ELSEVIER

Journal of Microbiological Methods 51 (2002) 217–226

Journal of Microbiological Methods

www.elsevier.com/locate/jmicmeth

# A probabilistic neural network approach for modeling and classification of bacterial growth/no-growth data☆

M. Hajmeer [a,*], I. Basheer [b]

[a] *Department of Population Health and Reproduction, School of Veterinary Medicine, One Shields Avenue, University of California, Davis, CA 95616, USA*

[b] *Department of Transportation, 5900 Folsom Blvd., Sacramento, CA 95819, USA*

Received 7 January 2002; received in revised form 21 March 2002; accepted 16 April 2002

**Abstract**

In this paper, we propose to use probabilistic neural networks (PNNs) for classification of bacterial growth/no-growth data and modeling the probability of growth. The PNN approach combines both Bayes theorem of conditional probability and Parzen's method for estimating the probability density functions of the random variables. Unlike other neural network training paradigms, PNNs are characterized by high training speed and their ability to produce confidence levels for their classification decision. As a practical application of the proposed approach, PNNs were investigated for their ability in classification of growth/no-growth state of a pathogenic *Escherichia coli* R31 in response to temperature and water activity. A comparison with the most frequently used traditional statistical method based on logistic regression and multilayer feedforward artificial neural network (MFANN) trained by error backpropagation was also carried out. The PNN-based models were found to outperform linear and nonlinear logistic regression and MFANN in both the classification accuracy and ease by which PNN-based models are developed. 

*Keywords:* Bacterial growth; Logistic regression; Modeling; Probabilistic neural networks

## 1. Introduction

Traditionally, logistic statistical regression is used for modeling the probability of growth and no-growth of bacteria. For example, Presser et al. (1998) derived a logistic regression equation for determining the probability of growth of nonpathogenic *Escherichia coli* M23 as function of temperature, pH, lactic acid concentration, and water activity. The proposed form of the regressed equation was based on Bělehrádek type of growth rate model developed by Presser et al. (1997). The formulation of the logistic growth/no-growth regression model followed the procedure suggested by Ratkowsky and Ross (1995) in which the first derivative of an arbitrary growth rate or kinetic model is used as the basis for the probability model. Salter et al. (2000) used a similar approach to derive an equation for predicting the probability of pathogenic *E. coli* R31 in response to temperature and water activity. Also, following the same approach, Tienungoon et al.

☆ The opinions appearing in the article and the accuracy of results are the sole responsibility of the authors.

* Corresponding author. Tel.: +1-530-754-7373; fax: +1-530-752-5845.

*E-mail address:* mnhajmeer@ucdavis.edu (M. Hajmeer).

0167-7012/02/$ - see front matter 
PII: S0167-7012(02)00080-5

(2000) developed equations for probability of growth of *Listeria monocytogenes* as affected by temperature, pH, NaCl, and lactic acid concentration. Using logistic regression, Lopez-Malo et al. (2000) derived regression equation for estimating probability of growth of *Saccharomyces cerevisiae* as affected by pH, water activity, and potassium sorbate concentration. These studies utilized the standard logistic regression procedure (Hosmer and Lemeshow, 1989) to obtain the optimal values of the various free parameters in the regression equation. Unfortunately, the logistic regression method has its own inherent limitations (Marzpan and Stumpf, 1996; Tu, 2000). In this paper, we propose to use a conceptually different approach based on probabilistic neural networks that are not prone to the departure of data from statistical requirements such as normality (Masters, 1995), and to investigate whether such approach offers an advantage over the traditional statistical logistic regression.

Artificial neural networks (ANNs) have recently started to gain much interest in predictive microbiology modeling due to their flexibility and higher accuracy as compared to other traditional modeling techniques such as statistical regression (Basheer and Hajmeer, 2000). ANNs are highly nonlinear computational systems capable of approximating the behavior of poorly understood phenomena and explicitly unmodellable systems, such as the complex microbial growth, by learning from examples, without the need to assume the type of relationship and the degree of nonlinearity between the various independent (explanatory) and dependent (response) variables. In predictive microbiology area alone, many papers have been published following the earliest work by Hajmeer et al. (1996). For example, Hajmeer et al. (1996, 1997, 1998) used multilayer feedforward ANNs (MFANNs) trained by the error backpropagation method to predict the Gompertz growth parameters of *S. cerevisiae*, *Shigella flexneri*, and *E. coli* O157:H7 from various extrinsic biochemical and environmental conditions such as temperature, pH, water activity, and NaCl and $NaNO_2$ concentrations. Geeraerd et al. (1998) also used MFANNs to obtain growth parameters that were subsequently inserted in a dynamic growth equation. Similarly, Lou and Nakai (2001) used the same type of ANNs for predicting the inactivation rate of *E. coli* as affected by temperature, pH, and water activity. Jeyamkondan et al. (2001) used another type of supervised ANNs called general regression neural networks for modeling the generation time and lag phase duration of *Aeromonas hydrophilia*, *S. flexneri*, and *Brochothrix thermosphacta* in response to changes in temperature, pH, and NaCl and $NaNO_2$ concentrations. Rather than just modeling the growth parameters, Hajmeer et al. (2000) proposed a generalized approach using MFANNs for modeling the complete bacterial growth curves as affected by operating and environmental parameters. The generalized approach was applied to model growth curves of *E. coli* O157:H7 and *S. flexneri* as affected by temperature, pH, and NaCl concentration. Simon and Karim (2001) used probabilistic neural networks in conjunction with a modified Gompertz model to classify the lag, logarithmic, and stationary phases in a batch culture of *Bacillus subtilis*. Basheer and Hajmeer (2000) and Najjar et al. (1997) discussed in detail ANNs with regard to their types, their relation to biological networks, their mathematics, and their potential benefits to the area of predictive microbiology along with applications to bacterial growth parameters estimation and growth curve modeling. In almost all applications, ANNs were found to outperform traditional statistical regression methods.

The primary objective of this study is to investigate the use of probabilistic neural networks (PNNs) for the classification of bacterial growth and no-growth states as well as for deriving the probability of growth as affected by changing operating conditions. A PNN combines statistical theory with neural networks and, as such, results in a powerful method for classification where traditional statistical counterparts failed. A discussion of PNNs and their underlying mathematics and statistical basis is presented in the following section. The proposed approach is then applied to data pertaining to growth of a non-typeable Shiga toxin-producing pathogenic strain of *E. coli* (R31) to demonstrate the use of PNNs and the development of probabilistic models. Finally, the PNN-based models derived are compared to linear and nonlinear logistic regression models as well as MFANNs developed from the same data, and the advantages and disadvantages of the proposed approach are discussed.

## 2. Probabilistic neural networks (PNNs)

### 2.1. Basics

The PNN is a Bayes–Parzen classifier (Masters, 1995). The foundation of the approach is well known decades ago (1960s); however, the method was not of a widespread use because of the lack of sufficient computation power until recently. The PNN was first introduced by Specht (1990), who showed how the Bayes–Parzen classifier could be broken up into a large number of simple processes implemented in a multilayer neural network each of which could be run independently in parallel. Because the PNN is primarily based on Bayes–Parzen classification, it is of interest to discuss briefly both Bayes theorem for conditional probability and Parzen's method for estimating probability density function of random variables.

To understand Bayes' theorem, consider a sample $\boldsymbol{x}=[x_1,x_2,\ldots, x_p]$ taken from a collection of samples belonging to a number of distinct populations (1,2,..., $k$,..., $K$). Assuming that the (prior) probability that a sample belongs to the $k$th population (class) is $h_k$, the cost associated with misclassifying that sample is $c_k$, and that the true probability density function of all populations $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$,..., $f_k(\boldsymbol{x})$,..., $f_K(\boldsymbol{x})$ are known, Bayes theorem classifies an unknown sample into the $i$th population if

$$h_i c_i f_i(\boldsymbol{x}) > h_j c_j f_j(\boldsymbol{x}) \tag{1}$$

for all populations $j \neq i$. The density function $f_k(\boldsymbol{x})$ corresponds to the concentration of class $k$ examples around the unknown example. As seen from Eq. (1), Bayes' theorem favors a class that has high density in the vicinity of the unknown sample, or if the cost of misclassification or prior probability is high.

The biggest problem with the Bayes' classification approach lies in the fact that the probability density function $f_k(\boldsymbol{x})$ is not usually known. In nearly all standard statistical classification algorithms, some knowledge regarding the underlying distribution of the population of all random variables used in classification should be known or reasonably assumed. Most often, normal (Gaussian) distribution is assumed; however, the assumption of normality can not always be safely justified. When the distribution is not known (which is often the case) and the true distribution deviates considerably from the assumed one, the traditional statistical methods normally run into major classification problems resulting in high misclassification rate. There is a need to derive an estimate of $f_k(\boldsymbol{x})$, from the training set composed of the training example, rather than just assume normal distribution. The resulting distribution will be a multivariate probability density function (PDF) that combines all the explanatory random variables.

To derive such distribution estimator from a set of training examples, the Parzen's (1962) method is usually used. The univariate case of PDF was proposed by Parzen (1962) and then was extended to the multivariate case by Cacoullos (1966). The multivariate PDF estimator, $g(\boldsymbol{x})$, may be expressed as:

$$g(x_1,x_2,\ldots,x_p) = \frac{1}{n\sigma_1\sigma_2\ldots\sigma_p}\sum_{i=1}^{n} W \left(\frac{x_1 - x_{1,i}}{\sigma_1}, \frac{x_2 - x_{2,i}}{\sigma_2}, \ldots, \frac{x_p - x_{p,i}}{\sigma_p}\right) \tag{2}$$

where $\sigma_1$, $\sigma_2$,..., and $\sigma_p$ are the smoothing parameters representing standard deviation (also called window or kernel width) around the mean of $p$ random variables $x_1$, $x_2$,..., $x_p$, $W$ is a weighting function to be selected with specific characteristics (Specht, 1990; Masters, 1995), and $n$ is the total number of training examples. If all smoothing parameters are assumed equal (i.e., $\sigma_1=\sigma_2=\sigma_3=\cdots=\sigma_p=\sigma$), and a bell-shaped Gaussian function is used for $W$, a reduced form of Eq. (2) results:

$$g(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2} n\sigma^p}\sum_{i=1}^{n} \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{2\sigma^2}\right) \tag{3}$$

where $\boldsymbol{x}$ is the vector of random variables (explanatory variables), and $\boldsymbol{x}_i$ is the $i$th training vector. Eq. (3) represents the average of the multivariate distributions where each distribution is centered at one distinct training example. It is worth mentioning that the assumption of a Gaussian weighting function does not imply that the overall PDF will be Gaussian (normal); however, other weighting functions such as the reciprocal function $[W(d)=1/1+d^2]$ may be used (Masters, 1995). As the sample size, $n$, increases, the Parzen's PDF estimator asymptotically approaches the true underlying density function.

### 2.2. Network operation

Consider the simple network architecture shown in Fig. 1 with four input nodes ($p=4$) in the input layer, two population classes (class 1 and class 2), five training examples belonging to class 1 ($n_1=5$), and three examples in class 2 ($n_2=3$). The pattern layer (see Fig. 1) is designed to contain one neuron (node) for each training case available and the neurons are split into the two classes. The summation layer contains one neuron for each class. The output layer contains one neuron that operates a trivial threshold discrimination; it simply retains the maximum of the two summation neurons. The PNN executes a training case by first presenting it to all pattern layer neurons. Each neuron in the pattern layer computes a distance measure between the presented input vector and the training example represented by that pattern neuron. The PNN then subjects this distance measure to the Parzen window (weighting function, $W$) and yields the activation of each neuron in the pattern layer. Subsequently, the activation from each class is fed to the corresponding summation layer neuron, which adds all the results in a particular class together. The activation of each summation neuron is executed by applying the remaining part of the Parzen's estimator equation (e.g., the constant multiplier in Eq. (3)) to obtain the estimated probability density function value of population of a particular class. If the misclassification cost and prior probabilities are equal between the two classes, and the classes are mutually exclusive (i.e., no case can be classified into more than one class) and exhaustive (i.e., the training set covers all classes fairly), the activation of the summation neurons will be equal to the posterior probability of each class. The results from the two summation neurons are then compared and the largest is fed forward to the output neuron to yield the computed class and the probability that this example will belong to that class.

The most important parameter that needs to be determined to obtain an optimal PNN is the smoothing parameters ($\sigma_1$, $\sigma_2$,..., and $\sigma_p$) of the random variables. A straightforward procedure involves selecting an arbitrary values of $\sigma$'s, training the network, and testing it on a test (validation) set of examples. This procedure is repeated for other $\sigma$'s and the set of $\sigma$'s that produces the least misclassification rate (percentage of examples that were misclassified) is chosen. A better and more efficient procedure for searching for the optimal smoothing

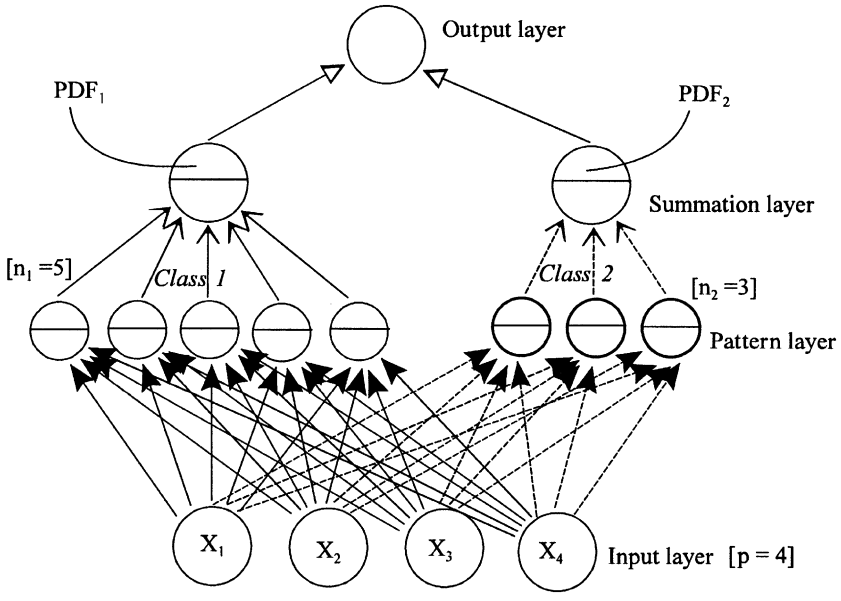


Fig. 1. A simple probabilistic neural network (PNN) with four input variables, two classes, and eight training examples (five belonging to class 1 and three to class 2).

parameter of random variables and classes is proposed by Masters (1995). This procedure prevents any bias in the network to the correctly classified examples and, thus, will be followed in this study. Other details on the mathematics as well as advanced variations of PNNs is given in Specht (1990) and Masters (1995).

## 3. Application

### 3.1. Data and methodology

In order to demonstrate the use of PNNs, we used data from Salter et al. (2000) on growth/no-growth of an *E. coli* strain R31 under effect of temperature ($T$) and water activity ($a_w$). Salter et al. (2000) used the data to derive a nonlinear logistic model for classification of growth/no-growth data and identification of the probabilistic interface (boundary) between growth and no-growth states. The data consisted of experimental testing of a large number of combinations (total of 179) of $T$ and $a_w$ in the range $T=7.7-37.0$ °C and $a_w=0.943-0.987$. All samples were observed daily, and a sample was scored positive if it showed an increase in turbidity or deposit in the base of the tube. If after 50 days there was neither turbidity nor deposit, a loopful of culture was streaked onto plate count agar to determine if any growth is present. The details of the experimental program and materials and testing methods are available in Salter et al. (2000). For any ($T$, $a_w$) combination, growth was recorded as 1 if it occurred and 0 if it did not. In the database, there were 99 cases of growth and 80 cases of no-growth. The objective of the modeling phase in this application was to develop a predictive probabilistic model that enables identification of any ($T$, $a_w$) as belonging to either one of the two classes: growth or no-growth. In addition, the posterior probability of growth ($P_1$) is determined directly from the model. The arbitrary function $f$ by which the probability of growth can be determined as function of the explanatory variable $T$ and $a_w$ [i.e., $P_1=f(T, a_w)$] will be replaced by a predictive PNN.

A PNN computer software (Masters, 1995) was used to develop the probabilistic model. Four different PNN training schemes were examined: (1) a Gaussian kernel (i.e., $W$) and one smoothing parameter $\sigma$ for all input random variables (i.e., $T$ and $a_w$), (2) a reciprocal kernel and one $\sigma$ for all input variables, (3) a Gaussian kernel and a different $\sigma$ for each input random variable, and (4) a Gaussian kernel with a separate $\sigma$ for each input random variable and classes (i.e., growth and no-growth).

The prior probabilities of growth ($p_1$) and no-growth ($p_0$) were assumed to be equal to the respective proportion of growth and no-growth examples in the entire database; namely $p_1=0.55$ and $p_0=1-p_1=0.45$. With exclusive and exhaustive classes, the posterior probability of growth ($P_1$) for a given input vector $\boldsymbol{x}$ is calculated from Bayes' theorem according to:

$$P_1(\boldsymbol{x}) = \frac{p_1 g_1(\boldsymbol{x})}{p_1 g_1(\boldsymbol{x}) + p_0 g_0(\boldsymbol{x})} \tag{4}$$

In Eq. (4), $g_1(\boldsymbol{x})$ and $g_0(\boldsymbol{x})$ are the computed Parzen's probability density functions for growth and no-growth classes (Eq. (3)).

The methodology for computing the accuracy of the developed PNN and the method to compare the various PNNs using the four different training schemes were based on computing the confusion matrix. Because the PNN output (probability of event) is essentially continuous and the developed PNN involves classification, the probabilities were transformed into crisp values of 1 for growth and 0 for no-growth using 0.5 as the threshold probability between the two states. This will allow deriving the confusion matrix (C-matrix) expressed as:

$$\text{C}-\text{matrix} = \begin{pmatrix} \text{a} & \text{b} \\ \text{c} & \text{d} \end{pmatrix} = \begin{pmatrix} 0 \rightarrow 0 & 0 \rightarrow 1 \\ 1 \rightarrow 0 & 1 \rightarrow 1 \end{pmatrix} \tag{5}$$

where $a$, $b$, $c$, and $d$ are the matrix elements representing the number of cases as classified by the PNN. $a$ is the number of nonevent (i.e., no-growth) cases, denoted by 0, that were classified by the PNN as nonevent and referred to in Eq. (5) as $0 \rightarrow 0$, $b$ is the number of nonevent cases classified by PNN as event (i.e., growth) cases denoted by 1 and referred to in Eq. (5) as $0 \rightarrow 1$ (i.e., number of false alarm cases), $c$ is the number of event cases classified by the PNN as nonevent cases and referred to as $1 \rightarrow 0$, and $d$ is the number of event cases classified by the PNN as event cases and referred to as $1 \rightarrow 1$. The sum of $a$ and $d$ is the total number of cases that were classified correctly,

and $(b+c)$ is the total number of misclassified cases. Two performance measures obtained from the C-matrix were used in gauging the quality of the network's prediction. These were FC=$(a+d)/N$ and FAR = $b/(a+b)$, where FC is fraction correct representing the fraction of cases classified correctly, FAR is the false alarm rate (false positive) representing the fraction of negative cases classified as positive by the model, and $N$ is the total number of cases ($N=a+b+c+d$). These parameters were obtained for both the training and test sets for each PNN developed using the four schemes. A perfect classifier network is one with $b=c=0$, implying FC = 100% and FAR = 0%.

The 179 ($T$, $a_w$) growth/no-growth examples were randomly split into 143 examples (73 growth and 70 no-growth) to be used for training the PNN, and the remaining 36 examples (26 for growth and 10 for no-growth), never used in PNN development, for testing the validity of the trained PNN. The testing data comprised about 20% of the entire database. A typical PNN for this model is shown in Fig. 2 with two neurons in the input layer, 143 neurons in the pattern layer distributed into 73 neurons for class 1 (growth) and 70 neurons for class 2 (no-growth), two summation neurons (one for each class), and one output neuron that gives the posterior probability of growth, $P_1$.

### 3.2. Results

Both training and execution of the developed networks with architecture shown in Fig. 2, and using the four training schemes described earlier were almost instantaneous. Table 1 summarizes the elements of confusion matrices and the performance measures of the training, validations (testing) sets, and the full database for the four training schemes used. As seen in Table 1, all PNN training schemes yielded highly accurate PNNs, with training scheme 2 (reciprocal kernel and an equal smoothing parameter for both $T$ and $a_w$) resulting in the best classification network at an optimal $\sigma=0.0009$. This PNN, referred to herein as PNN2, proved to be a perfect classifier with accuracy measures for the training data as FC = 100% and FAR = 0.0%, and an excellent accuracy for the test data with FC = 94.44% and FAR = 0.0%. Using training and test data combined, the accuracy of PNN2 was still excellent at FC = 98.88% and FAR = 0.0%. It is

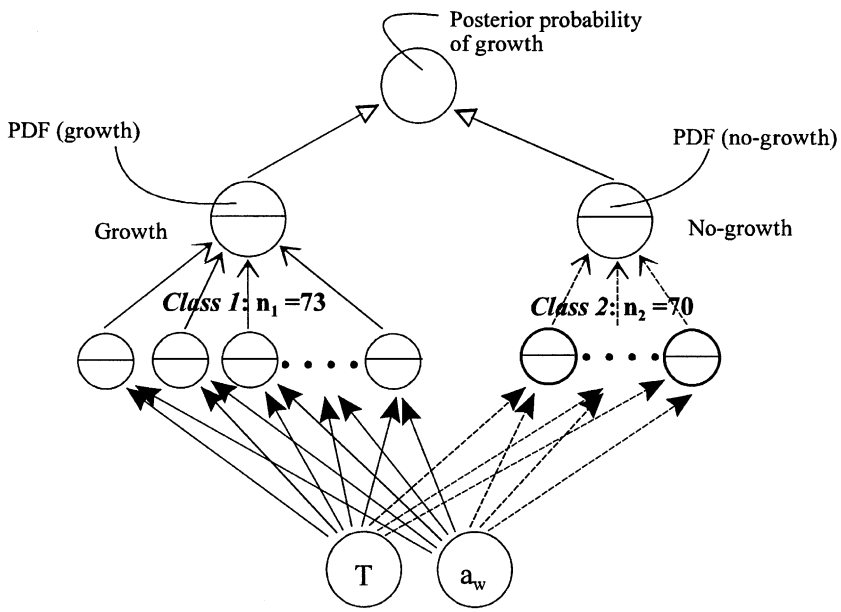


Fig. 2. The architecture of *E. coli* R31 growth classification probabilistic neural network (PNN) with two input variables ($T$, $a_w$), two classes (growth, no-growth), and 143 neurons in the pattern layer representing training examples (73 growth cases and 70 no-growth cases).

Table 1
Comparison of accuracy of three approaches using confusion matrix performance measures

| Model | Training scheme | Training data (143 examples) | | | Validation data (36 examples) | | | Combined data (179 examples) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | C-matrix coefficients {*a*, *b*, *c*, *d*} | FC (%) | FAR (%) | C-matrix coefficients {*a*, *b*, *c*, *d*} | FC (%) | FAR (%) | C-matrix coefficients {*a*, *b*, *c*, *d*} | FC (%) | FAR (%) |
| PNN | 1 | {68, 2, 0, 73} | 98.60 | 2.86 | {9, 1, 1, 25} | 94.44 | 10.00 | {77, 3, 1, 98} | 97.77 | 3.75 |
| | 2 | {70, 0, 73, 0} | 100.0 | 0.00 | {10, 0, 2, 24} | 94.44 | 0.00 | {80, 0, 2, 97} | 98.88 | 0.00 |
| | 3 | {68, 2, 0, 73} | 98.60 | 2.86 | {9, 1, 1, 25} | 94.44 | 10.00 | {77, 3, 1, 98} | 97.77 | 3.75 |
| | 4 | {69, 1, 1, 72} | 98.60 | 1.43 | {10, 0, 7, 19} | 80.56 | 0.00 | {79, 1, 8, 91} | 94.97 | 1.25 |
| NLLR (Eq. (9)) | | {61, 9, 5, 68} | 90.21 | 12.86 | {10, 0, 3, 23} | 91.67 | 0.00 | {71, 9, 8, 91} | 90.50 | 11.25 |
| LLR (Eq. (8)) | | {54, 16, 19, 54} | 75.52 | 22.86 | {8, 2, 6, 20} | 77.78 | 20.00 | {62, 18, 25, 74} | 75.98 | 22.50 |
| MFANN (Eq. (10)) | | {65, 5, 5, 68} | 93.00 | 7.14 | {10, 0, 2, 24} | 94.44 | 0.00 | {75, 5, 7, 92} | 93.30 | 6.25 |

C-matrix: confusion matrix; FC: fraction correct; FAR: false alarm rate.
PNN: probabilistic neural network; MFANN: multilayer feedforward artificial neural network.
NLLR: nonlinear logistic regression; LLR: linear logistic regression.

interesting to notice that the false alarm rate (proportion of no-growth cases being misclassified into growth ones) was zero on both training and test data. The Probability of Detecting (POD) an event (growth) for the PNN2 model is also computed from the confusion matrix as $\mathrm{POD} = d/(c+d) = 97.98\%$, indicating the high accuracy and reliability of the model to identify critical (i.e., growth) cases.

As a comparison of the PNN approach with traditional statistics, we selected the logistic regression modeling approach (Hosmer and Lemeshow, 1989) frequently used by many researchers in growth/no-growth studies (e.g., Ratkowsky and Ross, 1995; Presser et al., 1998; Bolton and Frank, 1999; Salter et al., 2000; Lopez-Malo et al., 2000), and the more frequently used type of neural networks, the MFANNs trained by error backpropagation method (Masters, 1995). As for logistic regression, both linear and nonlinear logistic models were examined. Statistically, a logistic regression model relates the probability of occurrence of an event conditional on a vector of explanatory variables. We developed the linear and nonlinear models in this study for comparison with the PNN approach. Similar to PNN development method, both logistic models were developed by using the 143 training examples, while the remaining 36 examples for validation of the models. The linear model developed is expressed as:

$$\mathrm{logit}(P_1) = \beta_0 + \beta_1 T + \beta_2 a_w \tag{6}$$

where $\mathrm{logit}(P_1) = \ln(P_1/1 - P_1)$, and $\beta_0$, $\beta_1$, and $\beta_2$ are the model free parameters determined by the logistic regression procedure that maximizes some likelihood function (Hosmer and Lemeshow, 1989). The probability of growth is computed from

$$P_1 = \frac{1}{1 + e^{-\mathrm{logit}(P_1)}} \tag{7}$$

The resulting logistic regression equation on the *E. coli* training data was:

$$\mathrm{logit}(P_1) = -190.45 + 0.1852T + 195.05a_w \tag{8}$$

with a standard error of 0.0354 and 34.605 for $\beta_1$ and $\beta_2$, respectively.

The second model used in this study was the nonlinear logistic model developed by following Salter et al. (2000) model; however, unlike Salter et al. (2000) model which used the entire database, only the 143 data sets were used in developing the model, and the remaining data sets were kept aside for model validation. The resulting nonlinear logistic regression equation was:

$$\begin{aligned}\mathrm{logit}(P_1) = {} & 6.949 + 8.441\ \ln(a_w - 0.943) \\ & + 11.894\ \ln(T - 3.411) \\ & + 385.30\ \ln(1 - \exp(0.3(T - 49.23)))\end{aligned} \tag{9}$$

The standard errors of the free parameters in Eq. (9) are 1.935, 2.713, and 108.886 for the 8.441, 11.894, and 385.30 coefficients, respectively. It is worth to

mention that nonlinear logistic models like Eq. (9) are hard to develop because there is no rigorous procedure that will enable the modeler to choose the shape of the nonlinear functions, the degree of nonlinearity, or the cross-interaction terms between the independent variables. The above two logistic models (Eqs. (8) and (9)) were run on the 143 and 36 training and validation ($T$, $a_w$) examples and $P_1$ computed.

As for the MFANN, the optimum network derived had three layers of nodes, namely, two inputs ($T$, $a_w$) in the input layer, two hidden nodes in the hidden layer, and one output node in the output layer representing the continuous activation value between 0 for no-growth and 1 for growth. The developed MFANN architecture with the optimal weights and thresholds is shown in Fig. 3. Because of the simple architecture of the derived MFANN, it was possible to derive a relatively simple equation to compute the output activation, $\hat{y}$, as function of temperature and water activity. This equation is expressed as:

$$\hat{y} = \left(1 + \exp\left[\frac{19.281}{1+\exp(-4.154T^* + 17.868a_w^* - 4.883)} + \frac{18.381}{1+\exp(7.021T^* + 6.408a_w^* - 5.281)} - 7.037\right]\right)^{-1} \quad (10)$$

where $T^*$ and $a_w^*$ are positive normalized (between 0.0 and 1.0) temperature and water activity computed from $T^*=0.029T-0.143$, where $T$ is in °C, and $a_w^*=10a_w-9.0$.

The computed $P_1$ from logistic equations (Eqs. (8) and (9)) and $\hat{y}$ from MFANN (Eq. (10)) were transformed into dichotomous variable representing whether growth has (1) or has not (0) occurred. A threshold of 0.5 for $P_1$ was used (i.e., growth if $P_1 \geq 0.5$, and no-growth if $P_1 < 0.5$). Then, the confusion matrices were formed and the associated performance measures were computed and summarized as shown in Table 1. It is seen in Table 1 that the PNN approach (regardless of the training scheme used) outperforms the linear and nonlinear logistic regression as well as the multilayer feedforward ANN in its prediction accuracy of the growth and no-growth states. The nonlinear logistic model also outperformed the linear model. Based on entire data, the linear model misclassified a total of 43 cases, the nonlinear model misclassified a total of 17 cases, the MFANN model misclassified 12 cases, and the PNN-based model misclassified only two cases. The best PNN model (PNN2) has the lowest false alarm rate of 0.0% compared to 22.50% for the linear model, 11.25% for the nonlinear model, and 6.25% for the MFANN model. A low false alarm rate is a major attribute when dealing with the economics of process under plant settings.

The PNN2 computed posterior probability outputs ($P_1$) for the 179 ($T$, $a_w$) examples were plotted against the corresponding $T$ and $a_w$, as displayed in the three-dimensional surface depicted in Fig. 4. As seen in Fig. 4, there is an abrupt increase in probability of growth from 0 to 1 at high $a_w$ (left side of the 3-D block). This is to be expected as both high temperature and $a_w$ produce a favorable condition for the bacteria to grow.

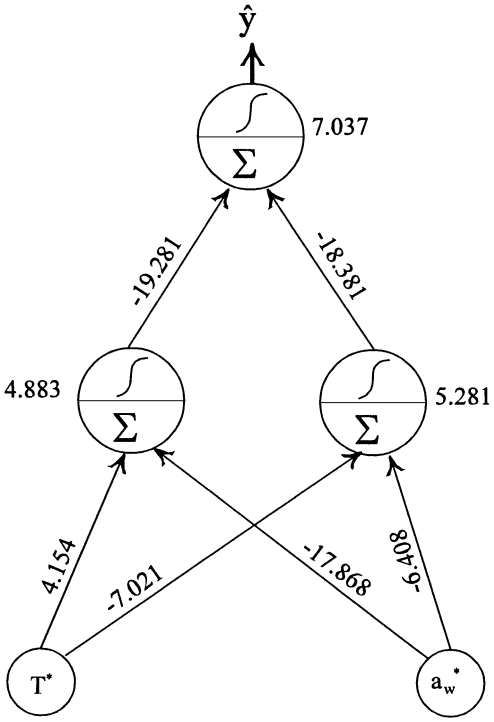


Fig. 3. The architecture of *E. coli* R31 growth classification multilayer feedforward ANN (MFANN) with two normalized input variables ($T^*$, $a_w^*$), two hidden nodes, and one output node representing the continuous activation with 1 for growth and 0 for no-growth ($T^*=0.029T-0.143$, where $T$ is in °C, and $a_w^*=10a_w-9.0$).

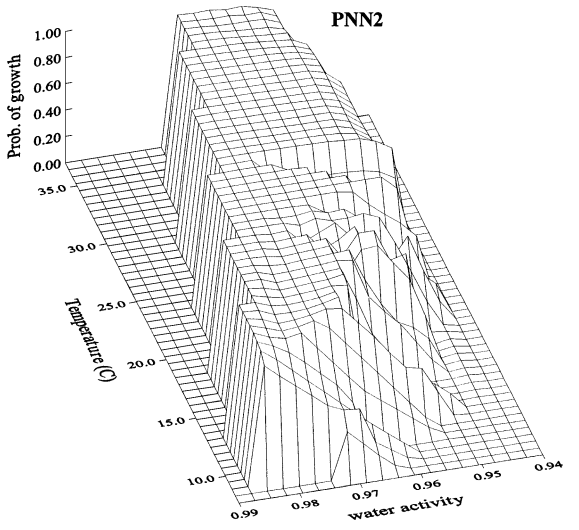


Fig. 4. Variation of posterior probability of growth of *E. coli* R31 in relation to temperature and water activity.

In contrast, at low $a_w$ (right side of the block), a gradual increase in probability of growth could be seen especially at low temperature. As the temperature gets higher, the system gradually moves to the abrupt behavior until $T=35$ °C is reached where it then experiences and maintains the abrupt trend. For any desired level of probability (confidence), Fig. 4 can also define the boundary between growth and no-growth.

## 4. Concluding remarks

In this paper, we proposed a new approach based on probabilistic neural networks (PNNs) for classifying the bacterial growth and no-growth state. The approach combines Bayes' conditional probability theorem and Parzen's method for density function estimation. The prime advantage of the PNN algorithm is that its ability as a statistical classifier is not restricted by the departure of the distribution of random variables from multivariate normality or the equal dispersion of data in each class. Additionally, PNNs are resilient to multimodal distributions where most nonparameteric statistical classification methods fail (Masters, 1995). Unlike other types of ANNs, the proposed approach is able to directly generate correct posterior event (e.g., bacterial growth) probabilities by virtue of PNNs' mathematical basis that does not impose any conditions or make assumptions as to the nature of the distribution of random variables. Rather, unlike most traditional statistical methods (e.g., logistic regression), the probability density functions (regardless of their complexity) are derived directly from the data and, thus, are better representations of the true distributions.

The PNN is a Bayesian classifier implemented in parallel, and thus, it is not trained iteratively. Thus, compared to other multilayer feedforward neural networks training paradigms, PNNs train orders of magnitude faster, and as such, it is easier to derive and involves less parameters that are essential for optimizing network performance. Essentially, training a PNN consists of copying training examples into the network, and as such, it is almost instantaneous. Compared to nonlinear logistic regression, PNNs are easier to build where one starts with no knowledge as to the best combination of the parameters or the shape and degree of nonlinearity needed to produce a reasonably accurate logistic regression model. Other advantages of PNNs over standard neural networks and traditional statistical techniques include their robustness to noisy data (with outliers), which can harm many types of ANNs and can severely hamper most traditional statistical methods. Beside accuracy, the fact that the PNN output is probabilistic makes PNNs the best choice for classification and probabilistic modeling among many other types of MFANNs.

The only disadvantage of PNNs relates actually to the availability of a good computer with a large memory and high processing speed. Basically, the training vectors are transformed into connection weight vectors, and thus, the weight matrix can become quite large when the training set size is quite large. Thus, the PNN can be computer memory-consuming and slow to execute when used in real-time systems. However, with most average computer speeds available today, both training and execution occur rapidly. Thus, a PNN-based model can be fast and efficiently re-trainable classifier (when additional data become available) when implemented in hardware systems.

As an application utilizing *E. coli* R31 data, the PNN approach was compared to the frequently used linear and nonlinear logistic regression as well as the multilayer feedforward ANN trained by error back-

propagation method. The PNN-based models were found to outperform the other examined methods in their classification accuracy.

## References


Basheer, I.A., Hajmeer, M.N., 2000. Artificial neural networks: fundamentals, computation, design and application. Journal of Microbiological Methods 43, 3–31.

Bolton, L.F., Frank, J.F., 1999. Defining the growth/no-growth interface for *Listeria monocytogenes* in Mexican-style cheese based on salt, pH, and moisture content. Journal of Food Protection 62, 601–609.

Cacoullos, T., 1966. Estimation of multivariate density. Annals of the Institute of Statistical Mathematics 18 (2), 179–189.

Geeraerd, A.H., Herremans, C.H., Cenens, C., Van Impe, J.F., 1998. Application of artificial neural networks as a non-linear modular modeling technique to describe bacterial growth in chilled food products. International Journal of Food Microbiology 44, 49–68.

Hajmeer, M.N., Basheer, I.A., Najjar, Y.M., 1996. The growth of *Escherichia coli* O157:H7—a backpropagation neural network approach. In: Dagli, C.H., et al. (Eds.), Intelligent Engineering Systems Through Artificial Neural Networks, Proc. ANNIE'96, vol. 6. ASME Press, New York, pp. 635–640.

Hajmeer, M.N., Basheer, I.A., Najjar, Y.M., 1997. Computational neural networks for predictive microbiology: II. Application. International Journal of Food Microbiology 34, 51–66.

Hajmeer, M.N., Basheer, I.A., Fung, D.Y.C., Marsden, J.L., 1998. A nonlinear response surface model based on artificial neural networks for growth of *Saccharomyces cerevisiae*. Journal of Rapid Methods and Automation in Microbiology 6, 103–118.

Hajmeer, M.N., Basheer, I.A., Marsden, J.L., Fung, D.Y.C., 2000. New approach for modeling generalized microbial growth curves using artificial neural networks. Journal of Rapid Methods and Automation in Microbiology 8 (4), 265–284.

Hosmer, D.W., Lemeshow, S., 1989. Applied Logistic Regression. Wiley, New York.

Jeyamkondan, S., Jayas, D.S., Holley, R.A., 2001. Microbial growth modelling with artificial neural networks. International Journal of Food Microbiology 64, 343–354.

Lopez-Malo, A., Guerrero, S., Alzamora, S.M., 2000. Probabilistic modeling of *Saccharomyces cerevisiae* inhibition under the effects of water activity, pH, and potassium sorbate concentration. Journal of Food Protection 63 (1), 91–95.

Lou, W., Nakai, S., 2001. Application of artificial neural networks for predicting the thermal inactivation of bacteria: a combined effect of temperature, pH, and water activity. Food Research International 34, 573–579.

Marzpan, C., Stumpf, G., 1996. A neural network for tornado prediction based on Doppler radar-derived attributes. Journal of Applied Meteorology 35, 617–626.

Masters, T., 1995. Advanced Algorithms for Neural Networks. Wiley, New York.

Najjar, Y.M., Basheer, I.A., Hajmeer, M.N., 1997. Computational neural networks for predictive microbiology: I. Methodology. International Journal of Food Microbiology 34, 27–49.

Parzen, E., 1962. On estimation of a probability density function and mode. Annals of Mathematical Statistics 36, 1065–1076.

Presser, K.A., Ratkowsky, D.A., Ross, T., 1997. Modelling the growth rate of *Escherichia coli* as a function of pH and lactic acid concentration. Applied and Environmental Microbiology 63, 2355–2360.

Presser, K.A., Ross, T., Ratkowsky, D.A., 1998. Modelling the growth (growth/no growth interface) of *Escherichia coli* as function of temperature, pH, lactic acid concentration, and water activity. Applied and Environmental Microbiology 64 (5), 1773–1779.

Ratkowsky, D.A., Ross, T., 1995. Modelling the bacterial growth/no growth interface. Letters in Applied Microbiology 20, 29–33.

Salter, M.A., Ratkowsky, D.A., Ross, T., McMeekin, T.A., 2000. Modelling the combined temperature and salt (NaCl) limits for growth of a pathogenic *Escherichia coli* strain using nonlinear logistic regression. International Journal of Food Microbiology 61, 159–167.

Simon, L., Karim, M.N., 2001. Probabilistic neural networks using Bayesian decision strategies and a modified Gompertz model for growth phase classification in the batch culture of *Bacillus subtilis*. Biochemical Engineering Journal 7 (1), 41–48.

Specht, D.F., 1990. Probabilistic neural networks. Neural Networks 3, 109–118.

Tienungoon, S., Ratkowsky, D.A., McMeekin, T.A., Ross, T., 2000. Growth limits of *Listeria monocytogenes* as a function of temperature, pH, NaCl, and lactic acid. Applied and Environmental Microbiology 66 (11), 4979–4987.

Tu, J.V., 2000. Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. Journal of Clinical Epidemiology 49 (11), 1225–1231.