Software description

Counter-propagation neural networks in Matlab

Igor Kuzmanovski a,b,⁎, Marjana Novič b

a Institut za hemija, PMF, Univerzitet “Sv. Kiril i Metodij”, P.O. Box 162, 1001 Skopje, Macedonia
b National Institute of Chemistry, Ljubljana, Hajdrihova 19, SLO-1115 Ljubljana, Slovenia

Received 9 May 2007; received in revised form 16 July 2007; accepted 20 July 2007
Available online 2 August 2007

Abstract

The counter-propagation neural networks have been widely used by the chemometricians for more than fifteen years. This valuable tool for data analysis has been applied for solving many different chemometric problems. In this paper the implementation of counter-propagation neural networks in Matlab environment is described. The program presented here is an extension of Self-Organizing Maps Toolbox for Matlab that is not widely used by chemometricians. This program coupled with the excellent visualization tools available in Self-Organizing Maps Toolbox and with other valuable functions in this environment could be of great interest for analysis of chemical data. The use of the program is demonstrated on the development of the regression and classification models.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Counter-propagation neural networks; Kohonen self-organizing maps; Classification; Regression; Matlab

1. Introduction

Kohonen self-organizing maps (SOM) [1,2] and counter-propagation neural networks (CPNN) [3–8] have become common tools in the chemometric community in the last fifteen to twenty years. Their principle is an experience-based modeling, which can be exploited for black box models; as such, they have shown good results in many different application in chemistry where analytical function which connects input with output variables does not exist or it is impossible to derive one (as in QSAR studies) or where there are deviations from linear behavior which is a case in their applications in analytical chemistry.

Self-Organizing Maps Toolbox [9–11] for Matlab [12] is available on internet for free [10]; nevertheless the authors have not found many applications of this software in chemometrics [13–16]. Due to its simplicity, comfort of use for their excellent incorporation in visualization tools and easy integration with other toolboxes available in Matlab environment, we decided to develop a program for counter-propagation neural networks (CPNN) based on this toolbox.

The algorithm for the counter-propagation neural network tool was written and elaborated for the use in Matlab environment. It was integrated into the SOM Toolbox to exploit the advantages of the standard visualization tools available. The performances of the program are demonstrated on the models built for prediction of the boiling points for the series of substances [17,18] and for classification of famous Italian olive oils data set [19–21].

2. Counter-propagation neural networks algorithm

From the didactical point of view this type of artificial neural networks are usually represented as consisting of two layers (Fig. 1): one layer is the competitive — Kohonen layer and the other is the output layer. The input layer in CPNN performs the mapping of the multidimensional input data into lower dimensional array (most often two-dimensional, since the exploration and visualization of more than two-dimensional SOMs are not easy for human perception). The mapping is performed by use of competitive learning — often called winner-takes-it-all strategy. The main steps of the training procedure of CPNN are presented on the flow chart given in Fig. 2.
The training process of the CPNN is performed in similar way as the training process of Kohonen self-organizing maps [7,8]. This means that the vectors with the N input variables \( \mathbf{x} = (x_1, \ldots, x_i, \ldots, x_N) \) are compared only with the weights \( \mathbf{w} = (w_1, \ldots, w_j, \ldots, w_N) \) of the neurons in the Kohonen layer. Once the winning (or central) neuron \( c \) is found among the neurons in the Kohonen layer only, the weights of both layers (Kohonen and output) are adjusted according to the pairs of input and target vectors \( (\mathbf{x}, y) \) using suitably selected learning rate \( \eta(t) \) and neighborhood function \( a \).

The learning rate \( \eta(t) \) is a non-increasing function, which defines the intensity of the changes of the weights during the training process. Some of the commonly used functions, which define the change of the learning rate with time, are given in Fig. 3. Beside the changes of the weights of the central neuron, the weights of the neighboring neurons are also corrected. The intensity of the correction is determined by the shape of the neighborhood function (Fig. 4). The width of the neighborhood defined by the topological distance between the central neuron and the furthest neuron affected by the correction, during the training, also decreases.

At the end of this part we should mention that from the programmer’s point of view the division of CPNN into two layers is only artificial. The necessary part is to exclude the weight levels, which correspond to dependent variable(s) in

\[
\begin{align*}
\mathbf{w}_{j_i}^{\text{new}} & = \mathbf{w}_{j_i}^{\text{old}} + \eta(t) \cdot a(d_j - d_c) \cdot (x_i - \mathbf{w}_{j_i}^{\text{old}}) \\
\mathbf{u}_{j_i}^{\text{new}} & = \mathbf{u}_{j_i}^{\text{old}} + \eta(t) \cdot a(d_j - d_c) \cdot (y_i - \mathbf{u}_{j_i}^{\text{old}}).
\end{align*}
\]

Fig. 1. Graphical representation of counter-propagation neural network. The Kohonen layer serves for mapping the (multidimensional) input data — \( \mathbf{x} \) into two-dimensional map and finding the winning neurons, while the weights in both layers (Kohonen and output) are adjusted under the same conditions (the learning rate and the neighbourhood function) using pairs of input and target vectors \( (\mathbf{x}, \mathbf{y}) \).

Fig. 2. Flow-chart with representation of some of the most important steps in the training process.
the phase of finding the winning neuron. Afterwards, in the correction of the weights process both layers are treated as one. In the test phase after the winning neuron is found (using only Kohonen layer) the root-mean-square-error of prediction (RMSEP) is calculated using only weights in the output layer.

After the training is finished the Kohonen layer serves as pointing device. After the sample vector \( x_s \) is introduced to the CPNN the comparison of the weights of the Kohonen layer is performed and the position of the winning neuron is determined (this time without further adjustment of the weights — training was finished earlier), the corresponding neuron in the output layer and the values of the weights stored in it, are selected as the best matching for the sample vector \( x_s \). Even if during the training phase some of the neurons in the Kohonen layer have never been excited by training samples, due to existing interactions between the neurons in this phase, the neurons in the output layer will have stored values for possible samples that have not been used during the training. These properties of CPNN together with suitably selected final neighborhood radius are important in development of model with good generalization performances for the interpolation of the properties modeled.

3. Data sets

As previously stated the data sets used for the demonstration of this program were taken from the literature [17–20].

The data set used for development of the regression model consists of 185 saturated acyclic compounds (ethers, diethers, acetals and peroxides, as well as their sulfur analogs) [17,18]. Twelve calculated descriptors were used for prediction of the boiling points of these substances.

The Italian olive oils data set [19,20], which was used for classification, consists of 572 samples of olive oils produced in nine different regions in Italy (North Apulia, Calabria, South Apulia, Sicily, Inner Sardinia, Costal Sardinia, East Laguria, West Laguria and Umbria). For each of the samples the percentage of the following fatty acids was determined: palmitic, palmitoleic, stearic, oleic, linoleic, arachidic, linolenic and eicosenoic. In this case, as dependent variables were used vectors \( y \) with length nine. If the sample belongs to region labeled as \( k \), then \( y_k = 1 \), and all the other elements of \( y \) are set to 0.

In order to perform the analysis, the data sets were randomly divided into training and test sets. In the case of development of the regression model, 40% of the structures were used as a test set. The remaining structures were used as a training set. In the case of development of the classification model, the Italian olive oil data set was divided into training set consisted of 1/3 of the sample, while all the other samples were used as a test set. In both cases before the optimization started the variables were autoscaled.

4. Software specifications and requirements

The CPNN program was developed in Matlab 6.5 (Release 13) [12] on the bases of SOM Toolbox [9] developed by J. Vesanto et al. All the features available for SOM in the toolbox are also available for our CPNN program. The Matlab function which executes the program is called \texttt{som_counter_prop}. The syntax for execution of CPNN program could be found by typing \texttt{help som_counter_prop} in Matlab’s Command Window after the installation of the program. The required inputs as well as the outputs are given in the help section of \texttt{som_counter_prop} program. The parameters used for definition of the shape of the CPNN and for its training are numerical. Only three input/output parameters are not numerical. These parameters are structured variables representing training data, test data and the trained CPNN.

The supporting documentation and demo scripts available in the SOM Toolbox [9] should be used in order to learn how to extract maximum information of the analyzed data by this counter-propagation neural networks program.

The input data file format used in our program is the same as the one used in the SOM Toolbox [9]. This data format is in details described in the SOM Toolbox documentation [9]. Our CPNN program is capable of handling the missing values in the same way as the other functions available in the SOM Toolbox. For this purpose the user should replace them with the label “NaN”.

The \textit{training} of the CPNN could be performed in two phases: rough (with large learning rate and large neighborhood radius)
and fine-tuning phase (with smaller learning rate and smaller neighborhood radius). If one prefers to train the networks in only one phase it could be achieved by adjusting the number of epochs in one of the phases to zero. Of course, all the adjustment of the weights of the map neurons are performed by finding the winning neurons while considering the weight levels only for the independent variables. While RMSEP for the training set and RMSEP for the test set were calculated comparing only the weights levels from the output layer with the corresponding values for the dependent variables.

The Matlab program presented here as input variables accepts almost all the parameters necessary for creation of highly customizable CPNN: training and test data sets, training parameters (epochs in rough and fine-tuning phase, neighborhood function), size of the CPNN, neighborhood (hexagonal or rectangular), shape of the network (sheet, cylinder, toroid) weights initialization function, the parameter called mask and the labels which will be used for labeling of the trained map.

The initialization of the weights of the network could be performed assigning random numbers to the weights in all the levels or by its initialization along the first two principal components. The later weight initialization algorithm is better, since in two consecutive optimizations the network will show the same performances. The user may chose between one of these initialization functions.

As we mentioned earlier from the programmer’s point of view the division of CPNN into two layers is only artificial. What is important is to exclude the weight levels which correspond to dependent variable(s) in the process of finding the winning neurons during the training. This is performed using the mask. The mask is an important input vector ($m$) — its length is equal to the total number of variables, independent and dependent ones. It is responsible for the division of the variables into dependent and independent ones. If its element is set to zero, the corresponding variable is excluded for the process of finding the winning neuron (it becomes dependent variable) and the corresponding weight level becomes part of the output layer of the CPNN. In addition, by setting a proper value for the elements of the mask ($0 < m_i \leq 1$) that correspond to the independent variables, the relative importance of these variables is adjusted.

The output variables of the trained map are: RMSEP values (calculated for the training and for the test set), the structured variable which contains the weights levels (called codebook vectors by the authors of the SOM Toolbox) of the trained CPNN as well as the predicted values for the input and output variables for the training and the test set.

The compatibility of the CPNN program was tested with Matlab 6.5 and 7.0 versions, using computers running under Windows XP operating system. The time required for training of the CPNN depends on the size of the analyzed data.

### 5. Demonstration

In order to present some features of the CPNN program, together with it, two demo scripts (called demo_class.m and som_counter_prop_demo_reg.m are provided) are provided.

For these demonstrations we used CPNN with hexagonal neighborhood, with plain boundary condition, Gaussian neighborhood function and linearly decreasing learning rate. In order to perform the finding of most suitable network size and optimal number of epochs (in both training phases) in automated manner the optimization was performed with genetic algorithms [22–28]. The explanation about how the optimization was performed using genetic algorithms is out of the scope of this article and will be published elsewhere.

#### 5.1. Regression demo

As previously stated genetic algorithms were used for finding the most suitable network size and the training parameters for prediction of the boiling points with CPNN.

For the sake of the demonstration, in this case as well as in the case of the demonstration of the use of this program for classification purposes, the data sets were randomly divided into a training and test set. However, in the case of using our CPNN program for research, the users are advised to perform the separation of the data set using an algorithm suitable for this purpose [29,30].

The optimization procedure was repeated several times. The network size and the training parameters for one of the best solutions are presented in Table 1. The agreement between the expected and calculated values for the samples in the test set is presented in Fig. 5. The correlation coefficients both, for the training set and the test set, are above 0.95 and show that the model presented here has acceptable performances.

#### 5.2. Classification demo

After few repetitions of the optimization the most suitable size of the network and training parameters were selected (Table 2). The unified distance matrix for this trained network, calculated considering only weight levels corresponding to dependent variables, is presented in Fig. 6. One can notice that all the regions are well differentiated on the map. Exceptions are the samples from Sicily, which are divided in two parts. Also, only 3 of the samples in the training set are misclassified or are in

<table>
<thead>
<tr>
<th>Network size</th>
<th>Width</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Training parameters</th>
<th>Rough training phase</th>
<th>Fine-tuning phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of epochs</td>
<td>18</td>
<td>230</td>
</tr>
<tr>
<td>Initial neighbourhood radius</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Final neighbourhood radius</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Initial learning rate</td>
<td>0.10</td>
<td>0.05</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CPNN parameters for the presented regression example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network size</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
the regions not occupied by most of the samples. One of the samples from South Apulia is in the upper right region of the map, which belongs to olive oil samples from Sicily. The second sample is from East Laguria and it is placed at the left bottom part of the map. Although this neuron is in the region, which belongs to olive oil samples from West Laguria, the topological distance of this and its neighboring neurons is larger than among other neurons in this region. The third sample is from Sicily and it is found in the region of the network that belongs to North Apulia, in the neuron, which is close to the region of the network, which belongs to Sicily. Since this sample is close to the borderline of the two clusters we decided to examine the weights of this neuron a little bit more carefully. In Table 3 the corresponding weights of the weight levels from the output layer for this neuron and of the surrounding neurons are presented. Simple inspection of the weights presented in this table shows the reasons for misclassification of the particular sample.

We tested the trained CPNN with the test samples (Fig. 7). Only 16 samples from the test set (which represent only 4.1% of all samples) were misclassified. Most of the misclassified samples have origin from Sicily (8 samples), three misclassified samples were from South Apulia and two were from Calabria. Another three misclassified samples belong to three different regions (East Laguria, North Apulia and Costal Sardinia).

6. Conclusion

The presented program for counter-propagation neural networks, according to authors of this manuscript, is valuable extension of SOM Toolbox and it could be of great interest not only for chemists. Using this program maximum information could be extracted from your data when it is used in combination with other functions of SOM Toolbox, especially its excellent capabilities for different kinds of visualizations of the data and the trained networks.

Other advantage achieved by using this program is the openness of the algorithms developed in Matlab environment for easy integration with other functions available from other toolboxes. For example it could be used coupled with functions that could perform optimization of its parameters or variable selection using genetic algorithms. As mentioned earlier in this article the automated optimization procedure of the CPNN using

<table>
<thead>
<tr>
<th>Network size</th>
<th>Width</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>21</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Training parameters</th>
<th>Rough training phase</th>
<th>Fine-tuning phase</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of epochs</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>Initial neighbourhood radius</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Final neighbourhood radius</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Initial learning rate</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>Final neighbourhood radius</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Initial learning rate</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Table 2: CPNN parameters for the presented classification example

Fig. 5. Expected vs. predicted values for the boiling points for the (a) training set and (b) test set.
genetic algorithms is in progress in our laboratory. Additionally, our program can also be applied to data with missing values.

At the end we would like to say that the program, together with the demo scripts that describe some of its features as well as the Italian olive oil data sets could be downloaded from: http://www.hemija.net/chemometrics/.

7. Validation

The chemometric work groups from Skopje, Macedonia, and Ljubljana, Slovenia, developed a toolbox for analyzing and modeling data sets by means of a Counter Propagation Neural (CPN) network. A CPN network resembles two associated Kohonen networks which are trained in a supervised way.

The provided readme file described all the necessary steps to be taken for installing the toolbox successfully. This was evaluated by running the provided demo script `som_counter_prop_demo_class.m`. This demo shows in a nutshell how the CPN is trained and how the results for the training and test set can be inspected nicely in a graphical way. During run time no errors were encountered. The time to train the CPN network is only a few seconds (for the relatively small data sets I used). The toolbox was tested with Matlab 6.5.

There are just a few non-severe outstanding issues. I missed a kind of comprehensive tutorial which addresses how the most important scripts must be called, i.e., which input and output parameters and variables must be provided. Moreover, a description of the format of the input data must be provided.

Fig. 6. Unified distance matrix, calculated using weigh levels that correspond to output variables, for the trained CPNN together with the neurons labelled with the training samples (the number of the samples is in the brackets). North Apulia — NA, Calabria — Ca, South Apulia — SA, Sicily — Si, Inner Sardinia — IS, Costal Sardinia — CS, East Laguria — EL, West Laguria — WL and Umbria — Um.
Table 3
Comparison of the weights that correspond to dependent variables of the best matching neuron and its neighbouring neurons for the misclassified sample from Sicily (North Apulia — NA, Calabria — Ca, South Apulia — SA, Sicily — Si, Inner Sardinia — IS, Costal Sardinia — CS, East Laguria — EL, West Laguria — WL and Umbria — Um)

<table>
<thead>
<tr>
<th>Positions of the neurons</th>
<th>NA</th>
<th>Ca</th>
<th>SA</th>
<th>Si</th>
<th>IS</th>
<th>CS</th>
<th>EL</th>
<th>WL</th>
<th>Um</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper right (NA region)</td>
<td>0.503</td>
<td>0.001</td>
<td>0.000</td>
<td>0.496</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Upper left (Si region)</td>
<td>0.212</td>
<td>0.011</td>
<td>0.000</td>
<td>0.777</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Right (NA region)</td>
<td>0.765</td>
<td>0.000</td>
<td>0.000</td>
<td>0.235</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Lower right (NA region)</td>
<td>0.773</td>
<td>0.000</td>
<td>0.000</td>
<td>0.227</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Lower left (NA region)</td>
<td>0.680</td>
<td>0.000</td>
<td>0.000</td>
<td>0.320</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Left (Si region)</td>
<td>0.398</td>
<td>0.004</td>
<td>0.000</td>
<td>0.597</td>
<td>0.000</td>
<td>0.000</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Best matching neuron</td>
<td>0.534</td>
<td>0.001</td>
<td>0.000</td>
<td>0.465</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Dependent variables for the sample</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

And last, I minor one, in the 3D PCA plot coloured symbols are used for the data objects. I suggest using the same colouring in the legend of the figure as well.

In summary, the CPN toolbox is a useful tool for analyzing and modeling (large) data sets in a supervised way. The toolbox is easy to install and runs fast and error free.

Fig. 7. Unified distance matrix, calculated using weigh levels that correspond to output variables, together with the neurons labelled with the samples from the test set.
Acknowledgements

The financial support of the European Union in the framework of the Marie-Curie Research Training Network IBAAC project (MCRTN-CT-2003-505020) and of the Ministry of Education, Science, and Sport of Slovenia (P1-017 grant) as well as the financial support of the Ministry of Education and Science of Macedonia and Ministry of Education, Science, and Sport of Slovenia is gratefully acknowledged (project grant: BI-MK/04-06-12).

References