ANN-based Software Analyzer Design and Implementation in Processes of Microbial Ecology

SVETLA VASSILEVA¹, BONKA TZVETKOVA²

¹Department of Knowledge Based Control Systems, Institute of Control and Systems Research-Bulgarian Academy of Sciences (BAS), str. Akad.G.Bonchev, bl.2, POB 79, 1113 Sofia
²Institute of Microbiology - BAS, str. Akad.G.Bonchev, bl.26, POB 79, 1113 Sofia
BULGARIA
vasileva@icsr.bas.bg, bonka_ivanova@abv.bg, http://www.icsr.bas.bg

Abstract:- Present work is oriented to study of processes of microbial ecology by developing and implementing artificial neural networks (ANN)-based software analyzers. Microbial ecology investigates the function of microorganisms in the environment. Towards understanding the role of microorganisms in important ecosystem processes, one example is considered. The presented example is focused on continuous yeasts growth prediction by the developed software analyzer, which helps to discover the influence of initial formaldehyde concentrations on the continuous cell growth, correlated with the protein-synthesizing ability \( A, [h^{-1}] \) of yeasts strain *Candida didensii* 74-10 at two different dilution rates \( D, [h^{-1}] \). The Levenberg-Marquardt optimization technique has been used to upgrade the network by minimizing the sum square error (SSE). The performance of the network with two hidden layers for predicting cell mass and protein synthesizing ability is found to be very effective.

Key-Words: - intelligent software analyzers, artificial neural networks, microbial ecology, continuous yeast growth, formaldehyde

1 Introduction

Nowadays new methods and the fusion of ideas from microbial ecology and environmental microbiology, computer science and information technology are allowing ecologists and microbiologists to tackle previously unanswerable questions [6,8-12].

Microbial ecology is the science, which study the function of microorganisms in relation to the environment. Microorganisms are the most diverse and abundant group of organisms on Earth. Yet, within the field of ecology, microbial studies have typically focused on the processes microorganisms carry out, rather than on the ecology of the organisms themselves. In contrast, studies of environmental microbiology have primarily characterized the diversity of microbial populations. A true integration of classical ecological thinking and organism-based microbiological studies is a surprisingly recent phenomenon.

Contemporary efforts to link microbial ecology and environmental microbiology are taking advantage of new methods and technologies [3,9,10]. New tools allow researchers to solve problems of wastes elimination in water and soil, to describe influence of different chemical and biochemical reagents on microbial population, to distinguish large-scale patterns of microbial diversity and to compare such patterns to those of plant and animal taxa. Now scientists can describe microbial community composition and determine its stability across time and space. Additionally, modern tools for describing community composition can be combined with ecosystem and biogeochemical measurements to identify environmental controls on microbial processes and the specific roles of microbes in biogeochemical cycles.

Software sensors (known also as software analyzers, inferential or indirect measurements) belong to the knowledge-based tools, which helps the scientists to discover new particularities of microbial populations related to the environmental conditions. Software sensor models, which are the basis for soft-analyzer design, allow implementing various methods of sensitivity analysis to select the most influential operational variables to the target control variable [10,13]. The software sensors have received deserved attention in last years, because they provide a progressive automation technique when hardware sensors are not hopeful on-line to realize opportune monitoring and efficient control actions [10].
In comparison with the hardware sensors, the soft-sensors allow predicting with higher accuracy the hard to measure key process variables, to use model-bases (specially created for intelligent soft-sensor) and to select the most relevant sensor-models for operative control and monitoring of multiphase, complex and unstudied processes.

Main goal of our research is to demonstrate efficiency of software analyzer implementation in microbial ecology on an example of yeast growth influenced by various initial concentration of formaldehyde.

2 ANN-based soft-analyzer design

Modern concept of microbial ecology lies on the network of interacting intelligent systems for data acquisition, assessment, interpretation, decision support and control, to provide efficiency savings of data and expert knowledge.

The technology of an intelligent software analyzer design consists of the following six steps: data-base construction and analysis of data-relevance for software analyzer design; selection of input-output attributes for sensor-model structure determination; best-predictors choice by sensitivity analysis; validation of the designed sensor model; simulative investigations and tests of the developed soft-analyzer in the real conditions [10].

Methods, which are used for software analyzer design in this study, are based on the artificial neural networks (ANN). The algorithm and data processing are performed in the Neural Network Toolbox of Matlab with Simulink, ver.6.5.

2.1. Artificial Neural Networks

The foundation of a neural network is the neuron which is a processing element performing a weighted sum of all inputs variables that feed it. Depending on the value of weighted sum of the variables, the neuron gives a signal to the neurons in the adjacent layer through a non-linear transfer function. The choice of the architecture of the network depends on the task to be performed and the architecture of the model is specified by the node characteristics, network topology and learning algorithm. In standard architecture, neurons are grouped into different layers like-input, output and hidden layers. Generally, for modeling of physical systems, three-layered, feed-forward network is normally used. In the present study, a four-layered feed-forward network as shown in Fig.1 has been used which consists of a layer of input neurons, two layers of hidden neurons and one layer of output neurons. The ANN configuration is represented as 5:15:16:1, that is, the input layer consists of five inputs, each hidden layer consists of fifteen and sixteen neurons, respectively and the output layer consists of one output. The non-linear transfer functions implemented in our case are sigmoidal in the hidden layers and linear in the output layer.

2.2. Back Propagation Training Algorithm

The back propagation of error algorithm, based on multi-layered feed-forward net and considered to be the most versatile algorithm [6], was used to train the network for predicting correct outputs those obtained from experiments and generated one. The BP algorithm adjusts the network weights and bias values to minimize the square sum of the difference between the given output \( Y \) and output values calculated by the net \( Y' \) using gradient decent method as follows:

\[
SSE = \frac{1}{2N} \sum (Y - Y')^2, \tag{1}
\]

where \( N \) is the number of experimental data points used for the training. Levenburg-Marquardt technique is used to improve the learning rate and stability of the BP algorithm for searching minimum error.

Training and Testing Procedure is carried out as follows. The entire data were divided into two sets. The larger set consisting of 401 data was used for training and the rest 99 data were reserved for use in testing and validation of the ANN predicted output values. As the ANN inputs should be in the range of 0 to 1, so all inputs are normalized according the formula:

\[
X_{\text{normalized}} = \frac{(X - X_{\text{min}})}{(X_{\text{max}} - X_{\text{min}})} \tag{2}
\]

The following algorithm is then followed for training the network:

Step 0: Initialize weights (set random values between 0 and 1).

Step 1: While stopping condition is false then do steps 2-9.

Step 2: For each training pair of set do steps 3-8.

Step 3: Each input unit \( x_i, i = 1, 2, \ldots, 5 \) receives input signals \( x_i \) and sends this signal to all the nodes in the next hidden layer.

Step 4(a): Each hidden unit \( h_j, j = 1, 2, \ldots, 15 \) sums its weighted input signals and bias is added to this weighted sum to compute its output signal as:

\[
O_{Oj} = f(\omega_{Oj} + \sum o_{ij}x_i), i = 1, 2, \ldots, 5 \tag{3}
\]

and sends this signal to all the units in the following
layer \((h_k, \text{hidden layer})\).

Step 4(b): Each hidden unit \((h_k, k = 1,2,\ldots,16)\) sums its weighted input signals and bias is added to this weighted sum to compute its output signal as
\[
O_{ok} = f\left(\omega_{oj} + \sum \omega_{jk} O_{jk}\right), j=1,2,\ldots,15
\]  
(4)
and sends this signal to all the units in the following layer \((O_l, \text{output layer})\).

Step 5: The output unit \((O_l, l = 1)\) in the output layer sums its weighted input signals and applies its activation function to compute its output as
\[
O_{ol} = f\left(\omega_{oj} + \sum \omega_{kl} O_{ok}\right), k=1,2,\ldots,16
\]  
(5)
Step 6: The BP of errors starts at output layer \(O_l\) to compute its error information term as
\[
\delta_1 = (t - O_{ol}) f'\left(\omega_{oj} + \sum \omega_{kl} O_{ok}\right)
\]  
(6)
where \(t\) is the target value. It calculates weights correction and bias correction terms as
\[
\Delta \omega_{kl} = \delta_1 \delta_k O_{ok} \quad \text{and} \quad \Delta \omega_{oj} = \alpha \delta_1,
\]  
(7)
where \(\delta_1\) is the error sent to nodes in the previous layer.

Step 7(a): Each hidden unit \((h_k, k = 1,2,\ldots,16)\) sums its delta inputs from units in the next layer and calculate its error information term as
\[
\delta_{ok} = \delta_k f^l(O_{ok}),
\]  
(8)
\[\delta_k = \sum \delta \omega_{kl}, \text{ for } l=1.\]  
It calculates its weights correction and bias correction term as
\[
\Delta \omega_{jk} = \delta_{ok} O_{kj} \quad \text{and} \quad \Delta \omega_{ok} = \alpha \delta_k,
\]  
(9)
Step 7(b): As step 7(a), each hidden unit \((h_j, j=1,2,\ldots,15)\) sums its delta inputs from units in the next layer and calculate its error information term as
\[
\delta_{oj} = \delta_j f^l(O_{oj}),
\]  
(10)
where \(\delta_j = \sum \delta \omega_{jk}, \text{ for } k=1,2,\ldots,16.\) It calculates its weights correction term and bias correction term as
\[
\Delta \omega_{jk} = \delta_{oj} O_{kj} \quad \text{and} \quad \Delta \omega_{oj} = \alpha \delta_j.
\]  
(11)
Step 8: Each output node \((O_l, l = 1)\) updates its weights and bias as
\[
\omega_{jk}(\text{new}) = \omega_{kl}(\text{old}) + \Delta \omega_{kl},
\]  
(12)
where \(k=1,2,\ldots,16\) and
\[
\omega_{ok}(\text{new}) = \omega_{ol}(\text{old}) + \Delta \omega_{ol}.
\]  
(13)
Each hidden node \((h_k, k=1,2,\ldots,16)\) updates its weights and bias as
\[
\omega_{ij}(\text{new}) = \omega_{jk}(\text{old}) + \Delta \omega_{jk},
\]  
(14)
\[
\omega_{oj}(\text{new}) = \omega_{ok}(\text{old}) + \Delta \omega_{ok}.
\]  
(15)
Step 9: Test of stopping condition.
As noted earlier, the Levenberg-Marquardt variation of nonlinear least squares optimization technique is used to upgrade the back propagation algorithm. It involves some additional computations in the step 6 to calculate its weight correction term as
\[
\Delta \omega_{jk} = (H + \lambda \times I)^{-1} \delta_{ok} Z_j
\]  
(16)
where \(k\) is the row number; \(L\) is the number of neurons in that layer and \(l\) is the identity matrix
\[
(H)_{kl} = [J'J]_{kl}
\]  
(17)
and
\[
J_{kl} = \partial(t_{k} - y_{L})\delta_{k}.
\]  
(18)
where \(I\) is the identity matrix of the function; \(H\) is the Hesian matrix of the function; \(J\) is the Jacobian matrix of the function; and \(l\) is step length, which is the parameter for Levenberg- Marquardt method.

After training, the network is tested by introducing the testing input data sets. Experimental data are then compared with simulated data. If the network predictions are in close agreement with the experimental data, then network topology is accepted. Else the training process is repeated with new parameters.

3 Example – results and discussion
The aim of this example is to explain some aspects of the influence of formaldehyde on continuous yeast growth and protein formation ability by implementing ANN-based software analyzers.

3.1 Problem statement
To explain the reason of designing the software analyzer of biomass content and protein-synthesizing ability, some explanation concerning the process will be introduced.
Acid wood hydrolisates or other wastes of plant origin can be used in some cases as culture media for cultivation of fodder yeasts. In addition to reducing sugars, acid wood hydrolisates also contain formaldehyde, furfural [3], organic acids and other substances that cannot be completely separated.
Thus, it is interesting to study their effects on the growth and protein synthesis of fodder yeasts [3]. On the other hand, formaldehyde is a common product of the chemical industry that is often dispersed in ecosystems and because of its toxic and mutagenic properties; it presents a certain ecological hazard [4]. Therefore, formaldehyde degradation by bacteria and yeasts has been a subject of many recent studies.
Formaldehyde can be assimilated by
methylotrophic microorganisms via the serine pathway as a sole source of energy.

It is known that when the formaldehyde content in the cell is increased it cannot be metabolized efficiently by inclusion in the ribulose monophosphate pathway, so it acts as an inhibitor. Formaldehyde is also an inhibiting agent when the activity of formaldehyde dehydrogenase is low and limits the consumption of formaldehyde as a substrate.

The presented example is focused on process of continuous yeasts growth prediction by the developed software analyzer, which helps to discover the influence of initial formaldehyde concentrations on specific growth rate and protein-synthesizing ability of yeasts strain Candida didensii 74-10 at different dilution rates (D, [h⁻¹]).

The experimental results involve data about the biomass (X, [g/dm³]), yield of biomass (YX, [%]), rate of substrate assimilation (qs, [h⁻¹]), protein content (P, [%]), yield of protein (YP, [%]), RNA [%] and protein-synthesizing ability (A, [h⁻¹]). The experimental results from 6-time averaged data are presented in Table 1 for D=0.1[h⁻¹] and in Table 2 for D=0.25[h⁻¹] at three various initial concentrations of formaldehyde of 0, 0.02 and 0.04 [%].

Table 1. Data instances at D=0.1[h⁻¹]

<table>
<thead>
<tr>
<th>№</th>
<th>Variables</th>
<th>Formaldehyde, [%]</th>
<th>X [g/dm³]</th>
<th>YX [%]</th>
<th>qS [h⁻¹]</th>
<th>P [%]</th>
<th>RNA [%]</th>
<th>A [h⁻¹]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>0.00</td>
<td>8.14</td>
<td>68.5</td>
<td>6.50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0.02</td>
<td>6.10</td>
<td>58.0</td>
<td>56.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0.04</td>
<td>1.30</td>
<td>1.50</td>
<td>1.80</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>0.00</td>
<td>34.40</td>
<td>47.60</td>
<td>31.60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>0.02</td>
<td>3.70</td>
<td>4.80</td>
<td>1.98</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>0.04</td>
<td>1.14</td>
<td>0.99</td>
<td>1.59</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Data instances at D=0.25[h⁻¹]

<table>
<thead>
<tr>
<th>№</th>
<th>Variables</th>
<th>Formaldehyde, [%]</th>
<th>X [g/dm³]</th>
<th>YX [%]</th>
<th>qS [h⁻¹]</th>
<th>P [%]</th>
<th>RNA [%]</th>
<th>A [h⁻¹]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>0.00</td>
<td>7.84</td>
<td>7.12</td>
<td>3.95</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0.02</td>
<td>63.50</td>
<td>63.00</td>
<td>56.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0.04</td>
<td>3.80</td>
<td>3.90</td>
<td>4.20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>0.00</td>
<td>33.70</td>
<td>36.20</td>
<td>34.20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>0.02</td>
<td>3.34</td>
<td>6.29</td>
<td>1.97</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>0.04</td>
<td>1.02</td>
<td>1.43</td>
<td>1.73</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The experimental microbiological materials and methods are presented in details in [4].

3.2. Results and discussion

In this investigation, the ANN simulations are performed two times for protein-synthesizing ability at dilution rates D=0.1 and D=0.25[h⁻¹]. In the both cases, the ANN-employed has five input nodes corresponding to the five input variables, namely, X, YX, qS, P and RNA (Fig.1). Out of 500 experimental data, first 401 data sets are used to train the network and the last 99 data are used for testing and validation of the NN model. During training the network, SSE has been kept at 0.001 and the frequency of progress displays (in epochs) is set at 50 with maximum epochs of 10000 to train the network.

The principal component analysis (PCA) was used to reduce the dimension of the input vectors [1,4,7,10,13]. This technique is efficient to orthogonalize the components of the input vectors, to order the resulting orthogonal components (principal components) so that those with the largest variation come first and to eliminate those components that contribute the least to the variation in the data set. In our case this procedure is implemented to eliminate those components that contribute less than 2% to the total variation in the data set.

ANN sensor-models corresponding to different numbers of hidden layers and number of neurons in the hidden layers are tried to find the network architecture that provides the least error. An ANN-architecture with 2 hidden layers containing 15 neurons and 16 neurons is found to be optimum for protein-synthesizing ability predictions.

![Fig.1 Four-layered feed-forward ANN for prediction of protein-synthesizing ability (A,[h⁻¹])](image)

The simulation results show that introduction of two hidden layers improve the forecasting performance of ANN compare to the use of single hidden layer. The average SSE is observed to be reached the error goal of 0.001, when two hidden layers are used instead of 0.05 reached with only one hidden layer containing 16 neurons. It is then decided to evaluate the network model using 99 testing data sets. The results show that the predicted
protein synthesizing ability, $A[h^{-1}]$, of ANN results obtained are as close as with training and testing data given to the network. The performance of the trained networks are measured to some extent by the errors on the training, validation and test sets, but often is useful to investigate the network response in more details. One option is to perform a regression analysis between the network response and the corresponding targets. The following Fig.2 and 3 illustrate the network outputs for protein-synthesizing ability prediction at $D=0.1$ and $D=0.25$ $[h^{-1}]$, plotted versus the targets as open circles. The best linear fit is indicated by a dashed line and the perfect fit of the output equal to targets is indicated by the solid line. The correlation coefficient $R=0.996$ for $D=0.1$ and $R=0.98$ for $D=0.25[h^{-1}]$ are results very close to $R=1$ which is the measure of a perfect correlation between outputs and targets. It shows that the received ANN-sensor models for protein-synthesizing ability $(A[h^{-1}])$ at various dilution rates $(D, [h^{-1}])$ have satisfactory fit.

Formaldehyde at higher content leads to the inhibition of yeast strain Candida didensii 74-10 growth and decreasing of protein content as it was shown by microbiological experiments (Table 1 and 2). This inhibition is stronger for biomass $(X)$ growth and protein synthesis $(P)$, respectively for protein-synthesizing ability $(A[h^{-1}])$ under higher dilution rate $D=0.25[h^{-1}]$. These results are important in connection with the investigation of processes of microbial ecology its representative in

4 Conclusions

In the present investigation, artificial neural networks-based software analyzers have been designed and demonstrated to predict the state of continuous yeast growth of strain Candida didensii 74-10, influenced by formaldehyde at its different initial concentrations and various dilution rates. A propagation algorithm using the Levenberg-Marquardt approach for training the network is found to be very effective to generalize and predict the protein synthesizing ability during continuous fermentation. The configuration of the back propagation neural network that gives the best prediction is the one with two hidden layers consisting of 15 neurons and 16 neurons in each layer. ANN predicted results are very close to the experimental values as follows from the statistical analysis, as is shown in Fig.2 and Fig.3. The average SSE is observed to be reached the error goal of 0.001 and the maximum percentage relative errors [$\%$] are found to be close to zero for protein-synthesizing ability prediction in both cases – for $D=0.1$ and $D=0.25[h^{-1}]$ (see Fig. 4).
Acknowledgement This work was supported by the Grant I-1406/2004 allocated by the National Fond for Scientific Research to the Bulgarian Ministry of Education and Science.

References: