Schedae Informaticae Vol. 24 (2015): 133–142
doi: 10.4467/20838476SI.15.013.3035

# Subspace Memory Clustering

Łukasz Struski[1], Jacek Tabor[1], Przemysław Spurek[1]
Faculty of Mathematics and Computer Science
Jagiellonian University, ul. Łojasiewicza 6, 30-348 Kraków, Poland
e-mail: *lukasz.struski@im.uj.edu.pl*, *jacek.tabor@ii.uj.edu.pl*, *przemyslaw.spurek@ii.uj.edu.pl*.

**Abstract.** We present a new subspace clustering method called SuMC (Subspace Memory Clustering), which allows to efficiently divide a dataset $D \subset \mathbb{R}^N$ into $k \in \mathbb{N}$ pairwise disjoint clusters of possibly different dimensions. Since our approach is based on the memory compression, we do not need to explicitly specify dimensions of groups: in fact we only need to specify the mean number of scalars which is used to describe a data-point. In the case of one cluster our method reduces to a classical Karhunen-Loeve (PCA) transform. We test our method on some typical data from UCI repository and on data coming from real-life experiments.

**Keywords:** subspace clustering, projection clustering, PCA.

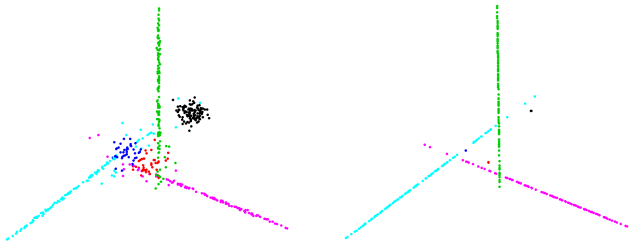
## 1. Introduction

In many data engineering problems we deal with high-dimensional data, which causes challenges to existing clustering methods, especially those based on density approach. Therefore, there is a need to simultaneously cluster the data into multiple subspaces and find low-dimensional subspaces optimally fitting each group. This problem, known as subspace clustering, or projection clustering [1], has found numerous applications in computer vision (e.g., image segmentation, motion segmentation, face clustering), image processing (e.g., image representation and compression), and systems theory (e.g., hybrid system identification). In general, subspace clustering algorithms

[1]The paper was supported by the National Centre for Research and Development under Grant no. WND-DEM-1-153/01.

can be divided into two main classes: those who construct axis parallel subspaces (this approach is especially useful for highly dimensional data), and those who are able to build arbitrary subspaces.

The two basic types of axis-parallel clustering methods are distinguished by the way they proceed. The bottom-up search method takes advantage of the downward closure property of density to reduce the searched space. Algorithms first create a histogram for each dimension, and by their analysis construct higher dimensional clusters. CLIQUE [2] was one of the first algorithms which combines density and grid based clustering. Similar approach was presented in ENCLUS [3], MAFIA [4], CLTree [5], DOC [6]. On the other hand, the top-down subspace clustering approach starts by finding an initial approximation of the clusters in the full feature space with equally weighted dimensions. Next each dimension is assigned a weight for each cluster. The updated weights are then used in the following iteration to regenerate clusters. PROCLUS [7] was the first top-down subspace clustering algorithm. Similar to CLARANS [8] and FINDIT [9], PROCLUS samples the data, then selects a set of $k$-medoids and iteratively improves the clustering. These algorithms are not able to capture local data correlations and find clusters of correlated objects since the principal axes of correlated data are arbitrarily oriented.



(a) All points of a given dataset together with its clusters.

(b) Projections onto the subspaces.

**Figure 1.** Dividing a dataset into two zero-dimensional (points) and three one-dimensional subspaces (lines).

The second type of subspace clustering algorithms concentrate on finding subspaces in arbitrary position in space. The first implementation of this approach was given by ORCLUS [10] using ideas similar to the axis-parallel approach PROCLUS. In [11] the algorithm 4C, a combination of DBSCAN [12] and PCA, is presented to find correlation clusters. COPAC [13] is based on similar ideas as 4C but deals with some problems like meaningless similarity matrices due to sparse $\varepsilon$-neighborhoods, instead taking a fixed number $k$ of neighbors. Similar idea is present in algorithms based on classical $k$-means approach [14,15]. By considering different objective functions which take into account the inherent trade-off between the dimension of a subspace and the induced clustering error we obtain clusters in arbitrary subspaces.

In this paper we present a novel projection clustering method which is based on information theory. More precisely, our goal is to divide data into subspaces of possibly various dimensions, in such a way to minimize the mean squared error, while keeping fixed the compression level (the number of used scalars). This approach allows us to efficiently divide a data into fixed $k \in \mathbb{N}$ pairwise disjoint clusters (which are described in an optimal way by subspaces) and to determinate their dimensions, see Figure 1, where the data is divided into two zero-dimensional (points) and three one-dimensional affine subspaces (lines).

The main advantage of the presented method is that it automatically finds the optimal dimensions of the clusters. However, we need to specify the number of clusters $k$ and the compression level $p$.

## 2. Theory: cost function and the clustering algorithm

In this section we describe the clustering method. Our goal is to divide dataset $D \subset \mathbb{R}^N$, $N \in \mathbb{N}$ into $k \in \mathbb{N}$ pairwise disjoint clusters $D_1, \ldots, D_k$ ($D = D_1 \cup \ldots \cup D_k$) such that each cluster is well represented by an affine subspace.

Let us consider a situation of one group $D \subset \mathbb{R}^N$ and an affine space $V \subset \mathbb{R}^N$ of dimension $n \leq N$. We replace each point $x \in D$ by its projection onto $V$. Consequently the total squared-error in cluster is given by

$$E[D; V] := \sum_{x \in D} \operatorname{dist}(x, V)^2,$$

where $\operatorname{dist}(x, V)$ describe distance between point and its orthogonal projection on subspace $V$. Observe, that we need $|D| \cdot n$ scalars to describe a point after projection, and $|D| \cdot N$ for original data. In such a case the compression level of memory (measured by the number of used scalars) which was used for representing data in lower dimensional subspaces, is defined by

$$p := \frac{|D|n}{|D|N} = \frac{n}{N} \in [0, 1],$$

where $|D|$ is the cardinality of the dataset $D$. There appears a natural question how to construct the subspace $V$ which minimizes the total squared-error with fixed level of allowed memory. The answer is given by the PCA algorithm [16, 17]. The optimal subspace is given by $V = \mathrm{m}_D + \operatorname{span}(v_1, \ldots, v_n)$, where $\mathrm{m}_D$ is a mean of $D$ and $v_1, \ldots, v_n$ are the $n$ consecutive eigenvectors of the covariance matrix $\mathrm{cov}_D$ of $D$ (ordered decreasingly with respect to eigenvalues). This allows us to define the minimal projection error

$$E[D, n] := E[D, \mathrm{m}_D + \operatorname{span}(v_1, \ldots, v_n)]. \tag{1}$$

It is possible to calculate the value of $E[D, n]$ by analyzing only the eigenvalues of covariance matrix.

**Proposition 1** [17, Propetries A1–A5], *Let $D \subset \mathbb{R}^N$ be a given dataset, where $N \in \mathbb{N}$, and let $\lambda_1, \ldots, \lambda_N$ be the decreasingly ordered eigenvalues corresponding to eigenvectors $v_1, \ldots, v_N$ of covariance matrix $\mathrm{cov}_D$. Then the squared error is given by*

$$E[D, n] = |D| \cdot \sum_{i \in \mathbb{N},\ i > n} \lambda_i. \tag{2}$$

The $n$ in the equation (2) denotes that we use $n \in \mathbb{N}$ parameters (scalars) in the compression of every element of the group $D$. However, in the search for the solution of the optimization problem it is better to allow the use of non-integer $n \in \mathbb{R}$. That is why we extend $E[D, n]$ in an affine way for arbitrary $n \in \mathbb{R}$ and $n > 0$:

$$E[D, n] := (\lfloor n \rfloor + 1 - n) E[D, \lfloor n \rfloor] + (n - \lfloor n \rfloor) E[D, \lfloor n \rfloor + 1], \tag{3}$$

where $E[D, \cdot]$ for natural numbers is defined in (1), and extended by $E[D, n] := 0$ for $n \geq N$. This allows us to consider non-integer dimensions for subspaces.

Now we are ready to formalize our main optimization problem.

**Optimization problem** *Let dataset $D \subset \mathbb{R}^N$, the number of cluster $k$ and the compression level $p \in [0, 1]$ be given. Our goal is to find the splitting of $D$ into $k$-clusters $D_1, \ldots, D_k$ and numbers[2] $n_1, \ldots, n_k$, which minimize the total squared error*

$$E[D_1, n_1; \ldots; D_k, n_k] := E[D_1, n_1] + \ldots + E[D_k, n_k],$$

*under the condition*

$$\frac{n_1 |D_1| + \ldots + n_k |D_k|}{N |D|} \leq p. \tag{4}$$

Observe that the left hand side of (4) denotes the proportion of the total number of scalars used for storage orthogonal projection of points from clusters $D_1, \ldots, D_k$ on subspaces of dimension $n_1, \ldots, n_k$ with respect to the total initial number of scalars needed to describe the dataset (in other words this can be seen as the compression level).

Let us proceed by the description of the procedure we use to find an approximate solution to our Optimization Problem. Suppose, for simplicity, that we want to divide the data $D$ into $k = 2$ clusters with fixed compression level $p$. This means that the total amount of scalars we are able to use approximately equals $M = p \cdot |D| \cdot N$.

First, our algorithm assigns each point of $D$ randomly to one of two clusters $D_1$ and $D_2$. For each of those clusters we reserve the amount of memory which is proportional to their number of elements $M_1 = p \cdot |D_1| \cdot N$ and $M_2 = p \cdot |D_2| \cdot N$.

In other words, we initially use the same amount of memory to the compression of a point in both clusters (or equivalently, that we approximate both clusters with subspaces of equal dimensions). In the next step, we shift the memory from one cluster to another, if it increases the compression level (decreases the squared error). In the last step we proceed through all elements of the dataset $D$ and verify whether

[2] They correspond to the dimensions of the subspaces.

we obtain smaller error by changing the belonging of $x$. We repeat above steps until we do not switch the position of any point from our dataset. In the case $k > 2$ we proceed by the following steps:

1. We assign each point of $D$ randomly to one of $k$ clusters $D_1, \ldots, D_k$ and calculate the amount of memory $M_i = p \cdot |D_i| \cdot N$ for $i = 1, \ldots, k$, which are related to clusters.

2. We proceed through all elements of the dataset $D$.

   (a) We create clones $\tilde{D}_1, \ldots, \tilde{D}_k$ of groups $D_1, \ldots, D_k$. We switch position[3] of element $x \in D$ from the cluster where belongs $x \in \tilde{D}_i$ to another clusters. We calculate error of $E[\tilde{D}_i, \tilde{n}_i] + E[\tilde{D}_j, \tilde{n}_j]$ for all $j \neq i$, $j = 1, \ldots, k$ and we choose minimum from these. Assume that minimum is realized for $j = t$.

   (b) We compare $E[D_i, n_i] + E[D_t, n_t]$ from this minimum and if we obtain smaller error by changing the belonging of $x$ than we switch $x \in D_i$ from $D_i$ to $D_t$.

3. We repeat Step 2 until we do not switch the position of any point from our dataset.

## 3. Experiments

In this section we present comparison of our algorithm, which we denote by SuMC (**Su**bspace **M**emory **C**lustering), with classical approach given by ORCLUS. Both of these methods detect clusters in arbitrarily-oriented subspaces and are able to find dimension of each component. Moreover, these algorithms require similar parameters: number of clusters and their joint dimension (ORCLUS) or total number of parameters (SuMC). The difference between these methods is that ORCLUS divides a dataset into given number of clusters of the same dimensions, while our method can switch the memory between clusters, which result in the automatic discovery of the optimal dimension of each cluster. It should be highlighted that our method needs parameter $p \in [0, 1]$ directly associated with the dimensions of clusters.

Let us start with synthetic datasets, which we are randomly generated on the spaces $[0, 1]^{\dim}$, where dim denotes dimension of a space. First, we create two datasets $X_1, X_2 \subset \mathbb{R}^3$, which contain two 1-dimensional (each have 100 points) and two 2–dimensional (each have 200 points) clusters. Note that we need $1 \cdot 100 \cdot 2 + 2 \cdot 200 \cdot 2 = 1000$ scalars to remember each of these datasets. In the Table 1 we show results of comparison two methods: SuMC and ORCLUS. Both algorithms give comparably good results.

Our method allows to detect the optimal dimensions of the clusters. If we take as a number of clusters 4 and $p = 0.63$ in SuMC algorithm for dataset $X_1$ then we

[3] Note that if we change the position of an element from one cluster to other cluster then we also change memories of these clusters.

**Table 1.** Comparison of SuMC and ORCLUS algorithms on synthetic datasets $X_1, X_2 \subset \mathbb{R}^3$, which have two 1-dimensional and two 2-dimensional subspaces (we divide on four clusters).

| Dataset | | $X_1$ | | $X_2$ | |
|---|---|---|---|---|---|
| Dimensions of clusters | | 1 | 2 | 1 | 2 |
| Compression level for SuMC (4) | | 0.33 | 0.66 | 0.33 | 0.66 |
| Rand index | SuMC | 0.8728381 | 1 | 0.7586923 | 1 |
| | ORCLUS | 1 | 0.7374569 | 0.9950417 | 0.8364218 |

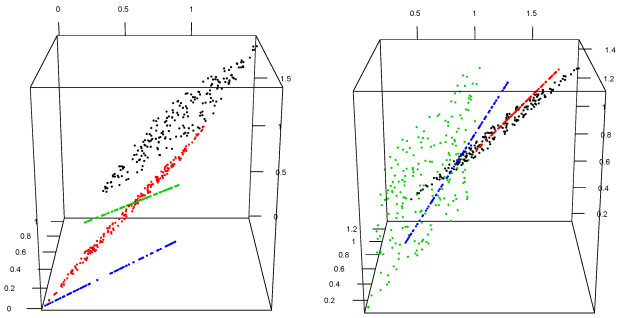


**Figure 2.** Results of algorithm SuMC for datasets $X_1$ and $X_2$ respectively.

obtain the original clusters, see Figure 2. Moreover, dimensions of these clusters equal $1, 1, 2, 2.04$. Similarly, if we put $p = 0.56$ for dataset $X_2$ then we obtain 100% compatibility with original clusters and dimensions of this clusters are given by $1, 1, 2, 2.67$, see Figure 2.

On Figure 3, we show how the mean square error varies depending on the compression level $p$ (since for each $p$ we made only few random starts in our experiments, we are not able to find the global minimum, which results in the local increases in the error level with the decrease of compression level). As one can see, the mean square error is stabilized in the range from 0.5 to 1.

We have also generated datasets $X_3 \subset \mathbb{R}^4$ and $X_4 \subset \mathbb{R}^{20}$ in the same way as datasets $X_1, X_2$. The dataset $X_3$ contains one 1-dimensional (100 points), one 2-dimensional (each have 200 points) and one 3-dimensional (300 points) subspaces. In Table 2 we present results of comparison between SuMC and ORCLUS.

**Table 2.** Comparison of SuMC and ORCLUS algorithms on synthetic dataset $X_3 \subset \mathbb{R}^4$ (we divide on three clusters).

| Dimensions of clusters | | 1 | 2 | 3 |
|---|---|---|---|---|
| Compression level for SuMC | | 0.25 | 0.50 | 0.75 |
| Rand index | SuMC | 1 | 0.6364886 | 1 |
| | ORCLUS | 0.9149972 | 0.7886867 | 0.9399221 |

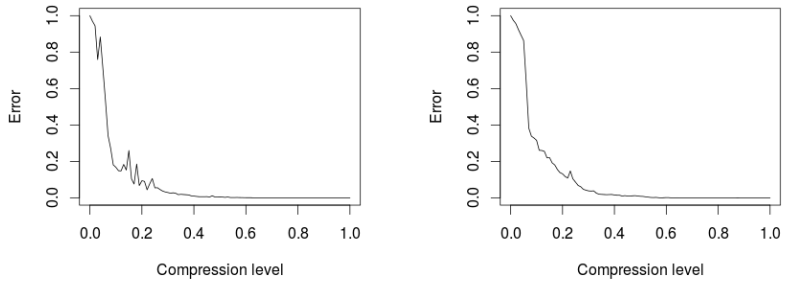



**Figure 3.** Graphs of the error function depending on the compression level.

The dataset $X_4 \subset \mathbb{R}^{20}$ contains 2-dimensional (100 points), 6-dimensional (200 points), 10-dimensional (400 points), 15-dimensional (600 points) subspaces. In the Table 3 we present results of comparison between SuMC and ORCLUS.

**Table 3.** Comparison of SuMC and ORCLUS algorithms on synthetic dataset $X_4 \subset \mathbb{R}^{20}$ (we divide on four clusters).

| Dimensions of clusters | | 2 | 6 | 10 | 15 |
|---|---|---|---|---|---|
| Compression level for SuMC | | 0.1 | 0.3 | 0.5 | 0.75 |
| Rand index | SuMC | 0.6368911 | 0.8643075 | 0.9988204 | 0.9968982 |
| | ORCLUS | 0.8447646 | 0.9616083 | 0.8680002 | 0.8123634 |

As one can see from above examples, our method compared to ORCLUS gives better results when we search for clusters near the highest dimensional subspace included in dataset. In the case when we want to find higher-dimensional subspaces than their are in fact present in the data, then our method can assign any point to any cluster because we have more memory than we in fact need.

In the next example we compare methods on datasets form the uci-repository http://archive.ics.uci.edu/ml. Let us consider three datasets from this repository: *glass*, *wine* and *yeast*, see Table 4.

From the above table we can conclude that our method gives in most cases better results than ORCLUS method.

At the end of this section we present a possible application of subspace clustering for possible preprocessing of turbine wind states. The experiment was performed using 4-D data from one of vertical axis wind turbine prototypes. The data covered the period from 18.04.2014 till 21.04.2014 and were recorded for every second by the on-line register system. The dataset contains the basic values that define the operational state of the turbine: wind speed, rotational speed of the rotor and the AC/DC power generated by the turbine. The dataset included 214306 measurements.

**Table 4.** Comparison of SuMC method with ORCLUS method on several datasets: *glass* (divide on 7 clusters), *wine* (divide on 3 clusters) and *yeast* (divide on 10 clusters) from the uci-repository ($p$ is compression level for SuMC method).

| | Rand index | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Glass | | | Wine | | | Yeast | | |
| **Dim.** | $\boldsymbol{p}$ | SuMC | ORCLUS | $\boldsymbol{p}$ | SuMC | ORCLUS | $\boldsymbol{p}$ | SuMC | ORCLUS |
| 1 | 0.11 | 0.67158 | 0.48690 | 0.07 | 0.53304 | 0.55881 | 0.12 | 0.73253 | 0.64056 |
| 2 | 0.22 | 0.69734 | 0.59094 | 0.15 | 0.62928 | 0.55678 | 0.25 | 0.72776 | 0.49860 |
| 3 | 0.33 | 0.68150 | 0.59200 | 0.23 | 0.64343 | 0.53400 | 0.37 | 0.70630 | 0.63279 |
| 4 | 0.44 | 0.68729 | 0.64126 | 0.31 | 0.66045 | 0.53660 | 0.5 | 0.68570 | 0.69806 |
| 5 | 0.55 | 0.68036 | 0.66202 | 0.38 | 0.58217 | 0.55843 | 0.62 | 0.62723 | 0.70411 |
| 6 | 0.66 | 0.63916 | 0.65210 | 0.46 | 0.55907 | 0.567767 | 0.75 | 0.63632 | 0.69944 |
| 7 | 0.77 | 0.64104 | 0.6690 | 0.54 | 0.55628 | 0.54840 | 0.87 | 0.70354 | 0.70313 |
| 8 | 0.88 | 0.52981 | 0.68957 | 0.61 | 0.56224 | 0.64235 | 1 | 0.72156 | 0.70088 |
| 9 | 1 | 0.65464 | 0.64420 | 0.69 | 0.55710 | 0.5890307 | — | — | — |
| 10 | — | — | — | 0.77 | 0.5509427 | 0.6494001 | — | — | — |
| 11 | — | — | — | 0.85 | 0.568971 | 0.6478131 | — | — | — |
| 12 | — | — | — | 0.92 | 0.5589412 | 0.6065511 | — | — | — |
| 13 | — | — | — | 1 | 0.5534819 | 0.6621596 | — | — | — |

The attempts of classification of this kind of vertical axis wind turbines data using ART-2 neural network were already taken in [18–20]. The results of classification carried out by that neural network were as good as the one done by a human expert. ART-type neural networks were also used in analysis of horizontal axis wind turbines operational states [21–24]. Since a turbine states are visualized by liner components in data cloud we use subspace clustering method for extracting them. Clusters obtained by subspaces clustering correspond with different phases of wind turbine precess. Consequently our method can be applied to preprocessing in the system which uses ART-type neural networks.

## 4. Conclusions

In this paper the projection clustering algorithm SuMC was presented. The method is based on information theory. Consequently we do not need to explicitly specify dimensions of groups, but on to define the mean number of scalars which is used to describe a data-point. This approach allows us to construct effective algorithm of subspace clustering which approximates groups by subspaces. The method changes on-line dimensions of subspaces corresponding with clusters and finds the optimal ones.

## 5. References


[1] Vidal R., *Subspace clustering.* Signal Processing Magazine, IEEE, 2011, 28(2), pp. 52–68.

[2] Agrawal R., Gehrke J., Gunopulos D., Raghavan P., *Automatic subspace clustering of high dimensional data for data mining applications.* vol. 27. ACM, 1998.

[3] Cheng C.H., Fu A.W., Zhang Y., *Entropy-based subspace clustering for mining numerical data.* In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 1999, pp. 84–93.

[4] Goil S., Nagesh H., Choudhary A., Mafia: *Efficient and scalable subspace clustering for very large data sets.* In: *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999, pp. 443–452.

[5] Liu B., Xia Y., Yu P.S., *Clustering through decision tree construction.* In: *Proceedings of the ninth international conference on Information and knowledge management.* ACM, 2000, pp. 20–29.

[6] Procopiuc C.M., Jones M., Agarwal P.K., Murali T., *A monte carlo algorithm for fast projective clustering.* In: *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, ACM, 2002, pp. 418–427.

[7] Aggarwal C.C., Wolf J.L., Yu P.S., Procopiuc C., Park J.S., *Fast algorithms for projected clustering.* In: *ACM SIGMOD Record.* vol. 28, ACM, 1999, pp. 61–72.

[8] Ng R.T., Han J., *Clarans: A method for clustering objects for spatial data mining.* Knowledge and Data Engineering, IEEE Transactions on, 2002, 14(5), pp. 1003–1016.

[9] Woo K.G., Lee J.H., Kim M.H., Lee Y.J., *Findit: a fast and intelligent subspace clustering algorithm using dimension voting.* Information and Software Technology, 2004, 46(4), pp. 255–271.

[10] Aggarwal C.C., Yu P.S., *Finding generalized projected clusters in high dimensional spaces.* vol. 29. ACM, 2000.

[11] Böhm C., Kailing K., Kröger P., Zimek A., *Computing clusters of correlation connected objects.* In: *Proceedings of the 2004 ACM SIGMOD international conference on Management of data.* ACM, 2004, pp. 455–466.

[12] Ester M., Kriegel H.P., Sander J., Xu X., *A density-based algorithm for discovering clusters in large spatial databases with noise.* In: Kdd., 1996, 96, pp. 226–231.

[13] Achtert E., Böhm C., Kriegel H.P., Kröger P., Zimek A., et al., *Robust, complete, and efficient correlation clustering.* In: SDM, SIAM, 2007, pp. 413–418.

[14] Spurek P., Śmieja M., Misztal K., *Subspaces clustering approach to lossy image compression.* In: *Computer Information Systems and Industrial Management.* Springer 2014, pp. 571–579.

[15] Spurek P., Tabor J., Misztal K., *Weighted approach to projective clustering.* In: *Computer Information Systems and Industrial Management.* Springer 2013, pp. 367–378.

[16] Bingham E., Mannila H., *Random projection in dimensionality reduction: applications to image and text data.* In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2001, pp. 245–250.

[17] Jolliffe I., *Principal component analysis.* Wiley Online Library, 2005.

[18] Barszcz T., Bielecki A., Wójcik M., *Art-type artificial neural networks applications for classification of operational states in wind turbines.* In: *Artifical Intelligence and Soft Computing.* Springer 2010, pp. 11–18.

[19] Barszcz T., Bielecki A., Wójcik M., *Vibration signals processing by cellular automata for wind turbines intelligent monitoring.* Diagnostyka, 2013, 14.

[20] Barszcz T., Bielecki A., Bielecka M., Wójcik M., Wuka M., *Vertical axis wind turbine states classification by an art-2 neural network with a stereographic projection as a signal normalization.* In: *Applied Condition Monitoring.*

[21] Barszcz T., Bielecka M., Bielecki A., Wójcik M., *Wind turbines states classification by a fuzzy-art neural network with a stereographic projection as a signal normalization.* In: *Adaptive and Natural Computing Algorithms.* Springer 2011, pp. 225–234.

[22] Barszcz T., Bielecki A., Wójcik M., Bielecka M., *Art-2 artificial neural networks applications for classification of vibration signals and operational states of wind turbines for intelligent monitoring.* In: *Advances in Condition Monitoring of Machinery in Non-Stationary Operations.* Springer 2014, pp. 679–688.

[23] Bielecka M., Barszcz T., Bielecki A., Wójcik M., *Fractal modelling of various wind characteristics for application in a cybernetic model of a wind turbine.* In: *Artificial Intelligence and Soft Computing.* Springer, 2012, pp. 531–538.

[24] Bielecki A., Barszcz T., Wójcik M., Bielecka M., *Hybrid system of art and rbf neural networks for classification of vibration signals and operational states of wind turbines.* In: *Artificial Intelligence and Soft Computing.* Springer, 2014, pp. 3–11.