στο σύνολό τους, σε διάφορους συνδυασμούς είναι οι εξής εννέα: θόρυβος, δείκτης ισχύος του ληφθέντος σήματος (RSSI), εισερχόμενα και εξερχόμενα πακέτα, λάθη και Bytes και το bitrate.

Από την σύγκριση των περιπτώσεων ελέγχου προέκυψαν τα εξής:

- Σε κανένα από τα τρία σενάρια κανονικοποίησης τα αποτελέσματα δεν εξαρτώνται από το εάν τα δείγματα δεδομένων είναι σε διάταξη ανάλογης του bitrate τους ή όχι. Και στις δύο περιπτώσεις τα αποτελέσματα είναι ίσα μεταξύ τους.
- Στο σενάριο των μη κανονικοποιημένων δειγμάτων δεδομένων, η ύπαρξη ή όχι του bitrate ως μεταβλητή του δείγματος δεδομένων δεν επηρεάζει τα αποτελέσματα πάντοτε προς την ίδια κατεύθυνση, σε άλλες περιπτώσεις

μείωνε το ποσοστό εσφαλμένων εκτιμήσεων bitrate ενώ σε άλλες προκαλούσε την αύξησή του. Επιπλέον, σε αυτό το σενάριο η περίπτωση κατά την οποία λάβαμε το μικρότερο ποσοστό εσφαλμένων εκτιμήσεων bitrate δεν ήταν κάποια από εκείνες που το περιείχαν ως μεταβλητή. Αντίθετα, στα άλλα σενάρια κανονικοποίησης, η ύπαρξη του bitrate ως μεταβλητή προκαλούσε πάντοτε την μείωση του ποσοστού εσφαλμένων εκτιμήσεων bitrate. Παρ' όλ' αυτά, η χρήση του bitrate ως μεταβλητή είναι λανθασμένη επιλογή καθώς το bitrate είναι η υπό εξέταση μεταβλητή και σε καμία περίπτωση δεν μπορεί να θεωρηθεί γνωστή η τιμή του.

- Τέλος, οι επιλεγμένες μεταβλητές διαφέρουν από σενάριο σε σενάριο κανονικοποίησης. Κατά το σενάριο των μη κανονικοποιημένων δεδομένων, η περίπτωση με το ελάχιστο ποσοστό εσφαλμένων εκτιμήσεων του bitrate, ίσο με 28,6%, ήταν εκείνο στο οποίο οι μεταβλητές των δειγμάτων δεδομένων ήταν ο δείκτης έντασης του ληφθέντος σήματος και τα εισερχόμενα και εξερχόμενα πακέτα ενώ στο σενάριο κανονικοποίησης των δεδομένων μεταξύ μηδενός και ένα, η περίπτωση με το ελάχιστο ποσοστό εσφαλμένων εκτιμήσεων του bitrate, ίσο με 24,6%, ήταν εκείνο στο οποίο οι μεταβλητές των δειγμάτων δεδομένων ήταν ο δείκτης έντασης του ληφθέντος σήματος και τα εισερχόμενα πακέτα, λάθη και Bytes και τα εξερχόμενα πακέτα. Έτσι, στα πρώτα δύο σενάρια κανονικοποίησης η επιλογή των μεταβλητών ήταν ξεκάθαρη. Αντίθετα, η επιλογή των μεταβλητών των δειγμάτων δεδομένων στο σενάριο κανονικοποίησης της απόκλισης των δεδομένων δεν ήταν τόσο ξεκάθαρη. Σε αυτό το σενάριο, οι περιπτώσεις που δεν περιελάμβαναν το bitrate ως μεταβλητή και είχαν το χαμηλότερο ποσοστό εσφαλμένων εκτιμήσεων bitrate, ίσο με 25,4%, είναι δύο. Στην πρώτη περίπτωση οι μεταβλητές που χρησιμοποιήθηκαν ήταν ο δείκτης έντασης του ληφθέντος σήματος και τα εισερχόμενα πακέτα, λάθη και Bytes και τα εξερχόμενα πακέτα ενώ στην δεύτερη περίπτωση ήταν ο δείκτης έντασης του ληφθέντος σήματος και τα εισερχόμενα και εξερχόμενα πακέτα. Ωστόσο, στην πρώτη περίπτωση απαιτείται μεγαλύτερη υπολογιστική

ισχύς σε σχέση με την δεύτερη καθιστώντας την δεύτερη επιλογή βέλτιστη λύση και τις μεταβλητές της, αυτές που θα χρησιμοποιηθούν στην συνέχεια της έρευνας.

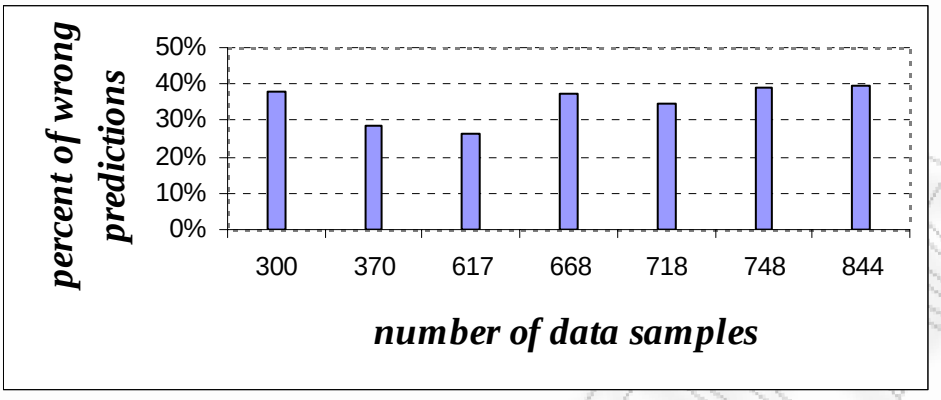
4.3 Επιλογή του Αριθμού των Δειγμάτων Δεδομένων

Έχοντας επιλέξει τις μεταβλητές του δείγματος δεδομένων, το επόμενο στάδιο είναι η επιλογή του αριθμού των δειγμάτων δεδομένων που θα συμμετάσχουν στην εκπαίδευση του χάρτη. Για τον σκοπό αυτό δημιουργήθηκαν περιπτώσεις ελέγχου που περιελάμβαναν τις ίδιες μεταβλητές των δειγμάτων δεδομένων (ανάλογα με τα αποτελέσματα της ενότητας 4.2 για το κάθε σενάριο κανονικοποίησης) αλλά διαφορετικό αριθμό δειγμάτων δεδομένων.

Για την λήψη αποτελεσμάτων χρησιμοποιήθηκαν και πάλι οι ίδιες παράμετροι εκπαίδευσης και η έκδοση FREQ του προγράμματός μας.

4.3.1 Σενάριο Μη Κανονικοποιημένων Δειγμάτων Δεδομένων

Σύμφωνα με τα αποτελέσματα, ο αριθμός των δειγμάτων δεδομένων επηρεάζει τα αποτελέσματα του ποσοστού εσφαλμένων εκτιμήσεων αλλά όχι πάντοτε με τον ίδιο τρόπο. Το διάγραμμα αυτών των αποτελεσμάτων φαίνεται στην Εικόνα 4.

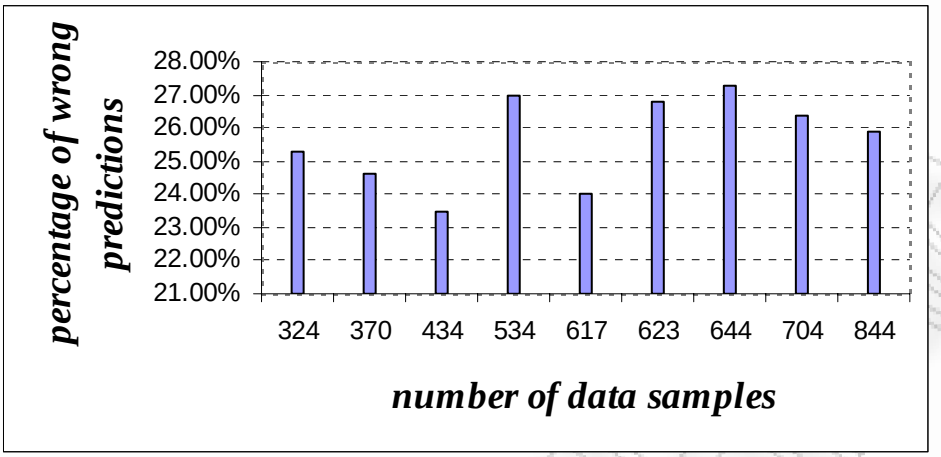


Εικόνα 4. Σενάριο μη κανονικοποιημένων δειγμάτων δεδομένων: Διάγραμμα του ποσοστού των εσφαλμένων εκτιμήσεων bitrate σε συνάρτηση με τον αριθμό των δειγμάτων δεδομένων που χρησιμοποιήθηκαν στην εκπαίδευση του χάρτη SOM.

Τελικά, το ελάχιστο αποτέλεσμα ήταν 26,4% και εμφανίζεται για αριθμό δειγμάτων δεδομένων ίσο με 617. Η αύξηση του αριθμού των δειγμάτων δεδομένων μέχρι την τιμή 617 προκαλεί την μείωση του ποσοστού εσφαλμένων εκτιμήσεων του bitrate αλλά η περαιτέρω αύξησή του προκαλεί την χειροτέρευση των αποτελεσμάτων. Έτσι, στην συνέχεια της έρευνας για το σενάριο των μη κανονικοποιημένων αποτελεσμάτων ο αριθμός των δειγμάτων δεδομένων που χρησιμοποιούνται κατά την εκπαίδευση του χάρτη είναι 617.

4.3.2 Σενάριο Κανονικοποιημένων Δεδομένων Μεταξύ του Μηδενός και της Μονάδας

Όμοια με το σενάριο των μη κανονικοποιημένων δεδομένων, το διάγραμμα των αποτελεσμάτων απεικονίζεται στην Εικόνα 5.

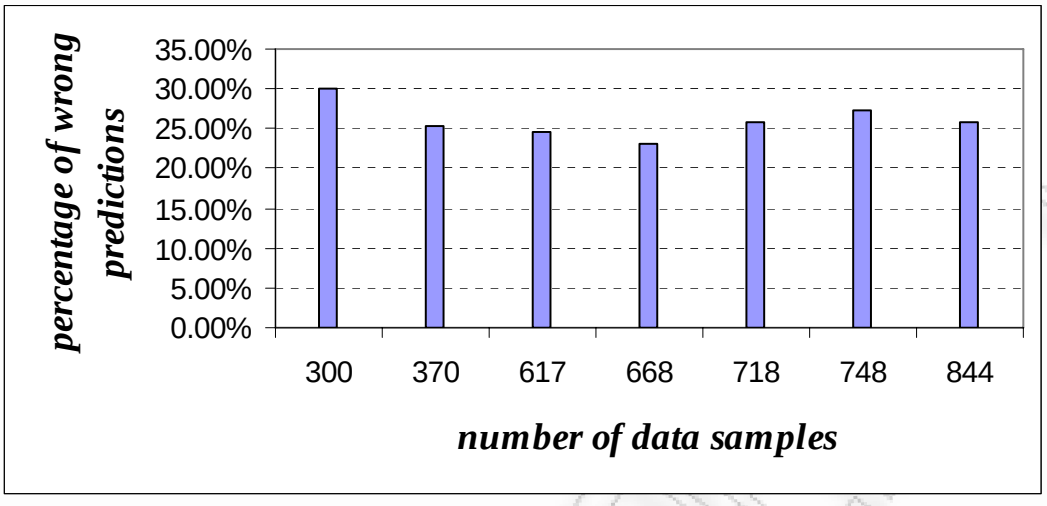


Εικόνα 5. Σενάριο κανονικοποιημένων δεδομένων στο διάστημα [0,1]: Διάγραμμα του ποσοστού εσφαλμένων εκτιμήσεων του bitrate σε συνάρτηση με τον αριθμό των δειγμάτων δεδομένων.

Αναλύοντας το παραπάνω διάγραμμα παρατηρούμε ότι τα αποτελέσματα διακυμαίνονται μεταξύ των τιμών 23,5% και 27,3%. Το διάγραμμα έχει δύο τοπικά ελάχιστα ((434, 23,5), (617, 24,0)) και δύο τοπικά μέγιστα ((534, 27,0), (644, 27,3)). Τέλος, το ελάχιστο αποτέλεσμα είναι ίσο με 23,5% και εμφανίζεται για αριθμό δειγμάτων δεδομένων ίσο με 434. Έτσι, για αυτό το σενάριο, στο υπόλοιπο της έρευνάς μας χρησιμοποιήθηκαν αρχεία δεδομένων που περιέχουν 434 δείγματα δεδομένων.

4.3.3 Σενάριο Κανονικοποιημένης Απόκλισης των Δειγμάτων Δεδομένων

Τέλος, το διάγραμμα των αποτελεσμάτων αυτού του σεναρίου κανονικοποίησης είναι αυτό της παρακάτω εικόνας (Εικόνα 6).



Εικόνα 6. Σενάριο κανονικοποιημένης διακύμανσης των δειγμάτων δεδομένων στο διάστημα [0,1]: Διάγραμμα του ποσοστού εσφαλμένων εκτιμήσεων του bitrate σε σχέση με τον αριθμό των δειγμάτων δεδομένων.

Το ελάχιστο αποτέλεσμα σε αυτό το σενάριο είναι ίσο με 23,1% και προκύπτει για αριθμό δειγμάτων δεδομένων ίσο με 668. Η αύξηση του αριθμού των δειγμάτων δεδομένων μέχρι την τιμή 668 μειώνει το ποσοστό εσφαλμένων εκτιμήσεων του bitrate αλλά η περαιτέρω αύξηση από αυτήν την τιμή προκαλεί αύξηση του ποσοστού εσφαλμένων εκτιμήσεων bitrate και χειροτέρευση του αποτελέσματος. Έτσι, για το υπόλοιπο της έρευνάς μας υπό αυτό το σενάριο κανονικοποίησης χρησιμοποιήθηκαν αρχεία δεδομένων που απαρτίζονταν από 668 δείγματα δεδομένων.

4.4 Επιλογή του Αλγορίθμου Εκπαίδευσης και των Παραμέτρων του

Τελευταία στάδιο της έρευνας είναι η επιλογή μεταξύ των δύο αλγορίθμων εκπαίδευσης της τεχνικής SOM και των τιμών των παραμέτρων τους. Για τον σκοπό αυτό, και στα τρία σενάρια κανονικοποίησης, αρχικά καθορίστηκε ο καλύτερος συνδυασμός των παραμέτρων ανά αλγόριθμο εκπαίδευσης και στην συνέχεια συγκρίθηκαν τα βέλτιστα αποτελέσματα των δύο αλγορίθμων εκπαίδευσης.

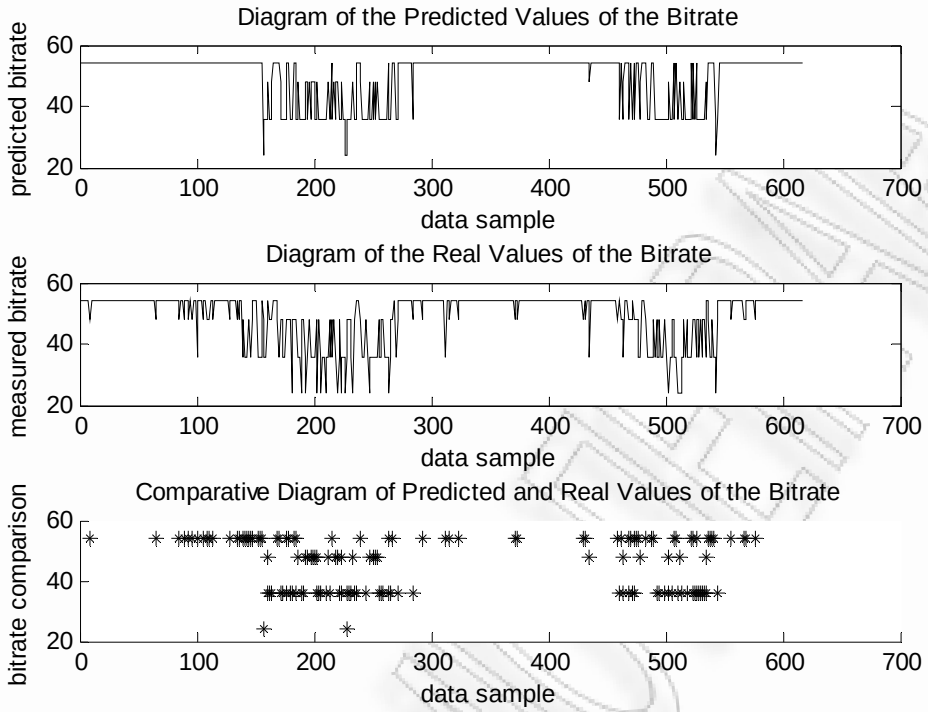
4.4.1 Σενάριο των Μη Κανονικοποιημένων Δεδομένων

Για την εύρεση των βέλτιστων τιμών των παραμέτρων του αλγορίθμου μαζικής εκπαίδευσης δοκιμάστηκαν διάφορες περιπτώσεις ελέγχου αλλάζοντας κάθε φορά μόνο μία εκ των παραμέτρων. Οι παράμετροι δοκιμάστηκαν με τυχαία σειρά.

Συγκρίνοντας τα αποτελέσματα ο βέλτιστος συνδυασμός που προέκυψε ήταν εκείνος στον οποίο ισχύουν τα παρακάτω:

- ✓ Συνάρτηση γειτνίασης: Γκαουσιανή
- ✓ Αρχική ακτίνα για την «πρόχειρη» φάση: 5
- ✓ Τελική ακτίνα για την «πρόχειρη» φάση: 1
- ✓ Διάρκεια εκπαίδευσης για την «πρόχειρη» φάση: 6
- ✓ Αρχική ακτίνα για την φάση τελειοποίησης: 1
- ✓ Τελική ακτίνα για την φάση τελειοποίηση: 1
- ✓ Διάρκεια εκπαίδευσης κατά την φάση τελειοποίησης: 48

Το διάγραμμα των εκτιμήσεων του bitrate, των πραγματικών μετρήσεων του και το συγκριτικό διάγραμμα αυτών απεικονίζονται στην Εικόνα 7.



Εικόνα 7. Αλγόριθμος μαζικής εκπαίδευσης στο σενάριο μη κανονικοποιημένων δεδομένων: Διάγραμμα εκτιμήσεων του bitrate, Διάγραμμα των πραγματικών μετρήσεων και συγκριτικό διάγραμμα αυτών. Το σύμβολο * απεικονίζει τα δείγματα δεδομένων που οι τιμές των εκτιμήσεων και οι μετρήσεις του bitrate διαφέρουν.

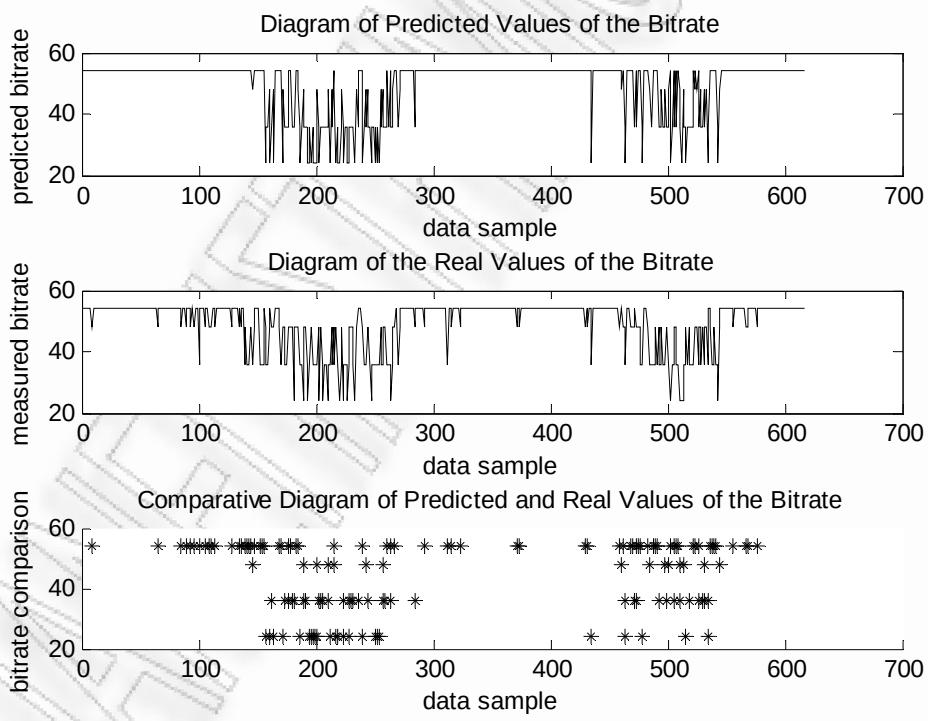
Στην περίπτωση του αλγορίθμου σειριακής εκπαίδευσης, ακολουθώντας την ίδια τεχνική με αυτήν που χρησιμοποιήθηκε στην περίπτωση του αλγορίθμου μαζικής εκπαίδευσης δημιουργήθηκαν και πάλι διαφορετικές περιπτώσεις ελέγχου. Αν και η τεχνική κατά βάση είναι η ίδια σε αυτήν την περίπτωση υπάρχει μία βασική διαφορά: στον αλγόριθμο σειριακής εκπαίδευσης τα δείγματα δεδομένων δεν εισάγονται όλα ταυτοχρόνως και ως εκ τούτου η σειρά με την οποία εισέρχονται στο σύστημα προκαλεί διαφοροποιήσεις. Για την αποφυγή τέτοιων καταστάσεων, κατά την διάρκεια των ελέγχων επιλέχθηκε να εισέρχονται με την σειρά που εμφανίζονται στο αρχείο δεδομένων.

Τελικά, ο βέλτιστος συνδυασμός των τιμών των παραμέτρων για τον αλγόριθμο σειριακής εκπαίδευσης ήταν ο εξής:

- ✓ Συνάρτηση Γειτνίασης: Γκαουσιανή

- ✓ Είδος μήκους (Length type): epochs
- ✓ Συνάρτηση Εκπαίδευσης (Learning function): inv
- ✓ Αρχική Ακτίνα για την «πρόχειρη» φάση: 3
- ✓ Αρχική Ακτίνα για την «πρόχειρη» φάση: 1
- ✓ Διάρκεια Εκπαίδευσης για την «πρόχειρη» φάση: 4
- ✓ Αρχικό Άλφα για την «πρόχειρη» φάση: 0.5
- ✓ Αρχική Ακτίνα για την φάση τελειοποίησης: 1
- ✓ Τελική Ακτίνα για την φάση τελειοποίησης: 1
- ✓ Διάρκεια Εκπαίδευσης για την φάση τελειοποίησης: 21
- ✓ Αρχικό Άλφα για την φάση τελειοποίησης: 0.05

Το διάγραμμα των εκτιμήσεων του bitrate, των πραγματικών μετρήσεων του και το συγκριτικό διάγραμμα αυτών απεικονίζονται στην Εικόνα 8.



Εικόνα 8. Αλγόριθμος σειριακής εκπαίδευσης στο σενάριο μη κανονικοποιημένων δεδομένων: Διάγραμμα των εκτιμήσεων του bitrate, των πραγματικών μετρήσεων του και το συγκριτικό διάγραμμα αυτών. Το σύμβολο * απεικονίζει τα δείγματα δεδομένων που οι τιμές των εκτιμήσεων και οι μετρήσεις του bitrate διαφέρουν.

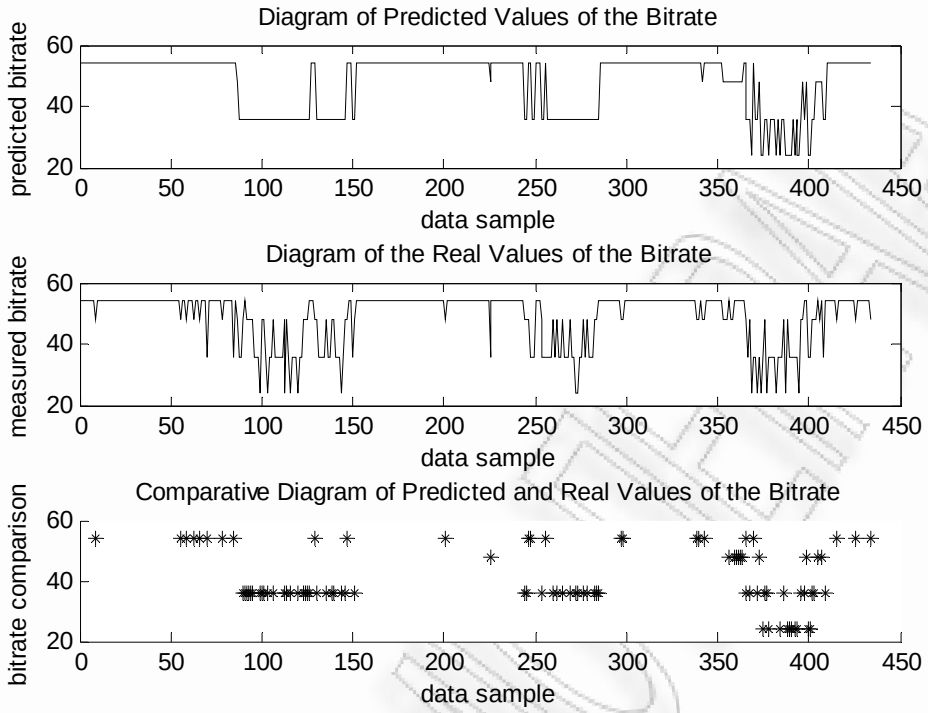
Η σύγκριση του αποτελέσματος για την περίπτωση του αλγορίθμου μαζικής εκπαίδευσης με αυτό του αλγορίθμου σειριακής εκπαίδευσης, ίσο με 25,6%, είναι λίγο υψηλότερο από το δεύτερο, ίσο με 24,6%. Καθώς λοιπόν το αποτέλεσμα αναφέρεται σε λάθος εκτιμήσεις, η πρώτη εντύπωση είναι ότι η βέλτιστη επιλογή είναι ο αλγόριθμος σειριακής εκπαίδευσης. Παρ' όλ' αυτά, ο χρόνος που απαιτείται για την φάση εκπαίδευσης είναι κάποιες φορές ιδιαίτερα σημαντικός. Έτσι, αν και το αποτέλεσμα του αλγορίθμου σειριακής εκπαίδευσης είναι καλύτερο, ο αλγόριθμος μαζικής εκπαίδευσης είναι γρηγορότερος. Σύμφωνα με το πρόγραμμα, ο αλγόριθμος μαζικής εκπαίδευσης απαιτεί περίπου 3 με 4 δευτερόλεπτα για να ολοκληρωθεί η φάση εκπαίδευσης ενώ ο αλγόριθμος σειριακής εκπαίδευσης απαιτεί σχεδόν τον διπλάσιο χρόνο (7-8 δευτερόλεπτα). Ως εκ τούτου, και επειδή η διαφορά των αποτελεσμάτων των δύο αλγορίθμων είναι σχετικά μικρή, η επιλογή μεταξύ των δύο αλγορίθμων είναι υποκειμενική και εξαρτάται από την ύπαρξη ή όχι της ανάγκης μιας γρήγορης εκπαίδευσης του χάρτη SOM.

4.4.2 Σενάριο Κανονικοποιημένων Δεδομένων Μεταξύ του Μηδενός και της Μονάδας

Χρησιμοποιώντας την ίδια διαδικασία όπως στο σενάριο μη κανονικοποίησης των δεδομένων, ο βέλτιστος συνδυασμός που προέκυψε για τον αλγόριθμο μαζικής εκπαίδευσης ήταν ο εξής:

- ✓ Συνάρτηση γειτνίασης: Γκαουσιανή
- ✓ Αρχική ακτίνα για την «πρόχειρη» φάση: 3
- ✓ Τελική ακτίνα για την «πρόχειρη» φάση: 1
- ✓ Διάρκεια εκπαίδευσης για την «πρόχειρη» φάση: 1
- ✓ Αρχική ακτίνα για την φάση τελειοποίησης: 1
- ✓ Τελική ακτίνα για την φάση τελειοποίηση: 1
- ✓ Διάρκεια εκπαίδευσης κατά την φάση τελειοποίησης: 9

Το διάγραμμα των εκτιμήσεων του bitrate, των πραγματικών μετρήσεων του και το συγκριτικό διάγραμμα αυτών απεικονίζονται στην Εικόνα 9.

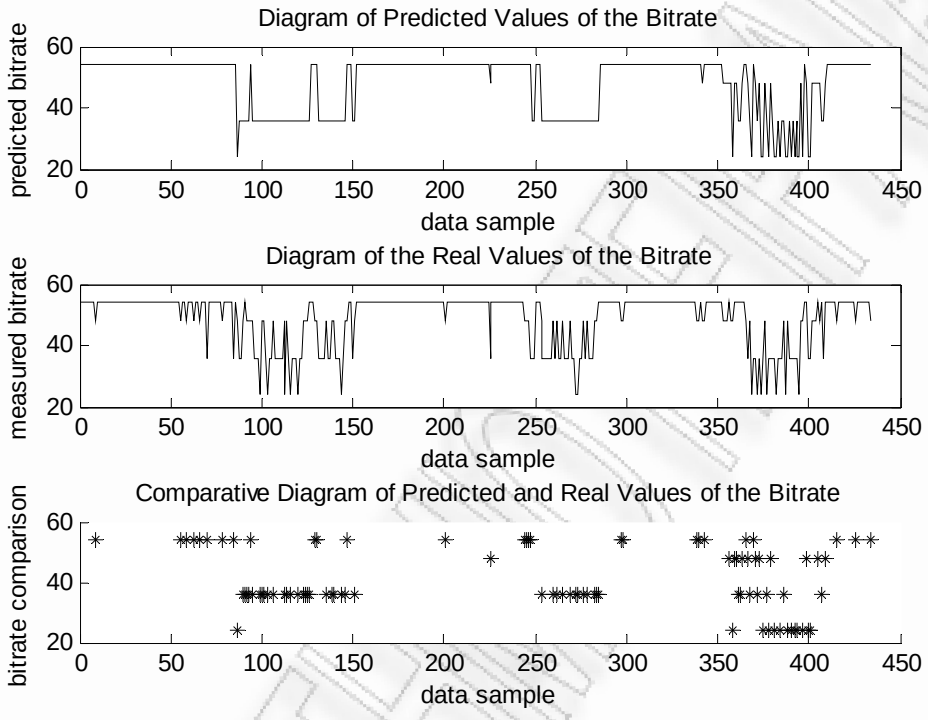


Εικόνα 9. Αλγόριθμος μαζικής εκπαίδευσης στο σενάριο κανονικοποιημένων δεδομένων στο διάστημα [0,1]: Διάγραμμα εκτιμήσεων του bitrate, Διάγραμμα των πραγματικών μετρήσεων και συγκριτικό διάγραμμα αυτών. Το σύμβολο * απεικονίζει τα δείγματα δεδομένων που οι τιμές των εκτιμήσεων και οι μετρήσεις του bitrate διαφέρουν.

Αντίστοιχα, ο βέλτιστος συνδυασμός των τιμών των παραμέτρων εκπαίδευσης για τον αλγόριθμο σειριακής εκπαίδευσης σε αυτό το σενάριο ήταν ο εξής:

- ✓ Συνάρτηση Γειτνίασης: Γκαουσιανή
- ✓ Είδος μήκους (Length type): epochs
- ✓ Συνάρτηση Εκπαίδευσης (Learning function): inv
- ✓ Αρχική Ακτίνα για την «πρόχειρη» φάση: 4
- ✓ Αρχική Ακτίνα για την «πρόχειρη» φάση: 1
- ✓ Διάρκεια Εκπαίδευσης για την «πρόχειρη» φάση: 5
- ✓ Αρχικό Άλφα για την «πρόχειρη» φάση: 0.5
- ✓ Αρχική Ακτίνα για την φάση τελειοποίησης: 1
- ✓ Τελική Ακτίνα για την φάση τελειοποίησης: 1
- ✓ Διάρκεια Εκπαίδευσης για την φάση τελειοποίησης: 21
- ✓ Αρχικό Άλφα για την φάση τελειοποίησης: 0.055

Το διάγραμμα των εκτιμήσεων του bitrate, των πραγματικών μετρήσεων του και το συγκριτικό διάγραμμα αυτών απεικονίζονται στην Εικόνα 10.



Εικόνα 10. Αλγόριθμος σειριακής εκπαίδευσης στο σενάριο κανονικοποιημένων δεδομένων στο διάστημα [0,1]: Διάγραμμα των εκτιμήσεων του bitrate, των πραγματικών μετρήσεων του και το συγκριτικό διάγραμμα αυτών. Το σύμβολο * απεικονίζει τα δείγματα δεδομένων που οι τιμές των εκτιμήσεων και οι μετρήσεις του bitrate διαφέρουν.

Σε αυτό το σενάριο κανονικοποίησης, από την σύγκριση του αποτελέσματος του αλγορίθμου μαζικής εκπαίδευσης με αυτό του αλγορίθμου σειριακής εκπαίδευσης προκύπτει ότι τα δύο αποτελέσματα είναι ίσα μεταξύ τους με τιμή 22,6%. Ως εκ τούτου, η πρώτη εντύπωση είναι ότι οι δύο αλγόριθμοι μπορούν να χρησιμοποιηθούν εξίσου. Ωστόσο, όπως αναφέρθηκε και στο προηγούμενο σενάριο κανονικοποίησης, ο χρόνος που απαιτείται για την ολοκλήρωση της φάσης της εκπαίδευσης είναι κάποιες φορές ιδιαίτερα σημαντικός και ο αλγόριθμος μαζικής εκπαίδευσης είναι γρηγορότερος. Σύμφωνα με το πρόγραμμα, σε αυτό το σενάριο κανονικοποίησης, ο αλγόριθμος μαζικής εκπαίδευσης απαιτεί λιγότερο από 1 δευτερόλεπτο ενώ ο αλγόριθμος σειριακής

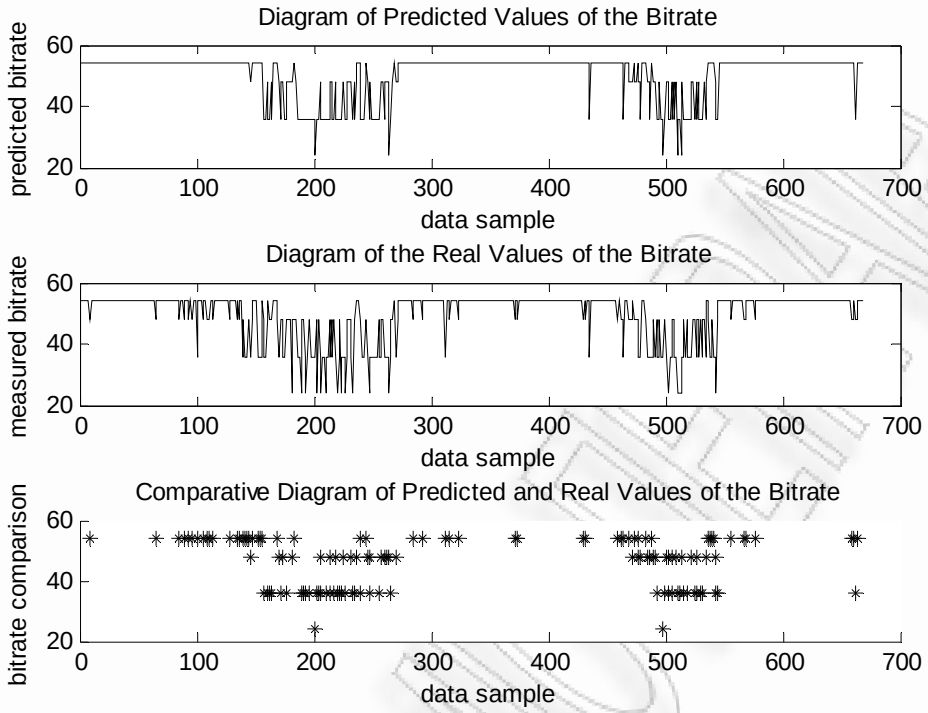
εκπαίδευσης απαιτεί περίπου 4 με 5 δευτερόλεπτα. Συνεπώς, η βέλτιστη επιλογή σε αυτό το σενάριο κανονικοποίησης είναι ο αλγόριθμος μαζικής εκπαίδευσης.

4.4.3 Σενάριο Κανονικοποιημένης Απόκλισης των Δειγμάτων Δεδομένων

Σε αυτό το σενάριο, ο βέλτιστος συνδυασμός που προέκυψε για τον αλγόριθμο μαζικής εκπαίδευσης ήταν εκείνος για τον οποίο οι τιμές των παραμέτρων φαίνονται παρακάτω:

- ✓ Συνάρτηση γειτνίασης: Γκαουσιανή
- ✓ Αρχική ακτίνα για την «πρόχειρη» φάση: 4
- ✓ Τελική ακτίνα για την «πρόχειρη» φάση: 1
- ✓ Διάρκεια εκπαίδευσης για την «πρόχειρη» φάση: 2
- ✓ Αρχική ακτίνα για την φάση τελειοποίησης: 1
- ✓ Τελική ακτίνα για την φάση τελειοποίηση: 1
- ✓ Διάρκεια εκπαίδευσης κατά την φάση τελειοποίησης: 10

Το διάγραμμα των εκτιμήσεων του bitrate, των πραγματικών μετρήσεων του και το συγκριτικό διάγραμμα αυτών απεικονίζονται στην Εικόνα 11.

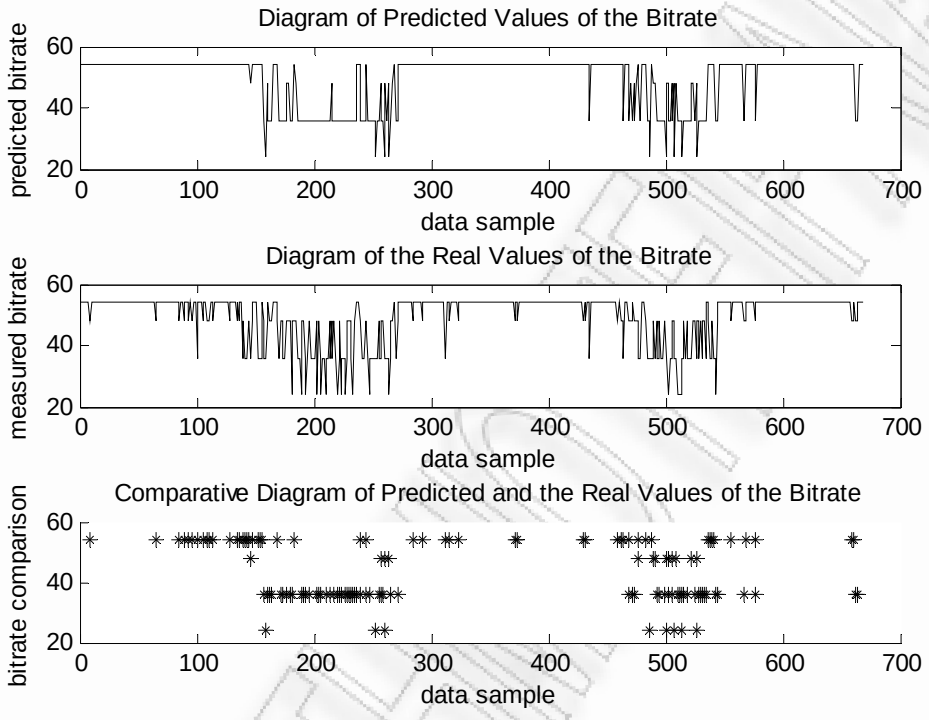


Εικόνα 11. Αλγόριθμος μαζικής εκπαίδευσης στο σενάριο κανονικοποιημένης απόκλισης των δεδομένων στο διάστημα [0,1]: Διάγραμμα εκτιμήσεων του bitrate, Διάγραμμα των πραγματικών μετρήσεων και συγκριτικό διάγραμμα αυτών. Το σύμβολο * απεικονίζει τα δείγματα δεδομένων που οι τιμές των εκτιμήσεων και οι μετρήσεις του bitrate διαφέρουν.

Αντίστοιχα, ο βέλτιστος συνδυασμός των τιμών των παραμέτρων εκπαίδευσης για τον αλγόριθμο σειριακής εκπαίδευσης σε αυτό το σενάριο ήταν ο εξής:

- ✓ Συνάρτηση Γειτνίασης: Ep
- ✓ Είδος μήκους (Length type): epochs
- ✓ Συνάρτηση Εκπαίδευσης (Learning function): inv
- ✓ Αρχική Ακτίνα για την «πρόχειρη» φάση: 3
- ✓ Αρχική Ακτίνα για την «πρόχειρη» φάση: 1
- ✓ Διάρκεια Εκπαίδευσης για την «πρόχειρη» φάση: 8
- ✓ Αρχικό Άλφα για την «πρόχειρη» φάση: 0.5
- ✓ Αρχική Ακτίνα για την φάση τελειοποίησης: 1
- ✓ Τελική Ακτίνα για την φάση τελειοποίησης: 1
- ✓ Διάρκεια Εκπαίδευσης για την φάση τελειοποίησης: 20
- ✓ Αρχικό Άλφα για την φάση τελειοποίησης: 0.05

Το διάγραμμα των εκτιμήσεων του bitrate, των πραγματικών μετρήσεων του και το συγκριτικό διάγραμμα αυτών απεικονίζονται στην Εικόνα 12.



Εικόνα 12. Αλγόριθμος σειριακής εκπαίδευσης στο σενάριο κανονικοποιημένης απόκλισης των δεδομένων στο διάστημα [0,1]: Διάγραμμα των εκτιμήσεων του bitrate, των πραγματικών μετρήσεων του και το συγκριτικό διάγραμμα αυτών. Το σύμβολο * απεικονίζει τα δείγματα δεδομένων που οι τιμές των εκτιμήσεων και οι μετρήσεις του bitrate διαφέρουν.

Τέλος, σε αυτό το σενάριο κανονικοποίησης, η σύγκριση του αποτελέσματος του αλγορίθμου μαζικής εκπαίδευσης με αυτό του αλγορίθμου σειριακής εκπαίδευσης αποκάλυψε ότι το πρώτο αποτέλεσμα, ίσο με 21,1%, είναι ελαφρώς μικρότερο από το δεύτερο, ίσο με 22,5%. Αντίστοιχα μετρήθηκαν και σε αυτό το σενάριο οι χρόνοι εκπαίδευσης του χάρτη. Συγκεκριμένα, σε αυτό το σενάριο, σύμφωνα με το πρόγραμμα ο αλγόριθμος μαζικής εκπαίδευσης απαιτεί λιγότερο από 1 δευτερόλεπτο ενώ ο αλγόριθμος σειριακής εκπαίδευσης απαιτεί περίπου 7 με 8 δευτερόλεπτα. Συνδυάζοντας τα παραπάνω αποτελέσματα, η βέλτιστη επιλογή είναι ο αλγόριθμος μαζικής εκπαίδευσης.

ΚΕΦΑΛΑΙΟ 5: Συμπεράσματα

Η γρήγορη εξέλιξη των ασύρματων επικοινωνιών απαιτεί την χρήση συστημάτων που έχουν την δυνατότητα να προσαρμόζονται έξυπνα στα ποικίλα περιβάλλοντα της σύγχρονης εποχής. Τα cognitive συστήματα είναι μια πολλά υποσχόμενη τεχνολογία προς αυτήν την κατεύθυνση αλλά η «γνωστική» διαδικασία που χρησιμοποιούν για τον έλεγχο, την αξιολόγηση και την επιλογή μίας παραμετροποίησης είναι συχνά ιδιαίτερα χρονοβόρα οδηγώντας στην ανάγκη χρήσης μιας τεχνικής εκπαίδευσης για να την επιταχύνει. Στην εργασία μας χρησιμοποιήσαμε μια μη ελεγχόμενη τεχνική εκπαίδευσης, τους χάρτες αυτό-οργάνωσης, για να εκπαιδεύσουμε το CRS να προβλέπει το bitrate που μπορεί να επιτευχθεί υπό μία παραμετροποίηση βάσει της προηγούμενης εμπειρίας του. Πραγματοποιώντας αρκετές περιπτώσεις ελέγχου καταφέραμε να προβλέψουμε σωστά το bitrate σε ποσοστό 79,9% των δειγμάτων δεδομένων που ελέχθησαν. Η τεχνική που προτείνουμε αναμένεται να βοηθήσει το cognitive σύστημα στην επιλογή μεταξύ διαφορετικών υποψήφιων παραμετροποιήσεων λαμβάνοντας υπόψην τις εκτιμήσεις του bitrate που μπορεί να επιτευχθεί.

ANNEX B: MATLAB CODES

version12.m

```
training
centers2
dataevaluation
```

training.m

```
echo off
ans='error!';

%selection of version
while ans=='error!'
input('type 1 for the 1st version or 2 for the 2nd version:', 's');
if ans=='1'
   labeltype='vote';
elseif ans=='2'
   labeltype='add1';
else sprintf('error!')
end
end

%call of som user interface in order to select the data file which contains
%the data samples for the training and to set the initialization and the
%training variables

%initialization and load of the data file which was used for the training
input('insert the name of the data file which will be used for the training:', 's');
```

```
sd = som_read_data([ans,'.data']);   %data struct
clear ans

sd = som_normalize (sd, 'range');

som_gui

echo on
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
echo off

%initialization and load of the som map which was created after the
%training phase
```

```
input('insert the name of the cod file:', 's');
sm = som_read_cod([ans,'.cod']);  %som map struct
clear ans

%autolabel of each data sample according the data file
auto = som_autolabel(sm, sd, labeltype);

%som grid struct, from this structure is possible to load both the labels of the
%cells, from the .label field, and their coordinations from the .coord field.
%Thus, it is possible to sort out cells with the same labels and to define
%the center of their cluster.

sg = som_grid(sm);
sg.label = auto.labels;

% image of an empty som map
som_show(sm,'umat', '', 'bar', 'none')

%echo on
%pause % Strike any key to continue...
%echo off

%label addition on the cells of the empty som map
som_show_add('label', auto)
```

centers2.m

```
%calculation of the centers of the clusters
mapsize=(sg.msize(1,1))*(sg.msize(1,2));
k1=0;
x1=0;
```

```
y1=0;
k2=0;
x2=0;
y2=0;
k3=0;
x3=0;
y3=0;
k4=0;
x4=0;
y4=0;
labelsize = size(sg.label);
for v=1:mapsize
   for q=1:labelsize(2)
      if sg.label{v,q}~=""
         c(v,q) = str2num(sg.label{v,q});
      else c(v,q)=0;
      end
      q=q+1;
   end
end

for m=1:mapsize
   for i=1:labelsize(2)
         if c(m,i)==24          %selection of cells with
                                %label equal to 24
            x1=x1+(sg.coord(m,1));   %participation of the
                                %cell to the calculation
                                %of x coordinate of the
                                %center of the cluster
                                %whose cells have label
                                %equal to 24
           y1=y1+(sg.coord(m,2));   %participation of the
```

```
                    %cell to the calculation
                    %of y coordinate of the
                    %center of the cluster
                    %whose cells have
                    %label equal to 24
    k1=k1+1;            %number of cells with
                    %label equal to 24
    m;                  %cell that has been
                    %tested

elseif c(m,i)==36           %selection of cells with
                    %label equal to 36
    x2=x2+(sg.coord(m,1));  %participation of the
                    %cell to the calculation
                    %of x coordinate of the
                    %center of the cluster
                    %whose cells have label
                    %equal to 36
    y2=y2+(sg.coord(m,2));  %participation of the
                    %cell to the calculation
                    %of y coordinate of the
                    %center of the cluster
                    %whose cells have
                    %label equal to 36
    k2=k2+1;            %number of cells with
                    %label equal to 36
    m;                  %cell that has been
                    %tested

elseif c(m,i)==48           %selection of cells
                    %with label equal
                    %to 48
```

```
x3=x3+(sg.coord(m,1));   %participation of the
                         %cell to the calculation
                         %of x coordinate of the
                         %center of the cluster
                         %whose cells have label
                         %equal to 48
y3=y3+(sg.coord(m,2));   %participation of the
                         %cell to the calculation
                         %of y coordinate of the
                         %center of the cluster
                         %whose cells have label
                         %equal to 48
k3=k3+1;                 %number of cells with
                         %label equal to 48
m;                       %cell that has been
                         %tested

elseif c(m,i)==54        %selection of cells
                         %with label equal to 54
x4=x4+(sg.coord(m,1));   %participation of the
                         %cell to the calculation
                         %of x coordinate of the
                         %center of the cluster
                         %whose cells have label
                         %equal to 54
y4=y4+(sg.coord(m,2));   %participation of the
                         %cell to the calculation
                         %of y coordinate of the
                         %center of the cluster
                         %whose cells have
                         %label equal to 54
k4=k4+1;                 %number of cells with
```

```
                        %label equal to 54
            m;          %cell that has been
                        %tested
          end
        end
        i=i+1;
        m=m+1;
    end
%center of the cluster whose cells have label equal to 24
cx1=x1/k1;
cy1=y1/k1;

%center of the cluster whose cells have label equal to 36
cx2=x2/k2;
cy2=y2/k2;

%center of the cluster whose cells have label equal to 48
cx3=x3/k3;
cy3=y3/k3;

%center of the cluster whose cells have label equal to 54
cx4=x4/k4;
cy4=y4/k4;

%all the centers of the clusters
cx = [cx1 cx2 cx3 cx4];
cy = [cy1 cy2 cy3 cy4];
sprintf(' center of the cluster whose cells have bitrate equal to 24 = (%0.3g, %0.3g) \n center of the cluster whose cells have bitrate equal to 36 = (%0.3g, %0.3g) \n center of the cluster whose cells have bitrate equal to 48 = (%0.3g, %0.3g) \n center of the cluster whose cells have bitrate equal to 54 = (%0.3g, %0.3g)', cx1, cy1, cx2, cy2, cx3, cy3, cx4, cy4)
```

```
 %display of the centres of the clusters on the last datagram of the map
hold on
plot(cx, cy, 'rv')
```

dataevaluation.m

```
%estimation of the cluster to which a data sample belongs and as
%a result estimation of its bitrate as well

%read and load of the data file whose data samples will be tested
input('insert the name of the data file whose data samples will be tested:', 's');
ed = som_read_data([ans,'.data']);    %data struct
clear ans

ed = som_normalize (sd, 'range');

%som_bmus depicts the number of the cell to which the data sample
%belongs
bmus = som_bmus(sm, ed);

%size of the table of bmus
l = size(bmus);

%for each bmus, and thus for each cell that represents a data sample,
%is calculated its distance from the center of each cluster

for i=1:l(1)

   %distance from the center of the cluster which has label/bitrate
   %equal to 24
   d1 = sqrt((cx1-sg.coord(bmus(i),1))^2 + (cy1-sg.coord(bmus(i),2))^2);
```

```
    %distance from the center of the cluster which has label/bitrate
    %equal to 36
    d2 = sqrt((cx2-sg.coord(bmus(i),1))^2 + (cy2-sg.coord(bmus(i),2))^2);

    %distance from the center of the cluster which has label/bitrate
    %equal to 48
    d3 = sqrt((cx3-sg.coord(bmus(i),1))^2 + (cy3-sg.coord(bmus(i),2))^2);

    %distance from the center of the cluster which has label/bitrate
    %equal to 54
    d4 = sqrt((cx4-sg.coord(bmus(i),1))^2 + (cy4-sg.coord(bmus(i),2))^2);

    %selection of the center of the clusters which is closer to the
    %cell and classification of the data sample to the relevant
    %cluster/bitrate

    if d1<d2 && d1<d3 && d1<d4
       bitrate(i) = 24;
    elseif d2<d3 && d2<d1 && d2<d4
       bitrate(i) = 36;
    elseif d3<d2 && d3<d1 && d3<d4
       bitrate(i) = 48;
    elseif d4<d2 && d4<d3 && d4<d1
       bitrate(i) = 54;
    end

end

%display of the estimated bitrate of each data sample
k = bitrate';

%comparative diagram of the estimated values of the bitrate
```

```
%according to the SOM analysis per data sample and the real
%values that have been measured during the collection of
%the data samples per data sample

figure(2);
subplot(3,1,1)
plot(k,'k-')
title('Diagram of Bitrate Estimations')
xlabel('data sample')
ylabel('bitrate')

a=size(ed.labels);
b=a(1,1) ;

for i=1:b
z(i,1)=str2num(ed.labels{i,1});
i=i+1;
end
figure(2);
subplot(3,1,2)
plot(z,'k-')
title('Diagram of the Real Values of the Bitrate')
xlabel('data sample')
ylabel('bitrate')

%calculation of the estimations that differ from the real
%measurements and the ones that are the same. ed.data field
%is a data struct field from which the data samples were
%loaded

w = size(k);
same = 0;
```

```
different = 0;

for n = 1:w(1)
   if k(n) == z(n)
      same = same+1;
      n = n + 1;
   elseif k(n)~=z(n)
      different = different + 1;
      n = n + 1;
   end
end

same_per_cent = same * 100 /w(1);
different_per_cent = different * 100 /w(1);
sprintf(' the number of the correct estimations is %d and in percent %0.3g%% and \n the number of the wrong estimations is %d and in percent %0.3g%%', same, same_per_cent, different, different_per_cent)


figure(2)
subplot(3,1,3)
plot(k,'k-.')
hold on
plot(z, 'k*-')
title('Comparative Diagram of Bitrate Estimations (dashdot line) and their Real Values (solid line)')
xlabel('data sample')
ylabel('bitrate')
```

version3.m

training1
brandfr
totalfr
centers3
dataevaluation2

training1.m

```
echo off

%call of som user interface in order to select the data file which contains
%the data samples for the training and to set the initialization and the
%training variables

%initialization and load of the data file which was used for the training
input('insert the name of the data file which will be used for the training:', 's');
sd = som_read_data([ans,'.data']);   %data struct
clear ans

sd = som_normalize (sd, 'range');

som_gui

echo on
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
```

```
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
pause % Strike any key to continue...
echo off
%initialization and load of the som map which was created after the
%training phase
input('insert the name of the cod file:', 's');
sm = som_read_cod([ans,'.cod']);  %som map struct
clear ans

%autolabel of each data sample according the data file
auto = som_autolabel(sm, sd, 'freq');

%som grid struct, from this structure is possible to load both the labels of the
%cells, from the .label field, and their coordinations from the .coord field.
%Thus, it is possible to sort out cells with the same labels and to define
%the center of their cluster.

sg = som_grid(sm);
sg.label = auto.labels;
```

```
% image of an empty som map
som_show(sm,'umat', '', 'bar', 'none')
colormap(gray)

%echo on
%pause % Strike any key to continue...
%echo off

%label addition on the cells of the empty som map
som_show_add('label', auto, 'TextColor', 'cyan')
```

brandfr.m

```
%seperation of the label/bitrate and the frequency/ocurrences
lsize = size(sg.label);
hlsize = lsize(2);
vlsize = lsize(1);

for vls = 1:vlsize
   for hls = 1:hlsize
      lab = sg.label{vls, hls};
      if lab~=""
         if lab(5)==')'
            stbr = [lab(1), lab(2)];
            br(vls, hls) = str2num(stbr);
            stfr = lab(4);
            fr(vls, hls) = str2num(stfr);
         else
            stbr = [lab(1), lab(2)];
            br(vls, hls) = str2num(stbr);
            stfr = [lab(4), lab(5)];
```

```
            fr(vls, hls) = str2num(stfr);
        end
    else
        br(vls, hls)=0;
        fr(vls, hls)=0;
    end
    hls = hls + 1;
  end
  vls = vls + 1;
end
```

totalfr.m

```
%calculation of the total frequency/number of occurrences of
%all labels per cell

for vls = 1:vlsize
  tfr(vls,1) = 0;
  for hls = 1:hlsize
    tfr(vls,1) = tfr(vls,1) + fr(vls, hls);
    hls = hls + 1;
  end
  vls = vls + 1;
end
```

centers3.m

```
%calculation of the centers of the clusters
k1=0;
x1=0;
y1=0;
```

```
k2=0;
x2=0;
y2=0;
k3=0;
x3=0;
y3=0;
k4=0;
x4=0;
y4=0;

for vls = 1:vlsize
   for hls = 1:hlsize
     if tfr(vls,1)~=0
        %selection of cells with label equal to 24
        if br(vls, hls)==24
           %participation of the cell to the calculation of x
           %coordinate of the center of the cluster whose cells
           %have label equal to 24
           x1=x1+((fr(vls,hls))/tfr(vls,1))*(sg.coord(vls,1));
           %participation of the cell to the calculation of y
           %coordinate of the center of the cluster whose
           %cells have label equal to 24
           y1=y1+((fr(vls,hls))/tfr(vls,1))*(sg.coord(vls,2));
           %number of cells with label equal to 24
           k1=k1+1;
        %selection of cells with label equal to 36
        elseif br(vls, hls)==36
           %participation of the cell to the calculation of x
           %coordinate of the center of the cluster whose cells
           %have label equal to 36
           x2=x2+((fr(vls,hls))/tfr(vls,1))*(sg.coord(vls,1));
           %participation of the cell to the calculation of y
```

```
        %coordinate of the center of the cluster whose cells
        %have label equal to 36
        y2=y2+((fr(vls,hls))/tfr(vls,1))*(sg.coord(vls,2));
        %number of cells with label equal to 36
        k2=k2+1;
    %selection of cells with label equal to 48
    elseif br(vls, hls)==48
        %participation of the cell to the calculation of x
        %coordinate of the center of the cluster whose cells
        %have label equal to 48
        x3=x3+((fr(vls,hls))/tfr(vls,1))*(sg.coord(vls,1));
        %participation of the cell to the calculation of y
        %coordinate of the center of the cluster whose cells
        %have label equal to 48
        y3=y3+((fr(vls,hls))/tfr(vls,1))*(sg.coord(vls,2));
        %number of cells with label equal to 48
        k3=k3+1;
    %selection of cells with label equal to 54
    elseif br(vls, hls)==54
        %participation of the cell to the calculation of x
        %coordinate of the center of the cluster whose cells
        %have label equal to 54
        x4=x4+((fr(vls,hls))/tfr(vls,1))*(sg.coord(vls,1));
        %participation of the cell to the calculation of y
        %coordinate of the center of the cluster whose cells
        %have label equal to 54
        y4=y4+((fr(vls,hls))/tfr(vls,1))*(sg.coord(vls,2));
        %number of cells with label equal to 54
        k4=k4+1;
    end
    hls = hls + 1;
else
```

```
        hls = hls + 1;
      end
    end
    vls = vls + 1;
end

%center of the cluster whose cells have label equal to 24
cx1=x1/k1;
cy1=y1/k1;
%center of the cluster whose cells have label equal to 36
cx2=x2/k2;
cy2=y2/k2;
%center of the cluster whose cells have label equal to 48
cx3=x3/k3;
cy3=y3/k3;

%center of the cluster whose cells have label equal to 54
cx4=x4/k4;
cy4=y4/k4;

%all the centers of the clusters
cx = [cx1 cx2 cx3 cx4];
cy = [cy1 cy2 cy3 cy4];
sprintf(' center of the cluster whose cells have bitrate equal to 24 = (%0.3g, %0.3g) \n center of the cluster whose cells have bitrate equal to 36 = (%0.3g, %0.3g) \n center of the cluster whose cells have bitrate equal to 48 = (%0.3g, %0.3g) \n center of the cluster whose cells have bitrate equal to 54 = (%0.3g, %0.3g)', cx1, cy1, cx2, cy2, cx3, cy3, cx4, cy4)

%display of the centers of the clusters on the last datagram of
%the map
hold on
```

```
plot(cx, cy, 'rv')
```

dataevaluation2.m

```
%estimation of the cluster to which a data sample belongs and
%as a result estimation of its bitrate as well

%read and load of the data file whose data samples will be
%tested
input('insert the name of the data file whose data samples will be tested:', 's');
ed = som_read_data([ans,'.data']);    %data struct
clear ans

ed = som_normalize (sd, 'range');

%som_bmus depicts the number of the cell to which the data
%sample belongs
bmus = som_bmus(sm, ed);

%size of the table of bmus
l = size(bmus);
%for each bmus, and thus for each cell that represents a data
%sample, is calculated its distance from the center of each
%cluster

for i=1:l(1)

   %distance from the center of the cluster which has label/bitrate
   %equal to 24
   d1 = sqrt((cx1-sg.coord(bmus(i),1))^2 + (cy1-sg.coord(bmus(i),2))^2);
```

```
    %distance from the center of the cluster which has label/bitrate
    %equal to 36
    d2 = sqrt((cx2-sg.coord(bmus(i),1))^2 + (cy2-sg.coord(bmus(i),2))^2);

    %distance from the center of the cluster which has label/bitrate
    %equal to 48
    d3 = sqrt((cx3-sg.coord(bmus(i),1))^2 + (cy3-sg.coord(bmus(i),2))^2);

    %distance from the center of the cluster which has label/bitrate
    %equal to 54
    d4 = sqrt((cx4-sg.coord(bmus(i),1))^2 + (cy4-sg.coord(bmus(i),2))^2);

    %selection of the center of the clusters which is closer to the
    %cell and classification of the data sample to the relevant
    %cluster/bitrate
    if d1<d2 && d1<d3 && d1<d4
       bitrate(i) = 24;
    elseif d2<d3 && d2<d1 && d2<d4
       bitrate(i) = 36;
    elseif d3<d2 && d3<d1 && d3<d4
       bitrate(i) = 48;
    elseif d4<d2 && d4<d3 && d4<d1
       bitrate(i) = 54;
    end
end
%display of the estimated bitrate of each data sample
k = bitrate';

%comparative diagram of the estimated values of the bitrate
%according to the SOM analysis per data sample and the real
%values that have been measured during the collection of the
%data samples per data sample
```

```
figure(2)
subplot(3,1,1)
plot(k,'k-')
title('Diagram of Bitrate Estimations')
xlabel('data sample')
ylabel('bitrate')

a=size(ed.labels);
b=a(1,1) ;
t24=0;
t36=0;
t48=0;
t54=0;
for i=1:b
z(i,1)=str2num(ed.labels{i,1});
if z(i,1)==24
   t24=t24+1;
elseif z(i,1)==36
   t36=t36+1;
elseif z(i,1)==48
   t48=t48+1;
elseif z(i,1)==54
   t54=t54+1;
end
i=i+1;
end

figure(2)
subplot(3,1,2)
plot(z, 'k-')
title('Diagram of the Real Values of the Bitrate')
xlabel('data sample')
```

```
ylabel('bitrate')

%calculation of the estimations that differ from the real
%measurements and the ones that are the same. ed.data field
%is a data struct field from which the data samples were
%loaded

w = size(k);
same = 0;
different = 0;
br24=0;
br36=0;
br48=0;
br54=0;

for n = 1:w(1)
   if k(n) == z(n)
      same = same+1;
      r(n)= NaN;
      n = n + 1;
   elseif k(n)~=z(n)
      different = different + 1;
      r(n)= k(n);
      if z(n)==24
         br24=br24+1;
      elseif z(n)==36
         br36=br36+1;
      elseif z(n)==48
         br48=br48+1;
      elseif z(n)==54
         br54=br54+1;
      end
```

```
        n = n + 1;
    end
end

same_per_cent = same * 100 /w(1);
different_per_cent = different * 100 /w(1);
sprintf(' the number of the correct estimations is %d and in percent %0.3g%% and \n the number of the wrong estimations is %d and in percent %0.3g%%', same, same_per_cent, different, different_per_cent)
figure(2)
subplot(3,1,3)
plot(k,'k-.')
hold on
plot(z, 'k-')
title('Comparative Diagram of Bitrate Estimations (dashdot line) and their Real Values (solid line)')
xlabel('data sample')
ylabel('bitrate')
hold on
plot(r, 'k*')
```