The Concept of a New Neural Map for Clustering, Data Visualization and Prediction with Probability Distribution Approximation

Janusz Morajda(✉)

Krakow University of Economics, Krakow, Poland
morajdaj@uek.krakow.pl

Abstract. The paper submits a proposition of a new data analysis tool (named PDCM – ***P****robability* ***D****istribution generating and* ***C****lustering* ***M****aps*) constructed in the form of an extended neural map and dedicated to a variety of tasks, such as specific clustering, visualization, prediction (together with its possible visual analysis and justification) and generation of approximated Bayesian *a posteriori* probability density distribution for dependent variable. Basic theoretical aspects concerning the structure and training process of the proposed model have been presented. Also, research involving application of the PDCM method for real estate market data (Boston dataset) has been shown together with promising research results and conclusions.

Keywords: neural maps · neural networks · SOM · Self-Organizing Prediction Maps · data analysis · data exploration · clustering · prediction · data visualization · probability distribution · management · real estate

1 Introduction

In data analysis and data mining processes, various models can be constructed depending on the goal of data processing. Frequently considered tasks concern:

a) data clustering – aimed to explain group structure of data patterns,
b) data visualization – in order to better understanding of (multidimensional) data sets,
c) prediction of values or classes of certain (dependent) variable for new patterns,
d) justification/explanation of such predictions.

These tasks (when performed on the same dataset for a given problem) are usually executed by different and unrelated tools (dedicated to specific, separate kinds of issues), what can make interpretation difficulties for a researcher or practitioner during data analyses. For example, such diverse techniques as:

- k-means classifiers, hierarchical clustering methods, or Kohonen neural networks (Self Organizing Maps - SOM) [8] are intended exclusively to clustering problems

M. Papadaki et al. (Eds.): EMCIS 2022, LNBIP 464, pp. 17–32, 2023.
https://doi.org/10.1007/978-3-031-30694-5_2

[28]; some of them are also equipped with visualisation capabilities (i.e. dendrograms in hierarchical methods, or 2-dimensional groups presentation on a plane (map) in case of Kohonen SOMs).
- all linear and nonlinear regression tools, and all machine learning regression techniques (e.g. perceptrons, GRNN networks and others) are aimed to solve prediction problems concerning forecasting/evaluation of real values of a dependent variable,
- all pattern classifiers (like decision trees, k-nearest neighbour method, naive Bayes classifier, specific types of neural networks, etc.) are dedicated to assigning proper class (from a previously approved finite set of classes) to a new (multidimensional) pattern (vector of features).

In paper [16] Morajda and Paliwoda-Pękosz presented a concept of Self-Organizing Prediction Map (SOPM) that is a kind of a neural map based on standard Kohonen SOM network (see Sect. 2), and links (within one tool) all tasks a) b) c) d) listed above. The SOPM model is described here in Sect. 3.

The main goal of present article is introducing a concept of another neural map – an extension of SOPM, which apart from tasks a) b) c) d) generates the probability distributions for predictions of dependent variables for new patterns. This new model is named PDCM (from ***P****robability* ***D****istribution generating and* ***C****lustering* ***M****aps*). The idea of PDCM (as certain enhancement of SOPM) is presented in Sect. 4. Section 5 submits a research concerning application of proposed PDCM model in real estate data analysis.

2 Neural Maps – Literature Background

Kohonen neural network, particularly Self-Organizing Map (SOM) was proposed by T. Kohonen (see e.g. [8]) and has been accepted as a basic type of a neural map. In general SOMs are devoted to solve clustering problems (i.e. identification of groups of similar objects treated as multivariate vectors of features) with additional possibility of visualization of recognized groups in the plane (2-dimensional map). They perform the process of cluster analysis (patterns grouping) with the mapping of groups existing in a multidimensional feature space onto a two-dimensional map (rectangular structure of neurons in a plane), with maintaining topology of group distribution. Signals delivered by neurons placed in the rectangular map can then be analysed numerically and can be used for construction of graphs that visualize clusters arrangement.

Many publications report usefulness of these tools in various domains, e.g. in genetics (gene data clustering [20]); chemistry (antioxidants classification in biodiesel [7]); computer systems security (network intrusion detection [12]) and others.

A great many research works show usefulness of SOMs in management (e.g. in decision-support processes) and in business/economic data analysis. Such applications may particularly concern: business failure prediction [27], city infrastructure management [10], waste management [21], company performance analysis and clustering of companies [3], technological processes designing [25], analysis of social media with utilization in tourism businesses [9], generating of transaction strategies in financial markets [15], identification of bank risk profiles and failure prediction [24], detection of

tax evasion [1], protected area management [5], water resources management [2, 22], corporate behaviours analysis [6], and others.

Numerous modifications of original SOM networks have been proposed in literature, let us show here only a small sample of various published postulates: a structure composed of many hierarchically (layered) SOM maps, named HSOM, was proposed in [19], the SOM model, in which the coordinates of neurons on the map are not constant, but are subject to dynamic changes during the learning process, was proposed in [14]; multilayer, hierarchical neural architectures based on Kohonen networks implementing clustering, used to recognize certain types of images were proposed in [11]. A good review of various variants of SOM has been presented by Moshou in monography [18].

3 Self-Organizing Prediction Maps (SOPM)

Original SOM model and its derivative tools, dedicated to clustering and its visualization, are usually not applicable in prediction problems. In turn, prediction neural networks are not equipped with explanation and/or visualization capabilities. A certain solution to these problems is the concept of Self-Organizing Prediction Map (SOPM) – as a modification of SOMs – proposed by Morajda and Paliwoda-Pękosz in paper [16] (certain modifications of SOPM, called FLOPM, has also been submitted by the same authors in [17]). SOPMs enable:

- clustering of available (used for model training) patterns (features vectors) with special respect to a selected feature of special meaning (denoted here by χ)
- visualization of clustering results in the special map,
- making predictions of the special feature χ for new patterns, together with numerical and visual analysis of these predictions.

The basic assumption for SOPM is that a selected variable χ (one of features x_i describing each pattern included into a dataset undergoing analysis) has a special (key) meaning in a data mining process or is accepted as a dependent variable in prediction task. In clustering of patterns from a given dataset, the research inquiry can involve recognition and visualization of clusters' arrangement with separate, particular consideration of variable χ. Consequently in SOPM models, the idea of modification (in relation to SOM) of the projection between the multidimensional feature space (the analysed patterns are positioned in) and set of neurons placed in a rectangular XY map, relies on following rules (see Fig. 1):

a) the key variable χ is projected only on the coordinate Y,
b) other variables (features) x_j ($j = 1, 2, \ldots, n; x_j \neq \chi$) are mapped only on the coordinate X.

The projection is realized by a special training algorithm using analysed dataset.

If the variable χ is qualitative and expressed on the ordinal scale with finite set of ordered values, then the projection χ onto Y according to the rule a) is simple: subsequent rows in the SOPM map represent subsequent ordered values of χ (number of rows is equal to number of χ values).

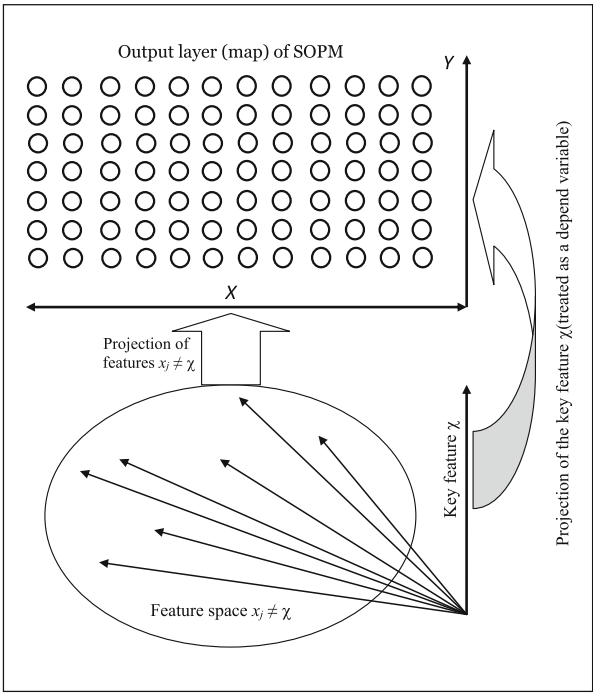

Fig. 1. Scheme of mapping of a multidimensional feature space onto the rectangular layer of neurons in SOPM model (small circles represent positions of neurons in the map). *Source:* [16]

Let us now assume that variable χ is continuous, i.e. takes real values from certain range $D \subset \Re$. Let N denotes the number of rows of neurons in SOPM. The projection χ onto Y is then executed as follows:

- all values of χ from training dataset are sorted into an ascending sequence, which then is cut into N equally numerous subsequences,
- each subsequence determines certain range R_i ($i = 1, 2,\ldots, N$) of χ values, so that for any i each value from R_i is not greater than any value from R_{i+1}, and $R_1 \cup R_2 \cup \ldots \cup R_N = D$,
- ranges R_i ($i = 1, 2,\ldots, N$) are assigned to subsequent rows of SOPM, in turn these rows have numerical coordinates y_i ($i = 1, 2,\ldots, N$) on Y axis, where y_i is the centre of R_i,
- consequently each value of variable χ can be assigned to a certain row (range R_i) and finally projected to respective value y_i.

The proposed training algorithm of SOPM is a simple modification of well-known SOM training procedure (see e.g. [8]), adjusted to the above mentioned concept of SOPM data mapping as follows:

Step 1. Consider a set $\{(\mathbf{x}_p, \chi_p), p = 1, 2, \ldots, J\}$ as a training dataset (representing analysed phenomenon), where $\mathbf{x}_p$ is a vector of features (variables) $\neq \chi_p$
Step 2. $p \leftarrow 1$
Step 3. Deliver $\mathbf{x}_p$ to the SOPM input; find the row of neurons corresponding to χ_p
Step 4. In the selected row find a neuron generating the lowest signal (as a distance between $\mathbf{x}_i$ and neuron's weight vector $\mathbf{w}$) and approve it as a *winning* neuron n_p
Step 5. In the SOPM output map determine the neighbourhood for (around) n_p
Step 6. Adjust weights $\mathbf{w}$ (by adding $\Delta\mathbf{w}$ to $\mathbf{w}$) for all neurons from this neighbourhood according to the rule:

$$\Delta\mathbf{w} = \eta \cdot s(n_m) \cdot (\mathbf{x}_p - \mathbf{w}) \tag{1}$$

where η is a learning coefficient ($0 < \eta < 1$), and $s(n_m)$, where $0 < s(n_m) \leq 1$ and $s(n_p) = 1$, is the value of neighbourhood function for a being trained neuron n_m belonging to the determined neighbourhood of n_p
Step 7. $p \leftarrow p + 1$; if $p \leq J$ go to **step 3**, otherwise go to **step 8**
Step 8. If the end-of-training condition is not fulfilled go to **step 2**, otherwise **stop**.

It should be noted that main and crucial difference between SOPM training algorithm and the classical training procedure for Kohonen's SOM relies on constraint of selection the winning neuron (and then its neighbourhood) from strictly selected row of map neurons, i.e. the row corresponding to the current value of χ_p. It should also be noted that a classic SOM training algorithm is fully unsupervised, but in SOPM this procedure is mixed: supervised as regards the key variable χ, and unsupervised with respect to all other features (see [16] for details).

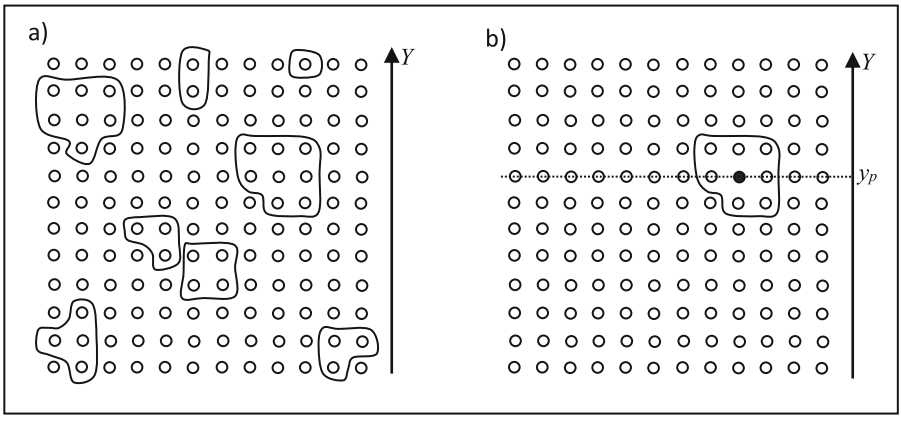

Fig. 2. Interpretation of SOPM maps (small circles represent particular neurons distributed in a rectangular map). a) Hypothetical effect of clustering process (indicated groups of neurons represent multidimensional clusters of patterns). b) Utilisation of a trained model in prediction of χ variable for a new pattern (explanation in text).

After completing the training algorithm, each pattern from the analysed dataset is assigned to a certain neuron in the output map of SOPM – it is the finally winning

neuron (see Step 4) for the given pattern. Consequently, particular neurons "collect" assigned patterns, then groups of such neighbouring neurons in the map, having large "collections" of patterns, represent corresponding clusters of objects placed in multi-dimensional feature space (see hypothetical example in Fig. 2a). However, in case of SOPMs, such a clustering is executed also with respect to special variable χ projected onto a separate (vertical) axis in the map of neurons.

Apart from such special clustering and its visualization, a trained SOPM can also be utilised for prediction of χ variable for new patterns (see example in Fig. 2b)). After delivering input vector $\mathbf{x}$ of a new pattern to the model input, a winning neuron (i.e. generating the lowest signal) out of the whole map is being found (black circle in Fig. 2b)). The coordinate y_p of the row it belongs to (dotted line in Fig. 2b)) is a predicted value of χ for this pattern. Moreover, if a winning neuron belongs to a certain previously identified group, the considered pattern can be assigned to the corresponding cluster of objects. It delivers additional information that better explains the prediction result. Also, further fine-tuned prediction process applied only to the identified cluster (with use of other methods) is possible.

Additionally, numerical (or graphical) analysis of signals from a neighbourhood of winning neuron can (informally) show uncertainty of the prediction: if the winning neuron is distinctly identified then the uncertainty is lower, however if there are many neighbouring neurons (belonging to many rows) generating similar signals – the uncertainty of the prediction is higher.

4 Proposition of PDCM Neural Map as an Extended Version of SOPM, Enabling Probabilistic Prediction

This section presents the concept of modification (expansion) of the SOPM method, named PDCM, which (preserving all SOPM capabilities) allows additionally generation of *a posteriori* probability distribution (in the Bayesian sense) for the value of the predicted variable χ.

4.1 Approximation of the Probability Distribution by Machine Learning Models

Let us consider any machine learning model designed to solve the classification problem and trained:

- by minimisation of SSE (*sum of squared errors*):

$$SSE = \frac{1}{2} \sum_{p} \sum_{i} (\theta_i^{(p)} - y_i^{(p)})^2 \tag{2}$$

where: $y_i^{(p)}$ – signal of i-th output neuron for p-th training pattern,
$\theta_i^{(p)}$ – desired training value of i-th output neuron (related to i-th class) for p-th learning pattern.

- using training output (desirable) values as 1 an 0 as follows:

$$\theta_i^{(p)} = \begin{cases} 1 \text{ if } i \text{ indicates correct class for } p\text{th pattern} \\ 0 \text{ otherwise} \end{cases} \tag{3}$$

Ruck et al. in [26] showed that a multilayer perceptron (or other supervised machine learning model - there are no formal limitations to its structure) designed to solve the classification problem and trained according to postulates (2) and (3), approximates the Bayesian optimal discriminant function. Moreover it was shown that the output signals y_i of such a model approximate (in the sense of minimisation of SSE) the Bayesian *a posteriori* probabilities of belonging of the input vector $\mathbf{x}$ to particular classes, i.e.:

$$y_i(\mathbf{x}) \approx P(\omega_i|\mathbf{x}) \quad \text{for outputs (classes)} \quad i = 1, 2, \ldots, N \tag{4}$$

where ω_i denotes i-th class.

If a predicted output variable χ is continuous, i.e. takes real values, a given class ω_i is related to a specific range R_i of the variable χ (see Sect. 3). Then the set of all outputs y_i ($i = 1, 2,\ldots, N$) can be used to approximate the entire conditional Bayesian probability density distribution (*a posteriori*) for the predicted variable χ (under the condition that the vector $\mathbf{x}$ has appeared). However, in order to approximate the probability distribution for χ, it is necessary to scale the y_i signals linearly. The scaling factor depends on the length of the range D (D is the set of all χ values, $D = R_1 \cup R_2 \cup \ldots \cup R_N$) and on the value N; this scaling should ensure that the area under the probability distribution graph (i.e. total probability) is equal to 1. This condition can be written as:

$$\lim_{N\to\infty} \sum_{i=1}^{N} (\lambda \cdot y_i) \frac{\chi_{\max} - \chi_{\min}}{N} = 1, \tag{5}$$

where: λ – scaling factor (multiplier of y_i signals),
$\chi_{\min}$, $\chi_{\max}$ – limits of range D.

As $\sum_{i=1}^{N} P(\omega_i|\mathbf{x}) = 1$, and consequently $\sum_{i=1}^{N} y_i = 1$ (with exact approximation (4)), condition (5) becomes fulfilled (independently from value N) for scaling factor:

$$\lambda = \frac{N}{\chi_{\max} - \chi_{\min}}. \tag{6}$$

It should be noted that despite theoretical considerations, in machine learning practice many elements can influence accuracy of approximation (4). For example model architecture, training parameters, selection of patterns in training set or (here) accepted number N of ranges R_i may have significant impact on this accuracy.

4.2 Main Assumptions of the PDCM Model as a Modified SOPM Network

In order to adopt the SOPM model to additional task of creating probability distribution for undergoing prediction variable χ, the structure of the model should be expanded with a new layer of nodes that generate output signals y_i ($i = 1, 2,\ldots, N$) approximating

probabilities according to (4), see Fig. 3. During the training stage, desirable signals for these nodes are 0 or 1 according to (3). Each node aggregates signals from map neurons belonging to a certain map row representing given range R_i of the variable χ. There are no weights assigned to connections between middle layer and output layer (such connections are shown – for clarity reasons – only for the first and last rows in Fig. 3).

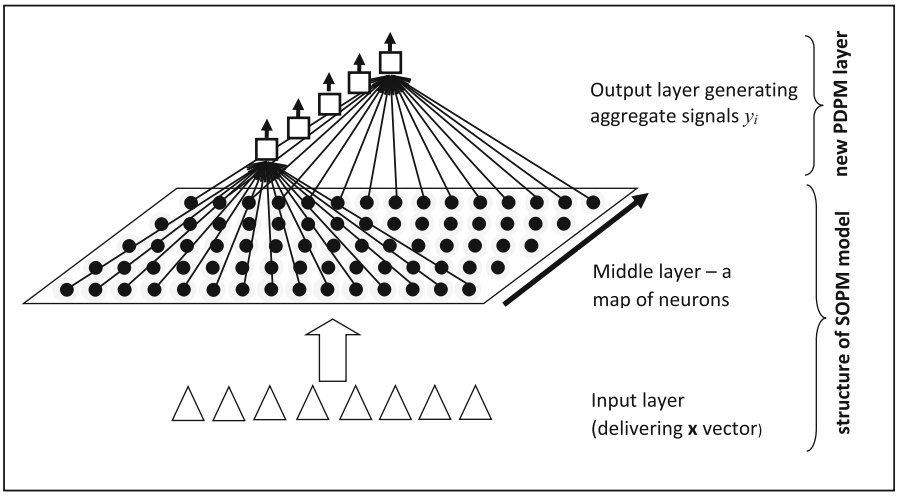

Fig. 3. Postulated structure of PDCM network.

Firstly, let us consider signals generating by neurons of middle layer (map). So far, in SOPM, this signals represented (for a given input vector $\mathbf{x}$) values of $d = \|\mathbf{x} - \mathbf{w}\|$, i.e. distance between $\mathbf{x}$ and neuron weight vector $\mathbf{w}$. Now let us introduce an exponential activation function for these neurons, so that they generate signals:

$$y_{ik} = \mathrm{e}^{-d} = \mathrm{e}^{-\|\mathbf{x}-\mathbf{w}\|} \tag{7}$$

where i, k are coordinates of the neuron in the map.

In PDCM model, signals (7) from all neurons belonging to a given i-th map row (corresponding to a range R_i, $i = 1, 2,\ldots, N$) are aggregated by an output node (see Fig. 3) to an output signal y_i according to the postulated formula:

$$y_i = \sqrt[K]{y_{i1} \cdot y_{i2} \cdot \ldots \cdot y_{iK}} = (y_{i1} \cdot y_{i2} \cdot \ldots \cdot y_{iK})^{\frac{1}{K}} \tag{8}$$

where K is the number of neurons in the row (horizontal size of the map).

4.3 Proposed Training Procedure of PDCM

As in the SOPM model, the training of PDCM network is dual: supervised due to the key variable χ (which is dependent variable in estimation and forecasting problems), and unsupervised due to other variables. This approach enables (as in SOPMs) the

implementation of the clustering process (cluster analysis) in the feature map along with its visualization, and solving the problem of estimating (predicting) the variable χ. However, while in the case of SOPM, the supervision relies only on indicating the appropriate row of the map (corresponding to the value of χ for a considered pattern $\mathbf{x}$), in the PDCM model, similarly to classic supervised neural networks, desired output signals θ (according to rule (3)) must be given for all nodes of output layer.

Let us note that due to specific character of connections between middle layer and output layer of PDCM (Fig. 3), and assuming (at first) no relations between rows (no neighbourhood at all) in the map, it is possible to consider separately N parts of the whole model (each containing one – i-th – row of the map and one corresponding output node); let us name such i-th part of the model by PDCMi ($i = 1, 2,\ldots, N$), (the neighbourhood aspect during the training stage of whole PDCM model will be considered later in Subsect. 4.4).

Below, a single step of PDCMi training for p-th training pattern $\mathbf{x}_p$ is presented (indexes i and p are then omitted due to better clarity). The minimized error function (2), denoted here by E, is now given by formula:

$$E(\mathbf{W}) = \frac{1}{2}(\theta - y)^2 = \frac{1}{2}\delta^2 \tag{9}$$

where: $\delta = \theta - y$ denotes output error value for a given (p-th) pattern

$\mathbf{W}$ denotes the vector of all weights of PDCMi.

The training procedure (based on classic backpropagation algorithm) aim to minimize function (9) using desired output signals θ determined according to rule (3).

Let $\mathbf{w}_k$ ($k = 1, 2, \ldots, K$) denotes weight vector of k-th neuron of PDCMi – on the whole PDCM map it is the neuron having coordinates (i, k). A formula describing one step of adjusting weights $\mathbf{w}_k$ (by a correction vector $\mathbf{\Delta w}_k$), based on steepest gradient method (used also in classic backpropagation algorithm), is given by equation:

$$\Delta\mathbf{w}_k = \eta \cdot (-\nabla E(\mathbf{w}_k)) \quad (k = 1, 2, \ldots, K) \tag{10}$$

where: $\nabla E(\mathbf{w}_k)$ is a part (relating to $\mathbf{w}_k$) of gradient of error function (2) in point $\mathbf{w}_k$,

η is a value of training coefficient, $0 < \eta < 1$.

Now the problem of specifying the training algorithm for the PDCM network (as a modification of the error backpropagation algorithm in the version with independent weight correction for each training pattern) relies on finding the vectors $\nabla E(\mathbf{w}_k)$. As:

$$\nabla E(\mathbf{w}_k) = \frac{\partial E(\mathbf{W})}{\partial \mathbf{w}_k} = \frac{\partial E(\mathbf{W})}{\partial y} \cdot \frac{\partial y}{\partial y_k} \cdot \frac{\partial y_k}{\partial \mathbf{w}_k} \tag{11}$$

then, considering dependencies (7), (8) and (9) we obtain (note that for clarity reasons the index i has been omitted everywhere, particularly for y, y_k and $\mathbf{w}_k$):

$$\nabla E(w_k) = -\delta \frac{1}{K}(y_1 \cdot y_2 \cdot \ldots \cdot y_K)^{\frac{1}{K}-1} \cdot \frac{y_1 \cdot y_2 \cdot \ldots \cdot y_K}{y_k} \cdot e^{-d} \cdot \left(-\frac{\partial d}{\partial \mathbf{w}_k}\right) \tag{12}$$

and, after simplification, taking again into consideration (7) and (8):

$$\nabla E(\mathbf{w}_k) = \frac{1}{K}\delta \cdot y \cdot \left(\frac{\partial d}{\partial \mathbf{w}_k}\right). \tag{13}$$

Assuming the Euclidean metric to determine the distance in the weights space, the distance $d = \|\mathbf{x} - \mathbf{w}_k\|$ is expressed by:

$$d = \sqrt{\sum_j (x_j - w_{jk})^2} \tag{14}$$

where j is the index for all subsequent elements of vectors $\mathbf{x}$ and $\mathbf{w}_k$.

Now, assuming that $d \neq 0$, we obtain

$$\frac{\partial d}{\partial w_{jk}} = \frac{1}{2d} 2 \cdot (x_j - w_{jk}) \cdot (-1) \tag{15}$$

and then, after applying it in (13), the gradient is determined as

$$\nabla E(w_k) = -\frac{1}{K} \delta \cdot y \cdot \frac{\mathbf{x} - \mathbf{w}_k}{\|\mathbf{x} - \mathbf{w}_k\|} \tag{16}$$

Finally, considering (10) and (16), the one-step weight correction vector $\mathbf{\Delta w}_k$, is determined as:

$$\Delta \mathbf{w}_k = \frac{1}{K} \eta \cdot \delta \cdot y \cdot \frac{\mathbf{x} - \mathbf{w}_k}{\|\mathbf{x} - \mathbf{w}_k\|} \qquad (k = 1, 2, \ldots, K). \tag{17}$$

It should be noted that the last factor in Eq. (17) represents a unit-length vector directed from the point $\mathbf{w}_k$ towards the point $\mathbf{x}$. The direction determined in this way is the direction of the entire weight correction vector $\mathbf{\Delta w}_k$ (anyway its orientation and length may vary and depend on other factors in (17)).

Such training steps are repeated for all training patterns from a considered dataset.

4.4 Generalized Training Procedure Taking into Account Neighbourhood Aspects

Following the methodology of training SOM and SOPM networks, let us now consider – for the PDCM network – the possibility of introducing the idea of neighbourhood and the related principle of similar method of training for topologically adjacent map neurons in middle layer (mapping adjacent areas of the feature space).

For a single PDCMi network ($i = 1, 2, \ldots, N$) the neighbourhood concept involves the requirement to differentiate the lengths of weight correction vectors $\Delta \mathbf{w}_k$ for $k = 1, 2, \ldots, K$ (both when the considered PDCMi contains a winning neuron and when it does not). Then the training rule (17) should be modified as follows:

$$\Delta \mathbf{w}_k = a_k \frac{1}{K} \eta \cdot \delta \cdot y \cdot \frac{\mathbf{x} - \mathbf{w}_k}{\|\mathbf{x} - \mathbf{w}_k\|} \quad (k = 1, 2, \ldots, K), \tag{18}$$

where a_k are neighbourhood coefficients (values of a neighbourhood function s) responsible for differentiating lengths of vectors $\Delta \mathbf{w}_k$.

Certain theoretical analyses executed by author has led to the conclusion that for a single PDCM*i* network the dependency:

$$\sum_{k=1}^{K} a_k = K \tag{19}$$

should be ensured.

Considering now the neighbourhood aspect for the whole PDCM during the training stage (i.e. the essential aspect for ensuring a proper organization of the map – PDCM middle layer – in order to perform clustering process), there is a need for introducing a certain neighbourhood function *s*, like in models SOM and SOPM. Here, the function *s* should be responsible for determining the neighbourhood coefficients a_k for all neurons in the map, during a given training step. The „centre" of function *s* is always the winning neuron, selected separately for each training pattern exactly according to rules accepted in SOPM (see Sect. 3). However in PDCM, the neighbourhood function *s* should additionally take into account the rule (3) and Eq. (19). Optimal selection of function *s* is the matter of on-going experiments, current results of such exploratory analyses were implemented by author in researches shown in next section.

The definition of learning rule (formula (18)) for PDCM network, supplemented by approving a method of determining the neighbourhood coefficients ensuring the implementation of the pattern grouping process, allows creation the network training algorithm. The algorithm has been implemented in form a computer program written by the author in C++, which is the basis for the research discussed in the next section.

5 Application of the PDCM Model in Real Estate Market Analysis

Below, results of application of the PDCM model in the issue of real estate value estimation are presented. A Boston housing dataset, available in the UCI ML Repository (https://archive.ics.uci.edu/ml/machine-learning-databases/housing/), has been used in presented research. This set has been often exploited in many researches concerning clustering or regression problems (see e.g. [4, 13, 23]).

The Boston dataset contains 506 patterns described by 14 numeric variables. The dependent feature χ represents the median of the estate prices in the given census area (MEDV). The remaining 13 variables describe certain features influencing real estate values and, after standardisation, constitute the input vector to PDCM. Six observations out of 506 (their numbers in the original Boston dataset are: 48, 137, 197, 314, 435, 457) were selected randomly for ultimate testing (creating the test set); the remaining 500 patterns create the training set.

The following PDCM model parameters were adopted:

- number of training epochs (presentations of whole training set): 300,
- PDCM map dimensions: *X* axis – 10 ($K = 10$), *Y* axis – 20 ($N = 20$),
- each range $R_1, R_2, \ldots, R_N$ contains 25 values of χ, taken from the training set,
- the rule (18) has been adopted for model training,
- training coefficient η has decreased linearly during the training from 0.7 to 0.07

- initial weights for map neurons were selected randomly from range [−1.5, 1.5].

The effect of clustering, expressed by the numbers of training patterns **x** assigned to particular neurons of the map (middle layer) of the PDCM network, is presented in tabular and graphical form in Fig. 4 (note also a relation to Fig. 2a)).

Real estate value y_i (for particular map rows)	Columns of PDCM map									
46,8	3	0	0	7	0	3	2	3	5	2
39,2	2	5	6	0	0	1	4	0	5	2
32,9	1	2	0	1	6	1	2	1	1	**10**
29,7	0	0	3	0	2	0	8	11	0	1
26,7	6	1	2	7	0	1	2	0	5	1
24,6	1	2	0	1	0	0	11	3	2	5
23,8	1	5	2	3	1	1	7	4	1	0
23,1	2	0	1	0	4	0	5	8	0	5
22,4	1	3	5	8	1	0	5	0	2	0
21,6	9	0	0	0	6	1	0	0	0	9
20,8	7	5	3	1	2	0	1	0	4	2
20,1	1	0	1	0	6	0	3	4	0	10
19,5	12	1	4	0	0	0	2	1	3	2
18,7	4	0	2	5	7	1	1	1	4	0
17,7	1	3	0	8	6	0	2	3	0	2
16,2	7	2	5	0	2	0	4	0	0	5
14,7	11	0	0	2	3	0	1	0	6	2
13,6	2	2	0	5	3	2	2	0	9	0
11,6	1	3	0	3	4	0	8	4	1	1
7,6	0	1	0	0	10	0	0	3	10	1

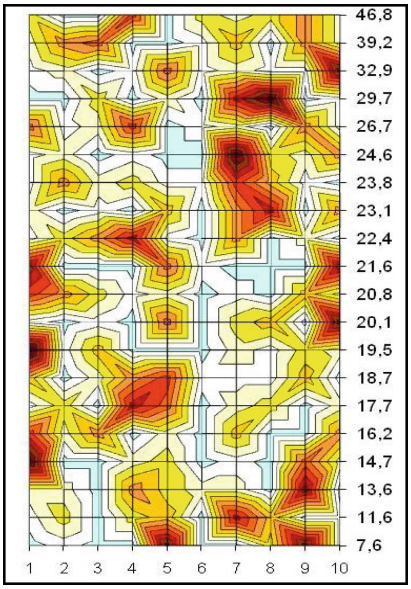

Fig. 4. Clustering results of the Boston real estate dataset

Analysis of clustering results (Fig. 4) shows the tendency to create clusters of training patterns assigned to the topologically adjacent PDCM map neurons. This method also allows (like SOPM) the identification of clusters with respect to values of a particular feature (variable χ – here median of the estate prices MEDV).

The PDCM method also (like SOPM) allows, for a given new pattern, determining a prediction value of variable χ (along with a visual or numerical informal assessment of prediction uncertainty), based on identification of winning neuron (and signal analysis of adjacent neurons). However in PDCM (contrary to SOPM) it is additionally possible to generate (approximate) formal *a posteriori* probability density distribution for the predicted (estimated) variable χ – this property is a key functional feature of this model. The results of testing the PDCM network in the real estate valuation process for six test cases are presented below.

Figure 5 shows graphically output signals generated according to formula (7) by neurons of the PDCM map (middle layer) in response to selected (exemplary) test patterns 1 and 3. The darker area in the graph, the stronger neuron's signal. The winning neuron (generating the highest signal – see formula (7), and indicating the point prediction of χ for a given pattern) is placed in the black area. For the test pattern 1 the prediction value is 18.7, for the test pattern 3 the prediction is 32.9.

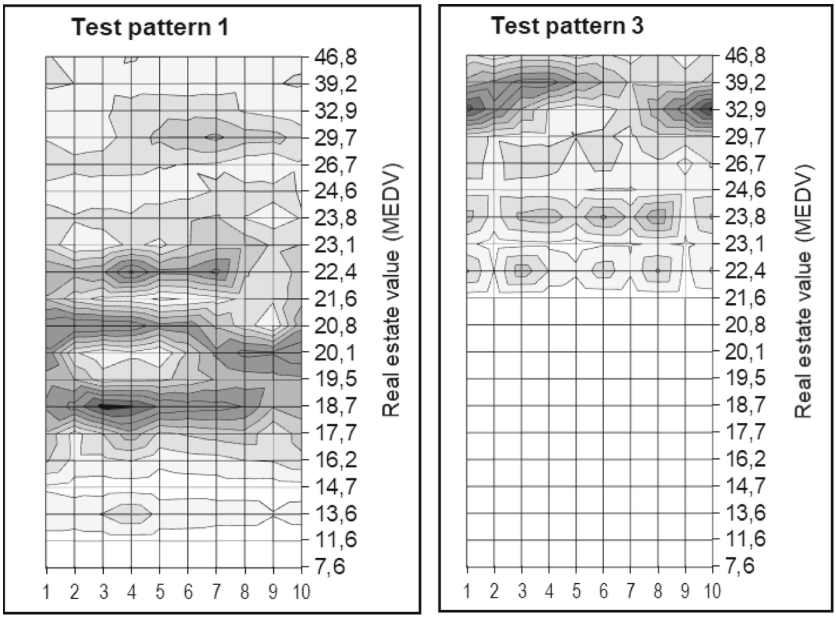

Fig. 5. Output signals of map neurons for two selected test pattern 1 and 3

The testing effects (based on point predictions generated by winning – i.e. generating the strongest signal – neuron) for all six testing patterns are presented in Table 1. The mean absolute error in estimating the MEDV value for the test set is **1.55**, what (for this specific dataset) should be appointed as a good result.

Table 1. Testing results (based on point predictions) for all six testing patterns

Test pattern	Actual values		Predicted values		Absolute error	
	MEDV	Range R_i	MEDV	Range R_i	MEDV	Range R_i
1	16.6	5	18.7	7	2.1	2
2	17.4	6	17.7	6	0.3	0
3	33.3	18	32.9	18	0.4	0
4	21.6	11	23.1	13	1.5	2
5	11.7	2	14.7	4	3.0	2
6	12.7	2	14.7	4	2.0	2

Figure 6 show the graphs of estimated probability density distributions for MEDV for six test patterns. These distributions have been approximated on the basis of signals generated by output nodes (output layer) of the PDCM model, according to formula (4). Black triangular mark on the horizontal axis shows the actual value of the MEDV.

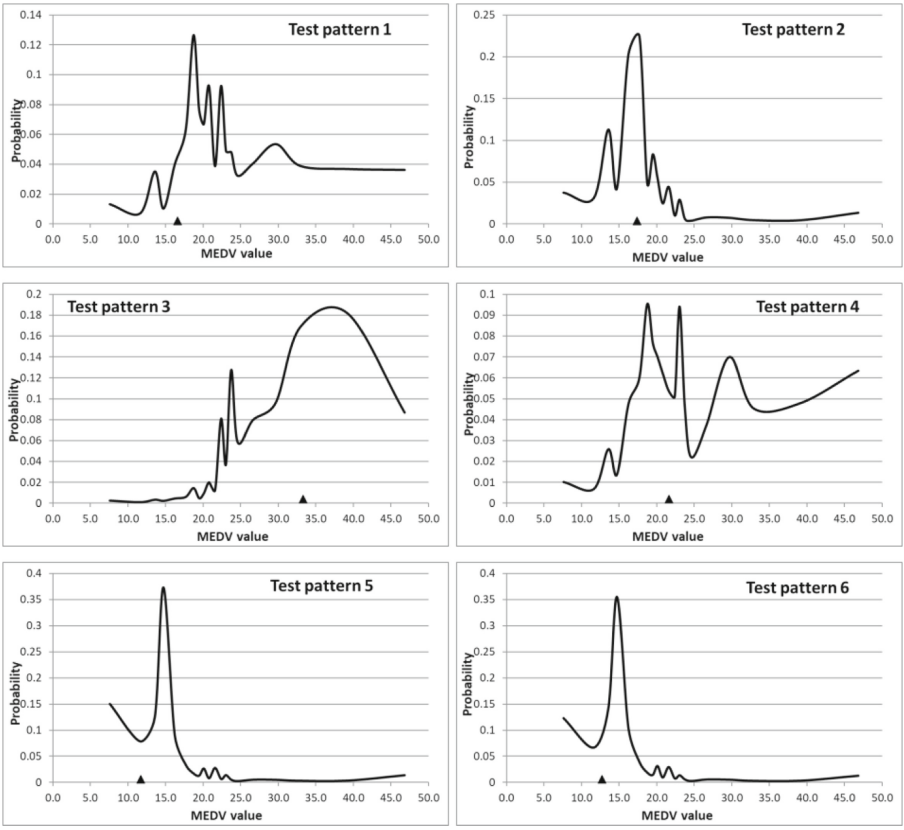

Fig. 6. Approximated probability density distributions for MEDV for six test patterns

It should be noted that apart from above mentioned possibilities of data analyses performed by proposed PDCM model, next data exploration options, based on further investigation in signals of all map neurons, are delivered by this method. For example, for the test pattern 3, a fairly large area of strong signals can be identified at the top of PDCM map (see Fig. 5); this fact is also confirmed by the shape of the probability density function (Fig. 6, pattern 3). The neuron generating the strongest signal in this area (the winning neuron) with coordinates: $X = 10$, $Y = 18$ (column = 10, MEDV = 32.9) collects 10 training patterns assigned to it (see table in Fig. 4, bold number **10**). These 10 cases may constitute (for a real estate market analyst) a comparative base helpful in additional justification of the estate price estimation.

6 Conclusions

The paper submits a concept of new neural tool PDCM, dedicated to relatively wide range of data analysis and data exploration tasks, i.e. special clustering, clusters' visualization, dependent variable prediction (together with its possible visual analysis and justification)

and probabilistic prediction on the basis of approximations of *a posteriori* probability density distribution. Basic theoretical considerations concerning the proposed model have been shown. Also, the presented analyses of application of the PDCM model for the real estate market data indicate the significant effectiveness of this tool and quite rich possibilities of this method in data mining.

However, much future research should yet be done; especially desirable analyses should concern selection of: PDCM parameters (e.g. map dimensions), coefficients used in training algorithm and shape of neighbourhood function (neighbourhood coefficients). Also, testing the model with use of other datasets will be beneficial.

Acknowledgements. The publication is financed by the Krakow University of Economics under the program "Support for Conference Activity - WAK-2022".

References

1. Assylbekov, Z., Melnykov, I., Bekishev, R., Baltabayeva, A., Bissengaliyeva, D., Mamlin, E.: Detecting value-added tax evasion by business entities of Kazakhstan. In: International Conference on Intelligent Decision Technologies, pp. 37–49. Springer, Cham (2016)
2. Chang, F.J., Huang, C.W., Cheng, S.T., Chang, L.C.: Conservation of groundwater from over-exploitation – scientific analyses for groundwater resources management. Sci. Total Environ. **598**, 828–838 (2017)
3. Dameri, R.P., Garelli, R., Resta, M.: Neural networks in accounting: clustering firm performance using financial reporting data. J. Inf. Syst. **34**(2), 149–166 (2020)
4. Dembczyński, K., Kotłowski, W., Słowiński, R.: Solving regression by learning an ensemble of decision rules. In: Artificial Intelligence and Soft Computing ICAISC 2008, pp. 533–544. Springer, Berlin, Heidelberg (2008)
5. Hossu, C.A., Ioja, I.C., Nita, M.R., Hartel, T., Badiu, D.L., Hersperger, A.M.: Need for a cross-sector approach in protected area management. Land Use Policy **69**, 586–597 (2017)
6. Iamandi, I.E., Constantin, L.G., Munteanu, S.M., Cernat-Gruici, B.: Mapping the ESG behavior of European companies. A holistic Kohonen approach. Sustainability **11**, 3276 (2019)
7. Kimura, M., et al.: Application of the self-organizing map in the classification of natural antioxidants in commercial biodiesel. Biofuels **12**, 673–678 (2018)
8. Kohonen, T.: Self-Organizing Maps, Series in Information Sciences (31). Springer-Verlag, Heidelberg (1995)
9. Le, T., Pardo, P., Claster, W.: Application of Artificial Neural Network in Social Media Data Analysis: A Case of Lodging Business in Philadelphia, in Artificial Neural Network Modelling, pp. 369–376. Springer, Cham (2016)
10. Liang, Y.M., Yin, X.F., Chang, D.O.U., Yang, L.I.U.: Application of SOM Neural Network in the Construction of Urban Ramp Driving Cycle, DEStech Transactions on Computer Science and Engineering, International Conference on Artificial Intelligence and Computing Science (ICAICS 2019), pp. 240–244 (2019)
11. Lis, B., Szczepaniak. P.S., Tomczyk .A.: Multi-layer Kohonen network and texture recognition. In: Soft Computing Tools, Techniques and Applications. AOW EXIT, Warsaw (2004)
12. Liu, J., Xu, L.: Improvement of som classification algorithm and application effect analysis in intrusion detection. In: Patnaik, S., Jain, V. (eds.) Recent Developments in Intelligent Computing, Communication and Devices. AISC, vol. 752, pp. 559–565. Springer, Singapore (2019). https://doi.org/10.1007/978-981-10-8944-2_65

13. Lydia, E.L., Bindu, G.H., Sirisham, A., Kiran, P.P.: Electronic governance of housing price using Boston dataset implementing through deep learning mechanism. Int. J. Recent Technol. Eng. **7**(6S2), 560–563 (2019)
14. Merkl, D., Rauber, A.: Alternative ways for cluster visualization in self-organizing maps. In: Proceedings of the Workshop on Self-Organizing Maps (WSOM'97), Helsinki (1997)
15. Morajda, J., Domaradzki, R.: Application of cluster analysis performed by SOM neural network to the creation of financial transaction strategies. J. Appl. Comput. Sci, **13**(1), 87–98 (2005)
16. Morajda, J., Paliwoda-Pękosz, G.: An enhancement of Kohonen neural networks for predictive analytics: Self-Organizing Prediction Maps. In: AMCIS 2020 Proceedings, vol. 6 (2020)
17. Morajda, J., Paliwoda-Pękosz, G.: A concept of FLOPM: neural maps with floating nodes for classification and prediction. In: AMCIS 2021 Proceedings, vol. 16 (2021)
18. Moshou, D.: Artificial Neural Maps. Applications. VDM Verlag Dr. Müller, Saarbrücken, Concepts, Architectures (2009)
19. Mükkulainen, R.: Script recognition with hierarchical feature maps. Connection Sci. **2**, 83–101 (1990)
20. Nan, F., Li, Y., Jia, X., Dong, L., Chen, Y.: Application of improved SOM network in gene data cluster analysis. Measurement **145**, 370–378 (2019)
21. Niska, H., Serkkola, A.: Data analytics approach to create waste generation profiles for waste management and collection. Waste Manage. **77**, 477–485 (2018)
22. Padulano, R., Del Giudice, G.: Pattern detection and scaling laws of daily water demand by SOM: an application to the WDN of Naples, Italy. Water Resour. Manag. **33**(2), 739–755 (2019)
23. Peng, J., Xia, Y.: A cutting algorithm for the minimum sum-of-squared error clustering. In: Proceedings of the Fifth SIAM International Conference on Data Mining. Newport Beach, California, pp. 150–160 (2005)
24. Rashkovan, V., Pokidin, D.: Ukrainian banks' business models clustering: application of Kohonen neural networks. Visnyk of the National Bank of Ukraine **2016**(238), 13–38 (2016)
25. Rojek, I.: Technological process planning by the use of neural networks. AI EDAM **31**(1), 1–15 (2017)
26. Ruck, D.W., Rogers, S.K., Kabrisky, M., Oxley, M.E., Suter, B.W.: The multilayered perceptron as an approximation to a Bayes optimal discriminant function. IEEE Trans. Neural Networks **1**, 296–298 (1990)
27. Wang, L., Wu, C.: Business failure prediction based on two-stage selective ensemble with manifold learning algorithm and kernel-based fuzzy self-organizing map. Knowl.-Based Syst. **121**, 99–110 (2017)
28. Wierzchoń, T., Kłopotek, M.: Algorithms of Cluster Analysis. Institute of Computer Science, Polish Academy of Sciences, Warsaw (2015)