Incremental Classification of Process Data for Anomaly Detection Based on Similarity Analysis

Stefan Byttner
Intelligent Systems Laboratory
Halmstad University
SE-301 18 Halmstad, Sweden
Stefan.Byttner@hh.se

Magnus Svensson
Volvo Technology
Gotaverksgatan 10
405 08 Goteborg, Sweden
Magnus.Svensson@volvo.com

Gancho Vachkov
Reliability-based Information Systems Engineering, Kagawa University
761-0396 Kagawa, Japan
vachkov@eng.kagawa-u.ac.jp

Abstract—**Performance evaluation and anomaly detection in complex systems are time consuming tasks based on analyzing, similarity analysis and classification of many different data sets from real operations. This paper presents an original computational technology for unsupervised incremental classification of large data sets by using a specially introduced similarity analysis method. First of all the so called compressed data models are obtained from the original large data sets by a newly proposed sequential clustering algorithm. Then the data sets are compared by pairs not directly, but by using their respective compressed data models. The evaluation of the pairs is done by a special similarity analysis method that uses the so called Intelligent Sensors (Agents) and data potentials. Finally a classification decision is generated by using a predefined threshold of similarity. The applicability of the proposed computational scheme for anomaly detection, based on many available large data sets is demonstrated on an example of 18 synthetic data sets. Suggestions for further improvements of the whole computation technology and a better applicability are also discussed in the paper.**

Keywords - anomaly detection, compressed data models, sequential clustering; incremental classification; similarity analysis; intelligent sensors

I. INTRODUCTION

Performance evaluation and anomaly detection of large complex systems and machines, such as trucks, buses, cars, excavators etc. are very important engineering activities that are quite challenging from a research point of view. Here a variety of clustering and classification algorithms are most often applied, which use different concepts of similarity analysis in order to produce plausible results from comparing the large data sets.

Generally speaking, the classification and pattern recognition problems, as well as various methods and algorithms for their solution are well presented and discussed in the literature [1-7]. However in the most often cases the problem is viewed as an off-line classification of preliminary given data set (patterns). Then the typical task is to classify every single pattern from the given data set as belonging to one or another class. This is actually the case of supervised classification.

In the practical area of anomaly detection and performance evaluation the problem is stated somewhat differently, in a sense of unsupervised classification. Here we usually have many large data sets available, with each of them representing the real operation of one machine for a limited period of time (e.g. one day operation). Then it is needed to discover which of the machines runs in a *faulty mode*, i.e. to perform the anomaly detection in an incremental way, by examining all the available data sets one by one and discovering the levels of dissimilarity between them.

In this paper we present a special *two-stage* computation scheme for solving the problem of incremental classification, The *first stag*e is information compression of the “raw operation data” representing the operation of a certain machine into a respective compressed data model, which consists of a small number of model parameters. For this purpose we use in this paper an original sequential clustering algorithm, which is able to extract the clusters as a sequence in decreasing order of the cluster volumes. The *second stage* is the incremental classification decision, which is based on an originally proposed method for similarity analysis that uses the so called Intelligent Sensors.

All the computational details of the proposed incremental classification procedure, as well as an example based on synthetic data sets are given in the sequel of the paper.

II. THE PROPOSED INCREMENTAL CLASSIFICATION SCHEME

First of all we assume that there is a data acquisition procedure for gathering a large number of process data for the current operation of the machine (Bus, Truck etc.) during a limited period of time (e.g. a few hours or a full day operation). Since the obtained data set typically consists of very large number of data, it is inconvenient to keep all the original data for direct comparison with other data sets for analysis and classification purposes. Therefore we propose here a *data compression* procedure that replaces the original data set with much smaller number of *representative values* that keep approximately the main characteristics (density distribution) of the original data set. For such purpose we use here a novel *sequential clustering* algorithm, explained in details in Section III of the paper. This algorithm produces the so called *Compressed Data Model* (CDM), which is kept as representative model of the original process data set for each operation.

The next step is to compare all gathered data sets by pairs. Here this is performed not by direct comparison of the original “raw” data sets, but rather by using their respective CDMs.

This comparison is done by another original procedure for *Similarity Analysis*, explained in Section IV of the paper.

The block diagram of the proposed two-step incremental classification of process data is presented in Fig. 1.

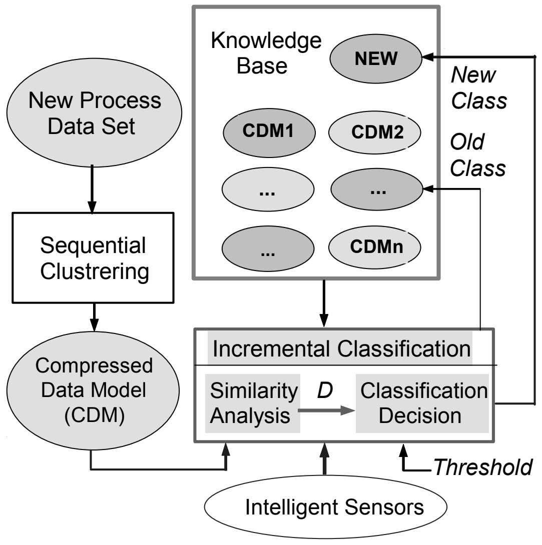


Figure 1. The proposed Incremental Classification Procedure.

The proposed procedure for *Incremental Classification* uses the CDM of each available process data set for comparison and classification. As seen from Fig. 1, it keeps the current status of all classification results in a kind of *Knowledge Base* (KB) that contains the CDMs of all new classes that have been extracted so far.

The incremental classification procedure consists of *steps*, each of them being performed when a new process data set from the current operation of the machine is available. Therefore it is not exactly a real time procedure, but rather *online* computation that is performed, when a new data set is available.

In order to start, the incremental classification procedure needs at least one CDM to be available and saved in the KB. This CDM represents one available class (one typical operation of the machine). Then during each step of the classification, a *new* process data set is submitted from the acquisition procedure and is first compressed by the sequential clustering, in order to create the respective CDM for this operation. After that the CDM is compared on similarity against all the existing CDMs in the KB, by using the *similarity analysis* procedure, explained in Section IV of the paper. Finally, a *Classification Decision* is made, according to a predetermined similarity *Threshold*, as shown in Fig. 1. According to this *binary* classification decision, the current operation (saved as current CDM) could be classified as *Existing (Old) Class* (existing operation), or as *New Class* (new operation), which forms a new item in the KB, thus enlarging its contents by one.

It is seen that the proposed classification is an *incremental* procedure, allowing the Knowledge Base to gradually grow, if a new data set with a *low enough* similarity to all currently existing CDMs in KB is discovered.

The result from the similarity analysis procedure in Fig. 1 is the so called *Dissimilarity* $D, 1 \leq D \leq 0$. It is obvious that $D \rightarrow 0$ refers to *very similar* data sets, while $D \rightarrow 1$ means that the data sets are far apart from each other (*very different* operations).

All the computational details for the incremental classification are given in Sections III and IV of the paper.

III. Sequential Clustering for Creating the Compressed Data Models

As seen from Fig. 1, the first step before the similarity analysis and classification of the process data is to reduce the large amount of the "raw data" information in the original data set. We refer to such computation step as *Data Compression*. Our idea for data compression is to use the original large data set: $\mathbf{x}_i = [x_{i1}, x_{i2}, ..., x_{iK}], i = 1, 2, ..., M$ that consist of *M* data in the *K*-dimensional input space for creating a respective *Compressed Data Model*. Then the parameters of this CDM are further on used as *representative values* of the original process data set.

Such data *compression* can be performed by using different unsupervised competitive learning algorithms, such as clustering algorithms [1,2], Neural-Gas and its modifications [8,9,10], Support Vector Machines [11] and Self-Organizing Maps [12,13] etc. All these algorithms try to find the most appropriate positions of the preliminary fixed number of *N* clusters (neurons) in the *K*-dimensional data space so that the resulting group of clusters resembles as much as possible the density distribution of the original data in the same space. The clusters can be viewed as *information granules*, which make a kind of generalization of the raw data, as discussed in [14].

The most often used clustering algorithms, such as the very popular *Fuzzy C-means* clustering [1,2] and also some other unsupervised learning algorithms use the concept of S*imultaneous Clustering*. This means that the number N_c of the clusters is predetermined and available before the calculations.

The real problem here is that this number is rarely known in advance, which leads to obtaining some *implausible solutions* that have smaller or larger number of clusters than the real ones. To alleviate this problem various criteria for *optimal clustering* have been introduced and often used, such as the *Dunn*'s Index [15], *Davies-Bouldin* Index [16], and some others. However all these criteria give a *posterior* solution of the clustering problem, in a sense that the optimal number of clusters *N** is known after performing unnecessary computation of many possible solutions.

Another problem with the simultaneous clustering methods is that the extracted clusters are not arranged in any special (increasing or decreasing) order of their characteristics (i.e. size, volume) but rather randomly, depending on the initial conditions.

Another group of clustering algorithms uses the idea of *Sequential Clustering*. Here the number of clusters is not predetermined, but the clusters are gradually extracted (one after another) in a kind of sequence, according to a given criterion, until appropriate stopping conditions are satisfied.

The clear advantages of such computation strategy for clustering are twofold. First, there are *no redundant computations* for unnecessary large number of clusters. Second, the clusters are extracted in an *ordered sequence*, starting with the most significant cluster (e.g. the cluster with the largest volume) and proceeding to the least significant (the smallest) cluster. Such process produces a clearer and more physically understandable structure of the original data set.

One of the most famous and original sequential clustering algorithm is the *Mountain Clustering* [17,18] with some of its versions and modifications [19,20]. The general concept here is to use of the so called *mountain* (or *potential*) function, in order to estimate the current areas of highest density in the data space. This function decreases gradually in a sequence with each new extracted cluster. This algorithm is easy to implement, but has some problems with the proper selection of the parameters (especially the *width*) of the new subtracted mountain function, after the extraction (and removal) of the current cluster.

The C-Fuzzy Decision Trees [21] is another approach to sequential data clustering, which uses the *depth-and-breadth* tree expansion, where each node of the tree represents the current extracted cluster. This algorithm finally produces a growth pattern of the cluster-based tree structure.

In this paper we use the idea of our previously proposed sequential clustering algorithm in [22], which needs relatively small number of tuning parameters and has proven to be robust in proper discovering of noisy clusters. As a result of the computations, the clusters are automatically arranged in decreasing order of their sizes (volumes). The computational details of this clustering algorithm are given below.

We suppose that a data set of *M* process data in the *K*-dimensional space is available. Our objective is to extract the centers (prototypes) $\mathbf{C}_i = [c_{i1}, c_{i2}, ..., c_{iK}],\ i=1,2,...,n$ of a sequence of clusters, arranged in *decreasing order* of their volumes $V_S, s=1,2,...,n$, i.e. $V_1 \geq V_2 \geq ... \geq V_n$.

The *cluster volume* V_S can be defined in different ways, but in general this is a measure of the *density* of the cluster or a measure of the *size* of the cluster in the *K*-dimensional data space. This measure is defined in the sequel of the paper.

The typical clustering algorithms are from the group of the *unsupervised learning* algorithms, but in our proposed sequential clustering algorithm we solve a direct *optimization* problem, namely maximizing the cluster volume V_S. Therefore we are dealing here with a *supervised learning* algorithm.

We define the so called *Cover Function* H_i, which is a standard *Gaussian* function with a current location of the center denoted as $\mathbf{c}$ and a fixed (predefined) *width* σ as follows:

$$H_i = \exp\left(-\frac{\|\mathbf{c}-\mathbf{x}_i\|^2}{2\sigma^2}\right),\ i=1,2,...,M \tag{1}$$

The *Cover Function* calculates the *proximity level* between the data point $\mathbf{x}_i$ and the current center $\mathbf{c}$ of the function. Here $H_i \rightarrow 0$ refers to *Low Proximity* (*far* distance between the center $\mathbf{c}$ and the data point $\mathbf{x}_i$), while $H_i \rightarrow 1$ means *High Proximity* (*short* distance between $\mathbf{c}$ and $\mathbf{x}_i$).

Then the *volume V* of the current cluster is defined by the following function, which sums the weighted proximities of all data $\mathbf{x}_i$ to the current $\mathbf{c}$ location of the cover function, as follows:

$$V = \sum_{i=1}^{M} P_i H_i = \sum_{i=1}^{M} P_i \exp\left(-\frac{\|\mathbf{c}-\mathbf{x}_i\|^2}{2\sigma^2}\right) \tag{2}$$

Here $P_i \in [0,1],\ i=1,2,...,M$ is a kind of *weight* parameter, called *Power* (*Capacity*) of the data point $\mathbf{x}_i$. The natural assumption is that at the beginning of the computation process all data have a *full power* (full capacity), as follows: $P_i = 1.0,\ i=1,2,...,M$.

Once a cluster *s* is extracted from the data set, then the power of all data points is *reduced* by the following recursive calculation:

$$P_i = P_i - P_i H_i = P_i(1-H_i),\ i=1,2,...,M \tag{3}$$

Then the problem of finding the current cluster $s,\ s=1,2,...$ becomes the *optimization* problem of *maximizing* the volume *V* of the cluster, computed by (2) at each step of the sequential clustering procedure.

A simple physical model that explains the main idea of the proposed clustering algorithm is to use a "*moving cup*" with a fixed shape (i.e. a Cover Function (1) with a fixed spread σ) in the *K*-dimensional space, and try to accumulate the heaviest *Vmax* amount of "material" (data points) under the cup. Then the location of the cup in the space represents the center of the current extracted cluster.

Here it is obvious that the type and accuracy of the optimization algorithm used for the *cup moving* would affect the accuracy of the solution *Vmax,* as well as the total computation time. However this does not change the general idea of the proposed sequential algorithm.

Here, as in our previous work [22], we also use the popular *Particle Swarm Optimization* (PSO) algorithm [23] with *Inertia Weight.* We have modified the original PSO algorithm to include constraints in the process search space. The computational details are omitted here because of space limitations. In all further simulations we have assumed the following parameters for the PSO: number of *particles Np = 12; Inertia weight* parameter, linearly decreasing from $\omega = 1.6$ to $\omega = 0.4$ and the maximal number of *Iterations: Iter = 600.* In our modification of the PSO algorithm, if the criterion *V* is *stabilized* within a small predetermined threshold, the algorithm terminates automatically, thus completing a less number of iterations and saving computation time.

At each step s, $s = 1, 2, \ldots$ of the sequential clustering, a new cluster is being extracted, when the PSO algorithm terminates. Then the *average capacity* AC of all data points, as well as the *total volume* TV of all s currently extracted clusters can be calculated, as follows:

$$AC_s = \sum_{i=1}^{M} P_i^s \Big/ M \quad \text{and} \quad TV_s = \sum_{i=1}^{s} V_i \tag{4}$$

Here P_i^s is the *remaining capacity* of the *i*-th data after the end of the *s*-th step (i.e after extracting the *s*-th cluster), calculated by (3).

When performing the proposed sequential clustering, the following trends can be easily noticed, namely: $AC_s \to 0$ and $TV_s \to M$ with $s \to \infty$. They provide us with the following idea for a meaningful *stopping criterion* of the sequential clustering process. If we need to compress the original data set into a respective CDM with a predetermined (small) *information loss* ε, we have to perform the sequential clustering in s steps (clusters), until the following inequality is hold:

$$TV_s \le M(1-\varepsilon) \tag{5}$$

In all further computations we have used the above stopping criterion with $\varepsilon = 0.03$. For the example of *18* data sets that are further on analyzed in Section V, the use of this criterion hade led to extraction of number of clusters s in the range $s \in [11, 22]$ with a predefined width of the Cover Function $\sigma = 0.07$.

Fig. 2 depicts a numerical example of a test process data set with $M = 900$ data. The above proposed sequential clustering algorithm was performed with $\varepsilon = 0.03$ and $\sigma = 0.07$, and it resulted in extracting $s = 21$ clusters, shown in the next Fig. 3.

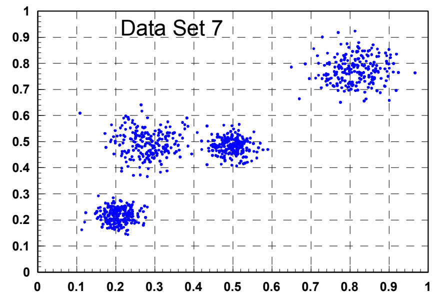


Figure 2. Example of test Process Data Set.

The large curve symbols in Fig. 3 denote the *4* extracted clusters with the largest volumes. The proper extraction of these clusters can be visually confirmed by noticing that their locations are placed correctly within the four areas with the largest density of the original data from Fig. 2.

Volumes V of the extracted clusters gradually decrease when the number s of extracted clusters increases, as shown in Fig. 4. for several different assumed widths σ of the Cover Function (2). Also, the average capacity AC of all *900* data decreases monotonously with increasing the number s of clusters, as shown in Fig. 5 for the same variations of the widths σ of the Cover Function.

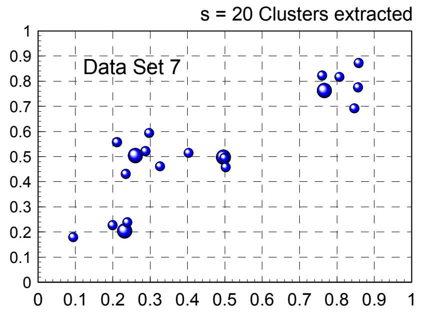


Figure 3. Results from the Sequential Clustering of the test data set in Fig. 2. The *4* large curve symbols denote the clusters with the largest volumes.

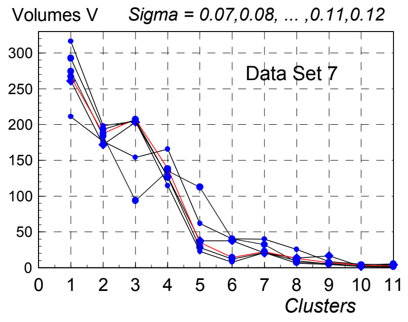


Figure 4. The cluster volumes V are almost monotonously decreasing with increasing the number of steps (clusters) s.

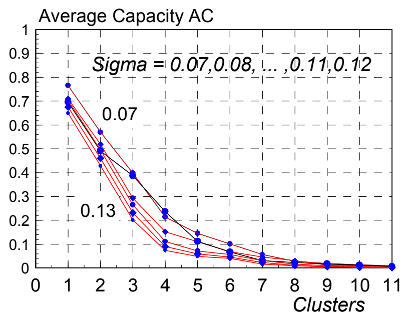


Figure 5. The average capacity of all data are monotonously decreasing with increasing the number of steps (clusters) s.

It becomes clear now that the data compression method that uses the above sequential clustering algorithm is a *lossy* compression, since it does not utilize the full amount of *100*% information from all data, but rather uses $(1-\varepsilon)$ of it, according to the stopping criterion (5). However, this information loss can be controlled by appropriate choice of the level ε. It is also obvious that smaller values of ε lead to a larger computation time.

As for the *compression rate*, it can be computed in the following way. The original (raw) data set consists of M K-dimensional data points, i.e. there are $M \times K$ data items in total. When this data set is processed (compressed) by the sequential clustering algorithm, we only need to keep the respective model parameters, namely: the number of clusters s;

the assumed width σ of the Cover Function; the volumes of all clusters $V_i, i=1,2,...,s$ and the *Cluster Coordinates:* $\mathbf{x}_{oi}, i=1,2,...,s$ that have the same dimension *K*, as the original data. This makes totally: $s\times(K+1)+2$ data to be saved for each data set, instead of the original number of $K\times M$ data items. Since *M* is normally a large number of original process data and *s* is usually a much smaller number of extracted clusters, the efficiency of the proposed compression method from a memory viewpoint is obvious.

The next step in the overall algorithm for incremental classification from Fig. 1 is to evaluate the *similarity* between all pairs of data, as discussed in the next Section of the paper.

IV. Similarity Analysis of Data Sets by Using Compressed Data Models

All process data sets that have to be compared to each other in order to find the difference between them could be considered as *data clouds* with different and unstructured (arbitrary) shape in the *K*-dimensional space, and with different data distribution. Therefore an appropriate and robust method for *plausible similarity analysis* of such data clouds with diverse shape and size in the multidimensional space is highly required.

In this paper we present an original idea for similarity analysis that uses the concept of *Intelligent Sensors* (IS). These sensors are also called *Agents* **A** in the sequel of the paper.

The idea is to locate a fixed number of *Na* agents in the *K*-dimensional space that calculate (estimate) the so called *Data Potential* DP of the whole data set at the location of the given agent. Since we suppose that the original data set is no more available, we have to use the respective CDM for calculating the DP for each agent. Then the dissimilarity degree between a given pair of data sets could be calculated as a difference in the estimated data potentials of all available *Na* agents for these two data sets. The details are given below.

Let us denote the set of all agents as $\mathbf{A}=\{\mathbf{a}_1,\mathbf{a}_2,...,\mathbf{a}_{Na}\}$, where $N_a=|\mathbf{A}|$. It is assumed that each agent $\mathbf{a}_i, i=1,2,...,N_a$ has fixed coordinates in the *K*-dimensional data space. In the most typical case, the agents could be located at *equal distances* from each other (a *grid-like* location). Such case is illustrated in the example of Fig. 6. for two data sets , named as Data set **7** and Data set **17**, with number of agents: *Na = 100* .

The *data potential* DP_i at the location of the *i*-th agent is calculated as a summation of the values of the cover functions (1) for this location, scaled by the normalized cluster volumes $RV_j,\ j=1,2,...,s$, i.e.

$$DP_i=\sum_{j=1}^{s}RV_j\exp\left(-\left\|\mathbf{c}_j-\mathbf{a}_i\right\|^2\Big/2\sigma^2\right),\ i=1,2,...,N_a \qquad (6)$$

Here $\mathbf{c}_j$ denotes the location of the *j*-th extracted cluster and $\mathbf{a}_j$ is the location of the *i*-th agent in the process space. The original cluster volumes $V_j,\ j=1,2,...,s$ have to be normalized to respective RV_j , in order to assure a *fair* comparison of compressed data sets that may contain different number of data. This normalization is performed simply as:

$$RV_j=V_j\Big/\sum_{l=1}^{s}V_l\in[0,1],\ j=1,2,...,s \qquad (7)$$

It should be noted that the width σ in (6) is the same, as used in (1) and (2) from Section III for the data compression procedure and s in (6) and (7) denotes the final number of the extracted clusters by the sequential clustering.

For clarity, the next Fig. 7 illustrates the calculation of the data potential (6) by *one agent* (one Intelligent Sensor), at its fixed location *x* in the one-dimensional space by assuming that the CDM of the original data set consist of just *s = 3* extracted clusters, denoted as C1, C2 and C3.

The *3*-dimensional plot of the DP of the original *Data Set 7* from Fig. 6a), calculated by (6) from the respective CDM is illustrated in Fig. 8. It is seen from this figure that the data potential represents in a plausible way the real *density distribution* of the original data set from Fig. 6a).

It is clear that agents with different locations in the process space will produce (estimate) different data potentials DP by using the same data set and its respective CDM. This fact is used here to compute the *dissimilarity degree Dif*(**S1,S2**) between a given pair of data sets {**S1,S2**}, by using their respective compressed data models CDM1 and CDM2. A simple way of such computation is given in (8), by taking the mean of all absolute differences between the DP, measured by all available sensors *Na*, that is:

$$Dif(\mathrm{S1,S2})=\sum_{i=1}^{Na}\left|DP_i^{S1}-DP_i^{S2}\right|\Big/N_a \qquad (8)$$

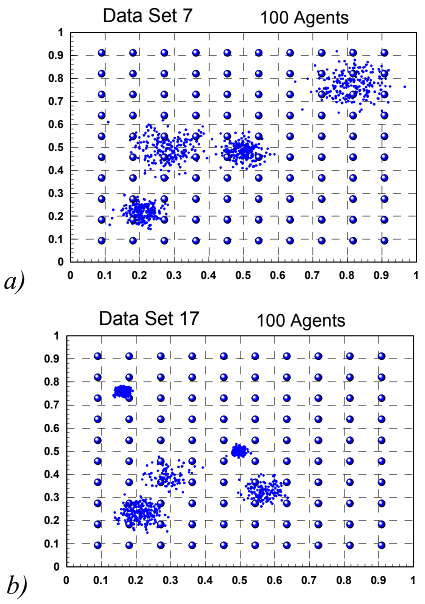


Figure 6. Example of locating *100* agents (*100* Intelligent Sensors) in the two-dimensional data space for two different data sets.

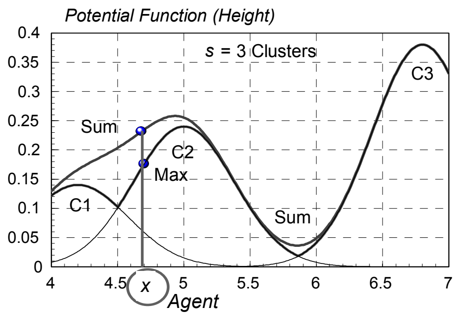


Figure 7. Example illustrating calculation of the Potential (6) for one agent, (one Intelligent Sensor) with location at *x*.

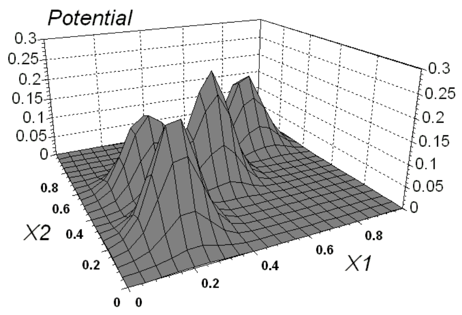


Figure 8. 3-dimensional plot of the data potential (6) for Data Set 7 from Fig. 6a), computed by using the respective compressed data model.

As example, the difference between the data sets **7** and **17**, shown in Fig. 8, is computed by (8) as: *Dif*(**7**,**17**) = *3.35*.

From the above explained idea for similarity analysis of complex data sets, it becomes clear that the number of assumed agents *Na*, as well as their locations in the data space may change the result (8) for the similarity (i.e the *dissimilarity* degree *Dif*(**S1**,**S2**)). This fact can be understood by looking at Fig. 6 and imagine that the number of agents (*Na = 100* in the figure) is *2* times bigger or *2* times smaller. Obviously a bigger number of agents would produce more plausible evaluation of the similarity between these two data sets. Finding the "optimal" number of agents is obviously a subjective problem, which solution depends on the trade-off between the desired accuracy of comparison and the affordable computation time.

It would be a good idea to convert the dissimilarity level *Dif*(**S1**,**S2**), computed by (8) into a *normalized dissimilarity* $0 \le D \le 1$. This could be achieved, for example, after some practical considerations about the "most possible or expected difference" between any pair of data sets, which could be used to determine the upper boundary of the dissimilarity level. Because this is a problem dependent case, we did not do such normalization in this paper.

V. Computation Steps of the Incremental Classification

The final and the most important step in Fig. 1 of Section II is the incremental classification decision. As mentioned before, the incremental classification consists of sequential steps, and at each of them the CDM of the *new* (unchecked) data set is compared to all the CDMs in the Knowledge Base (KB).

At each step the so called *minimal dissimilarity* $D^p_{\min}$ is calculated by (8) between the CDM of the new submitted data set *q* and the CDM of a certain data set *p* from the KB. Then, if $D^p_{\min} \le Th$ (a given *threshold*), decision is made that the data set *q belongs* to the class *p* in the KB is made. In the opposite case of $D^p_{\min} > Th$, the data set *q* is classified as a *new class,* which is saved in the KB as a *new entry* that enlarges the contents of the KB by one.

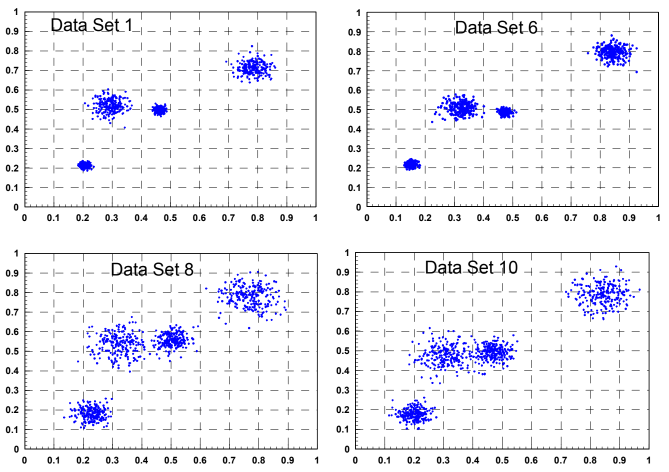
The KB is first initialized, so that to contain *at least one* known class before the incremental classification. This class represents a *known* data set that is called *Core Data Set.*

VI. Classification Results rfom Synthetic Data Sets

The above explained steps of the incremental classification from Fig. 1 are illustrated in this Section by using *18* two-dimensional artificially generated data sets, each of them with *900* normalized data. The data sets are numbered from **1** to **18** and have clearly visible *cluster structure*. For example, each of the data sets from **1** to **12** has four clusters that vary by size and location in the *2*-dimensional data space. Each of the remaining *6* data sets, numbered from **13** to **18** have 5 clusters that also vary by size and location in the space.

The presumption, when generating all these synthetic data sets was that each of them may represent a certain operation of a single machine (e.g. a Car, Bus, Truck etc.) for a limited period of time – several hours or a full day operation. Then the practical problem is to classify the data sets in an unsupervised way so that to discover whether all of them belong to the same class (normal condition), or there are some data sets (machines, vehicles) that may have deviated from the normal operation conditions and thus could be classified as *faulty* machines. Note that there could be several different types of faulty conditions that should also be classified and distinguished from the typical normal conditions.

Because of space limitations of the paper, the following Fig. 9 shows only *6* of the whole set of *18* data sets and some other data sets are shown further on as results of the classification.



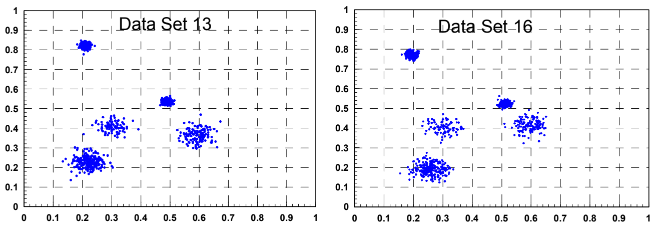


Figure 9. Example of six synthetic data sets (from the complete set of 18 data set), representing the daily operation of different machines (vehicles). They have been used for testing the proposed incremental classification.

First of all, a *pair-wise comparison* of all data sets (a total number of *(18 x 17)/2 = 153* pairs) was performed for two different number of agents, namely *Na = 81* and *Na = 100* agents, and the results are shown in Fig. 10.

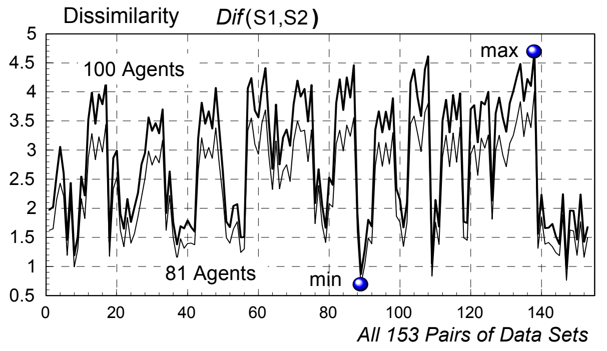


Figure 10. Pair-wise comparison of all *18* data sets, made by using two different numbers of agents, namely *Na = 81* and *Na = 100.* They indicate that the results from the relative comparison are not changed.

From this figure it becomes clear that the *relative comparison* results are not changed, i.e. the number of agents mainly affects the absolute values of the dissimilarity degrees, but not the relative results. The calculations shown in Fig. 10 have been also used for approximately defining the threshold *Th* for the incremental classification, already discussed in Section V.

The data sets pair **{7,9}** with the *minimal* dissimilarity of *D = 0.8657* and the data set pair **{12,18}** with the *maximal* dissimilarity of *D = 4.6862* are marked by two ball-type curve symbols in Fig. 10. These pairs are depicted in Fig. 11 and Fig. 12 respectively for the purpose a visual confirmation of the results.

The incremental classification of all *18* data sets has been performed *3* times, by assuming *3* different data sets, as *Core Data Sets*, namely: *Data Set* **1**, *Data Set* **5** and *Data Set* **7**. All the remaining *17* data sets are considered as *new sets* and are submitted in a sequence of *17* classification steps (one by one), as seen from the Table. An equal threshold of *Th = 3.0* has been assumed for all *3* classifications and the results are shown in Table I.

The results from the incremental classification are shown by two numbers in the column named *Class/**D*** in Table I. They denote the *decision* of the incremental classification, namely the *class*, to which the new data set from the current step belongs, and its *dissimilarity level* to the core data set of this class.

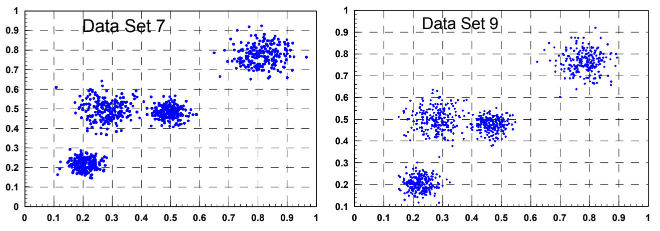


Figure 11. The most *Similar Data* Sets: **7** and **9** (*D = 0.8657*)

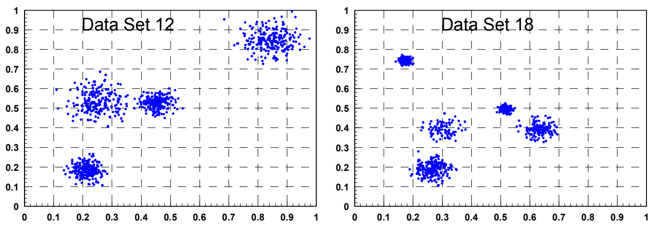


Figure 12. The most *Different Data* Sets: **12** and **18** (*D = 4.6862*)

The interpretation of the results from Table I is as follows:

- When the *Core Data Set* **1** is assumed in the beginning, *two* new classes, namely **5** and **13** are discovered. And while **5** is a "lonely class" (containing its own data set only), the new class **13** has *5* other data sets, associated to it, namely the data sets **14,15,16,17** and **18**. The core class **1** has *10* data sets, associated to it, namely the data sets **2,3,4,6,7,8,9,10,11** and **12**. Then a "soft" conclusion can be made that the classes **1, 5** and **13** are apart from each other.

TABLE I. CLASSIFICATION RESULTS FROM ALL 18 DATA SETS

Step *No.*	*Core Data Set* **1**		*Core Data Set: 5*		*Core Data Set: 7*	
	New Set	**Class/ *D***	New Set	**Class/ *D***	New Set	**Class/ *D***
1	2	**1**/*1.97*	1	**1**/0.00	1	**7**/1.45
2	3	**1**/2.02	2	**1**/1.97	2	**7**/1.65
3	4	**1**/2.57	3	**1**/2.02	3	**7**/1.38
4	5	**5**/0.00	4	**1**/2.57	4	**7**/1.76
5	6	**1**/2.59	6	**1**/2.59	5	**7**/2.67
6	7	**1**/1.45	7	**1**/1.45	6	**7**/2.08
7	8	**1**/2.42	8	**1**/2.43	8	**7**/1.92
8	9	**1**/1.19	9	**1**/1.19	9	**7**/0.87
9	10	**1**/1.51	10	**1**/1.52	10	**7**/1.28
10	11	**1**/2.54	11	**1**/2.54	11	**7**/1.80
11	12	**1**/2.22	12	**1**/2.22	12	**7**/1.68
12	13	**13**/0.00	13	**13**/0.00	13	**13**/0.00
13	14	**13**/1.55	14	**13**/1.55	14	**13**/1.55
14	15	**13**/2.23	15	**13**/2.23	15	**13**/2.23
15	16	**13**/1.66	16	**13**/1.66	16	**13**/1.66
16	17	**13**/1.66	17	**13**/1.66	17	**13**/1.66
17	18	**13**/1.73	18	**13**/1.73	18	**13**/1.73

- When the *Core Data Set* **5** is assumed at the start of the classification, all the remaining *17* data sets are classified into *two new* classes, namely the class **1** with *10* data sets (members) and class **13** (with *5* members). The core class **5** has *no members* (except itself). These results actually repeat the previous classification results, with the *Core Data Set* **1** and lead to the same conclusion.

- When the *Core Data Set* **7** is assumed at the start of the classification, the results are slightly different, namely only *one new class* **13** has been discovered (with *5* members). All the other data sets become members of the *core class* **7**. These results suggest that the class **7** stays somewhere *between* the classes **1** and **5** in the two previous classifications and incorporates a large number of similar data sets (*11* members in total). This can be regarded as a kind of compromise solution of the classification problem, under the assumed threshold of *Th = 0.3*.

The general conclusion from the above *3* incremental classifications is that *Data Set* **5** stays somewhat *apart* from all other data sets and therefore might be an indicator for a kind of anomaly (faulty condition) of the respective machine (system) under investigation.

VII. Conclusions

The proposed incremental classification scheme in this paper can be considered as a two-step computational procedure. The first step is the data compression, based on sequential clustering, which produces compact CDMs from the respective large original data sets. The second step is the actual incremental classification that is based on the specially proposed similarity analysis method, which uses a fixed number of so called Intelligent Sensors (Agents).

Our experience from many numerical examples gives us a confidence that the whole computational procedure for similarity analysis and classification is robust one and produces plausible results. It can be successfully used for classification of data sets with not only cluster structures, but also with other data structures, including linear, nonlinear and more complex structures in the data space.

There are also some important issues that have to be solved in the further research in order to improve the performance of the whole computational procedure. This includes refining the current optimization algorithm for the sequential clustering and finding a plausible way for normalizing the threshold that is used for the incremental classification.

Large number of data sets, obtained from real operations of multiple vehicles (Buses) is under consideration now for a possible real application of the proposed classification method for anomaly detection in complex systems.

References

[1] J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, New York: Plenum Press, 1981.

[2] A. Jain, "Data Clustering: 50 years beyond K-means", *Pattern Recognition Letters*, vol. 31, pp. 651-666, 2010.

[3] S. Ozawa, S. Pang and N. Kasabov, "An Incremental Principal Component Analysis for Chunk Data", Proc. of the 2006 IEEE Int. Conference on Fuzzy Systems, FUZZ-IEEE 2006, Vancouver, pp. 10493-10500, July, 2006.

[4] X. Zhou and P. Angelov, "Real-Time Joint Landmark Recognition and Classifier Generation by an Evolving Fuzzy System", Proc. of the 2006 IEEE Int. Conference on Fuzzy Systems, FUZZ-IEEE 2006, Vancouver, pp. 6314-6321, July, 2006.

[5] S. Soltic, S. Wysocki and N. Kasabov, "Evolving Spiking Neural Networks for Taste Recognition", Proc. of the 2008 *IEEE Int. Conference on Fuzzy Systems*, FUZZ-IEEE 2008, Hong Kong, pp. 2092-2098, June, 2008.

[6] S. Pang and N. Kasabov, "r-SVMT: Discovering the Knowledge of Association Rule over SVM Classification Trees", Proc. of the 2008 *IEEE Int. Conference on Fuzzy Systems*, FUZZ-IEEE 2008, Hong Kong, pp. 2487-2494, June, 2008.

[7] M. Svensson, S. Byttner and T. Rögnvaldsson, "Self-organizing maps for automatic fault detection in a vehicle cooling system", Proc. of the 4th Int. IEEE Conference on Intelligent Systems, IS 2008, pp. 24-8 – 24-12, Varna, Bulgaria, Sept., 2008.

[8] T. Martinetz, S. Berkovich and K. Schulten, "Neural-Gas Network for Vector Quantization and Its Application to Time-Series Prediction", *IEEE Trans. Neural Networks*, Vol. 4, No. 4, pp. 558-569. 1993.

[9] L. Xu, A. Krzyzak and A. Oja, "Rival Penalized Competitive Learning for Clustering Analysis, RBF Net and Curve Detection", *IEEE Trans. Neural Networks*, Vol. 4, No. 4, pp. 636-649, 1993.

[10] G. Vachkov, "Human-Assisted Fuzzy Image Similarity Analysis Based on Information Compression", *Journal of Advanced Comp. Intelligence*, JACIII, vol. 13, no. 3, pp. 255-261, 2009.

[11] C. Cortes and V. Vapnik, "Support vector network", *Machine learning,* vol. 20, pp. 273-297, 1995.

[12] T. Kohonen, *Self-Organizing Maps*, Third Edition, Springer Series in Information Sciences, Springer, Berlin, 2001.

[13] D. Alahakoon, S.K. Halgamuge and B. Srinivasan, "Dynamic Self-Organizing Maps with controlled growth for Knowledge Discovery", *IEEE Trans. Neural Networks*, vol. 11, No. 3: pp. 601-604, 2000.

[14] W. Pedrycz, *Knowledge-Based Clustering: From Data to Information Granules*, *Wiley-Intersciense*, 2005, pp. 316.

[15] J.C. Dunn, "Well Separated Clusters and Optimal Fuzzy Partitions", *Journal of Cybernetics,* vol. 4, pp. 95-104, 1974.

[16] D. Davies and D. Bouldin, "A Cluster Separation Measure", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 1, pp. 224-227, 1979.

[17] R. Yager and D. Filev, "Approximate Clustering via the Mountain Method", *IEEE Trans. on Systems, Man and Cybernetics*, vol. 24, No. 8, pp.1279-1284, 1994.

[18] A. Clark and D. Filev, "Clustering Techniques for Rule Extraction from Unstructured Text Fragments", CD-ROM Proc. of the NAFIPS'05 Conference, Ann Arbor, USA, July 2005.

[19] P. Angelov, "An Approach for Fuzzy Rule-based Adaptation using On-line Clustering", *International Journal of Approximate reasoning*", vol. 35, no. 3, pp. 275-289, 2004.

[20] K. Inoue and K. Urahama, "Sequential Fuzzy Cluster Extraction by a Graph Spectral Method", Patter Recognition Letters, vol. 20, pp. 699-705, 1999.

[21] W. Pedrycz and Z.A. Sosnowski, "C-Fuzzy Decision Trees", *IEEE Trans .SMC*, Part C, vol. 35, no. 4, pp. 498-511, 2005.

[22] G. Vachkov, "Temporal and Spatial Evolving Knowledge Base System with sequential Clustering", Proc. of the 2010 World Congress WCCI 2010 (FUZZ-IEEE 2010), Barcelona, Spain, pp. 3010-3017. July, 2010.

[23] R. Poli, J. Kennedy and T. Blackwell, Particle Swarm Intelligence. An Overview", *Swarm Intelligence*, vol. 1, pp. 33-57, 2007.