Bacterial Growth Classification with Support Vector Machines: A Comparative Study

*Emad A. El-Sebakhy*and Kanaan A. Faisal*

Information and Computer Science Department
College of Computer Science and Engineering
King Fahd University of Petroleum & Minerals
Dhahran 31261, Saudi Arabia
dodi05,faisal@ccse.kfupm.edu.sa

Abstract

In this paper, we propose to use support vector machines for classification of bacterial growth and non growth database and modeling the probability of growth. Unlike artificial neural networks paradigms, support vector machines use the kernel functions and support vectors with maximum margin, which allows a better performance. As a practical application of the new approach, support vector machines were investigated for their quality and accuracy 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 common used statistics, machine learning, and data mining schemes was carried out. The results shows that support vector machines classifier based on the Gaussian RBF Kernel was found to do better than most of logistic regression, K-nearest neighbor, probabilistic networks, and multilayer perceptron classifiers.

Keywords*: Bacterial growth; Logistic regression; Support Vector Machines; K-nearest neighbors,neural networks.*

1. Introduction

The practical applications Problems involving classification of outcome into one or more classes are plentiful in both bioinformatics and microarray technology. For instance, the classification of environmental factors, such as, temperature (T), water activity (Aw), pH as leading to growth or no-growth of a given microorganism is one but most common research problem in predictive microbiology [9, 18, 23, 20]. Generally, the existing classifiers attempt to answer questions like whether T of $151C$, Aw of 0.98, and pH of 4.5 would lead to growth of Escherichia coli $O157 : H7$, or what the probability of growth of this pathogen would be under such conditions.

Linear and nonlinear logistic regression has been used extensively for classification and estimation of the probability of bacterial growth under a set of conditions, see [11] for details. The authors in [18] derived a logistic regression equation for determining the probability of growth of non-pathogenic E. coli M23 as function of T, pH, Aw, and lactic acid concentration. [20] also used logistic regression to derive a nonlinear equation for estimating the probability of growth of pathogenic E. coli R31 in response to T and Aw, and [23] developed equations for growth of Listeria monocytogenes as affected by T, pH, $NaCl$, and lactic acid concentration. [14] developed a logistic regression equation for estimating probability of growth of Saccharomyces cerevisiae as affected by pH, Aw, and potassium sorbate concentration.

Recently, Artificial neural networks (ANNs) are powerful tools for solving a wide range of practical applications in both forecasting and classification problems [5, 7, 12, 17, 19]. ANNs are emerging powerful highly nonlinear computational techniques that have gained increasing popularity in predictive microbiology due to their flexibility and high accuracy in modeling complex data [2, 1]. Data pertaining to growth of pathogenic E. coli R31 as affected by temperature and water activity will be used to derive the various classifiers. In predictive microbiology, ANNs have been used for modeling the complex timedependent bacterial growth [2, 10, 21] or for predicting growth parameters such as lag time and exponential growth rate [15, 12] as affected by extrinsic biochemical and environmental conditions. The au-

*Corresponding author: dodi05@ccse.kfupm.edu.sa; fax: +966-3-860-2174.

thors in [2] proposed the feedforward neural networks based on the backpropagation minimization criterion to the area of predictive microbiology, along with applications to the estimation of bacterial growth parameters and growth curve modeling and they found that feedforward neural networks outperform the most traditional statistical classification approaches.

The primary objective of this study is to investigate the use of support vector machines (SVMs) classifier based on the theory of risk minimization and marginal distribution with the Gaussian radial basis (RBF) Kernel for the classification of bacterial growth and non-growth states as well as for investigate the probability of growth as affected by changing operating conditions. A discussion of SVMs and their underlying mathematics and statistical basis with the learning steps for both binary nonlinear separable and multicategory of the one one-against-all SVMs classifier is presented in Section 2. 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 SVMs and the development of models. Finally, the SVMs-based models derived are compared to linear and nonlinear logistic regression (LR), K-nearest neighbor, feedforward neural networks (FFNs) as well as probabilistic networks (PN) classifiers developed from the same data, and the advantages and disadvantages of the proposed approach are discussed. Finally, we learn the model parameters as it is shown in details in Sections 2 and 3.

The paper is organized as follows: Section 2 is a brief background of neuron-fuzzy systems and their use. Section 3 describes the implementation and comparative studies using the growth/no-growth state of a pathogenic Escherichia coli R31 in response to temperature and water activity database used for building and testing the support vector machines classifier. The conclusions and recommendations are presented in Section 4.

2. Support Vector Machines

Without loss of generality, the pattern classification problem can be restricted to consideration of the binary category, [8]. Consider $D = \{(\mathbf{x}_i, \mathbf{y}_i);\ \mathbf{x}_i \in \Re^p;\ \mathbf{y}_i \in \{-1, +1\};$ for all $i = 1, \ldots, n\}$ is the training set of vectors, where $\mathbf{x}_i$ is the i^{th} input vector. Here, we discuss the binary SVMs classifier for non-linear separable data and one-against-all support vector machines classifier. As it is known in both machine learning and data mining literatures, the idea of selecting the decision making function of the support vector machines is based on the risk function, which is the expected value of the loss due to classification given by the risk functional$R(\lambda)$. Generally, for $\lambda \in \Lambda$ and $n > h$, a typical uniform VC bound, which holds with probability $1-\eta$, has the form $R(\lambda) \leq R_{emp}(\lambda) + \sqrt{\frac{h(log(\frac{2n}{h}+1)-log(\frac{\eta}{4})}{n}}$, see [24, 25] for more details. The parameter h is called the Vapnik-Chervonenkis dimension of a set of functions and it describes the capacity of a set of functions to represent the data set.

2.1 Non-Linear Separable Binary Data

The basic idea in support vector machine is to map the inputs $\mathbf{x}$ to vectors $\varphi(\mathbf{x})$ in some high-dimensional feature space [24, 25]. Figure 1(a) and (b) showed an example of binary and multi-class categories in two dimensional space. The new decision hyper-planes have the form: $\omega^T \varphi(\mathbf{x}) + b = 0$, among all of it, the SVM solution is chosen as the one with the largest margin, *i.e.*, minimizes the margin, $\frac{\|\omega\|^2}{2}$, subject to the constraints $\mathbf{y}_i \left(\omega^T \varphi(\mathbf{x}) + b\right) \geq 1$, for $i = 1, \ldots, n$, where $\varphi : \Re^n \rightarrow \Re^m$ is the feature map, where the data points become linearly separable by a hyperplane defined by the pair $\omega \in \Re^m$; and $b \in \Re$. Figure 1(c) showed an example of the feature map from two dimensional space to a three-dimensional one: $\varphi(x_1, x_2) = (x_1^2, x_2^2, \sqrt{2} x_1\, x_2)$.

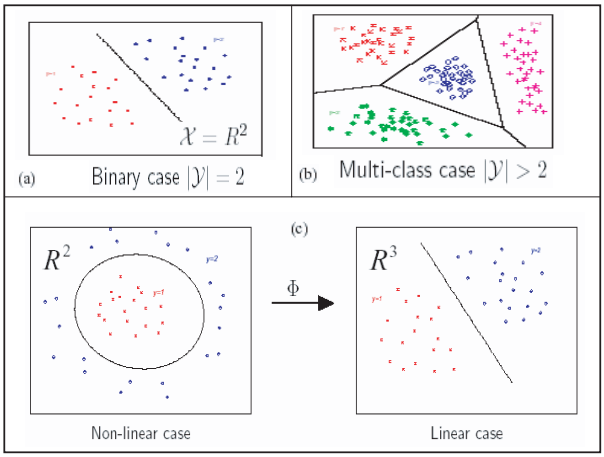

Figure 1. (a) binary category example in $2D$, (b) Multi-class categories in $2D$, (c) Optimal Non-Linear Separating Hyperplane in a 2–dimensional space

Since the problem is not linearly separable, then "slack" variables $\eta_i \geq 0$ are introduced which measure how much the margin constraints are violated, then the new constraint can be written as: $\mathbf{y}_i \left(\omega^T \varphi(\mathbf{x}_i) + b\right) \geq 1 - \eta_i$; for $i = 1, \ldots, n$. The amount of slack variables allowed can be controlled; then a penalty term $\frac{C}{\mu} \sum_{i=1}^{n} \eta_i^{\mu}$ is added to the objective function, where C, $\mu > 0$ are called the penalty coefficients [4]. The common values of the parameter μ are 1 and 2, giving linear and quadratic slack penalties, respectively. The binary SVM classification problem with the quadratic slack penalty can be considered as the following

optimization problem:

$$\begin{array}{l} \text{Min } J(\omega, b, \eta_i) = \frac{1}{2}\,\omega^T\,\omega + \frac{C}{2}\sum_{i=1}^{n}\eta_i^2, \\ \text{subject to} \\ \mathbf{y}_i\,\left(\omega^T\varphi(\mathbf{x}_i) + b\right) \geq 1 - \eta_i;\;\; \eta_i \geq 0; \end{array} \tag{1}$$

for $i = 1, \ldots, n$. Recently [22] have replaced the inequality constraints by equality constraints, a squared error term, and then the corresponding Lagrangian function, $L(\omega, b, \lambda, \eta, \gamma)$ for the system (1), that is,

$$\begin{array}{l} L = \frac{\|\omega\|^2}{2} + \frac{C}{2}\sum_{i=1}^{n}\eta_i^2 - \sum_{i=1}^{n}\gamma_i\,\eta_i \\ -\sum_{i=1}^{n}\lambda_i\{\mathbf{y}_i\,\left(\omega^T\,\varphi(\mathbf{x}_i) + b\right) - 1 + \eta_i\}, \end{array} \tag{2}$$

where $0 \leq \lambda_i \leq C$, and γ_i are the associated Lagrangian multipliers. The constant C is the user-defined constant, this constant C is the regularizing parameter. It determines the balance between the complexity of the network, characterized by the weight vector ω and the classification error of the data. For the normalized input signals the value of C is usually much bigger than 1 and adjusted by trials and errors, see [4] for more details. In practical real application, the dual quadratic optimization problem to the optimization problem (1) needed to be calculated. This can be achieved by computing the Kuhn, Tucker necessary conditions [13], that is,

$$\begin{array}{c} \frac{\partial L}{\partial b} = 0 \;\Rightarrow \sum_{i=1}^{n}\lambda_i\,\mathbf{y}_i = 0; \\ \frac{\partial L}{\partial \omega} = 0 \;\Rightarrow \omega = \sum_{i=1}^{n}\lambda_i,\,\mathbf{y}_i\,\varphi(\mathbf{x}_i); \\ \frac{\partial L}{\partial \eta_i} = 0 \;\Rightarrow C\,\eta_i - \lambda_i = 0; \\ \frac{\partial L}{\partial \lambda_i} = 0 \;\Rightarrow \mathbf{y}_i\,\left(\omega^T\,\varphi(\mathbf{x}_i) + b\right) - 1 + \eta_i = 0. \end{array}$$

This system can be written in a matrix format as:

$$\begin{array}{c} \mathbf{y}^T\,\lambda = 0;\;\; \omega = \mathbf{H}^T\,\lambda; \\ C\,\mathbf{I}\,\eta = \lambda\,\mathbf{I};\;\; \mathbf{H}\,\omega + \mathbf{Y}\,b + \mathbf{I}\,\eta = \widetilde{1}. \end{array}$$

By eliminating ω and η, we obtain:

$$\left[\begin{array}{c|c} 0 & \mathbf{Y}^T \\ \hline \mathbf{Y} & \mathbf{H}\,\mathbf{H}^T + C^{-1}\,I \end{array}\right]\,\left[\begin{array}{c} b \\ \hline \lambda \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \widetilde{1} \end{array}\right], \tag{3}$$

where $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^T$, $\lambda = [\lambda_1, \ldots, \lambda_n]^T$, $\mathbf{H} = [\mathbf{y}_1\,\phi(\mathbf{x}_1), \ldots, \mathbf{y}_n\,\phi(\mathbf{x}_n)]^T$, and $\widetilde{1} = [1, \ldots, 1]^T$. Thus, we only need to compute the inverse of $n \times n$ matrix, $\Gamma = (\mathbf{H}\,\mathbf{H}^T + C^{-1}\,I)$. Thus, we substitute for ω and η in the Lagrangian function, (2) with more simplification, the dual form [1] of the function to be optimized can be written as:

$$\mathbf{F}(\lambda) = \sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\,\lambda_j\,\mathbf{y}_i\,\mathbf{y}_j\,\kappa(\mathbf{x}_i, \mathbf{x}_j), \tag{4}$$

with the constraints: $\sum_{i=1}^{n}\lambda_i\,y_i = 0$, where $0 \leq \lambda_i \leq C$, and $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\,\phi(\mathbf{x}_j)$ is symmetric positive definite kernel function (satisfies the Mercer's condition, [3]. Therefore, we optimize the following problem with respect to λ_i:

$$\begin{array}{l} \text{Min } \{\sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\,\lambda_j\,\mathbf{y}_i\,\mathbf{y}_j\,\kappa(\mathbf{x}_i, \mathbf{x}_j)\}, \\ \text{subject to} \\ \sum_{j=1}^{n}\lambda_i\,\mathbf{y}_i = 0;\; \text{and } 0 \leq \lambda_i \leq C;\;\; i = 1, 2, \ldots, n, \end{array} \tag{5}$$

where $y_i \in \{-1, 1\}$ is the class label. Once the solution has been found, the decision can be constructed as:

$$f(\mathbf{x}) = \text{ sign }(\sum_{i=1}^{n}\lambda_i\,\mathbf{y}_i\,\kappa(\mathbf{x}_i, \mathbf{x}) + b). \tag{6}$$

We note that the kernel function $\kappa(\mathbf{x}, \mathbf{y})$ for the function $\varphi(\mathbf{x})$ in Figure 1 can be computed by squaring of their inner product, that is,

$$\begin{array}{rcl} \kappa(\mathbf{x}, \mathbf{y}) & = & (\varphi(\mathbf{x}), \varphi(\mathbf{y})) \\ & = & \left((x_1^2, x_2^2, \sqrt{2}x_1\,x_2), (y_1^2, y_2^2, \sqrt{2}y_1\,y_2)\right) \\ & = & \left((x_1^2\,y_1^2, x_2^2\,y_2^2, 2x_1\,x_2\,y_1\,y_2)\right) \\ & = & (\mathbf{x}, \mathbf{y})^2. \end{array} \tag{7}$$

The most common kernel functions in literatures can be summarized as follows:

- Linear: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \left(\mathbf{x}_i^T\,\mathbf{x}_j + \gamma\right)$
- Polynomial: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \left(\delta + \gamma\,\mathbf{x}_i^T\,\mathbf{x}_j\right)^q$;
- Gaussian (RBF): $\kappa(\mathbf{x}_i, \mathbf{x}_j) = exp\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{-2\,\sigma^2}\right)$,
- Sigmoid (MLP): $\kappa(\mathbf{x}_i, \mathbf{x}_j) = tansh\,(\gamma(\mathbf{x}_i.\,\mathbf{x}_j) - \delta)$, with gain γ and offset δ,
- Fourier Series: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \frac{sin(\frac{2\,n+1}{2})(\mathbf{x}_i - \mathbf{x}_j)}{sin((\frac{1}{2})(\mathbf{x}_i - \mathbf{x}_j))}$.

2.2 One-Against-All SVMs classifier for Multi-Classes

Dealing with multiclass pattern classification by support vector machines, both combining several binary classifiers or considering all classes at once is still an ongoing research issue, see [22] for further details. Figure 2 shows the common topology for the statistical pattern recognition under the support vector machine classifier. For the sake of simplicity, we only introduced the "*one - against-all*" method.

[1] $\omega(\lambda)$ has to be maximized with constraints $C \geq \lambda_i \geq 0$ and $\sum_{j=1}^{n}\lambda_i\,\mathbf{y}_i = 0$, for all $i = 1, 2, \ldots, n$.

See [22] for more information about the "*directed acyclic graph SVM*" (DAGSVM) technique.

The One-Against-All SVMs classifier constructs c support vector machine models, where c is the number of classes. The k^{th} SVM is trained with all of the examples in the k^{th} class with positive labels, and all other examples with negative labels. Thus, given n training set $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i \in \Re^p$ for all $i = 1, 2, \ldots, n$, and $y_i \in \{0, 1, \ldots, c-1\}$ is the class of $\mathbf{x}_i$. We use the symbol $y_i^{(k)}$ to refer to the i^{th} output unit for the class k, for all $k = 0, 1, \ldots, c-1$. To handle the pattern classification problem, we follow the same procedures as in the binary case, the support vector machines uses the least square technique in minimizing the activation function $J^{(k)}(\omega_k, b_k, \eta_{i,k})$, that is,

$$\begin{array}{l} \text{Min } \{\frac{1}{2}\sum\limits_{k=0}^{c-1}\omega_k^T\,\omega_k + \frac{C}{2}\sum\limits_{i=1}^{n}\sum\limits_{k=1}^{c}\eta_{i,k}^2\}, \\ \text{subject to} \\ \mathbf{y}_i^{(k)}\ \left(\omega_k^T\,\varphi_k(\mathbf{x}_i) + b_k\right) \geq 1 - \eta_{i,k}; \\ \eta_{i,k} \geq 0;\ i = 1, \ldots, n;\ k = 0, 1, \ldots, c-1. \end{array} \tag{8}$$

Figure 2 shows the most common least square SVM classifier architecture in the literature. In this graph, the results are linearly combined by weights ν_i, found by solving a quadratic program and compute: $\omega_i = \sum\limits_{i=1}^{n}\nu_i\,\mathbf{x}_i$,where ν_i is the support vectors, that is, ($\nu_i = y_i\,\lambda_i$ in the pattern classification; and $\nu_i = \lambda_i^\star - \lambda_i$ in the regression estimation). The linear combination is fed into the decision function $\sigma(.)$ or $f(.)$ (in pattern classification, $f(\mathbf{x}) = sign(\omega\,\mathbf{x} + b)$; and in regression estimation, $f(\mathbf{x}) = \omega\,\mathbf{x} + b$).

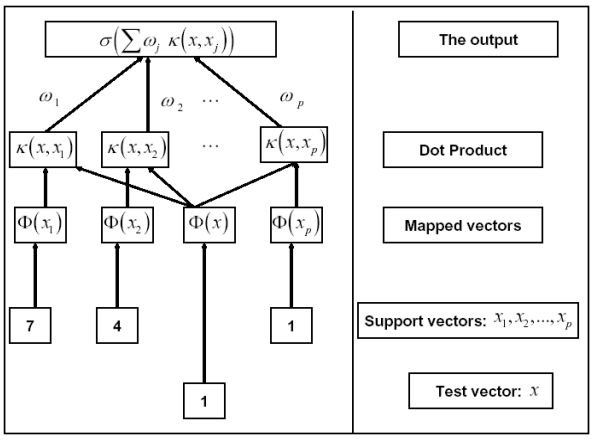

Figure 2. Architecture of SVM methods. The input $\mathbf{x} \in \Re^p$ and the support vectors $\mathbf{x}_i$

The corresponding Lagrangian function for the class k, $L^{(k)}(\omega_k, b_k, \eta_{i,k}, \lambda_{i,k})$ can be formulated as:

$$\begin{array}{l} L^{(k)} = J^{(k)}(\omega_k, b_k, \eta_{i,k}) \\ -\sum\limits_{i=1}^{n}\sum\limits_{k=1}^{c}\lambda_{i,k}\{\mathbf{y}_i^{(k)}\ \left(\omega_k^T\,\varphi_k(\mathbf{x}_i) + b_k\right) - 1 + \eta_{i,k}\}, \end{array} \tag{9}$$

for all $k = 0, 1, \ldots, c-1$, where $0 \leq \lambda_{i,k} \leq C$ are the associated Lagrangian multipliers. The dual quadratic optimization problem to the optimization problem (8) can be computed from the necessary conditions of the Lagrangian function in (9):

$$\begin{array}{c} \dfrac{\partial L^{(k)}}{\partial b_k} = 0 \Rightarrow \sum\limits_{i=1}^{n}\lambda_{i,k}\,\mathbf{y}_i^{(k)} = 0; \\ \dfrac{\partial L^{(k)}}{\partial \omega_k} = 0 \Rightarrow \omega_k = \sum\limits_{i=1}^{n}\lambda_{i,k}\,\mathbf{y}_i^{(k)}\,\varphi_k(\mathbf{x}_i); \\ \dfrac{\partial L^{(k)}}{\partial \eta_{i,k}} = 0 \Rightarrow C\,\eta_{i,k} - \lambda_{i,k} = 0; \\ \dfrac{\partial L^{(k)}}{\partial \lambda_{i,k}} = 0 \Rightarrow \mathbf{y}_i^{(k)}\ \left(\omega_k^T\,\varphi_k(\mathbf{x}_i) + b_k\right) - 1 + \eta_{i,k} = 0, \end{array}$$

for all $i = 1, 2, \ldots, n$ and $k = 0, 1, \ldots, c-1$. By eliminating ω_k and $\eta_{i,k}$, the above optimality conditions lead to $(n+1)^2 \times (n+1)^2$ linear system of equations. Therefore, we can determine the c decision hyperplanes using the decision functions: $\omega_k^T\,\varphi(\mathbf{x}) + b^k$, for all $k = 0, 1, \ldots, c-1$, and hence, we can classify the object $\mathbf{x}$ to the class k, if the class k has the largest value of the following decision function:

$$\text{class of } \mathbf{x} = \overset{\mathbf{Max}}{k}\ \left[\ \arg\ \left(\omega_k^T\,\varphi(\mathbf{x}) + b^k\right)\right],$$

for $k = 0, 1, \ldots, c-1$. Interestingly, when $\kappa(.)$ is a sigmoid kernel function, the SVM model produces a multilayer perceptron (MLP) discrimination function, but unlike the MLP model, a "global optimum" is guaranteed since the SVM model is formulated as a convex programming problem. On the other hand, when $\kappa(,)$ is a Gaussian kernel function, the SVM model leads to an radial basis function network, but unlike the RBF model, the centers and the weights are found at the same time, see [5] for more details. Although the SVM model has the above advantages over the MLP and RBF models, it is basically only for two-class classification, and the kernel functions are required to be to positive semi-definite matrices (Mercer's condition, see [16] for more details).

We note that for large number of observations, the matrix in (3) can not be stored, where we need an iterative solution method. The author in [6] suggested a fast algorithm o handle the computational complexity and running time for the system (3) using the large scale conjugate gradient algorithm. The conjugate gradient method for solving the system $\mathbf{AX} = \mathbf{B}$ with $\mathbf{A} \in \Re^n \times \Re^n$ symmetric positive definite and $\mathbf{B} \in \Re^n$ is given in details in [6]. The advantage of the conjugate gradient method is that if the matrix $\mathbf{A}$

is symmetric positive definite and it is written as: $\mathbf{A} = \mathbf{I}+\tau$, where $\text{rank}(\tau) = \delta$, the conjugate gradient algorithm in [6] converges in at most $\delta + 1$ steps.

Once the learning process in the support vector machines has finished, the judgment and evaluation of the quality and capability of the fitted model (validation step). To achieve this step, we compute numerous of quality measures, such as, *Correct Classification Rate (CCR)*, *Average Squared Classification Error (ASCE)*, *Time of execution*, and *Number of Parameters*; where the best model is the one with the highest *CCR*, see [5, 17, 7, 19] for more details. The performance of the support vector machines classifier is investigated against *K-nearest neighbor* (KNN), *feedforward neural networks* (FFN), *probabilistic networks* (PNN), and *linear/non-linear logistic regression* (LR) classifiers using both real-world applications and simulation studies. For the sake simplicity, we recorded only these investigations on growth/no-growth state of a pathogenic Escherichia coli R31 in response to temperature and water with some of the quality measures as it is shown in Section 3.

3 Implementation and Comparative Studies

3.1 Initialization

In this implementation and comparative study, the support vector machines with the Gauessian kernel function is tested using the growth and non growth state of a pathogenic Escherichia coli R31 in response to temperature and water database. This data set can be downloaded from UC-Irvine [2]. In this implementation, we use both *stratified Sampling* and *cross-validation* (internal and external validation) techniques to make sure that we get the same proportion from each group as in the original data. We repeat the estimation and validation processes for 1000 times using MATLAB V7 under Pentium M personal computer, then compute all the quality measures for the investigated classifiers. Next, we summarize the results by computing the average, the standard deviation, and the coefficient of variation of each quality measure over these 1000 runs.

We draw a graph for the mean of CCR versus its standard deviation over the 1000 runs. This graph helps us to decide which classifier is better in its performance. In this plot, each classifier is represented by a symbol. A good classifier should appear in the upper left corner of the graph. In addition, corresponding to these graphs, we summarize the results in Tables. In these Tables, the highest CCR's are given in boldface.

3.2 E. Coli Strain Database

We apply all eight classifiers to E. Coli Strain Data. This data set is taken from [20], pertaining to the growth of an E. Coli Strain $R31$ as affected by temperature and water activity. The data consist of experimental testing of a total of 179 cases. Each sample was scored positive (i.e., growth occurred) 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 loop-full of culture was streaked onto plate count agar to determine if any growth is present [9]. For any temperature and water activity, combination, growth was recorded as $(y = 1)$ if it occurred and $(y = 0)$ if it did not. The data contains 103 growth cases and 76.

As we can see from the scatter plot of the data in Figure 3, there is some overlap between the two groups.

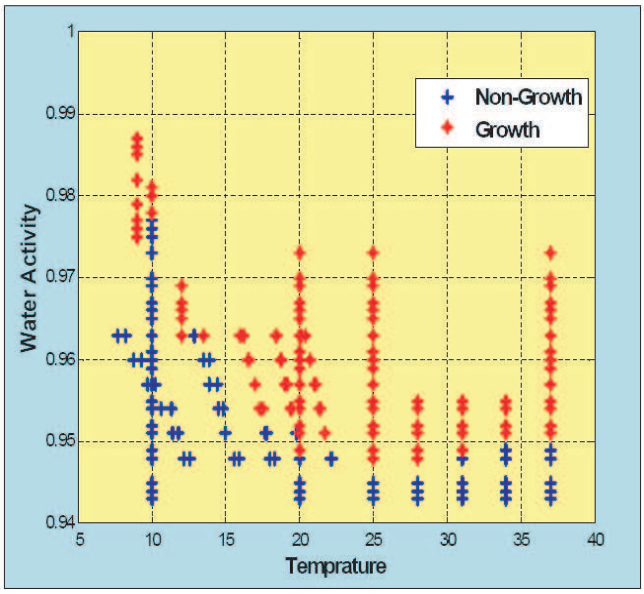

Figure 3. E. Coli Strain Data: The scatter plot of water activity and the temperature

To evaluate the performance of each classifier on E. Coli Strain data, we use 70% of the data for building the classification model (internal validation) and 30% of the data for testing/validation (external validation). The training set has 124 observations (50 from group 1 and 74 from group 2), and the validation set has 55 observations (27 from group 1 and 30 from group 2). We repeat the internal and external validation processes for 1000 times. The corresponding quality measures are shown in Table 1, the corresponding graphs are shown in Figure 5, and the results are compared with the one in [9] as well. In this application, numerous of kernel functions is investigated with different input parameters, we recorded only the best ones with the best results. Figure 4 shows the confusion matrix in both training and testing, respectively.

Figure 5 shows a scatter plot, where each of the investigated classifiers is represented by a symbol. The plot represents the average of CCR versus its standard deviation. A good

[2] URL: ftp://ftp.ics.uci.edu/pub/machine-learning-databases."

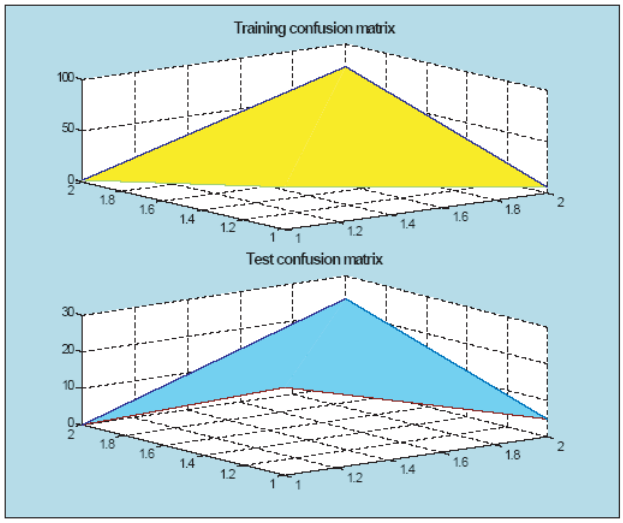

Figure 4. The confusion matrix in both training and testing for the SVMs classifier performance

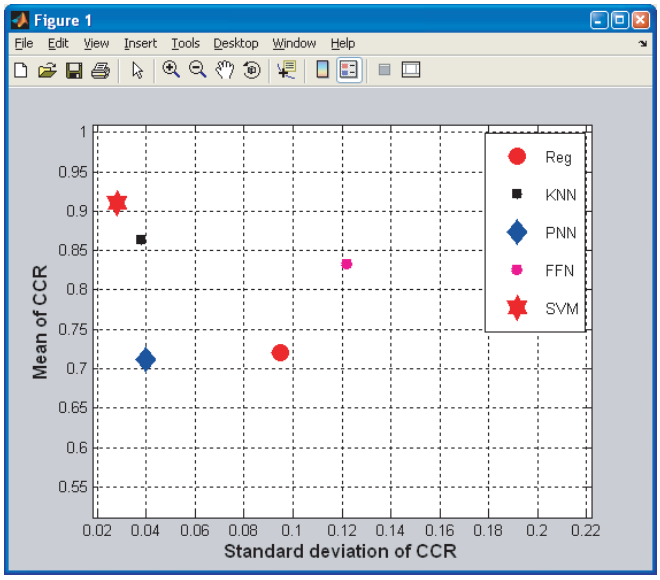

Figure 5. Average of CCR versus σ_{CCR}

classifier should appear in the upper left corner of the graph. Table 1 contains the summary of the classifiers quality measures. The high light CCR are given in boldface.

For the internal validation purpose, we summarize the output in Table 2. This table contains in the first two columns the number of correct classified observations in each class, and the last column contains the correct classification rate, the highest value is in boldface. The remaining columns contain the number of misclassified observations in each class. We observe that the PPN classifier gives the lowest average CCR value among the classifiers. The FFNN classifier has the highest value of CCR. Generally, this classifier work well for internal validation in low dimensions. On the other hand, it has the the highest execution time. The support vector machines classifier is still one of the highest values of the average CCR. All other classifiers perform more or less the same, with and multiple linear/nonlinear regression, and K-nearest neighbor.

In the internal validation, we observe that all classifiers giving close percentage of the *CCR*, but the *neur-fuzzy systems* performance is stable in the high dimensions. On the other hand, to evaluate the performance of each classifier, we divide the given data into testing and training sets. The results are summarized by computing average, standard deviation, and the coefficient of variation of each quality measure over all runs. From Table 1 and Figure 5, we observe that the three classifiers: logistic regression, support vector machines, and probabilistic neural network have the smallest average CCR. These are the worst performance among all classifiers. The linear discriminant analysis classifier and radial basis functions networks are the second worst average CCR. The FFNN has by far the highest execution time. This is mainly due to the large number of iterations. Yet, it the has the third worst performance among all classifiers. The support vector machines classifier is giving the highest values of the average CCR.

4. Conclusion and Future Work

Based on the obtained analytical results, we observe that the the support vector machines classifier with Gaussian sigmoidal kernel outperforms both feedforward neural networks, linear/nonlinear regression, K-nearest neighbors, and probabilistic networks classifiers in terms of bacteria growth/non-growth accuracy and misclassification costs and hence provide efficient alternatives to conduct food science tasks. The results can also be applied in other industry applications, such as, Micro-array Genomic, and biomedical industry.

Acknowledgements

The author would like to acknowledge the support and facilities provided by King Fahd University of Petroleum and Minerals (KFUPM) in the development of the paper.

References

[1] Almeida J. S., 2002. Predictive non-linear modeling of complex data by artificial neural networks. Curr. Opin. Biotechnol. 13 (1), 72-76.

[2] Basheer I. A. and Hajmeer M. N., 2000. Artificial neural networks: fundamentals, computation, design and application. J. Microbiol. Methods 43, 3-31.

[3] Boser B. E., Guyon I., and Vapnik V., (1992), "A training algorithm for optimal margin classifiers". *Proceedings of the*

Table 1. The External Validation Results

Classification Method	No. Parameter		Time of Exec.		CCR	
	mean	StDev	mean	StDev	mean	StDev
REGRESSION	3	0	0.206	0.083	0.720	0.095
KNN	3	0	0.022	0.008	0.863	0.038
PNN	2	0	0.323	0.032	0.711	0.040
FFN	2	0	14.099	11.160	0.832	0.122
SVM	3	0	1.59	1.019	**0.90**	0.052

Table 2. The Internal Validation Results

Classifier	COR-OBS1	COR-OBS2	MISS-OBS1	MISS-OBS2	CCR
Regression	55	86	21	17	0.788
KNN	68	95	8	8	0.911
PNN	44	86	32	17	0.726
FFN	74	102	2	1	**0.983**
SVM	75	101	1	2	**0.983**

Fourth Workshop on Computational Learning Theory, ACM Press, San Mateo, CA, pp. 144–152.

[4] Cristianini N., Shawe-Taylor J., (2000), *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press.

[5] Duda O. R., Hart E. P., and Stork G. D., 2001, *Pattern Classification*. Second Edition, , John Wiley & Sons, Inc, New York.

[6] El-Sebakhy E. A., (2004), "A Fast and Efficient Algorithm for Multi-class Support Vector Machines Classifier", ICICS2004, 28–30 November. IEEE Computer Society, pages: 397-412.

[7] Gordon, A. D. (1999), *Classification*, 2nd Edition, Chapman & Hall/ CRC.

[8] Gunn S., (1998), *Support vector machines for classification and regression*. Image, Speech and Intelligent Systems Tech., U.K.

[9] Hajmeer M. and Basheer I., Comparison of logistic regression and neural network-based classifiers for bacterial growth. *Food Microbiology*, 20:43–55, 2003.

[10] Hajmeer M. N., Basheer I. A., Marsden J. L., and Fung D. Y. C., 2000. New approach for modeling generalized microbial growth curves using artificial neural networks. J. Rapid Methods Automat. Microbiol. 8 (4), 265-284.

[11] Hosmer D. L. and Lemeshow S., (2000), *Applied Logistic Regression*, second edition. John Wiley & Sons, New York.

[12] Jeyamkondan S., Jayas D. S., and Holley R. A., 2001. Microbial growth modelling with artificial neural networks. Int. J. Food Microbiol. 64, 343-354.

[13] Kuhn H. W. and Tucker A. W., (1961), " Nonlinear Programming," In proceedings of the second Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley and Los Angeles, California, 481–492.

[14] Lopez-Malo A., Guerrero S., and Alzamora S.M., 2000. Probabilistic modeling of saccharomyces cerevisiae inhibition under the effects of water activity, ph, and potassium sorbate concentration. J. Food Prot. 63 (1), 91-95.

[15] Lou W. and 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 Res. Int. 34, 573-579.

[16] Mercer J., (1999), "Functions of positive and negative type and their connection with the theory of integral equations," *Philosophical Transactions of the Royal Society of London, Series*, 209, pp. 415–446.

[17] Mitchell T., Machine learning and data mining. *Communications of the ACM*, 4:2–11, 1999.

[18] Presser K. A., Ross T., and Ratkowsky D. A., 1998. Modelling the growth (growth/no growth interface) of E. coli as function of temperature, pH, lactic acid concentration, and water activity. Appl. Environ. Microbiol. 64 (5), 1773-1779.

[19] Quinlan J. R. 1986 " Induction of decision trees". *Machine Learning*, 1, 81–106.

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

[21] Schepers A. W., Thibault J., Lacroix C., 2000. Comparison of simple neural network and nonlinear regression models for descriptive modeling of lactobacillus helveticus growth in pH-controlled batch cultures. Enzyme Microbiol Technol. 26, 431-445.

[22] Suykens J. A. K. and Vandewalle J., (1999a), "Least squares support vector machine classifiers," *Neural Process. Lett.* 9, 293–300.

[23] 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. Appl. Environ. Microbiol. 66 (11), 4979-4987.

[24] Vapnik V., (1995), *The nature of statistical learning theory*, Springer, New York.

[25] Vapnik V., (1998), *Statistical learning theory*, John Wiley, New York.