METHODS AND ALGORITHMS OF TIME SERIES PROCESSING IN INTELLIGENT SYSTEMS

Sergey G. Antipov , Marina V. Fomina, Vadim V.Vagin,
Alexandr P. Eremeev, Vasilii A. Ganishev

Abstract: *Time series processing in intelligent is a complex cross–disciplinary problem that is posed in many research areas. In this paper two subproblems are considered: anomaly detection in time series and time series clustering on an example of speaker clustering.*

Keywords: *Time Series Processing, Anomaly Detection, Speaker Clustering, Neural Networks.*

Introduction

An Intelligent System (IS) is viewed as a computer system to solve problems that cannot be solved by human in real time, or a solution requires automated support. The solution should give results comparable to the decisions taken by a person who is a specialist in a certain domain. The most important class of problems whose solution requires the intelligent support is a complex technical object management. The main feature of such objects is that they are dynamic, have ability for developing, their state may change over time, so one needs to develop methods and algorithms that take into account a time factor. One of the basic tasks arising when processing temporal dependences is the task of a clustering and classification. The review of methods of such tasks solution will be given below. It is offered to consider a clustering problem on the example of the task of speaker recognition on a voice. The problem of classification is solved on the example of anomalies search in sets of time series.

The paper is structured as follows. In section 2, the concept of time series and problems of their processing are given. The most important problems arising in the case of time series analysis are considered. The classes of methods for their decision are numerated. Section 3 contains the review of the main clustering methods. In section 4, the clustering problem to audio signal processing is viewed. The offered method of a task solution and the practical implementation are described. Section 5 contains the description of the method and anomaly search algorithms in collections of time series. In subsection 5.1, setting up the anomaly search task is given. In subsection 5.2, the normalization of time series is presented. Subsection 5.3 represents search algorithms of exceptions in collections of time series for the cases of one and several classes. In subsection 5.4, results of computer modeling are given. In Section 6, there is given the temporal data model on the basis of which the temporal data model for a subsystem of time series processing in intelligent systems of real time can be described. Conclusions are presented in section 7.

The Problems of Processing Time Series

As for the analysis of complex technical objects behaviour requires the consideration of the time factor, there is a need to work with data that explicitly (or implicitly) contains time. In this regard, one has to deal with the problem of temporal data mining [*Roddick and Spiliopoulou, 1999, Weiqiang et al., 2002, Antunes and Oliveira, 2001*]. The most common case of this analysis is time series mining [*Perfilieva et al., 2013*]. Time series are used in many different areas (technics, economics, medicine, banking, etc.) and describe different processes that occur over time.

The problem of time series mining is important for solving the following tasks of process analysis.

1. A process state prediction depending on the qualitative evaluation of a current or previous state.
2. Abnormal event detection.
3. Time series trends or other change identification.

The following classes of methods are used to solve these problems: Associations, **S**equence, Classification, Clustering methods.

In combination with other methods of data mining, **prediction** or **forecasting** involves trend analysis, classification and model matching. The basis for all kinds of forecasting is the historical information stored in the database in the form of time series. If one can build or find patterns, that adequately reflect the object behaviour, it is likely that they can be used to predict the behaviour of a system in the future.

The problems of detecting trends, their qualitative assessment and forecast based on analysis of time series are of particular relevance due to the continuous growth of real-time data from the specific and complex technical objects, for example, sensors whose values change over time.

Consider the case where the object's behaviour is evaluated on the basis of particular parameter values observations.

In general, the time series TS is an ordered sequence of values $TS =< ts_1, ts_2, ts_{i,} .., ts_m >$ describing the flow of a long process, where the index *i* corresponds to a time mark. ts_i values can be sensor indications, product prices, exchange rates and so on.

Clustering Methods

The time series clustering problem belongs to a class of pattern recognition problems [*Vagin et al.,* 2008]. There are the following types of methods: hierarchical; methods based on statistical analysis and machine learning. The most commonly used methods are:

- agglomerative hierarchical clustering with increasing mixture of normal (Gaussian) distributions;

- Hidden Markov models;
- Kohonen networks;
- Histogram models;
- incremental self-organizing neural networks.

Agglomerative hierarchical clustering with increasing mixture of normal (Gaussian) distribution

A cluster corresponding to the time series can be described as a probability distribution. However, very often a cluster has a rather complicated shape, that can not be described by a single probability distribution. In such cases, a mixture of probability distributions is used for the description of a cluster.

For the exact cluster description, it is necessary to learn a mixture of distributions. For this very often the **EM-algorithm** (**E**xpectation-**M**aximization) is used [*Zhu et al., 2005*]. We introduce an auxiliary vector of hidden variables *G* that can be calculated for the known value of a parameter vector $\Theta = \{\theta_1, \ldots, \theta_s\}$. In its turn, a vector *G* helps to restore a vector Θ. The algorithm consists of two iterative repetition of steps [*Roweis, 1998*].

1. In the first step (**E**-step) the expected value of hidden parameters of a vector *G* is calculated for the current approximation of the vector Θ.
2. In the second step (**M**-step) the problem of maximizing the likelihood for a mixture of distributions is being solved, and a new vector Θ approximation is computed.

In the case when all mixture components have normal (Gaussian) probability density, one can represent the solution analytically. That is why mixtures of normal distributions (GMM - **G**aussian **M**ixture **M**odel) are used commonly in practice.

Agglomerative hierarchical clustering (AHC) is one of the most popular clustering methods, since it is simple to implement, but provides the accuracy sufficient for many applications. The basic idea is that at initialization, each time series are represented by separate clusters. Then, some clusters close one to another are merged together. This process is as long as certain criterion indicates that further merge does not lead to better results.

As a measure of the distance between clusters often distance-based Bayesian information criterion (BIC) is used.

However, the EM-algorithm can not be used at the early stages of the AHC, since most of initial clusters does not generally contain enough information for learning multicomponent GMM. The result is an overfitted model, which is correct only for the particular case.

In view of these shortcomings the method based on the incremental mixture of normal distributions (incremental GMM) was proposed in [*Han and Narayanan, 2008*]. The method is characterized by the following:

— a cluster, that is a union of two adjacent clusters, is represented by the distribution whose probability density is calculated as a weighted sum of merged clusters;

— the model is recursively updated after each combination of clusters using the hypothesis H_2, where the distance between clusters reaches a predetermined value.

Owing to the incremental GMM usage, researchers succeeded in increasing the quality of clustering at 4.47% [*Han and Narayanan, 2008*].

Hidden Markov models

Hidden Markov model (HMM) is a Markov process model, where the system initiating a process is in any state from $C = \{c_1, c_2, \dots, c_M\}$, but it is not known which one exactly. However, each of the state c_i with the likelihood b_{ix_j} produces an observed event x_j.

Formally HMM is defined as HMM = $\{C, X, \Pi, A, B\}$ [*Rabiner, 1989, Abdallah et al., 2012*], where:

- $C = \{c_1, c_2, \dots, c_M\}$ is a system states set (clusters);
- $X = \{x_1, x_2, \dots, x_N\}$ is a cluster centroids set;
- $\Pi = \{\pi_1, \pi_2, \dots, \pi_M\}$ is a list of initial (a priori) probabilities of membership to each cluster;
- $A = \{a_{ij}\}$ - the matrix of transitions between events: to which a cluster belongs the next feature vector;
- $B = \{b_{ix_c}\}$ is the matrix defined a communication probability of a feature *i-s* vector with a centroid x_c into a cluster c_i.

Each state consists of a set of substates. The probability density of each state is modelled by a GMM.

At absent of a priori knowledge about the number of time series types, the quantity of clusters is selected more than the maximum possible (over-clustering). In this approach, multiple clusters would correspond to one and the same type of time-series, that leads to the necessity of merging in the later stages of clustering. Let us describe the clustering algorithm.

In the first step of this algorithm, one needs to initialize HMM parameters. At this stage, one can use the classical *k*-means algorithm. For each cluster, GMM parameters are computed.

Then it is necessary to learn HMM using the **EM-algorithm** of clusterization.

On the second step of the algorithm it is necessary to converge several clusters belonging to the same type of time series to one. It is quite difficult to determine the optimal number of clusters analytically, so in practice one chooses $k = K$, where K is a value greater than the maximum possible number of clusters. In the learning process, the current cluster quantity k is reduced to k^*, the optimal number of clusters. Assuming that the correct cluster merge increases the value of the objective function and incorrect merge decreases it, one can use the BIC.

In [*Ajmera and Wooters, 2003*] it is shown that the BIC for this method can be optimized. Assumption is entered that an amount of parameters of every GMM must be a constant. This assumption greatly increases the speed of HMM learning.

Kohonen networks

In recent years, the usage of neural networks for time series clustering is of growing interest [*Ning et al., 2006*], [*Mori and Nakagawa, 2001*]. In particular, Kohonen network is frequently used because of the high rate of clustering.

This method is based on the projection from a multidimensional space in a two-dimensional one with some predefined structure. For Kohonen network learning, the Linde-Buso-Gray (LBG) algorithm is often used [*Linde et al., 1980*]. This algorithm helps to divide L feature vectors corresponding to the same time series type to M clusters. A centroids (cluster centres) set is called "codebook", it is unique for each time series type.

Several centroids in a codebook will match each time series type. In clustering step, a feature vectors are included to a cluster having the nearest centroid to this vector in its codebook [*Kumar and Rao, 2011*].

Incremental self-organizing neural networks

To remove the previous model limitations, self-organizing incremental neural networks (SOINN) have been developed [*Furao and Hasewaga, 2006*]. These neural networks are used for data clustering without a priori knowledge of its topology. Also, the model supports the learning without the ultimate goal for the whole period of the network operation (lifetime learning), it allows not to limit the maximum number of clusters. SOINN is the neural network with two layers. The first layer is used to determine the topological structure of the clusters, and the second is used to define the clusters number. First, the first layer is learned and then using the output of the first layer as an input, a second layer is trained. Researchers need to identify themselves the time to start learning the second layer, as well as to calculate the threshold T for each layer of the network. In the absence of a priori data structure knowledge, the threshold for the first layer is often selected adaptively.

The main idea of the algorithm is to build a probabilistic model of an input data. On the assumption that clusters form an area of high probability density, it is necessary to construct a graph that most accurately describes these areas and their relative positions in space. Its nodes lie in the areas of local likelihood maximum, and its edges join nodes belonging to the same cluster.

Large number of parameters that SOINN has and uncertainty in choosing a time to start the second layer learning makes this type of networks difficult to use in practice. Also, if there is any change in the first layer then the second layer must be completely retrained. It makes online learning or lifetime learning impossible.

To solve the above problems enhanced self-organizing incremental neural network (ESOINN) were developed [*Furao et al., 2007*]. The main difference of this approach is the use of single-layer neural network, which reduces the number of configurable settings.

Clustering Problem in Task Of Speaker Clustering

Problem Statement

Let us consider the problem of time series clustering on the example of audio signal processing.

Modern recording tools allow to introduce a sound signal into a time series, showing the change in sound intensity over time. However, this view is difficult to analyze because it contains a large amount of information noise. A signal spectrum is more informative for analysis than the signal itself. For calculation of the spectrum the Fast Fourier Transform algorithm is often used. It is easy to implement and has a complexity O($N\log_2 N$) less than the classical discrete Fourier transform algorithm O(N^2) [*Cooley and Tukey,* 1965].

Speaker clustering is a separation of voice recordings to some classes so that each class has only a voise of one user. Each recording contains a voice of a single person. This process is often an integral part of speaker recognition and speech recognition problems.

During evolution sounds in the lower frequency band contained the more useful information than ones in the higher frequency band. Mel-frequency cepstral coefficients (MFCC) were developed according to these characteristics of human hearing [*Vyas and Kumari, 2013*]. With these coefficients the information obtained from the low-frequency range is more carefully analysed, and the effect of high-frequency components typically containing extraneous noise is reduced.

Whole voice recording is divided into small intervals with duration ~ 10-30ms (signal quasi-stationary time) called frames. For each frame separately a set of mel-frequency cepstral coefficients is calculated.

The algorithm for computing mel-frequency cepstral coefficients can be described as follows [*Molau et al., 2001*]:

a) splitting a signal into frames;
b) an application of the weighting function (window) to each frame;
c) a use of the Fourier transform;
d) a use of mel-frequency filter;
e) a cepstrum calculation.

But MFCC change also contains unique information about the user's identity. In this work an extension of the previous acoustic vector by taking into account the dynamics of MFCC δ_i is used, which is expressed by the difference in a mel -frequency cepstral coefficients of the frame i and the previous one:

$$\delta_i\left(\tilde{C}_k\right)=\tilde{C}_k[i-1]-\tilde{C}_k[i]$$

Here each $\tilde{C}_k$ is a set of received mel-frequency cepstral coefficients, $k = 1,...,K$. K is quantity of mel - coefficients, the value of K is often chosen in a range from 12 to 24.

In this approach the first frame can not be used for clustering, since a change of its MFCC will be zero. L (the number of elements of the acoustic vector x) doubles:

$$L=|x|=\left|\left[\tilde{C}_1,...,\tilde{C}_k,\delta\left(\tilde{C}_1\right),...,\delta\left(\tilde{C}_k\right)\right]\right|=2\cdot K$$

A set of acoustic vectors will to be used for speaker clustering with ESOINN.

Practical implementation

For MFCC calculation freeware package Praat was used, it was developed at the University of Amsterdam [*Boersma and van Heuven, 2004*]. This tool implements many features required for speaker and speech recognition and has a simple interface and constant developers' support.

After building MFCC sets advanced acoustic vectors containing the information about the dynamic change of MFCC are computed based on them.

CMU Sphinx project [*CMU Sphinx*] provides three ready-made base of high quality voice recordings: CMU Arctic; CMU Chaplain; CMU Microphone Array Database.

The first two voice recording bases were used in the experiments for this paper.

CMU Arctic base is established in Language Technology Institute at Carnegie Mellon University [*CMU Arctic*] and is divided into 4 parts, each contains only one user speech in English. Records are presented in the format of Wave with 16 kHz sampling frequency and EGG, each of them is phonetically balanced.

Base CMU Chaplain was developed at Carnegie Mellon University together with the Lockheed Martin System Integration as part of a hardware and software system for automatic translation (Audio Voice Translation Guide System, also known as Tongues) [*CMU Chaplain*]. Recordings include a dialogue between two chaplains in the English, that lasts 4 hours 15 minutes.

For each frame, a set of 13 MFCC was calculated. Based on that the acoustic vector with 26 components was built.

To assess the quality of clustering a concept of precision A will be used (proportion of recordings correctly associated with the users), and the error E (the proportion of recordings incorrectly associated with the user), defined as follows:

$$V = V_T + V_F,\ A = \frac{V_T}{V},\ E = \frac{V_F}{V},$$

where V_T is the number of recordings correctly associated with the user, and V_F is the number of recordings incorrectly associated with the user.

The accuracy of speaker clustering method for each database CMU Arctic and CMU Chaplain can be represented by the following table:

Table 1. Experimental Results

Number of recordings	CMU Arctic	CMU Chaplain
100	84%	88%
200	86,5%	89,5%

It is worth noting that in the case of 100 recordings from CMU Arctic the system has identified five different users instead of four, but it is not mentioned in the calculation used for the accuracy of clustering users voice formula.

Anomaly Detection

The anomaly detection problem [*Chandola et al., 2009*] is set up as the task of searching for patterns in data sets that do not satisfy some typical behaviors. The ability to find abnormalities in a data set is important in a variety of subject areas: in the complex technical system analysis (e.g. satellite telemetry), in network traffic analysis, in medicine (analysis of MRT images) in the banking industry (fraud detection) and etc.

The anomaly, or "blowout" is defined as an element that stands out from the data set which it belongs to and differs significantly from the other elements of the sample. Informally, the problem of anomaly detection in time series sets is formulated as follows. There is a collection of time series describing some processes. This collection is used to describe normal processes. It is required to construct a model on the basis of the available data, that is a generalized the description of normal processes and allows to distinguish between normal and abnormal processes.

The task is complicated by the fact that a set of input data is limited and does not contain any examples of abnormal processes. It does not specify a criteria by which it would be possible to distinguish the <normal> and <abnormal> time series. In this regard, it is difficult to accurately assess the quality of the algorithm (the percentage of correctly detected anomalies, the number of false positives and the number of missed abnormalities). In addition, many algorithms are working well for some data sets will not fit well for other subject areas. It may also vary a criteria for determination the <correct> time series.

Let there be a set of objects, where each object has a time series:

$TSStudy = < TSstudy_1, TSstudy_2, .., TSstudy_m >$ is a learning set. Each the time series in the learning set is an example of <normal> process flow. Based on the analysis of time series of *TSStudy* one needs to build a model to refer the testing set of time series $TSTEST=<TStest_1, TStest_2, .., TStest_m>$ to <normal> or < abnormal> by some criterion.

This problem should be divided into two cases: the first case, when learning set contains examples of a single class; the second case, when the learning set contains examples of several classes. In the first case the fact of member of these objects to the class of the training set is important. In the second case one needs to further define an object belongs to a particular class.

Here are the main methods used to solve the problem of classification.

Classification is used to learn the model for the data assigned to different classes (learning stage), and to refer instances to one of the existing classes using the resulting model (test stage). Anomaly detection methods are based on the classification, it is assumed that if a classifier can be learned in an existing feature space, it can separate the normal and abnormal objects.

The advantage of anomaly detection methods based on classification includes the ability to use a lot of techniques and algorithms developed in the machine learning field, especially in the case where the learning set contains examples of several classes. Further test stage is fast compared to other classes of methods, as used originally constructed model (classifier).

Problem statement

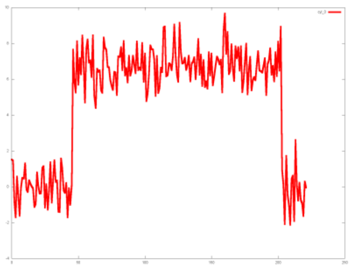
Let there be a set of objects, where each object is a time series:

$TSStudy = < TS\mathrm{study}_1, TS\mathrm{study}_2, \ldots , TSstudy_m >$ is a learning sample. Each of time series in the learning set is an example of a "normal" process flow. Based on time series mining from a set *TSStudy* one needs to build a model to refer time series from the test sample $TSTEST=<TStest_1, TStest_2, .., Tstest_l>$ to "normal" or "anomalies" on the basis of some criterion.

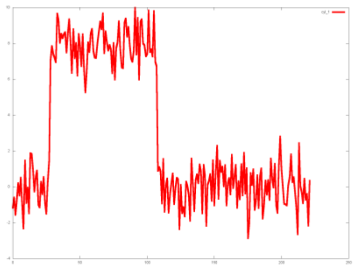
Let us consider this problem with a simple example. Let *TSStudy* learning set consists six time series (Fig. 1).

Test sample *TSTEST* consists of three time series (Fig. 2).

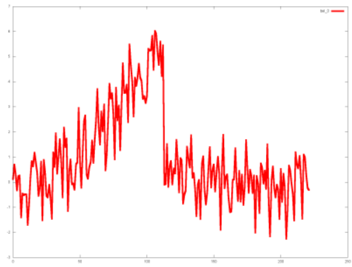
Based on the above problem statement it is clear that the time series (1), (2) and (6) of a study set are highly similar to each other, and therefore are members of the same class, let it be Class 1. Time series (3), (4) and (5) are also similar, but belong to another class, let it be Class 2. The test set (Fig. 2) shows that the time series (1) is likely to be a member of Class 1, a time series (2) is a member of Class 2. The third time series is significantly different from the previous two, and apparently "not similar" to any in the learning set. This suggests that the process by which time series (3) from the test sample was received is different from processes, by which time series from the learning set were obtained. On the contrary, the time series (1) and (2) from the test set (Fig. 2) are not anomalies, since their shapes are very "similar" to the individual time series in the learning set.



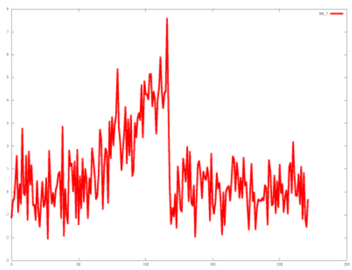
1st time series of a study set



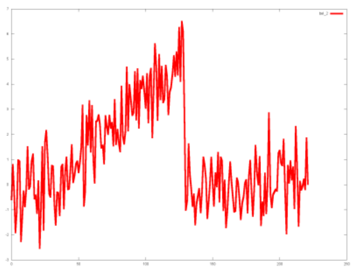
2nd time series of a study set



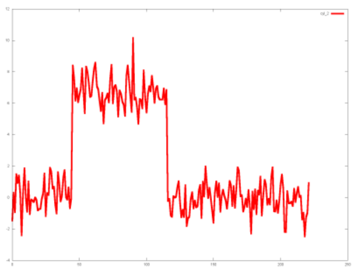
3rd time series of a study set



4th time series of a study set

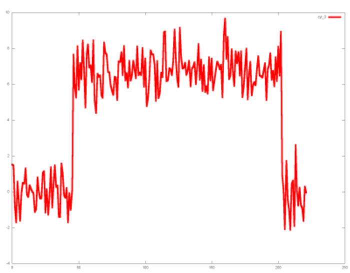


5th time series of a study set

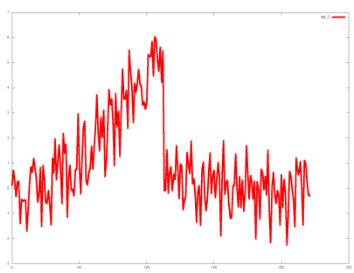


6th time series of a study set

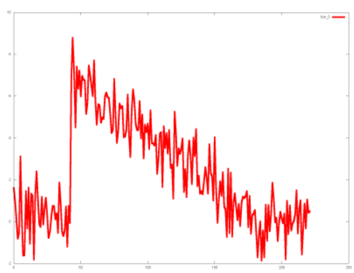
Figure 1. An example of a study set



1st time series of a test set



2nd time series of a test set



3rd time series of a test set

Figure 2. An example of a test set

Methods of time series representation

To create an algorithm summarizing an information provided by time series it is required, of course, to develop methods of time series pre-conversion. It is required to cause time series, that represent data from different areas, in different units to some forms convenient for further analysis. To work with time series, it is proposed to use two methods of representation a normalized and a symbolic representation. Normalization is bringing time series to a form that its mean value would be equal to zero and standard deviation would be one; this transformation is a necessary process of the data pre-processing [*Lin et al., 2003*]. Examples of current and normalized time series are shown in Table 2 (lines 1 and 2)

Table 2. Current and normalized time series representation

Time *t*	1	2	3	4	5	6	7	8	9
Absolute values	512	1448	88	1448	1448	1448	1448	1024	512
Normalized values	-1.0415	0.7478	-1.852	0.7478	0.7478	0.7478	0.7478	-0.0627	-0.0415
Symbolic representation of normalized values	*C*	*P*	*A*	*C*	*C*	*C*	*C*	*J*	*C*

Symbolic time series representation can be obtained from normalized one using the algorithm Symbolic Aggregate approXimation [*Lin et al., 2003*]. To perform this conversion the alphabet ***A*** - finite set of characters is introduced: $\boldsymbol{A} =\{a_1, a_2, .., a_{|A|-1}\}$.

An example of symbolic representation for the time series is given in Table 2 (line 3). Alphabet ***A*** was examined consist of 20 characters, ***A*** = {*A, B, C, ..., T*}.

Anomaly Detection Algorithm in time series sets

This paper proposes a method for the anomaly detection in the sets of time series, that is a modification of a method based on an accurate exceptions description [*Arning and Agrawal, 1996*]. The original formulation of this problem is given in [*Arning and Agrawal, 1996*]:

for a given set of objects *I* one needs to get an exclusion-set Ix. To do this, set for set *I* are introduced:

The function of dissimilarity D ($I_{j.}$), $I_j \in I$ defined on P (I), the set of all subsets of *I* and receiving positive real value;

The cardinality function $C(I_j)$: $I_j \in I$, defined on P (I) and receiving positive real value such that for any $I1 \subset I$, $I2 \subset I$ performed $I1 \subset I2 \Rightarrow C$ $(I1)$ <C $(I2)$;

The smoothing factor SF (Ij) = C $(I \setminus Ij)$ • $(D$ (I) - D $(I \setminus I_j))$, which is calculated for each $Ij \subseteq I$. Then $I_x \subset I$ will be considered an exclusion-set for I with respect to *D*, and *C*, if its smoothing factor SF (I_x) is maximal [*Arning and Agrawal, 1996*].

The algorithm TS-ADEEP that is based on this method was adapted for anomaly detection problem in sets of time series. As set *I* sets $TSStudy \cup \{ts\_test_j\}$ for each $ts\_test_j \in TStest$ are considered.

Dissimilarity function for time series will be set as follows:

$$D(I_j) = \frac{1}{N} \cdot \sum_{a \in I_j} \left| a - \overline{I_j} \right|^2 \quad \text{where} \quad \overline{I_j} = \sum_{a \in I_j} \frac{a}{|I_j|}$$

First I, the average for the time series of I_j is calculated. Dissimilarity function is calculated as the sum of squared distances between the mean and vectors of I_j. The cardinality function is given by the formula C $(I$ - $Ij)$ = 1 / | Ij | +1. The formula for calculating the smoothing factor is

SF (Ij) = C $(I$ - $Ij)$ · $(D$ (I) - D $(Ij))$. If an exclusion-set Ix, received for I = $TSStudy \cup \{ts\_test_j\}$ contains ts_test_j, then ts_test_j is an anomaly.

To determine anomalies in sets of time series based on the method described above the algorithm TS-ADEEP was developed.

In this paper, we propose the algorithm TS-ADEEP-Multi that is a generalization of the algorithm TS-ADEEP for the case of a learning set that contains examples of several time series classes. The generalization is quite obvious: dividing learning set into subsets containing examples of only one class and consistently applying to them and to each time series from a test set the algorithm TS-ADEEP, one can determine whether the considered time series are anomaly. For cases when

1) time series is an anomaly for each subset;
2) time series is not an anomaly for only a subset of the learning sample;

the answer is obvious. However, there is a case where the time series is not an anomaly for several classes of the learning set (but not all).

The pseudocode for algorithm TS-ADEEP-Multi is shown below:

The algorithm **TS-ADEEP-Multi**

input: (*TS Study*: learning set that contains examples of several classes;

TSTest: test set)

output: *TSAnom_Optimistic* - a set of anomaly time series of on the "optimistic" assessment

TSAnom_Pessimistic - a set of anomaly time series on the "pessimistic" assessment

begin

TSAnom_Optimistic = ∅; *TSAnom_Pessimistic* = ∅

Let *N* be a number of classes containing in the learning set

TSStudy_C = {$TSStudy\_C_1$, $TSStudy\_C_2$, .., $TSStudy\_C_N$} is a partition of *TSStudy* such that $TSStudy\_C_k$ contains only examples of class *k, k = 1..N*

for *j* from 1 to | *TSTest* |

choose $TSTest_j$ of *TSTest*

for *k* from *1* to *N*

$$I = TSStudy\_C_k \cup TSTest_j$$

Find the exclusion-set I_x in I

If the $TSTest_j \in I_x$ is an anomaly for the class *k* (it does not belong to him) then break

If $TSTest_j$ *does* not belong to any of classes $TSStudy\_C_k$, *k = 1 ..N* then

TSAnom_Optimistic = TSAnom_Optimistic ∪ $TSTest_j$

TSAnom_Pessimistic = TSAnom_Pessimistic ∪ $TSTest_j$

If *TSTestj* belongs to a unique class $TSStudy\_C_k$ then it is not an anomaly

If *TSTestj* belongs to several classes $TSStudy\_C_k$ then

TSAnom_Pessimistic = TSAnom_Pessimistic ∪ $TSTest_j$

print *TSAnom*

end

The simulation results for anomaly detection in time series

A simulation of anomaly detection process was conducted on synthetic and real data. As the synthetic data were taken classic time series description used in the scientific literature: «cylinder-bell-funnel» [*Naoki, 1994*] and «control chart» [*Pham and Chan, 1998*]. As the real was used data collected through special systems traffic analysis during files transfer via various protocols.

«Cylinder-bell-funnel» [*Naoki, 1994.*] contains three different classes - "cylinder", "bell", "funnel".

«Control chart» [*Pham and Chan, 1998*] contains six different classes that describe the trends may be presented in the process: cyclical, decreasing value, sharp drop, increasing value a constant, sharp increase.

In order to determine how well the proposed algorithm deals with anomaly detection in time series, several experiments were conducted. Let consider the simulation data set «cylinder-bell-funnel». First, as a learning set *TSStudy* considers as a set of time series belonging to two of the three classes, for example, "cylinder" and "bell". As test set *TSTEST* considers as set of time series belonging to all three classes «cylinder», «bell», «funnel». Time series $TStest_j$ is "normal" if it is a member of «cylinder» or «bell» classes and "abnormal" if it is not a member of them. Accordingly, the algorithm correctly finds anomalies, if it considers the time series of a class «funnel» from *TSTEST* as anomalies. It has been considered as a numerical representation of the time series and symbolic ones with a different alphabet sizes. Similarly, simulations carried out for the other pairs of classes: «bell» and «funnel», «cylinder» and «funnel».

The experiment showed that the proposed algorithm for anomaly detection does not always show good results: for some pairs of classes only a little more than half of the time series are correctly assigned to anomalies.

To improve this situation, it is proposed to further process the original time series by reduced or compressing it on normalization stage. This makes it possible to discard irrelevant details and to get rid from the noise. Example of time series compression is shown in Figure 3 (each ten points of original time series were assigned to a single point of new time series). The red lines in the figure connects points of original time series. The green line shows compressed time series: horizontal segments correspond to the 10 points of the original time series.

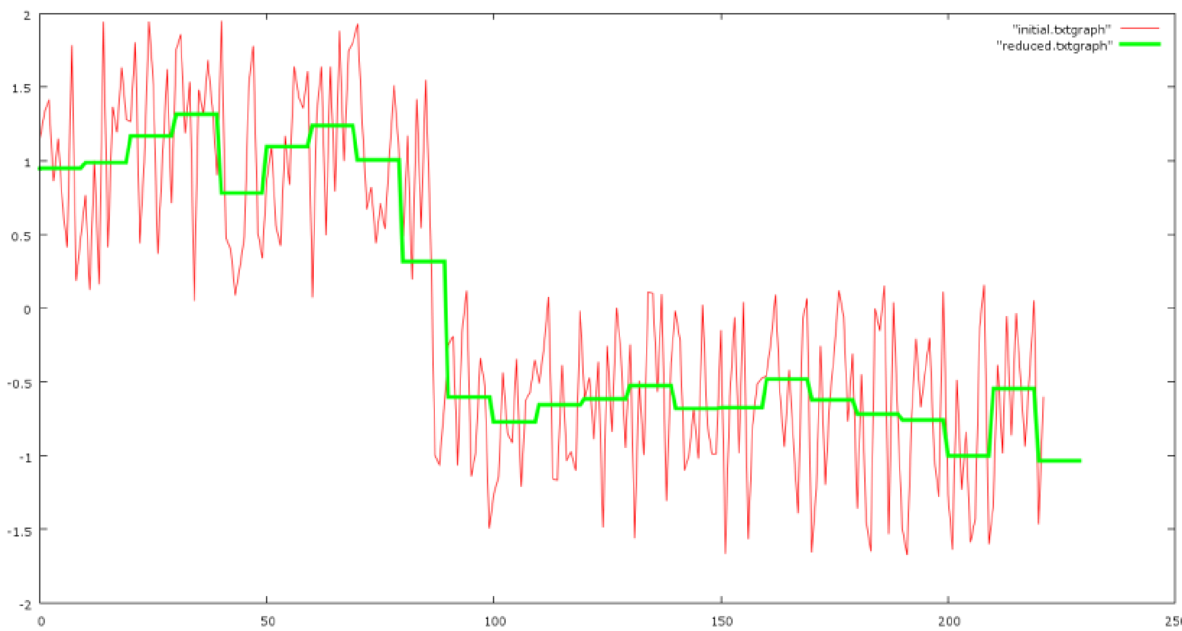


Figure 3. Example of time series compression

It has been verified in practice that the use of compressed time series algorithm for anomaly detection is significantly better than the use of it on the original data without compression (normal). Figure 4 shows a results comparison of a successful anomaly detection when using raw data without compression and compressed ones for «cylinder-bell-funnel». The *Y*-axis shows the percent of correctly recognized time series (normal or anomaly). The *X* axis presents four cases: a numerical representation of a value and a symbolic representation of a value (alphabet size 20, 35 or 50).

Figures 5 and 6 show results when using raw data without compression, and compression for the data sets «control chart».

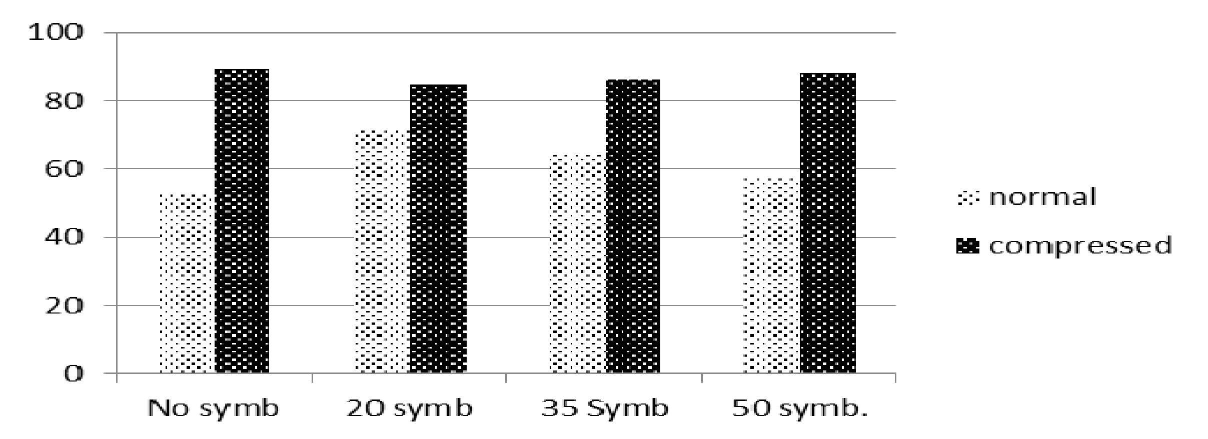


Figure 4. Comparison of a successful anomaly detection when using raw data without compression and compressed ones for «cylinder-bell-funnel»

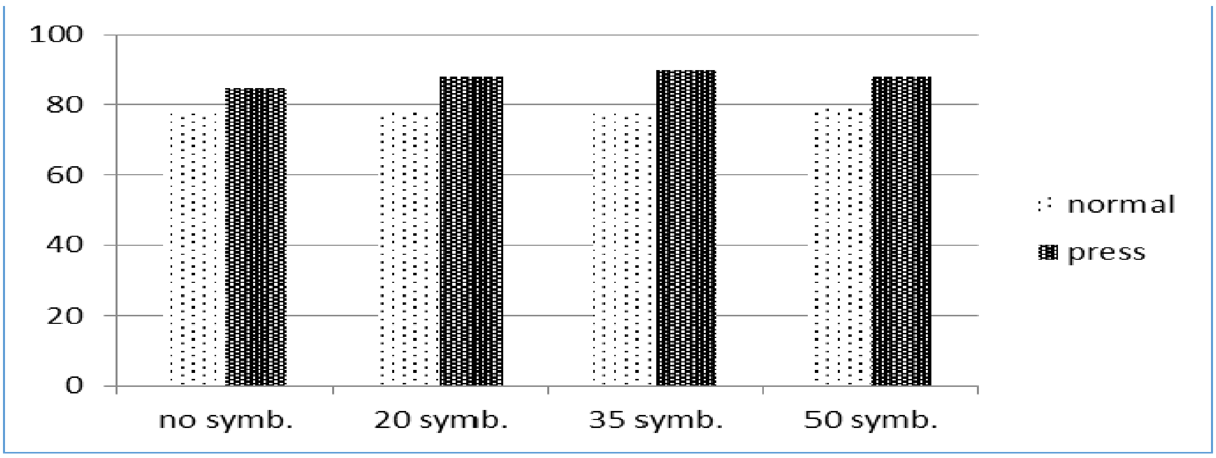


Figure 5. Comparison of a successful anomaly detection when using raw data without compression and compressed ones for «control chart», case of two classes

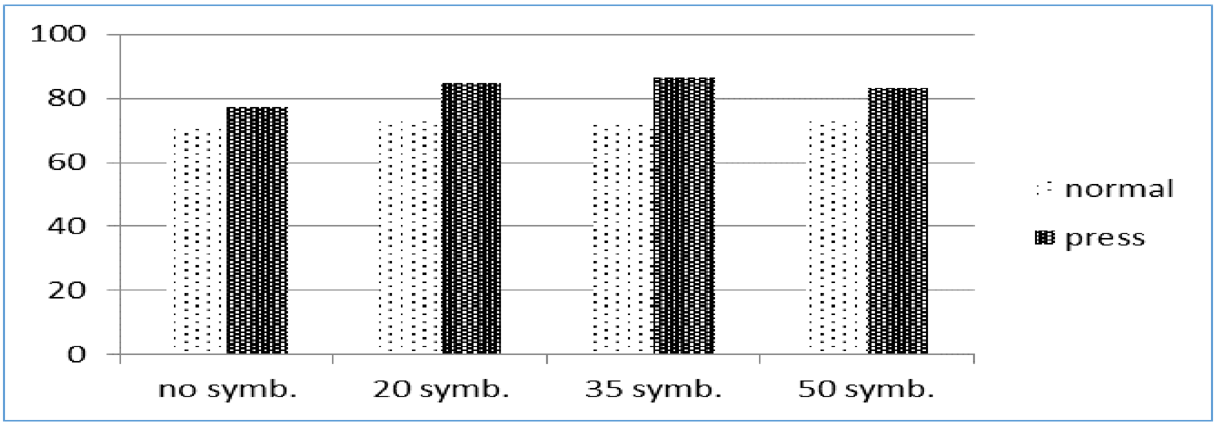


Figure 6. Comparison of a successful anomaly detection when using raw data without compression and compressed ones for «control chart», case of three classes

Temporal Data Model

One of important problems arising under treating temporal data is a problem of representing temporal data in a convenient form under processing. There is supposed a temporal data model that can be used in intelligent decision support systems for storing and converting temporal dependencies [Eremeyev et al., 2010].

The object *O* of a temporal data model (TDM) is any information or structural entity, for example, domain, attribute, relation defined in a time interval *I*. In TDM a structure or values of any object are changed in time (what is typical for time series), therefore a category of time is a basic entity. The operator of a range of the definition for *O* is $T: O \rightarrow I$ that returns a time interval when an object was defined. Let's consider TDM in a close analogy with a relation model since basic entities of both models coincide.

It is known that in a relation model, a domain is a set of one-typed values, for example, an integer set. A domain is simple if all its values are atomic (undecomposed) [Connolly, 2002]. A domain in TDM also satisfies these requirements, however it is defined in some time interval. A temporal cortege is determined in an interval *I* if in *I* domains of all attributes and its values are known in any moment from *I*.

In a relational model, a relation consists of cortege sets and each cortege has the same attribute set. For a temporal relation, a range of the definition (life time) is given through ranges of all corteges (records) inputting in a relation. In the general case, a cortege structure of a temporal relation can be arbitrary changed. In this connection, the identification problem of corteges appears. There is introduced an operator that for each cortege defines some unique key that is not changed under changing a cortege structure or values of cortege fields. A set of possible keys is associated with every relation R.

A key, as any element of TDM, has an own range of the definition, therefore, for choosing a primary key from the whole set of possible keys, it needs to choose a key that is defined in the whole required interval.

Constrains of TMD integrity. Rules of TMD integrity are intended for verifying entities and reference integrity. Integrity of entities means that at each moment, a value of primary key is one-valued determined, and reference integrity means that for each value of an external key appearing in a sibling relation, it needs to find a cortege with the same value of a primary key in a parent relation. And a definition range for values of an external key should be included in a definition range of a primary key.

Let's consider data processing in TMD. For this purpose, the operations for manipulating n-ary temporal relations are defined. Corresponding operations for a relational model are described in [Connolly, 2003]. Further *R* and *S* – relations; *A,B,C*, and so on (possible with indexes) – collections of attributes; *c* – a cortege of corresponding degree with corresponding domains.

The operation *THETA-SELECT (constrain)*. The result of performing the operation:

R[A θ c]– a cortege set from *R*, each of which satisfies a condition that *A* – component is in the relation with the *B* – component. If the relation *θ* is equality (widespread case), the operation *THETA-SELECT* is called simple *SELECT*.

The operation *PROJECTION.* The result of performing the operation: $R[A_1, A_2, \dots A_n]$ – the relation obtained by deleting all columns from *R* with the exception that are specified by attributes $A_1, A_2, \dots A_n$ and following deleting surplus line - duplicates and a range of definition projection coincides with a definition range of an original relation.

Operation *THETA-JOIN.* The result is a concatenation of relation lines R with relation lines S in accordance with the given condition defined by the relation *θ.* :For TDM, a range of definition *THETA-JOIN* correspond to an interval where the relation *θ* has been performed. If the relation *θ* is equality then the operation *THETA-JOIN* is called *EQUI-JOIN*.

Operation *NATURAL JOIN.* This operation is analogous to the operation *EQUI-JOIN* with the exception that in this case surplus columns generated under performing join are eliminated. Natural join is join used under normalization of relation collection.

Operation *DIVIDE*. Let relation $R(A,B_1)$ and $S(B_2,C)$ be given such that B_1 and B_2 are defined on the same domain (*S*). Then the result of this operation is a maximal subset *R[A]* such that its Cartesian product with $S[B_2]$ is included into *R*.

The possibility of a TDM implementation as a dialect of the widespread SQL on the basis of often used and capable of adaptation of DBMS with the open code SQLite has been considered in [Eremeeev et al., 2012]. The given model is a base for realization of the temporal DB of an intelligent decision support system (real time) and allows to operate with temporal dependencies including time series.

Conclusion

In this paper two approaches to processing of temporal data are considered. The approach based on clustering was applied to the solution of the problem of speaker clustering. Mel-frequency cepstral coefficients were used as speaker features. We propose a use of self-organizing incremental neural networks, because they have an ability for life leaning and no need in a priori knowledge about speakers or their quantity. A use of extended feature vectors with MFCC dynamics can improve the accuracy of speaker clustering. Currently the possibility of segmentation of a recording unit is under study for recordings that contain voice of two or more speakers.

Next we consider the problem of anomaly detection among sets of time series. We propose a nonparametric algorithm TS-ADEEP-Multi for anomaly detection in time series sets for the case when the learning set contains examples of several classes. The method for improving the accuracy of anomaly detection, due to "compression" of these time series is used to get rid of unnecessary detail and noise. In the future it is expected to modify the proposed algorithm to define abnormalities in the sets of time series for the case when the classes of time series are not known a priori.

Acknowledgements

The work is executed at financial support of RFBR (projects 14-07-00862, 14-01-00427, 15-01-05567) and in frame of Scientific research work T 2.737.2014/K of the project part of the government job in the sphere of scientific activity.

Bibliography

[Abdallah et al., 2012] S.J. Abdallah, I.N. Osman, M.E. Mustafa Text-Independent Speaker Identification Using Hidden Markov Model In: World of Computer Scienceand Information Technology Journal (WCSIT), 2012, pp. 203-208.

[Agrawal and Srikant, 1995] R. Agrawal, R. Srikant Mining Sequential Patterns In: Proceedings of the Eleventh International Conference on Data Engineering.ICDE ’95. Washington, DC, USA: IEEE Computer Society, 1995, pp. 3-14 .

[Ajmera and Wooters, 2003] J. Ajmera, C. Wooters A Robust Speaker Clustering Algorithm In: IEEE Workshop on Automatic Speech Recognition and Understanding, 2003, pp. 411-416.

[Arning et al., 1996] A. Arning, R. Agrawal, P. Raghavan A Linear Method for Deviation Detection in Large Databases In: Proceedings of KDD’1996. 1996. pp. 164–169.

[Antunes and Oliveira, 2001] M. Antunes, A. L. Oliveira Temporal data mining: an overview In: Eleventh International Workshop on the Principles of Diagnosis. 2001.

[Boersma and van Heuven, 2004] P. Boersma, V. van Heuven Praat, a system for doing phonetics by computer In: Glot International 5(9/10), 2004, pp. 341-345.

[CMU Arctic] CMU Arctic: http://www.festvox.org/cmu_arctic/

[CMU Chaplain] CMU Chaplain: http://www.speech.cs.cmu.edu/Tongues/

[CMU Sphinx] CMU Sphinx: http://cmusphinx.sourceforge.net/wiki/

[Connolly and Begg, 2002] T.M. Connolly, C.E. Begg Database Systems: A Practical Approach to Design, Implementation, and Management. Third Edition, Addison-Wesley, 2002, 1324 p.

[Cooley and Tukey, 1965] J.W. Cooley, J.W. Tukey An Algorithm for the Machine Calculation of Complex Fourier Series In: Mathematics of Computation, 1965, pp. 297-301.

[Eremeev et al., 2010] A.A. Eremeev, A.P. Eremeev, A.A. Panteleev Temporal data model and possibilities of its implementation on the basis of technology OLAP In: XII National Conference on Artifical Intelligence with international participation (CAI-2010):Proceedings in 4 volumns. Vol.3. – M.:Fizmatlit, 2010, pp.345-353 (in Russian).

[Eremeev and Panteleev, 2012] A.P. Eremeev, A.A. Panteleev About possibility of implementation of the temporal query languages In: J. MPEI Bulletin, № 2, 2012, pp.155-160 (In Russian).

[Furao and Hasegawa, 2006] S. Furao, O. Hasegawa: An incremental network for on-line unsupervised classification and topology learning // Neural Networks Vol. 19 Issue 1, 2006, pp. 90-106.

[Furao et al., 2007] S. Furao, T. Ogura, O. Hasegawa An enhanced self-organizing incremental neural network for online unsupervised learning In: Neural Networks Vol. 20 Issue 8, 2007, pp. 893-903.

[Han and Narayanan, 2008] K.J. Han, S.S. Narayanan Agglomerative Hierarchical Speaker Clustering using Incremental Gaussian Mixture Cluster Modeling In: Proceedings of InterSpeech, 2008, pp. 20-23.

[Kumar and Rao, 2011] Ch.S. Kumar, P. M. Rao: Design of an Automatic Speaker Recognition System using MFCC, Vector Quantization and LBG Algorithm In. International Journal on Computer Science and Engineering (IJCSE) Vol. 3 Issue 8, 2011, pp. 2942-2954.

[Lin et al., 2003] J. Lin, E. Keogh, S. Lonardi and B. Chiu A Symbolic Representation of Time Series, with Implications for Streaming Algorithms In: Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 2003, pp. 2-11.

[Linde et al., 1980] Y. Linde, A. Buzo, R.M. Gray An Algorithm for Vector Quantizer Design In: IEEE Transactions on Communications vol.28 iss.1, 1980, pp. 84-95.

[Molau et al., 2001] S. Molau, M. Pitz, R. Schlüter, H. Ney: Computing mel-frequency cepstral coefficients on the power spectrum In IEEE International Conference on Accoustics, Speech, and Signal Processing Vol.1, 2001, pp.73-76.

[Mori and Nakagawa, 2001] K. Mori, S. Nakagawa Speaker change detection and speaker clustering using VQ distortion for broadcast news speech recognition In: Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on Vol.1 , 2001, pp. 413-416.

[Ning et al., 2006] H. Ning, M. Liu, H. Tang, Th. Huang: A Spectral Clustering Approach to Speaker Diarization In: Proceedings ICSLP, 2006.

[Perfilieva et al., 2013] L. Perfilieva, N. Yarushkina, T. Afanasieva, A. Romanov Time series Analysis using Soft Computing Methods In: International Journal of General Systems, 42(6), 2013, pp.687-705.

[Pham and Chan, 1998] D.T. Pham, A.B. Chan Control Chart Pattern Recognition using a New Type of Self Organizing Neural Networkn In: Proceedings. Instn, Mech, Engrs. Vol 212, No 1, 1998, pp. 115-127.

[Rabiner, 1989] L. Rabiner: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition In: Proceedings of the IEEE vol.77 iss.2, 1989, pp. 257-286.

[Roddick and Spiliopoulou, 1999] J.F. Roddick, M. Spiliopoulou A bibliography of temporal, spatial and spatio-temporal data mining research In: SIGKDD Explor. Newsl. 1999, jun., Vol. 1, No. 1. pp. 34–38.

[Roweis, 1998] S. Roweis: EM Algorithms for PCA and SPCA In: Advances in Neural Information Processing Systems, 1998, pp. 626-632

[Saito, 1994] N. Saito Local feature extraction and its application using a library of basesn PhD thesis, Yale University, December 1994.

[Stefano et al., 2000] C. De Stefano, C. Sansone, M. Vento To reject or not to reject: that is the question — an answer in case of neural classifiers In: IEEE Transactions on Systems, Management and Cybernetics Vol. 1, 2000. pp. 84–94.

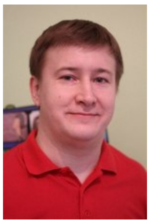
[Vagin et al., 2008] V. Vagin, E. Golovina, A. Zagoryanskaya, M. Fomina Exact and Plausible Inference in Intelligent Systems 2-nd Edition, 2008 (in Russian).

[Vyas and Kumari, 2013] G. Vyas, B. Kumari Speaker Recognition System Based on MFCC and DCT In: Internatonal Journal of Engineering and Advanced Technology(IJEAT) Vol. 2, Issue 5, 2013.

[Weiqiang et al., 2002] W.Lin, M.A. Orgun, G.J. Williams An Overview of Temporal Data Mining In: Proceedings of the 1st Australasian Data Mining Workshop, 2002.

[Zhu et al., 2005] X. Zhu, Cl. Barras, S. Meignier, J.-L. Gauvain Combining Speaker Identification and BIC for Speaker Diarization In: Interspeech'05 ISCA, 2005

Authors' Information



Antipov Sergey *– M.Sc., postgraduate student, Applied Mathematics Department of National Research University "Moscow Power Engineering Institute", 14, Krasnokazarmennaya Str., Moscow, 111250, Russia, Moscow, e-mail: antisergey@gmail.com*

Area of scientific interests*: artificial intelligence, time series processing, decision support system.*

Fomina Marina *– Associated Professor, Computer Science Department of National Research University "Moscow Power Engineering Institute", 14, Krasnokazarmennaya Str., Moscow, 111250, Russia, Moscow, e-mail: m_fomina2000@mail.ru*

Area of scientific interests*: Inductive notion formation, Knowledge discovery in Databases.*

Vagin Vadim *– Ph.D., Professor, Applied Mathematics Department of National Research University "Moscow Power Engineering Institute", 14, Krasnokazarmennaya Str., Moscow, 111250, Russia, Moscow, e-mail: vagin@appmat.ru*

Area of scientific interests*: artificial intelligence, logics, expert system.*

Eremeev Aleksandr *– Ph.D., Professor, Head of the Applied Mathematics Department of National Research University "Moscow Power Engineering Institute", 14, Krasnokazarmennaya Str., Moscow, 111250, Russia, Moscow, e-mail: eremeev@appmat.ru*

Area of scientific interests*: artificial intelligence, decision making, decision support system, expert system.*

Ganishev Vasilii *– M.Sc., postgraduate student, Computer Science and Embedded Systems, Ilmenau University of Technology, 5, Helmholtzplatz, Ilmenau, 98693, Germany, e-mail: vasily.ganishev@tu-ilmenau.de*

Major Fields of Scientific Research*: process mining, workflow management, artificial intelligence, neural networks.*