Neural Network Classification of Phase Equilibrium Methods

S. Oreški, J. Zupan,* and P. Glavič[+]
Faculty of Chemistry and Chemical Engineering, University of Maribor,
Smetanova 17, P.O.Box 219, SI-2000 Maribor, Slovenia
*National Institute of Chemistry, University of Ljubljana,
Hajdrihova 19, P.O.Box 34–30, SI-1000 Ljubljana, Slovenia

Original scientific paper
Received: 10. 1. 2001.
Accepted: 20. 2. 2001.

In the paper Kohonen neural network is described as an alternative tool for a fast selection of the most suitable physical property estimation method to be used in efficient chemical process design and simulation. Kohonen neural networks are trained to suggest the appropriate method of phase equilibrium estimation on the basis of known physical properties of samples (objects of the study). In other words, they classify the objects into none, one or more possible classes (possible methods of phase equilibrium) and estimate the reliability of the proposed classes (adequacy of different methods of phase equilibrium). Kohonen map with almost clearly separated clusters of vapor, vapor/liquid and liquid phase regions and 15 probability maps for each of the specific phase equilibrium method, were obtained. The analysis of the results confirmed the hypothesis that the use of Kohonen neural networks for separation of the classes was correct.

Keywords:
Physical properties, phase equilibrium, neural networks, neural network classification, learning procedure.

Introduction

During the past years, neural networks have become widely used in chemical engineering. First attempts appeared about a decade and a half ago. The application of neural networks in various areas of chemical process engineering, such as fault detection, diagnosis, process control, process design, and process simulation has been discussed[1]. Later on, a book was edited on neural networks for chemical engineers where principal fields of application within chemical engineering such as modeling and simulation of complicated processes, process identification and control, were represented[2]. Some very useful books which make readers acquainted with the basic concepts of networks and possible applications of them, have also been published[3–7]. Another promising field of application is to use a neural network as a tool for optimization, where neural networks substitute a mathematical solution of a set of complicated partial differential equations[8,9,10].

In the field of phase equilibria, we can mention the use of neural networks as a part of/or complete predictive tool for vapor/liquid equilibrium prediction, representing different separate phase equilibrium methods. For instance, a paper by *Alvarez*, et al.[11] can be mentioned where an intelligent mixing rule is formed by the combination of the Wong & Sandler and the Huron & Vidal mixing rules using the basic principles of neural networks. This new mixing rule, coupled with the Soave-Redlich-Kwong equation of state and the modified UNIFAC method, represents a new method, which provides successful prediction of vapor/liquid equilibria in isothermal processes. In their work a perceptron neural network with a least mean squares learning rule, is used[12, 13]. On the other hand, we can mention the attempt to use a trained neural network as a new group contribution method which is than compared with the UNIFAC method and experimental data[14]. Another example is the use of neural network method for predicting the phase equilibrium diagrams of the aqueous two-phase systems instead of the conventional thermodynamic methods[15]. In both works feedforward neural networks with error back-propagation learning algorithm, are employed[5,16,17].

In all the above mentioned fields of application, neural networks have been used for the simulation of complicated systems where the available information was experimental. In this study, we use neural networks as an alternative tool to help an engineer to choose a suitable phase equilibrium method for further process calculation. The physical property estimation method is crucial for good design and simulation of a chemical process. In our work, neural networks were trained with inputs describing several combinations of physical properties associated with the corresponding methods of phase equilibrium. Although many methods of phase

[+]Corresponding author. Tel.:+386 2 22 94 451; fax:+386 2 25 27 774; e-mail: glavic@uni-mb.si

equilibrium exist or are appearing day by day, a whole domain of all possible combinations of chemical components, their concentrations, temperatures, and pressures are not covered by them up to now. Our domain consists of a large number of data but, nevertheless, it still does not describe all the possibilities.

Problem consideration and neural network model selection

Artificial neural networks are a set of several different models featuring a wide variety of different architectures, learning strategies and applications. The nature of the problem we are trying to solve determines which neural network will be employed. In Figure 1 some hints about the kinds of problems that can be dealt with when using different neural network learning methods like back-propagation, counter-propagation and Kohonen network, are represented. The problems handled by neural networks can be quite diverse. Most generally they can be divided into five basic types:

Strategy Problem	Back- -propagation (supervised)	Counter- -propagation (supervised)	Kohonen network (unsupervised)
classification	+	+	+
modeling	+	(+)	
mapping	(+)	(+)	+
association	+	+	
moving window	+	+	

Table taken from Zupan, J. & Gasteiger, J. (1999). *Neural Networks in Chemistry and Drug Designs.* Weinheim: Wiley-VCH, page 168.

F i g . 1 – *Kinds of problems that can be dealt with when using different neural network learning methods.*

– ***classification***, which goal is to assign all given objects to appropriate classes (clusters) of objects, based on one or more properties that characterize a given class – the classification can be carried out in a supervised or in an unsupervised manner;

– ***modeling***, which is the search for an analytical function or a model; the advantage of the neural network model is that it does not require a knowledge of the mathematical function; it always requires supervised learning strategy;

– ***transformation*** or ***mapping***, where a multivariate space is transformed into another space of the same, lower or higher dimensionality); mapping can be carried out with not so efficient supervised or efficient unsupervised learning algorithm;

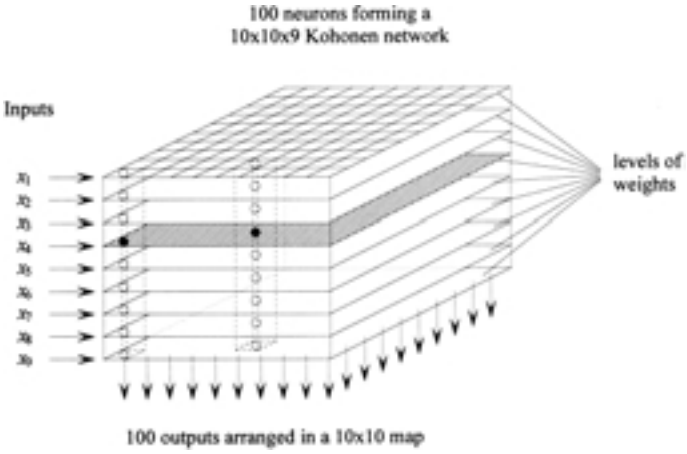
– ***auto*** (for reconstructing patterns) or ***hetero association*** (here we have association between members of two sets of patterns), which can be learned only through examples – i.e. through input-target pairs, so we are restricted to neural networks capable of performing supervised learning.

– the last neural network application is ***moving window*** or ***prediction of time-dependent events*** in time dependent modeling, where the input consists of present and past values of process variables, while the output predicts the future values of the same variables. We are again restricted to supervised learning procedure.

The choice of a supervised or an unsupervised approach depends on the problem and the data available to solve it. In both cases, objects with known answers are needed. In supervised learning, answers are directly used to influence the learning system; in unsupervised learning, answers are needed only to identify and label output neurons. While in supervised learning we want to force the system to adapt itself to an already selected representation of objects and classes, an unsupervised neural network method is more flexible due to its many possible outputs. For supervised learning, multivariate objects should be divided into three sets (a training, a control and a test set). In unsupervised learning we don't need a control set, since learning has to continue until the network stabilizes. The number of available objects is also a very critical factor, since a supervised learning procedure can take even hundreds of thousands of epochs, and for each object, the corrections of thousands of weights might be required even for medium sized net. When having the same number of available objects in unsupervised learning, a time for training is much shorter since single layer neural networks with much smaller number of epochs and weights, are trained.

In our application, three main characteristics could be exposed: a large number of data, the available data do not describe all the possibilities, and the classification is to be made by the neural networks. The first two characteristics of the problem studied, a large amount of data and an incomplete domain, exclude multi-layer neural networks with a supervised learning procedure like error back-propagation learning algorithm, mainly used in chemical engineering. A large number of data transformed into objects for training neural networks demands larger neural networks with more neurons and longer time for training, what is very discouraged for supervised learning. An incomplete do-

main requires an unsupervised approach because all the responses are not available. A neural network should also be able to classify objects into none, one or more classes, and not only into one out of several pre-defined (known in advance), existing classes. For classification of a large number of objects the unsupervised strategy seems to be more efficient than supervised one. A study by *Zupan* and *Gasteiger*[18] showed that almost 90 percent of all publications using neural networks in chemistry had used the back-propagation method, justified or not. According to the problem described, a Kohonen neural network was chosen among several different neural networks as one with the most appropriate architecture and learning strategy[5,19,20,21].



F i g . 2 – *Kohonen neural network represented as a block, containing neurons as columns and weights (circles) in levels. The level of weights which handles the fourth input variable x_{s4} is shaded. Dark shaded are the fourth weight on the neuron (1,1) and the fourth weight on the neuron (6,2), respectively.*

Kohonen network architecture and learning algorithm

As a rule, Kohonen type of the network is based on a single layer of neurons arranged in a two-dimensional plane having a well defined topology. The Kohonen neural network automatically adapts itself to the input data in such a way that similar input objects are associated with the topologically close neurons in the neural network. This means that neurons located physically close to each other react similarly to similar inputs, while the neurons that are far apart in the layout of the neural network can react quite differently even for the identical objects. In the Kohonen approach the neurons learn to determine the *location* of the neuron in the neural network that is most 'similar' to the input vector $\boldsymbol{X}_s$, while in the error back-propagation learning the neurons try to yield *quantitatively* an answer as close as possible to the target. 'Location of the most similar neuron' can mean the location of the closest neuron with the smallest or the largest Euclidean distance to the input object $\boldsymbol{X}_s$, or it can mean the neuron with the largest output in the entire network for this particular input vector $\boldsymbol{X}_s$, etc. The Kohonen neural network has only one layer of neurons; therefore, the specific input variable, let us say the i-th variable, x_{si}, in all the neurons of the neural network is always handled by the weights placed at the i-th position of each neuron. Because the neurons are presented as columns of weights all the i-th weights in all neurons can be regarded as forming an i-th level of weights (see Figure 2)[7].

During the training of the Kohonen neural network the weights of the m-dimensional neurons are self-organizing themselves in the two-dimensional plane in such a way that the the m-dimensional objects are mapped into a 2-dimensional plane of neurons with respect to some internal property. For the numerical values of this particular property one assumes that it is correlated to position of the object in the m-dimensional measurement space of objects[22] and consequently to the position of the object in the 2-dimensional plane of neurons. The algorithm for one cycle of Kohonen learning is as follows:

– an m-dimensional object $\boldsymbol{X}_s$ enters the Kohonen network consisting of N x N neurons;

– the responses of all N x N neurons (each having m weights) are calculated;

– the position c is found for the neuron whose output is the largest or the most similar to the input object $\boldsymbol{X}_s$ (using competitive comparison or the 'winner takes it all' approach);

– the weights of neuron c are corrected to improve its response for the same input $\boldsymbol{X}_s$ on the next cycle;

– the weights of all neurons in the neighborhood (which shrinks during the training) of the c-th neuron are corrected by an amount that decreases with increasing topological distance from c;

– the process is repeated by input of the next m-variate object $\boldsymbol{X}_s$.

In *competitive learning*, the network selects the winner c according to one of the two criteria:

– c is the neuron having either the largest output in the entire network

$$c \leftarrow \max\left\{\sum_{i=1}^{m} x_{si} w_{ji}\right\}, \qquad j = 1, 2, \ldots, n; \quad (1)$$

– or c is the neuron having the weight vector $\boldsymbol{W}_j(w_{j1}, w_{j2}, \ldots, w_{jm})$ the most similar to the input signal $\boldsymbol{X}_s(x_{s1}, x_{s2}, \ldots, x_{sm})$

$$c \leftarrow \min\left\{\sum_{i=1}^{m}(x_{si} - w_{ji})^2\right\}, \qquad j = 1, 2, \ldots, n; \quad (2)$$

The index j refers to a particular neuron, n is the number of neurons, m is the number of weights per neuron, and s identifies a particular input. After finding the neuron c, that best satisfies the selected criterion, its weights w_{ci} are corrected. The weights w_{ji} of the neighbouring neurons are also corrected. Their corrections are usually scaled down, depending on the distance from c. The corresponding scaling function is so called topology dependent function $a(d_c - d_j)$, where $d_c - d_j$ is the topological distance between the central neuron c and the current neuron j, and can have different forms. In Figure 3 a triangular form of the neighbourhood and correction function $a(d_c - d_j)$ for scaling corrections on neighbourweights is performed, which was also used in training the neural networks concerned. Corrections are also decreasing with each iteration step what is represented with another monotonically decreasing function $\eta(t)$ where t is the number of objects entered into the training process; t can be associated with time as well, since the time used for training is proportional to the number of objects entering the network. The corrections of the weights w_{ji} of the j-th neuron lying within the neighbourhood region defined depend on the criterion used to select the central neuron c. The correction function for the maximum signal criterion is:

$$w_{ji}^{(new)} = w_{ji}^{(old)} + \eta(t)a(d_c - d_j)(1 - x_{si}w_{ji}^{(old)}) \quad (3)$$

The correction function for the criterion with output signal the most similar to the input, signal $\boldsymbol{X}_s$ is evaluated similarly:

$$w_{ji}^{(new)} = w_{ji}^{(old)} + \eta(t)a(d_c - d_j)(x_{si} - w_{ji}^{(old)}) \quad (4)$$

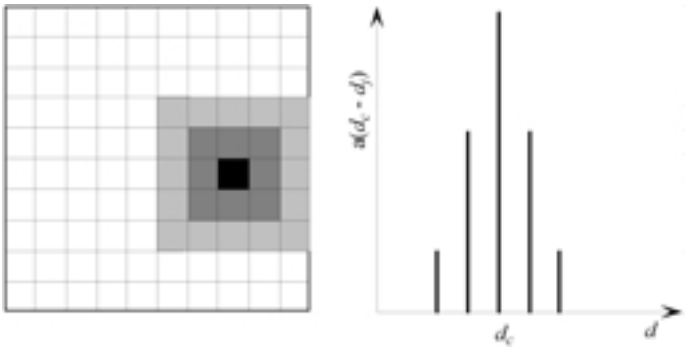


Fig. 3 – *The triangular function $a(d_c-d_j)$ for scaling corrections on neighbourweights. Other possible forms are constant, Mexican hat, etc.*

The Kohonen network or a Kohonen layer can be used alone or can be built into a more complex network as one of its constituent layers. One example of a such complex network is a counter-propagation neural network with supervised learning algorithm[7].

Data preprocessing

From the numerous phase equilibrium methods, we have chosen fifteen of the most often used in practice. They are divided into methods of phase equilibrium which use only equations of state and methods which employ activity coefficients. Among semi-empirical equations of state two cubic ones were chosen, *Soave-Redlich-Kwong* and *Peng-Robinson* method. Out of multiparametric equations *Benedict-Webb-Rubin*, *Starling* and *Lee-Kesler* methods were included. The last of the equations of state was the theoretical *Virial* equation. From activity coefficient methods *Margules-1* and *Margules-2* methods, slightly more complex *Van Laar* and complex *Wilson*, *NRTL*, *UNIQUAC*, *ASOG* and *UNIFAC* methods, were chosen. In the same group there was the *Regular Solution Theory*, too. The phase equilibrium methods mentioned are well represented in the corresponding literature, so they will not be described here again. The data about physical properties of samples (objects) represent information describing chemical bonds, structure of the components, working conditions, further calculations desired, accuracy of the methods, simplicity and speed of calculations, and data availability.

The data collected from the literature and assigned by experts were expressed as objects of the form $\boldsymbol{X} = (x_1, \ldots, x_9, y)$. The variables x_i of vector $\boldsymbol{X}$ are carrying information about physical properties like chemical bonds, working pressure, temperature of the mixture, etc. For instance, x_1 represents chemical bond; $x_1 = 1$ indicates nonpolar mixture of components, $x_1 = 2$ slightly polar mixture, $x_1 = 3$ polar, etc. The variable y represents one method of phase equilibrium out of the fifteen possible ones. For example, y with the value 1 represents *Soave-Redlich-Kwong* equation of state, y with the value 10 *Wilson* activity coefficient method, etc. Initially, there were more than 6000 objects assembled in the bank of data. To exclude multiple objects and to include possibilities overlooked, several preprocessing steps like sorting with 'bubble' and 'quick' sort techniques[23], detection and elimination of identical objects, and simple object classification, were executed.

Each of the nine x_i variables were checked and substituted by as many 'binary' (1/0) variables when it was necessary. To show and justify this procedure

Physical property	$\boldsymbol{X}$	$\boldsymbol{X}_s$			$\boldsymbol{W}_j$		
CHEMICAL BOND	x_1	x_{s1}	x_{s2}	x_{s3}	w_{j1}	w_{j2}	w_{j3}
NonPolar	1	1	0	0			
SlighPolar	2	2	0	0			
Polar	3	3	0	0			
Electrolyt	4	0	1	0			
Polymer	5	0	0	1			
⋮	⋮		⋮			⋮	
PRESSURE	x_3	x_{s9}	x_{s10}		w_{j9}	w_{j10}	
Low	1	1	0				
Medium	2	2	0				
High	3	3	0				
*P*c	4	0	1				
$1 < P\mathrm{r} < 10$	5	0	2				
TEMPERATURE	x_4	x_{s11}			w_{j11}		
$T < T\mathrm{b}$	1	1					
$T\mathrm{b} \le T \le T\mathrm{d}$	2	2					
$T\mathrm{d} < T < T\mathrm{c}$	3	3					
$T = T\mathrm{c}$	4	4					
$T\mathrm{c} < T < 4T\mathrm{c}$	5	5					
⋮	⋮		⋮			⋮	

Fig. 4 – *Some physical properties, their attributes, corresponding variables x_i, transformed variables x_{si} and associated weights w_{ji}.*

few examples are given in Figure 4. It can be seen that physical property *chemical bond* (variable x_1) in the initial object representation $\boldsymbol{X} = (x_1,\ldots,x_9, y)$ can asses values from 1 to 5, representing one of the five attributes: nonpolar, slightly polar or polar mixture, mixture with the presence of electrolyte, and mixture with the presence of polymer, respectively. The first three attributes indicate the increase of polarity from nonpolar to polar, hence they can be coded in one variable with values 1, 2, and 3, respectively. On the other hand, the last two attributes are not correlated neither among themselves nor with the first three ones. Therefore, they should be represented as separate, binary variables having (1/0) values instead of the original values of 4 and 5. This means that the old variable x_1 of the original vector $\boldsymbol{X}$ was transformed into three new variables x_{s1}, x_{s2} and x_{s3} of the new vector $\boldsymbol{X}_s$, suitable for training of the neural networks. The situation is similar when transforming variable x_3 representing the physical property *pressure*. Corresponding variable x_3 is transformed into two variables x_{s9} and x_{s10} of the new vector $\boldsymbol{X}_s$. However, for transformation of the variable x_4 (the *temperature*) having strongly correlated values, the binary substitution is not necessary; the variable x_4 of the original vector is simply transformed into one new variable, x_{s11}, having the same attribute values. Altogether from nine original variables 26 new ones were made.

Additionally, the values (in the range from 1 to 15) of the property y representing one of the 15 phase equilibrium methods were distributed into 15 new binary (1/0) variables y_{si} assigning "1" to the adequate method(s) and "0" to all other ones. The advantage of such notation is that the objects can be assigned by more methods at the same time. This reflects the actual state much better than the assignation of any objects with only one possibility (only one method). In fact the user is always eager to obtain the information abut alternative methods. Understandably, the number of weights in the neurons of the Kohonen network has to be adjusted correspondingly. Figure 5 shows the correspondence between the new variables in the new representation $\boldsymbol{X}_s$ and the "target" or method variable y.

Phase eqilibrium method	$\boldsymbol{Y}$	$\boldsymbol{X}_s$	$\boldsymbol{W}_j$
SRK	y_1	x_{s27}	w_{j27}
PR	y_2	x_{s28}	w_{j28}
BWR	y_3	x_{s29}	w_{j29}
STARLING	y_4	x_{s30}	w_{j30}
LKPKP	y_5	x_{s31}	w_{j31}
VIRIAL	y_6	x_{s32}	w_{j32}
MARGULES-1	y_7	x_{s33}	w_{j33}
MARGULES-2	y_8	x_{s34}	w_{j34}
VAN LAAR	y_9	x_{s35}	w_{j35}
WILSON	y_{10}	x_{s36}	w_{j36}
UNIQUAC	y_{11}	x_{s37}	w_{j37}
NRTL	y_{12}	x_{s38}	w_{j38}
ASOG	y_{13}	x_{s39}	w_{j39}
UNIFAC	y_{14}	x_{s40}	w_{j40}
REG SOLUTION	y_{15}	x_{s41}	w_{j41}

Fig. 5 – *Representation of fifteen phase equilibrium methods for which Kohonen neural networks were trained, their assigned variables x_i and x_{si}, and corresponding weights w_{ji}.*

After the transformation of old variables each object was represented as a multidimensional vector $\boldsymbol{X}_s = (x_{s1}, \ldots, x_{s26}, y_1 = x_{s27}, \ldots, y_{15} = x_{s41})$. Altogether the final representation consists of 41 variables x_{si} (26 and 15 representing 9 different x_i variables and a target vector $\boldsymbol{Y}$, respectively). Finally, in the data base of objects 3780, objects $\boldsymbol{X}_s$ arranged in 41-dimensional vectors appropriate for training

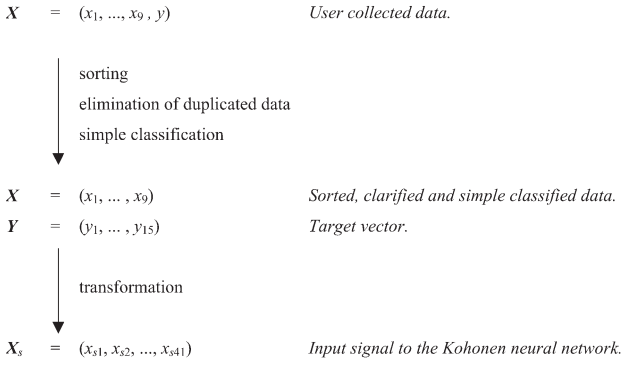


Fig. 6 – *Outline of preprocessing user collected data into net appropriate data.*

neural networks, were obtained. In Figure 6, an outline of data preprocessing is represented.

Training of neural networks and results

According to the number of objects, it was estimated that neural networks of size from 50x50 to 60x60 were needed, where the place for 2500 to 3600 neurons was available. For the present, several square Kohonen networks of the dimension 50 × 50 with 41 weights in each neuron were trained at different learning steps (epochs) using *competitive learning* (the 'winner takes it all' method) according to the criterion that the winner c is the neuron having the weight vector $\boldsymbol{W}_j(w_{j1},w_{j2},\ldots,w_{jm})$ most similar to the input signal $\boldsymbol{X}_s(x_{s1},x_{s2},\ldots,x_{sm})$ (Eq. 2). During the training, the correction of all the m (i=1,…,m) weights on the j-th neuron was carried out according to the correction function for the criterion with the most similar output signal to the input signal $\boldsymbol{X}_s$ (Eq. 4). The learning rate term $\eta(t)$ was decreasing linearly between 0.5 at the beginning of the training and 0.01 at the end of it. The triangular neighbourhood function $a(d_c\text{-}d_j)$ was used for scaling corrections on neighbourweights (see Figure 3).

When increasing the number of epochs in the training, the number of activated neurons increases, while errors of learning and the number of conflict situations (the situations in which two different objects excite the same neuron c), are reduced. Regardless of the prolongation of the training time the number of conflicts in any neural network could not be diminished below 16 %. When analyzing the conflict situations it was found that almost all of them were associated with the activity coefficient methods. In certain regions the Kohonen learning proposed a liquid phase when a two-phase vapor/liquid region was expected. Because the activity coefficient methods are applicable to the liquid part of the two-phase region (the vapor phase was ideal or it was simulated with an appropriate equation of state), these conflict situations were not considered as very bad ones. When inspecting several other conflict neurons it was found out that they where activated by the objects $\boldsymbol{X}_s$ which differed mutually only in values of the 'no-binary' variable x_4 (the temperature). The difference in the value of only one variable (temperature) is obviously not informative enough to enable the Kohonen neural networks to make more precise decisions.

From the original values in the interval (0.0–1.0), the final (trained) values of weights of neurons are scaled between 1 and 9. The neurons with the highest values of the weight w_{si} are the ones that satisfy the criterion of the corresponding variable x_{si} best. With the distance from this particular neuron the values of the weights w_{si} on the same level i of the neighboring neurons are diminishing. For each level i of weights w_{si} a 2-dimensional map of weight values can be drawn.

In Figure 7 the Kohonen map containing three different labels “V”, “S”, “L”, and no labels, is represented. The map is a result of the for 50 x 50 Kohonen neural network trained with 3780 41-dimensional objects after 900 epochs. Almost clearly separated clusters representing homogeneous vapor, heterogeneous vapor/liquid and homogeneous liquid phase regions (labels “V”, “S” and “L” respectively) can be seen. The empty places (with no labels) represent neurons not excited by any of the 3780 objects, thus every labeled neuron in the Kohonen map is activated with at least one phase equilibrium method. Figures from 8 to 15 shows eight out of all fifteen target maps (2-dimensional maps

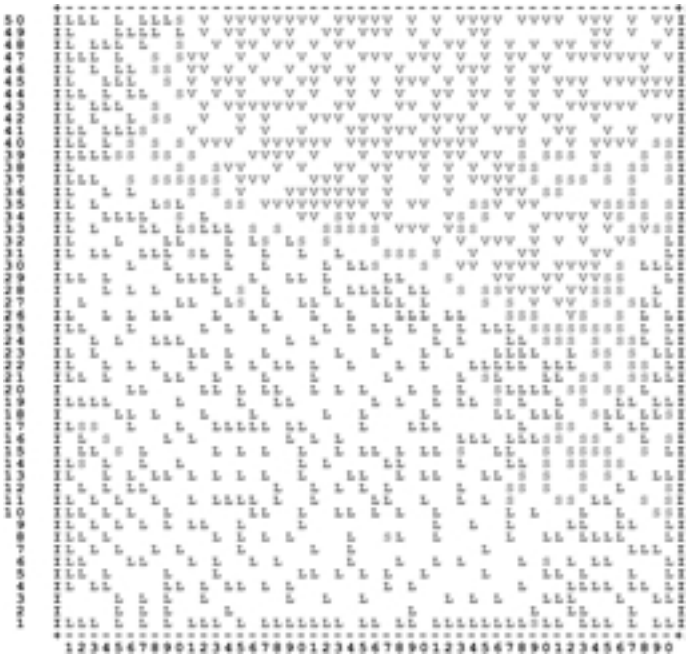


Fig. 7 – *Kohonen 50×50×41 map trained through 900 epochs.*

of weights w_{j27} to w_{j41}) representing eight phase equilibrium methods. In each of these eight maps, the positions of the activated neurons are exactly the same as the positions of the activated neurons in the Kohonen map in Figure 7. If the separate target map is overlapped with the Kohonen map it can be seen in which areas (vapor, vapor/liquid or liquid) the neurons are activated for the phase equilibrium method defined.

As expected the equations of state except the *Virial* method (which use is for engineering purposes limited only on vapor phase when theoretical mixing rules are recommended, Figure 11), appear in all the three regions "V", "S", "L" – see for example the *Soave-Redlich-Kwong* method in Figure 8, *Peng-Robinson* method in Figure 9, and *Benedict-Webb-Rubin* method in Figure 10. The activity coefficient methods appear only in regions "L" and "S" (*Margules-1* method in Figure 12, *Van Laar method* in Figure 13, *UNIQUAC* and *UNIFAC* methods in Figures 14 and 15 respectively).

When less neurons are activated on the target map, smaller is the range of applicability of an phase equilibrium method – (examples are shown on Figures 10, 11, 12 and 13 where the target maps y_3, y_6, y_7 and y_9 for the *Benedict-Webb-Rubin*, *Virial*, *Margules-1* and *Van Laar* methods, are represented).

Specific clusters on a target map (regions of activated neurons that can not be found on other target maps) indicate the specific use of the phase equilibrium method – see for example Figures 10 and 15 where the target maps y_3 and y_{14} for complex empirical Benedict-Webb-Rubin and group contribution UNIFAC methods, are represented.

When having similar phase equilibrium methods similar neurons are activated. For instance, semiempirical *Soave-Redlich-Kwong* and a little more useful semiempirical *Peng-Robinson* methods (Figures 8 and 9) are appropriate in similar situations. Somewhat bigger region of activated neurons in *Peng-Robinson* target map y_2 covers the region of activated neurons in the *Soave-Redlich-Kwong* target map y_1. Similar example are the target maps y_7 and y_9, which represents two mathematically easy to handle activity coefficient methods; one-parameter *Margules-1* method, applicable only for simple nonpolar mixtures, and more useful *Van Laar* method, applicable for nonpolar and moderate polar liquid mixtures and less accurate for strongly polar mixtures. The region of activated neurons of *Margules-1* method can be covered with the region of activated neurons of *Van Laar* method.

Equations of state and methods of activity coefficient can have the same activated neurons

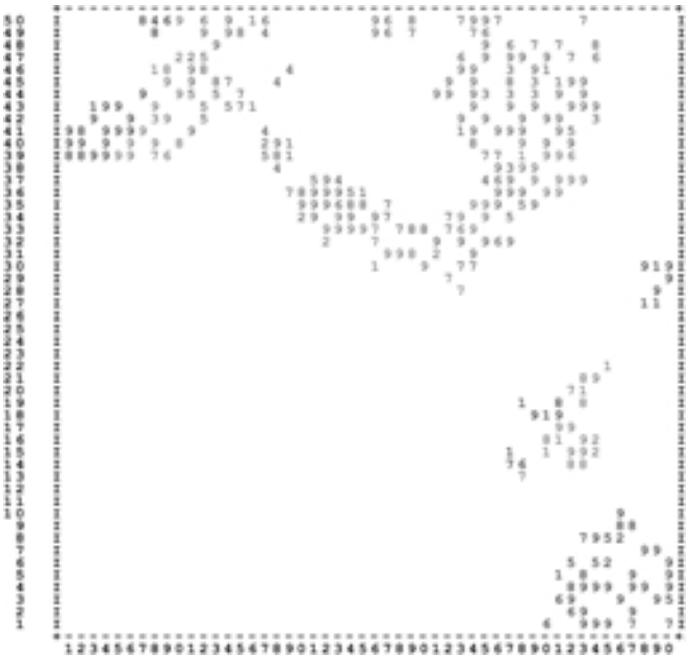


Fig. 8 – *Target map y_1 representing Soave-Redlich-Kwong method.*

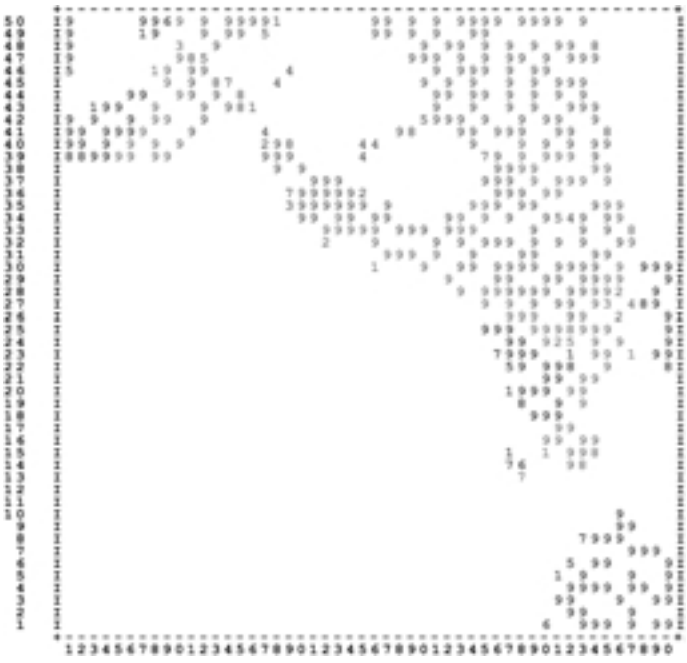


Fig. 9 – *Target map y_2 representing Peng-Robinson method.*

involved, when they are appropriate for the same objects. For example, when objects represent nonpolar or slightly polar liquid mixtures, the same regions "L" of activated neurons on different target maps can be found. We can find the same locations of the activated neurons if we overlap target maps y_1, y_2, y_7 and y_9 for *Soave-Redlich-Kwong* (Figure 8) and *Peng-Robinson* (Figure 9) equations of state, and *Margules-1* (Figure 12) and *Van Laar* (Figure 13) activity coefficient methods, respectively.

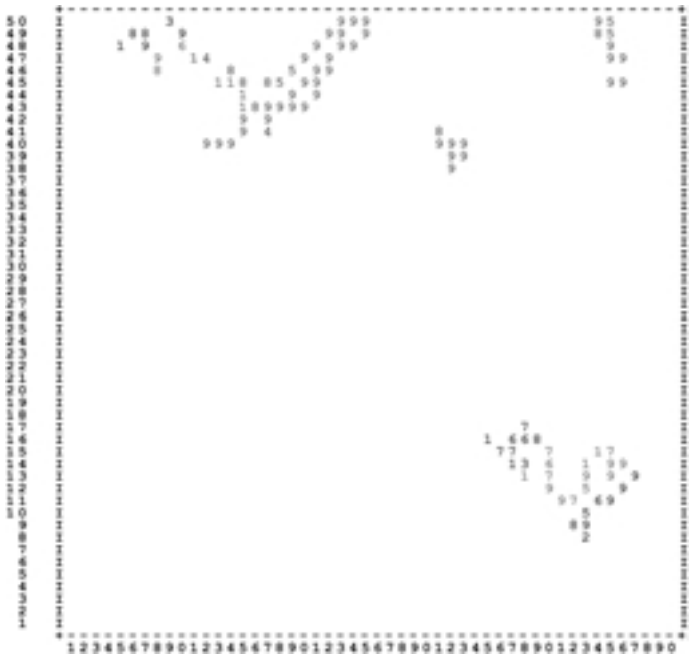


Fig. 10 – *Target map y_3 representing Benedict-Webb-Rubin method.*

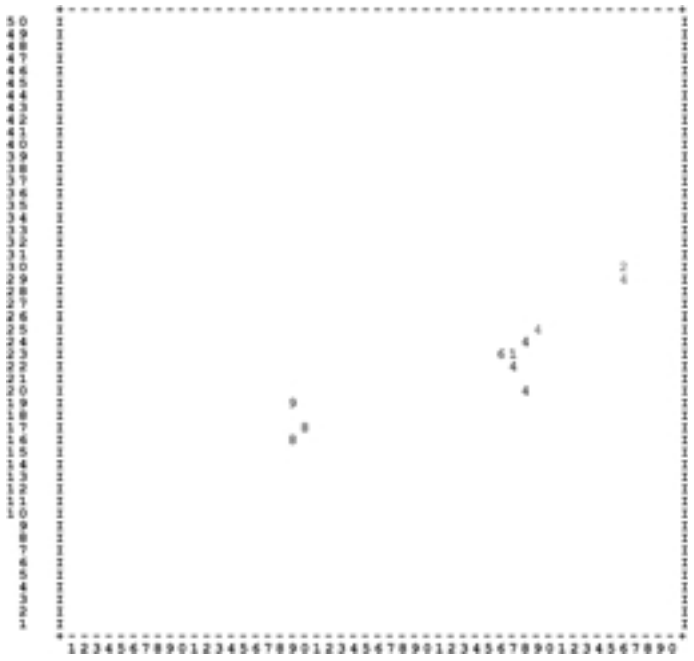


Fig. 12 – *Target map y_7 representing Margules-1 method.*

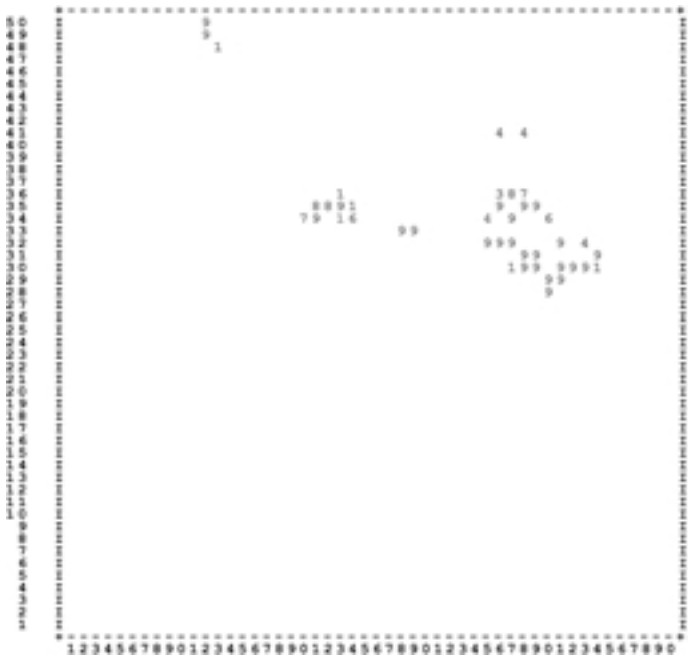


Fig. 11 – *Target map y_6 representing Virial method.*

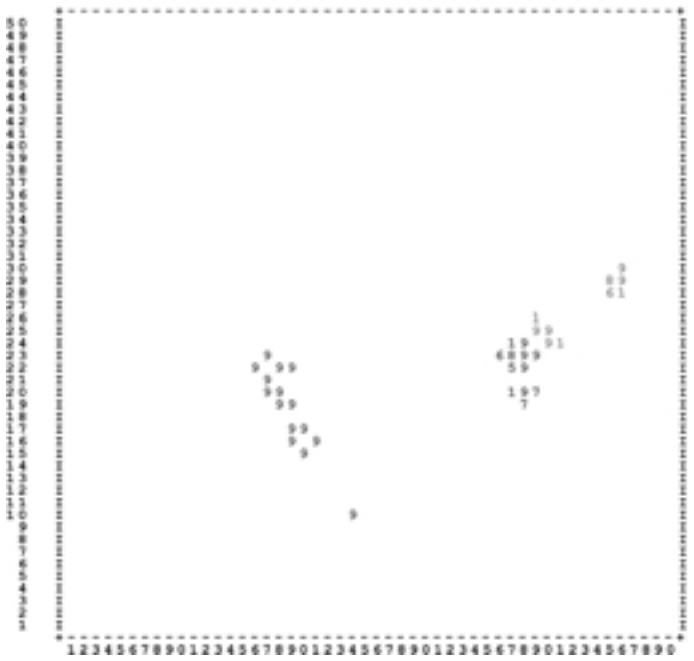


Fig. 13 – *Target map y_9 representing Van Laar method.*

UNIQUAC and UNIFAC methods are represented with target maps y_{11} and y_{14}, respectively (Figures 14 and 15). They are representatives of more complicated activity coefficient methods useful for nonpolar, moderate polar and strongly polar mixtures, and applicable to both vapor/liquid and liquid/liquid equilibrium. According to their wide applicability, the regions of the activated neurons are much bigger than, for example, the regions of activated neurons on *Margules-1* or *Van Laar* target maps (Figures 12 and 13, respectively). The Kohonen map and both, UNIQUAC and UNIFAC, target maps know the information about liquid/liquid equilibria, but information is hidden. With more careful selection of 'labels' on main Kohonen map the regions of liquid/liquid equilibria could be easily detected.

In the different target maps the same neurons can be activated but with different strengths. Such areas represent which phase equilibrium methods are appropriate for certain combinations of physical properties and with what levels of certainty. If in

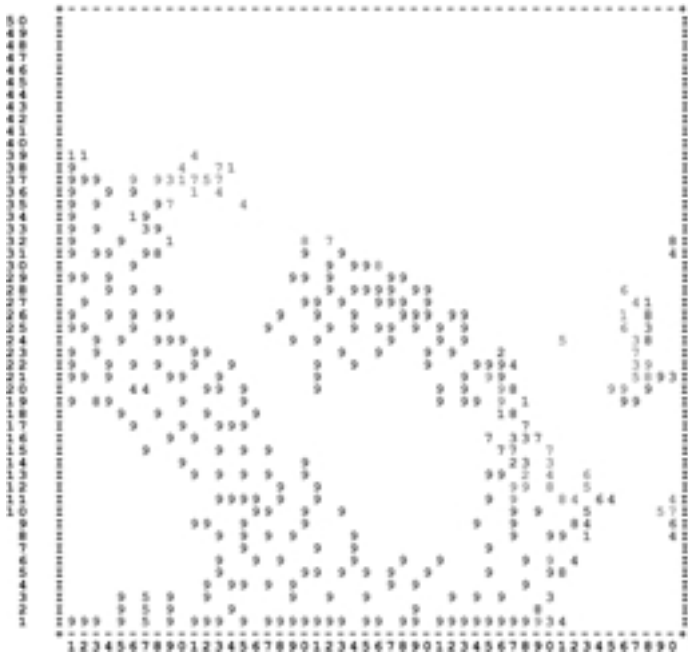


Fig. 14 – *Target map y_{11} representing UNIQUAC method.*

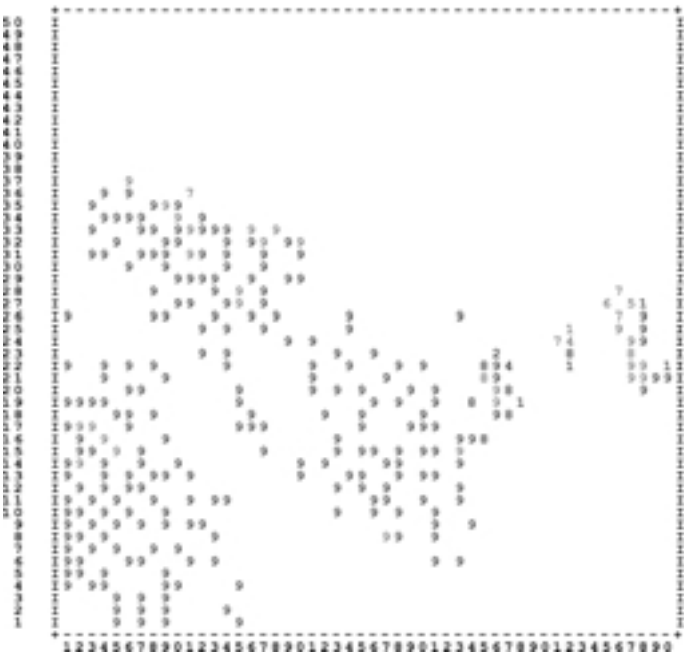


Fig. 15 – *Target map y_{14} representing UNIFAC method.*

the certain target map the same area is not activated, the corresponding method is not appropriate at all. When for a specific neuron (excited by an input object) all fifteen target maps are inspected, the information about adequacy of phase equilibrium methods for particular combinations of physical properties, is obtained.

Conclusions and future research

Using Kohonen learning it was possible to cluster vapor, liquid and vapor/liquid regions. It can self learn the characteristics and the applicability of the phase equilibrium methods. The trained neural network can estimate the reliability of appropriate phase equilibrium methods. Despite the less precise neural networks that have been trained, one can make a conclusion that the hypothesis to use Kohonen network, was correct. In future studies the conflict situations should be resolved with an intervention into the bank of objects to improve their informativity. According to the number of objects larger neural networks will be interesting to be trained in order to prevent too many objects to activate the same neuron. With more precise neural networks further sub-clusters inside the clustered homogeneous vapor, homogeneous liquid and heterogeneous vapor/liquid regions, are expected. With the use of more precise 'labels', regions of liquid/liquid equilibrium could be detected on the Kohonen map, and the corresponding target maps which represent the phase equilibrium methods that are appropriate for liquid/liquid equilibrium, too. Active neurons are distributed rather evenly through the whole Kohonen map. All the neurons (not only the activated ones) carry the weights trained. Hence, we expect to perceive important data about the knowledge missing in our bank of objects in return. This is an advantage over classical expert systems which, in the best case, can only warn the user against unsolvable situations.

Symbols

a – topology dependent scaling function or neighbourhood function
c – central neuron
d_c-d_j – topological distance between central neuron c and neuron j
j – current neuron
m – number of weights
n – number of neurons
s – particular input
t – number of objects which can be associated with time
$\boldsymbol{W}_j$ – weight vector
w_{ji} – weight of neuron j on level i
w_{ci} – weight of central neuron on level i
$\boldsymbol{X}$ – user's object
x_i – physical property i
$\boldsymbol{X}_s$ – input signal to the Kohonen neural network
$\boldsymbol{Y}$ – target vector
y_i – phase equilibrium method i

Greek symbols

$\eta(t)$ – learning rate term, dependent on t

References

1. *Hoskins, J. C., Himmelblau, D. M.*, Comput. Chem. Eng. **12** (1988) 881
2. *Bulsari, A. B., (Ed.)*, Neural Networks for Chemical Engineers, Elsevier, Amsterdam, 1995
3. *Schuster, H. G., (Ed.)*, Nonlinear Dynamics and Neural Networks, VCH Verlagsgesellschaft, Weinheim, VCH Publishers, New York, 1991
4. *Schuster, H. G., (Ed.)*, Applications of Neural Networks, Weinheim: VCH Verlagsgesellschaft, VCH Publishers, New York, 1992
5. *Zupan, J., Gasteiger, J.*, Neural Networks for Chemists, VCH Verlagsgesellschaft, Weinheim, 1993
6. *Ruan, D., (Ed.)*, Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks, and Genetic Algorithms, Kluwer Academic Publishers, Boston, 1997
7. *Zupan, J., Gasteiger, J.*, Neural Networks in Chemistry and Drug Design. Wiley- VCH, Weinheim, 1999
8. *Dong, D., McAvoy, J., Zafiriou, E.*, Ind. Eng. Chem. Res. **7** (1996) 35
9. *Calderon, Z., Espuña, A., Puigjaner, L.*, Comput. Chem. Eng. Supp. (1999) S463
10. *Acuña, G., Cubillos, F., Thibault, J., Latrille, E.*, Comput. Chem. Eng. Supp., (1999) S561
11. *Alvarez, E., Riverol, C., Correa, J. M., Navaza, J. M.*, Ind. Eng. ChemEng. Chem. Res. **38** (1999) 1706
12. *White, D., Sofge, D.*, Handbook of Intelligent Control, Van Nostrand, New York, 1992
13. *Hush, D., Home, B.*, Informatica Automatica **25** (1982) 19
14. *Petersen, R., Fredenslund, A., Rasmussen, P.*, Computers Chem. Engng. **18** (1994) S63
15. *Kan, P., Lee, C. H.*, Ind. Eng. Chem. Res. **35** (1996) 2015
16. *Rumelhart, D. E., Hinton, G. E., Williams, R. J.*, Parallel Distributed Processing: Explorations in the Microstructures of Cognition **1** (1986) 318
17. *Jones, W.P., Hoskins, J.*, Byte **Oct.** (1987) 155
18. *Zupan, J, Gasteiger, J.*, Anal. Chim. Acta. **248** (1991) 1
19. *Kohonen, T.*, Cybern., **43** (1982) 59
20. *Kohonen, T.*, Self-Organization and Associative Memory, Third Edition, Springer Verlag, Berlin, 1989
21. *Zupan, J.*, Algorithms for Chemists, John Wiley, Chichester, 1989
22. *Zupan, J.*, Acta Chimica Slovenica **41** (1994) 327
23. *Zupan, J.*, Uporaba računalniških metod v kemiji, Državna založba Slovenije, 1992