sensors

MDPI

Article

A False Alarm Reduction Method for a Gas Sensor Based Electronic Nose

Mohammad Mizanur Rahman [1,2,*], Chalie Charoenlarpnopparut [1], Prapun Suksompong [1], Pisanu Toochinda [3] and Attaphongse Taparugssanagorn [4,*]

1 School of Information Computer and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University, Khlong Luang 12121, Thailand; chalie@siit.tu.ac.th (C.C.); prapun@siit.tu.ac.th (P.S.)
2 Electronics and Communication Engineering Discipline, Science Engineering and Technology School, Khulna University, Khulna 9208, Bangladesh
3 School of Bio-chemical Engineering and Technology, Sirindhorn International Institute of Technology, Thammasat University, Khlong Luang 12121, Thailand; pisanu@siit.tu.ac.th
4 ICT Department, Telecommunications, School of Engineering and Technology, Asian Institute of Technology, Pathum Thani 12120, Thailand
* Correspondence: mizan.ku@gmail.com (M.M.R.); attaphongset@ait.asia (A.T.); Tel.: +66-8-8628-8820 (A.T.)

Received: 25 August 2017; Accepted: 9 September 2017; Published: 12 September 2017

Abstract: Electronic noses (E-Noses) are becoming popular for food and fruit quality assessment due to their robustness and repeated usability without fatigue, unlike human experts. An E-Nose equipped with classification algorithms and having open ended classification boundaries such as the *k*-nearest neighbor (*k*-NN), support vector machine (SVM), and multilayer perceptron neural network (MLPNN), are found to suffer from false classification errors of irrelevant odor data. To reduce false classification and misclassification errors, and to improve correct rejection performance; algorithms with a hyperspheric boundary, such as a radial basis function neural network (RBFNN) and generalized regression neural network (GRNN) with a Gaussian activation function in the hidden layer should be used. The simulation results presented in this paper show that GRNN has more correct classification efficiency and false alarm reduction capability compared to RBFNN. As the design of a GRNN and RBFNN is complex and expensive due to large numbers of neuron requirements, a simple hyperspheric classification method based on minimum, maximum, and mean (MMM) values of each class of the training dataset was presented. The MMM algorithm was simple and found to be fast and efficient in correctly classifying data of training classes, and correctly rejecting data of extraneous odors, and thereby reduced false alarms.

Keywords: electronic nose; false alarm; hyperspheric classification boundary; classification; correct rejection

1. Introduction

Trained human sniffers are employed as quality checkers in industries. However, an expert nose can fatigue due to sickness or exposure to an odor for long time. To overcome this prostration of human beings, an electronic nose (E-Nose) was introduced in 1982 [1]. E-Noses can also be deployed in chemical areas unreachable by humans or animals. An E-Nose performs its operation unceasingly through its lifetime without fatigue. The essential constituents of an E-Nose are a sensing system (combination of a sensor panel and a data acquisition system) and an automated pattern classification system [2]. An efficient, fast, and simple classification algorithm is required for training an E-Nose, which classifies types of target objects with an acceptable error rate.

Data from any trained odor class should be correctly classified, and data from any irrelevant odor class should be correctly rejected. Any incorrect classification of the data from trained classes or irrelevant classes results in classification error. E-Nose classification errors can be categorized into two following kinds:

(i) Misclassification or false negative: odor data of a training class is classified to another trained or irrelevant class, and,
(ii) False classification or false positive: odor data of an irrelevant class classified to a training class.

Any false classification causes false alarm. Along with classification efficiency (correct classification rate), and speed of classification and simplicity, false classification rate and correct rejection rate should also be considered as criteria to choose a classification algorithm for E-Nose applications. Until the recently, few studies addressed "false classification" or "false alarm" in E-Noses. The false alarm issue has been addressed for fire detection in [3–5]. In [3–5] multilayer perceptron neural network (MLPNN), principal component analysis (PCA), and linear discriminant analysis are used for classification, respectively. Chemical agent detection and concentration estimation has been studied in [6], where authors have used a threshold based approach for false alarm detection, not considering the presence of irrelevant gases or VOCs. The sensors used for E-Nose applications are sensitive to a wide range of gases or VOCs. A false alarm is generated by an E-Nose if it comes in contact with any irrelevant gases or VOCs to which the E-Nose sensors are sensitive. Therefore, once the odor data are converted to electronic signals and preprocessed, the pattern recognition and decision making stage of the E-Nose must play an effective role for correct classification of the odor, as well as false alarm reduction. A scrupulous pattern recognition method is essential for extracting meaningful information from the sensed data. In [7–9] PCA is used for dimensionality reduction and visualizing data in reduced diemnsions. Classification algorithms, such as *k*-nearest neighbor (*k*-NN) [10–13], support vector machine (SVM) [11–16], generalized regression neural network (GRNN) [17], radial basis function neural network (RBFNN) [18–22], and MLPNN [9,20–24], etc., have been applied for odor classification by different E-Noses. Among these algorithms, *k*-NN, SVM, and MLPNN separate the classes by unbound classification boundaries, i.e., lines, planes, and hyper-planes in two, three, and higher dimensional space, respectively. A hyperspheric classification algorithm presented by Cooper [25] classifies a test data to a class if it resides within a predefined Euclidean distance from the class center, i.e., mean point of the class; else it is decided to be out of the class. An inversion algorithm for MLPNN [26] and a Gaussian kernel based SVM [27] are shown to generate closed boundary around training data which classify two classes of data, i.e., one class inside the boundary and another outside the boundary. The classification boundaries produced by RBFNN and GRNN with Gaussian activation functions are bounded in two, three, or higher dimensions. For more than three dimensions, they produce a bounded hyperspheric classification boundary. It is shown in this paper that the classification methods with open ended classification boundaries falsely classify irrelevant data to training classes, which cause false alarms and thus are not a good choice for E-Nose applications. A classification algorithm with a hyperspheric classification boundary is a potential candidate for reducing false classification rates, and thereby improves "correct rejection" rate in E-Nose applications.

In this paper, the *k*-NN, SVM, GRNN, RBFNN, and MLPNN were compared in terms of computational speed, false classification rate, correct classification rate, and correct rejection rate. It as observed that RBFNN and GRNN are suitable to cope with false alarms, with GRNN performing better than RBFNN. A simple hyperspheric classification algorithm based on maximum, minimum, and means (MMM) of each feature or variable (sensor) for each class was also proposed. The design complexity of the proposed MMM method was lower compared to *k*-NN, SVM, GRNN, RBFNN, and MLPNN classification methods. In addition, the MMM method was found to potentially reduce false classification errors. For the analyses in this research, fruity odor data were collected from four kinds of fruits namely, banana, mango, sapodilla, and pineapple at three ripeness states. Ripeness and quality of different kinds of fruits are also studied in [28] to assess apple quality during shelf life, in [29] to classify ripeness states of peaches and pears, in [30–32] to assess peach quality, in [33] for disease detection of blueberry, in [34] to assess ripeness and damaged blueberries, in [35] to assess the ripeness state of pink lady apples, in [36] online discrimination among banana, wine and baseline response is presented, and in [37] a review on food and fruit process monitoring, shelf-life investigation, freshness evaluation, and authenticity assessment is presented.

An E-Nose design process, sample collection, sensor array, experimental setup, data acquisition, and classification algorithms are discussed in materials and methods section. The materials and methods section is followed by the results and discussion. Throughout this paper individual capital bold and small bold letters are used to represent matrices and vectors, respectively.

2. Materials and Methods

An E-Nose design process, sample collection, design of the sensor array, experimental setup, data acquisition, and classification algorithms are discussed in this section.

2.1. E-Nose Design Process

An E-Nose designing process has three main steps: (i) designing the sample and measurement chambers; (ii) data pre-processing; and (iii) classification and decision mapping. A block diagram of the design process is shown in Figure 1. The object under test was kept into the sample chamber, and the sensor array and data acquisition device were placed into measurement chamber. LabVIEW, and data processing and classification algorithms were installed in a Wi-Fi-connected computer to control the acquisition device, store data, pre-process the stored data to prepare signature patterns for training a classification algorithm, and classify odor data. The computer contained an Intel core i5 processor, 4 Gb random access memory, and ran with a 64 bit Windows 7 operating system. The outputs from the classification algorithm were mapped to the corresponding decisions set by the designer.

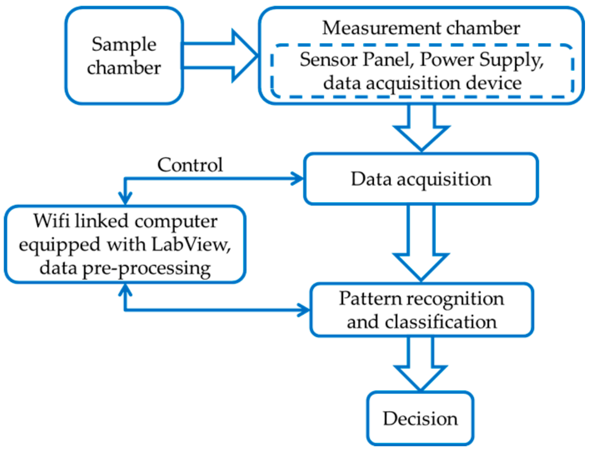
Sample chamber
Measurement chamber
Sensor Panel, Power Supply, data acquisition device
Control
Data acquisition
Wifi linked computer equipped with LabView, data pre-processing
Pattern recognition and classification
Decision

Figure 1. Design process of an electronic nose.

2.2. Sample Collection

Four categories of ripe fruits namely, banana, mango, sapodilla, and pineapple were chosen for this research. The photos of the fruits collected during experiment at their three ripeness stages are shown in Figure 2. Each fruit type was kept into the sample chamber in turn during the experiment. The samples were stored in different air tight boxes to prevent odor mix up. Throughout the experiment the fruits were preserved at 28 °C under normal conditions.

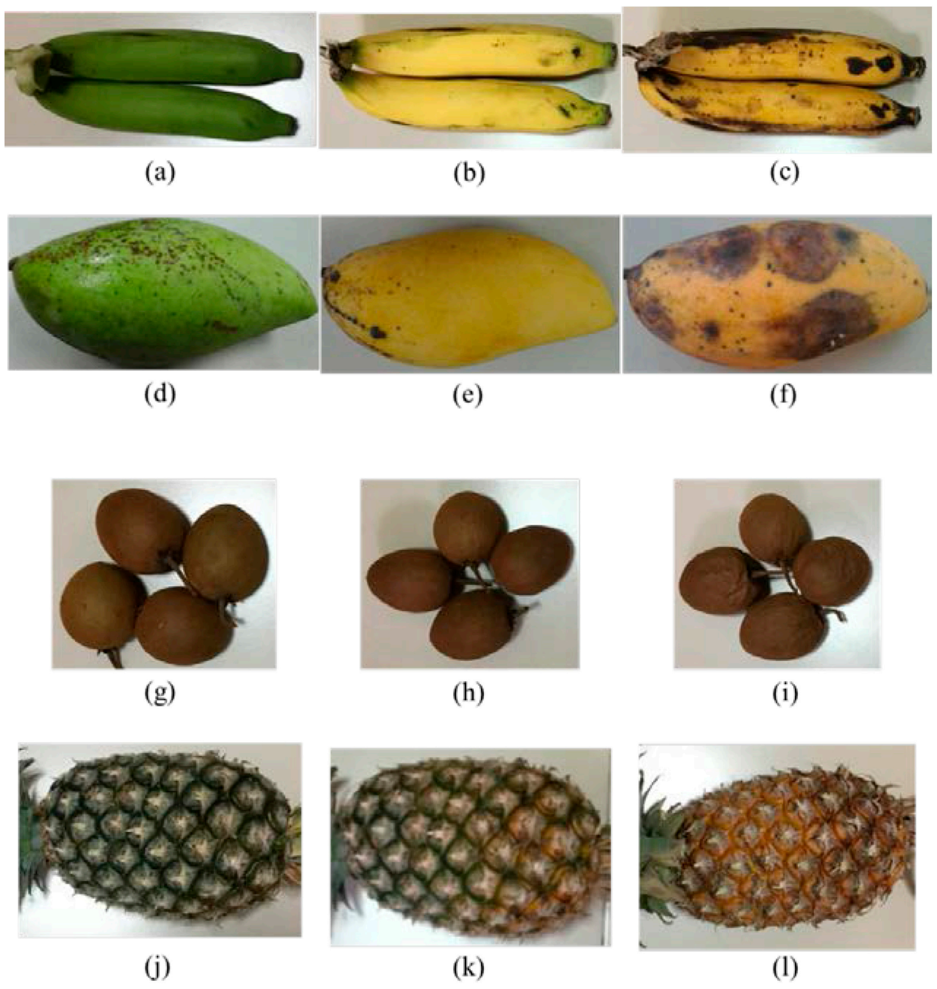

Figure 2. Fruit samples at different ripeness states: (**a**) unripe banana; (**b**) ripe banana; (**c**) rotten banana; (**d**) unripe mango; (**e**) ripe mango; (**f**) rotten mango; (**g**) unripe sapodilla; (**h**) ripe sapodilla; (**i**) rotten sapodilla; (**j**) unripe pineapple; (**k**) ripe pineapple; and (**l**) rotten pineapple.

2.3. Sensor Panel

Eight metal oxide gas (MOG) sensors from FIGARO Engineering Inc. (Osaka, Japan), namely, TGS2612, TGS821, TGS822, TGS813, TGS2602, TGS2603, TGS2620, and TGS2610 were combined to build the sensor panel. The sensitivity of the sensors to different VOCs are summarized in Table 1, and the printed circuit board of the E-Nose sensor panel is shown in Figure 3. In Table 1, S1 to S8 are labels assigned to the sensors in this research. The sensors in the panel were altogether sensitive to ethylene, acetylene, propane, butane, hydrogen, hydrogen sulfide, carbon monoxide, benzene, ammonia, acetone, hexane, trimethyl amine, and methyl mercaptan. The sensors' resistivity increased in the presence of free air or oxygen, and decreased in presence of target VOCs or gases. In presence of VOCs or gases, a current travelled through a sensor as in Figure 3, and the corresponding load resistor increased. As a result, the voltage across the load resistor of a sensor increased, corresponding to the concentration levels of VOCs or gases.

Table 1. Figaro gas sensors used in the E-Nose design and the gases or volatile organic compounds (VOCs) to which the sensors are sensitive.

Sensor Model	Sensor Label	Gases and VOCs											
		CH_2	C_2H_2	C_3H_8	C_4H_{10}	H_2	H_2S	CO	C_6H_6	NH_3	$(CH_3)_2CO$	C_6H_{14}	Trimethyl Amine and Methyl Mercaptan
TGS 2612	S1	√	√	√	√								
TGS 821	S2	√	√			√		√					
TGS 822	S3	√	√		√			√	√		√	√	
TGS 813	S4	√	√	√	√	√		√					
TGS 2602	S5		√			√	√			√			
TGS 2603	S6		√			√	√						√
TGS 2620	S7	√	√		√	√		√					
TGS 2610	S8	√	√	√	√	√							

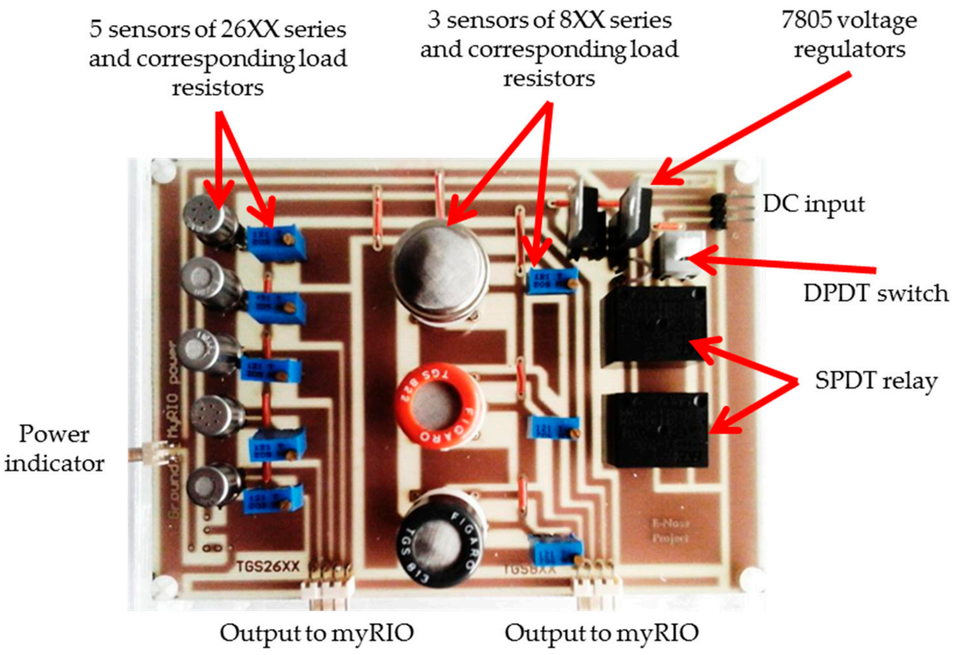
5 sensors of 26XX series and corresponding load resistors
3 sensors of 8XX series and corresponding load resistors
7805 voltage regulators
DC input
DPDT switch
SPDT relay
Power indicator
Output to myRIO
Output to myRIO

Figure 3. The electronic nose sensor panel. In the figure DC, DPDT, and SPDT stands for direct current, double pole double throw, and single pole double throw, respectively.

2.4. Experimental Setup

The fruit samples were kept in turn in the sample chamber shown in Figure 4. The sensor panel and the data acquisition device were kept into the measurement chamber. A LabVIEW installed personal computer (PC) was is wirelessly connected to the acquisition device to collect and pre-process the data. To prevent the mixing of undesired odors, both the chambers were air tightened. Connecting cables from a 5 volt and a 10 volt external direct current (DC) power supply are inserted through an airtight hole into the measurement chamber to power the sensors and the data acquisition device. The measurement process is comprised of two different phases, i.e., concentration and measurement. During concentration phase, Valves 3 and 4 were closed and VOCs were concentrated in the sample chamber. The Valves 1 and 2 were opened and Fans 1 and 2 were switched on during the concentration phase to clean the measurement chamber with free air and thereby neutralize the sensors. During the measurement phase, Valves 1 and 2 were closed, Valves 3 and 4 were opened, and Fans 3 and 4 were turned on to circulate odor VOCs between the samples and measurement chambers.

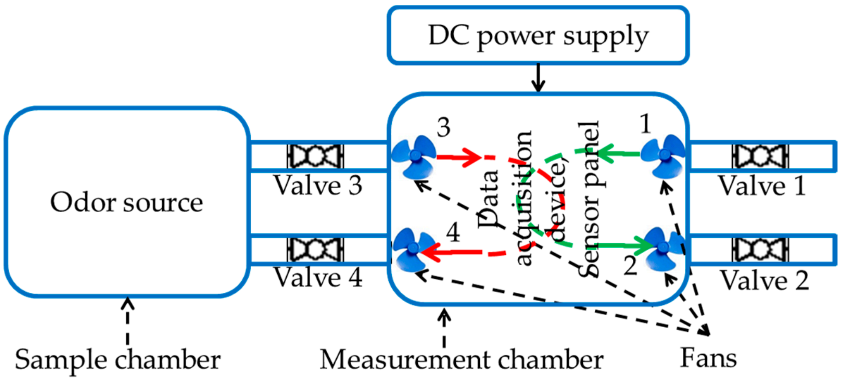

Figure 4. The E-Nose experimental setup. Valves 1 and 2, and Fans 1 and 2 are for air flow control, Valves 3 and 4, and Fans 3 and 4 are for circulating the odor between sample chamber and measurement chamber. A DC power supply powers the sensors, fans, and the data acquisition device. In the figure DC stands for direct current.

2.5. Data Acquisition

A data acquisition device from National Instruments Inc., named myRIO, was employed for data acquisition in the experiment. The acquisition device was controlled by a LabVIEW equipped PC via a wireless access point. Twenty repeated experiments were performed for each kind of fruits at three ripeness states, i.e., unripe, ripe, and rotten, which gave 240 measurements in total. The sensors were preheated for five minutes before beginning the experiment. The experiments were conducted in the following sequence: VOCs were concentrated for three minutes in the sample chamber, followed by two minutes of sensing. During the concentration phase, the measurement chamber was cleaned with free air to achieve sensor base level responses. To collect each data sample, this experiment cycle was repeated. At each ripeness state of the fruits, distinct transient and steady state responses were observed. Time domain sensor responses of an experiment to rotten sapodilla are given in Figure 4. The steady state responses of the sensors were used to produce the signature patterns for each fruit type at each ripeness state.

2.6. Pattern Recognition and Classification Algorithms

Classification algorithms as applied to E-Nose, namely, principal component analysis (PCA), *k*-NN, SVM, GRNN, RBFNN, MLPNN, and the proposed MMM method are presented in this subsection. The PCA is an unsupervised method of classification. In this research, supervised models were used for *k*-NN, SVM, GRNN, RBFNN, MLPNN, and MMM. The data matrix **X** and the corresponding target vector **t** (whose elements are class labels) are expressed as in Equations (1) and (2):

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ \vdots \\ \mathbf{X}_m \\ \vdots \\ \mathbf{X}_M \end{bmatrix} \text{ and } \mathbf{t} = \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \vdots \\ \mathbf{t}_m \\ \vdots \\ \mathbf{t}_M \end{bmatrix}, \quad (1)$$

where

$$\mathbf{X}_m = \begin{bmatrix} x_{1,m,1} & \cdots & x_{1,m,n} & \cdots & x_{1,m,N} \\ x_{2,m,1} & \cdots & x_{2,m,n} & \cdots & x_{2,m,N} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ x_{l,m,1} & \cdots & x_{l,m,n} & \cdots & x_{l,m,N} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ x_{L,m,1} & \cdots & x_{L,m,n} & \cdots & x_{L,m,N} \end{bmatrix} \text{ and } \mathbf{t}_m = \begin{bmatrix} t_{1,m} \\ t_{2,m} \\ \vdots \\ t_{l,m} \\ \vdots \\ t_{L,m} \end{bmatrix}, \tag{2}$$

$l = 1, 2, \ldots, L$ are the experimental indices of each class $m = 1, 2, \ldots, M$; L is the number of data samples in each class, M is the number of classes, $n = 1, 2, \ldots, N$ is the sensor index, and N is the number of sensors. The rows in $\mathbf{X}$ are experimental samples and the columns are feature or sensor variables. The training, validation and testing data are chosen from $\mathbf{X}$ as per the ratio defined by the designer. From each class, 70%, i.e., $0.7L$ samples are taken for training the algorithms.

2.6.1. Principal Component Analysis

PCA [4,7–11] is an unsupervised method of data analysis which reduces dimensionality while preserving maximum data variance. PCA converts the data to a new feature space through an orthogonal transformation process. The direction of maximum data variance in the original space is the first principal component (PC 1), the direction of second largest data variance which is orthogonal to the PC 1 is the second principal component (PC 2), PC 3 is orthogonal to both PC 1 and PC 2, and so on for higher dimensional PCs. The algorithm is given as follows:

Step 1. Find the mean vector $\mathbf{x} = [x_1, x_2, \ldots, x_N]$, where each element represents the mean of each column of $\mathbf{X}$. N is the number of sensors.
Step 2. Deduct the mean vector x from each row of matrix $\mathbf{X}$.
Step 3. Find the covariance matrix of the mean deducted matrix in Step 2.
Step 4. Find the eigenvalues and eigenvectors of the covariance matrix found in Step 3.
Step 5. Sort the eigenvectors in descending order of eigenvalues.
Step 6. The eigenvector with maximum eigenvalue is PC 1, the eigenvector corresponding to the second largest eigenvalue is PC 2, and so on for the descending eigenvalues.

PCA was used in this paper for visualizing higher dimensional data by reducing dimensionality.

2.6.2. *k*-Nearest Neighbor

k-NN [10–13] is the simplest pattern recognition and classification algorithm of all the machine learning algorithms. During the training phase a fraction, i.e., 70 percent of the experimental samples from matrix $\mathbf{X}$ in Equation (3) are chosen for training. The feature vectors in matrix $\mathbf{X}$ and the class labels in the target vector $\mathbf{t}$ in Equation (3) corresponding to training data, are stored during the training phase. The remaining 30 percent of the dataset is used for testing. The nearest k training samples from the test data vector are found first. An Euclidean distance metric is used to measure the closeness. The test data is assigned to the winning class by a majority voting between the closest k neighbors. To avoid any possible tie, k must be an odd number, and k must not be a multiple of the number of classes. The computational complexity of *k*-NN is $O(LM(N + k))$. A supervised *k*-NN model is used in this research.

2.6.3. Support Vector Machine

For the SVM [11–16] classification, the hyperplanes that provide the maximum margin between two classes, having no interior data, were found. The data that lie on the hyperplanes are known as support vectors. A hyperplane which lies between these two hyperplanes, so that the distances from it to the support vectors of both the classes are maximized, is known as a maximum-margin hyperplane. A linear classifier defined by the maximum-margin hyperplane is known as a maximum margin classifier. The SVM algorithm is given as follows:

Step 1. Maximize the Lagrangian [Equation (3)] such that $\sum_{m}\sum_{l}\alpha_{l,m}t_{l,m} \geq 0$ and $\alpha_{l,m} \geq 0$.

$$L(\alpha_{l,m}) = \sum_{l,m}\alpha_{l,m} - \frac{1}{2}\sum_{l,m,g,j} \alpha_{l,m}\alpha_{g,j}t_{l,m}t_{g,j}(\mathbf{x}_{l,m} \times \mathbf{x}_{g,j}), \tag{3}$$

where l and g are experiment indices within a class, m and j are class indices, $\alpha_{l,m}$ are Lagrange multipliers, $t_{l,m}$ are class labels (+1 or -1), and $\mathbf{x}$ are data vectors.

Step 2. Find the value of $\alpha_{l,m}$ from Equation (3).

Step 3. Put the values of $\alpha_{l,m}$ in Equation (5) to find the weight vector by Equation (4).

$$\mathbf{w} = \sum_{l,m}\alpha_{l,m}t_{l,m}\mathbf{x}_{l,m} \tag{4}$$

Step 4. By the Karush–Kuhn–Tucker condition [38] as shown in Equation 5 below and weights found from Equation (6), bias b is calculated.

$$\alpha_{l,m}[t_{l,m}(\mathbf{w} \times \mathbf{x}_{l,m} + b) - 1] = 0, \tag{5}$$

Step 5. Classify test data by looking at the sign of the solution to Equation (6) as given below:

$$f(\mathbf{u}) = \mathbf{w} \cdot \mathbf{u} + b = \left(\sum_{s=1}^{S}\alpha_s t_s \mathbf{x}_s \times \mathbf{u}\right) + b, \tag{6}$$

where, s is support vector index, S is the number of support vectors, b is the bias, and u is the data vector under testing.

The basic SVM is a binary classifier. In this research, multiple linear binary SVM classifiers were combined to obtain a multiclass SVM classifier. A multiclass SVM classifier classifies data with a majority voting between the outcomes from the binary SVM classifiers by the k-NN method.

The computational cost of solving the SVM problem has both a quadratic and cubic component. Based on small margin or large margin violations, SVM complexity varies between $O(NL^2M^2)$ and $O(NL^3M^3)$.

2.6.4. Multilayer Perceptron Neural Network

The MLPNN [9,20–24] is a supervised feed-forward neural network that uses back-propagation training and generalized delta rule learning. Each layer of an MLPNN is fully connected to the next layer as a directed graph. After every epoch, the mean squared errors (MSEs) are calculated from the differences between the outputs of the validation dataset to the corresponding targets. To minimize the error in every epoch the synaptic weights and activation thresholds are adjusted. A three layer MLPNN as shown in Figure 5 is applied for this research. The input layer has eight (N) neurons corresponding to eight sensors. The hidden layer consists of 10 (J) neurons with tan-sigmoid (h) activation function. The output layer has one (D) neuron with linear activation function to provide class decision corresponding to an input data. The MLPNN training algorithm is given below.

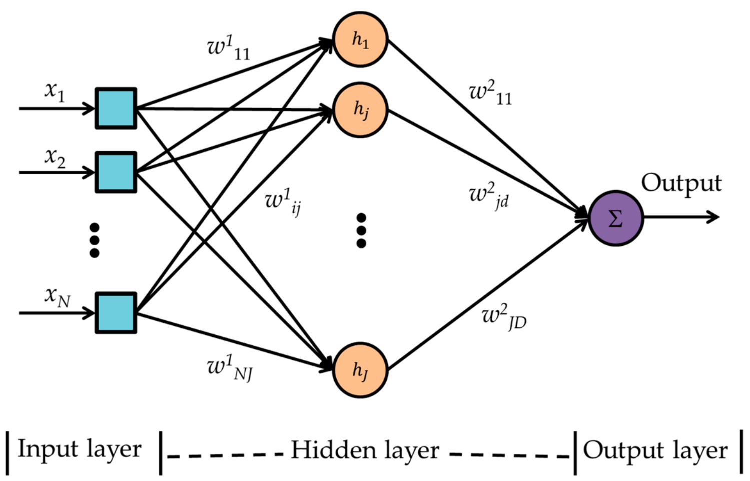

Figure 5. Multilayer perceptron neural network block diagram. *i* is the sensor index, *j* is the index for hidden layer neuron, and *d* is the index of output layer neuron.

Step 1. Initialize the weights to small positive and negative random values.

Step 2. Run the network forward with each training data to get the network outputs as given in Equation (7), and calculate the errors as shown in Step 3 to Step 6.

$$y^3_{l,m} = g\left(\sum_{j=1}^{J} w^2_{jd} h_j \left(\sum_{i=1}^{N} w^1_{ij} x_{l,m,i}\right)\right), \tag{7}$$

where $h_j(a) = 2/(1+\exp(-2a)) - 1$, i is the sensor index, j is the index for the hidden layer neuron, and d is the index of the output layer neuron.

Step 3. Compute the backpropagation error terms for the links to the output neuron as in Equation (8) below:

$$\delta^2_{l,m} = y^3_{l,m}\left(1 - y^3_{l,m}\right)\left(y^3_{l,m} - t_{l,m}\right). \tag{8}$$

Step 4. For each hidden node, calculate the backpropagation error term as in Equation (9) below,

$$\delta^1_j = y^2_j\left(1 - y^2_j\right)\sum_{d=1}^{D} \delta^2_{l,m} w^2_{jd}. \tag{9}$$

Step 5. Update the synaptic weights from a node in Layer 1 to a node (neuron) in Layer 2, as $\Delta w^1{}_{ij} = -\eta\delta^1{}_j x_{l,m,n}$, and apply, $w^1{}_{ij} = w^1{}_{ij} + \Delta w^1{}_{ij}$. Update the synaptic weights from a node at Layer 2 to a node in Layer 3, as $\Delta w^2{}_{jd} = -\eta\delta^2{}_d y_j$, and apply, $w^2{}_{jd} = w^2{}_{jd} + \Delta w^2{}_{jd}$.

Step 6. Calculate MSE by Equation (10) as follows:

$$MSE = \frac{1}{N_{ts}} \sum_{m=1,\ l=1}^{M,\ 0.15L} \left(y^3_{m,l} - t_{m,l}\right)^2, \tag{10}$$

where N_{ts} is total number of test samples, and L is multiplied by 0.15, as 15% of the data from each class in matrix X are utilized for validation.

Step 7. Repeat from Step 2 until the error limit or the number of the epoch limit is reached.

After training and validation, 15% remaining experimental data samples are passed through the network to test the network classification performance.

Assumptions:

- w^1_{ij} are weights from node *i* of layer 1 (input layer) to node *j* of layer 2 (hidden layer),
- w^2_{jd} are weights from node *j* of layer 2 to node d of layer 3 (output layer),
- hidden layer neurons' activation functions are sigmoid transfer functions,
- y^2_j is the output from node *j* of layer 2,
- $y^3_{l,m}$ is the output from the output layer,
- $t_{l,m}$ is the corresponding target,
- η is the learning rate,

Biases to any neuron in Layer 2 are considered to be 1 (not shown in the algorithm and in Figure 5 for simplicity.) The computational cost of MLPNN is $O(I^2_{\mathrm{MLPNN}})$where I_{MLPNN} is the total number of neurons in the MLPNN.

2.6.5. Generalized Regression Neural Network

A GRNN [17], a supervised neural network model, has three main layers namely, input, hidden, and output layers, as shown in Figure 6. At the beginning of the training phase, the synaptic weights (w^1_{ij}) from the input to the hidden layer neurons are initialized to the training data vectors and the synaptic weights (w^2_{jd}) from the hidden layer neurons to the output neurons are initialized to the targets corresponding to the training data. An input is fed by the input layer to the neurons in the hidden layer also known as pattern layer or radial basis layer. At the hidden layer, the Euclidean distances (weights) from the training data to the test data are calculated and processed by a Gaussian activation function to produce hidden layer outputs. A larger Euclidean distance results in smaller weights, and a smaller distance results in larger weights. At the output layer, the weighted summation of the hidden layer outputs (normalized by the summation of the hidden layer outputs) are calculated. This normalized output is passed through the linear activation function and the final output, i.e., the class of the test data, is achieved.

The GRNN training algorithm is given as follows:

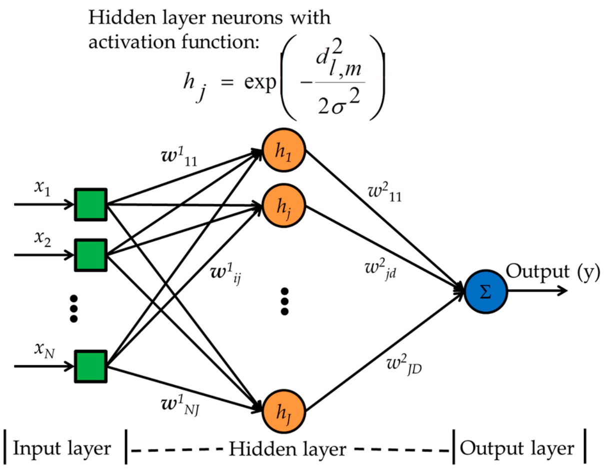

Figure 6. A generalized regression neural network block diagram. $d^2_{l,m}$ is Euclidean distance and σ is the spreading factor.

Step 1. Separate all the experimental samples in matrix **X** into 70% training samples, 15% validation samples, and 15% test samples.

Step 2. Initialize the synaptic weights from the input nodes to the hidden layer neurons to the training samples, and the weights from the hidden layer neurons to the output neuron with the corresponding training targets in vector **t**.

Step 3. Find the outputs to validation inputs $\mathbf{x}_{v,m}$ by Equation (11) as:

$$y(\mathbf{x}_{v,m}) = \frac{\sum\limits_{l,m} t_{l,m} \exp\left(-\frac{d^2_{l,m}}{2\sigma^2}\right)}{\sum\limits_{l,m} \exp\left(-\frac{d^2_{l,m}}{2\sigma^2}\right)}, \tag{11}$$

where $\mathbf{v} = 1, 2, \ldots, 0.15L$, is the validation data index in class m, σ is the spreading factor, and $d^2{}_{l,m} = (\mathbf{x}_{v,m} - \mathrm{x}_{l,m})^{\mathrm{T}}(\mathbf{x}_{v,m} - \mathbf{x}_{l,m})$, $[(.)^{\mathrm{T}}$ indicates transpose] is the square Euclidean distance from a training sample $\mathbf{x}_{i,m}$ to a validation sample $\mathbf{x}_{v,m}$.

Step 4. Calculate the MSE as:

$$E = \sum_{v,m} (y(\mathbf{x}_{v,m}) - t_{v,m})^2. \tag{12}$$

Step 5. If $E > E_{threshold}$, then adjust σ, and continue to Step 3, otherwise stop.

The computational cost of GRNN is $O(I_{\mathrm{GRNN}})$ where I_{GRNN} is the total number of neurons in the GRNN.

2.6.6. Radial Basis Function Neural Network

The design of a supervised RBFNN [18–22] is similar to the GRNN. However, the weights from the hidden layer neurons to the output neuron, $\mathbf{w}^2$, are not the true targets, but are initialized to small random numbers when the training begins, and are updated to optimal values during the training phase to reduce output error. The RBFNN algorithm is presented below.

Step 1. Choose a value for σ ranging randomly from 0 to 1.

Step 2. Choose random weights for the weights from the hidden layer neurons to the output neuron, i.e., w2.

Step 3. Calculate the entries of the Φ matrix, where Φ is defined as below:

$$\boldsymbol{\Phi} = \begin{bmatrix} \exp\left(-\frac{\|\mathbf{x}_{1,1}-\mathbf{x}_{1,1}\|^2}{2\sigma^2}\right) & \cdots & \exp\left(-\frac{\|\mathbf{x}_{1,1}-\mathbf{x}_{L,M}\|^2}{2\sigma^2}\right) \\ \vdots & \ddots & \vdots \\ \exp\left(-\frac{\|\mathbf{x}_{L,M}-\mathbf{x}_{1,1}\|^2}{2\sigma^2}\right) & \cdots & \exp\left(-\frac{\|\mathbf{x}_{L,M}-\mathbf{x}_{L,M}\|^2}{2\sigma^2}\right) \end{bmatrix}, \tag{13}$$

where, $\mathbf{x}$ are N dimensional vectors.

Step 4. Calculate the weight vector, w2 = $\Phi - 1$t.

Step 5. Calculate output, y = w2Φ.

Step 6. Evaluate, mean squared error, $E = \sum\limits_{v,m} (y(\mathbf{x}_{v,m}) - \mathbf{t}_{v,m})^2$

Step 7. If E > Ethreshold, change σ and continue to step 4 to adjust w2, otherwise exit.

The computational cost of RBFNN is $O(I_{\mathrm{RBFNN}})$ if optimal weights are found in one epoch. If the number of epochs are high, the complexity can be up to $O(I^2{}_{\mathrm{RBFNN}})$, where I_{RBFNN} is the total number of neurons in the RBFNN and is equal to I_{GRNN}.

2.6.7. Minimum-Maximum-Mean Hyperspheric Classification Method

The MMM method, a simplified hyperspheric classification method proposed in this paper for the E-Nose, is based on maximum, minimum, and mean responses of the sensors for different classes. This proposed MMM classification method is a supervised model. During training, the maximum vectors ($\vec{\mathbf{q}}_{C_n}$), minimum vectors ($\vec{\mathbf{v}}_{C_n}$), and mean vectors ($\vec{\mathbf{u}}_{C_n}$) are calculated from each class from the training data samples, and stored in matrices **Q**, **V**, and **U**, respectively as:

$$\mathbf{Q} = \begin{bmatrix} \vec{q}_1 \\ \vdots \\ \vec{q}_{C_n} \\ \vdots \\ \vec{q}_{C_N} \end{bmatrix}_{C_N \times 1} = \begin{bmatrix} q_{C_1,S_1} & \cdots & q_{C_1,S_N} \\ \vdots & \ddots & \vdots \\ q_{C_N,S_1} & \cdots & q_{C_N,S_N} \end{bmatrix}_{C_N \times S_N}, \tag{14}$$

$$\mathbf{V} = \begin{bmatrix} \vec{v}_1 \\ \vdots \\ \vec{v}_{C_n} \\ \vdots \\ \vec{v}_{C_N} \end{bmatrix}_{C_N \times 1} = \begin{bmatrix} v_{C_1,S_1} & \cdots & v_{C_1,S_N} \\ \vdots & \ddots & \vdots \\ v_{C_N,S_1} & \cdots & v_{C_N,S_N} \end{bmatrix}_{C_N \times S_N}, \tag{15}$$

$$\mathbf{U} = \begin{bmatrix} \vec{u}_1 \\ \vdots \\ \vec{u}_{C_n} \\ \vdots \\ \vec{u}_{C_N} \end{bmatrix}_{C_N \times 1} = \begin{bmatrix} u_{C_1,S_1} & \cdots & u_{C_1,S_N} \\ \vdots & \ddots & \vdots \\ u_{C_N,S_1} & \cdots & u_{C_N,S_N} \end{bmatrix}_{C_N \times S_N}. \tag{16}$$

In Equations (14)–(16), C_N and S_N are the number of classes and number of sensors, respectively. The maximum, minimum, and mean vectors are the rows of the matrices **Q**, **V**, and **U**, respectively. The columns of the matrices **Q**, **V**, and **U** represent sensor variables. A test data vector is compared element by element, i.e., every feature is compared to the maximum and minimum vectors of each class. A test data is classified to a class for which the following two criteria satisfy: (i) every element (feature variable) of the test data vector is less than, or equal to every element of the maximum vector of the corresponding class, and (ii) every element of the test data vector is greater than, or equal to every element of the minimum vector of the corresponding class. The criteria (i) and (ii) ensure the minimum-maximum range assessment for each sensor by the logical 'AND' operation between all the sensor variables. The data are assigned to the classes for which the logical examinations are true. If any odor data is classified to multiple classes, the Euclidean distances between the test data and the class mean vectors of the corresponding tied classes in Matrix **U** are calculated. The test data are assigned to the class in which the mean vector is the closest to the test data vector. The algorithm is given as follows:

Step 1. Compute the maximum, minimum, and mean matrices **Q**, **V**, and **U** from the whole dataset in **X**, where each row of the matrices **Q**, **V** and **U** corresponds to a class and each column corresponds to a sensor variable.

Step 2. Compare each feature of a test data vector with each feature of the maximum and minimum vectors of all the classes.

Step 3. Assign test data to a class for which the following two criteria satisfy: (i) each feature of the test data vector is less than or equal to the corresponding feature of the maximum vector of the corresponding class, and (ii) each feature of the test data vector is greater than or equal to each element of the minimum vector of the corresponding class. If the test data do not fall within the maximum-minimum range of any of the classes, then label it as "unclassified" or "correctly rejected", and stop.

Step 4. If any test data are within maximum-minimum limits of multiple classes, then the case of a tie occurs. To break this tie, the test data are assigned to the class whose mean vector (measured by Euclidean distance metric) is the closest to the test data vector. Once the test data are assigned to a class, exit the program.

Step 5. Run the steps from 1 to 4 for the validation dataset and calculate the percentage of error by $E_{MMM} = \sum_{l=1,m=1}^{0.15L,0.15M} (y_{l,m} - t_{l,m})^2$ If the error $E_{MMM} \leq E_{threshold}$ then exit, otherwise add $\eta \times \sigma_m$ to the corresponding rows of $\mathbf{Q}$ and deduct from mth row of $\mathbf{V}$ to expand the hyperspheric classification boundary and continue to Step 1, where, η is the learning rate, and σ_m is the standard deviation vector of class m.

All sensors in the sensor array are required to function properly. A malfunctioning sensor should be detected and replaced, and the E-Nose should be trained again to achieve good classification performance. Each of the mean, minimum, and maximum matrices has a computational cost of $O(M)$. If a tie occurs, the complexity becomes $O(M + I_{\text{ties}}(N + k))$, where Ities is the number of tied classes. The latter terms are added as per k-NN theory to break the tie, and the class with closest mean is needed to be found by the k-NN method. Due to the E-Nose data pattern, ties are less likely to occur, and thereby the computational complexity remains low.

3. Results

Electric responses, i.e., voltage changes across the sensor load resistors of the E-Nose sensor panel to the odors from four kinds of fruits, i.e., banana, mango, sapodilla, and pineapple, each at their three ripeness states forming twelve classes, were recorded for this research. One set of recorded time versus voltage response for ripe sapodilla is shown in Figure 7. Each set of response comprised responses from eight sensors. The data samples were prepared by averaging the steady state voltage responses from the sensors. The classification algorithms were trained with 70% of the data samples from banana, sapodilla, and pineapple, each at three different ripeness states (nine classes), and 30% of the data samples from each of the nine classes were used for validation and testing. Data samples from mango odor at three ripeness states were considered as three irrelevant classes, and were used to test false classification and correct rejection performances of the classification algorithms.

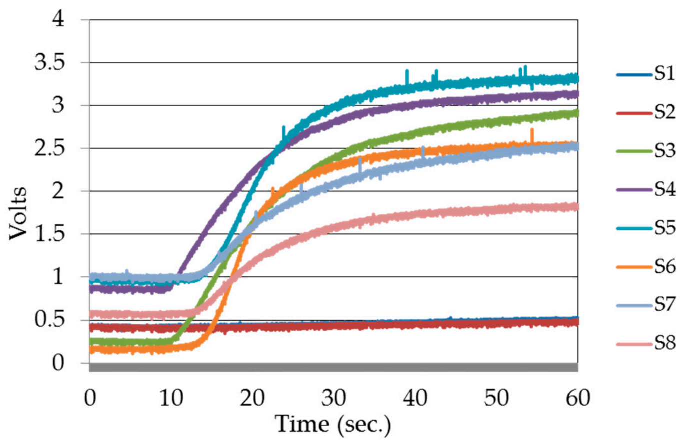

Figure 7. Time versus voltage responses of eight sensors to rotten sapodilla. S1 to S8 are sensor labels as listed in Table 1.

A three dimensional PCA scores plot of banana, mango, sapodilla, and pineapple at three ripeness states in Figure 8 shows that few samples of green banana and green mango, ripe mango and rotten mango, rotten banana and green sapodilla, rotten banana and rotten sapodilla overlapped with each other.

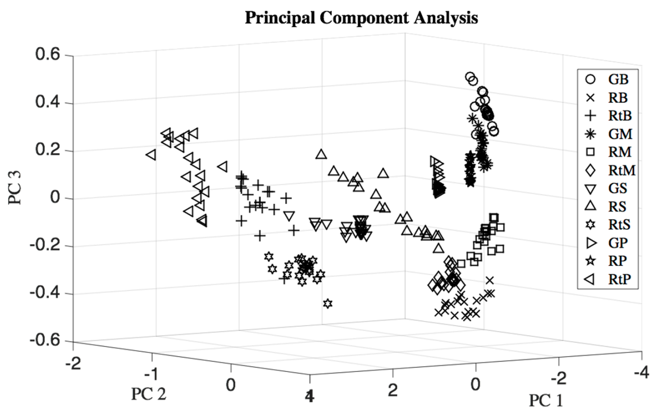

Figure 8. Three dimensional principal component analysis (PCA) scores plot of four types of fruits each at three ripeness states: GB = green banana, RB = ripe banana, RtB = rotten banana, GM = unripe mango, RM = ripe mango, RtM = rotten mango, GS = unripe sapodilla, RS = ripe sapodilla, RtS = rotten sapodilla, GP = unripe pineapple, RP = ripe pineapple, and RtP = rotten pineapple.

The signature patterns as depicted in Figure 9 were prepared by the mean responses of nine trained classes, i.e., green banana, ripe banana, rotten banana, green sapodilla, ripe sapodilla, rotten sapodilla, green pineapple, ripe pineapple, and rotten pineapple. The signatures were composed of eight features corresponding to the eight sensors. It was seen that the signatures were significantly different from each other. The responses of the sensors S1 and S2 were small, having insignificant variation for different classes. In contrast, the sensors S3 to S8 showed significant variations in responses for the different classes, and produced distinct signature patterns for different fruit odors and their ripeness states.

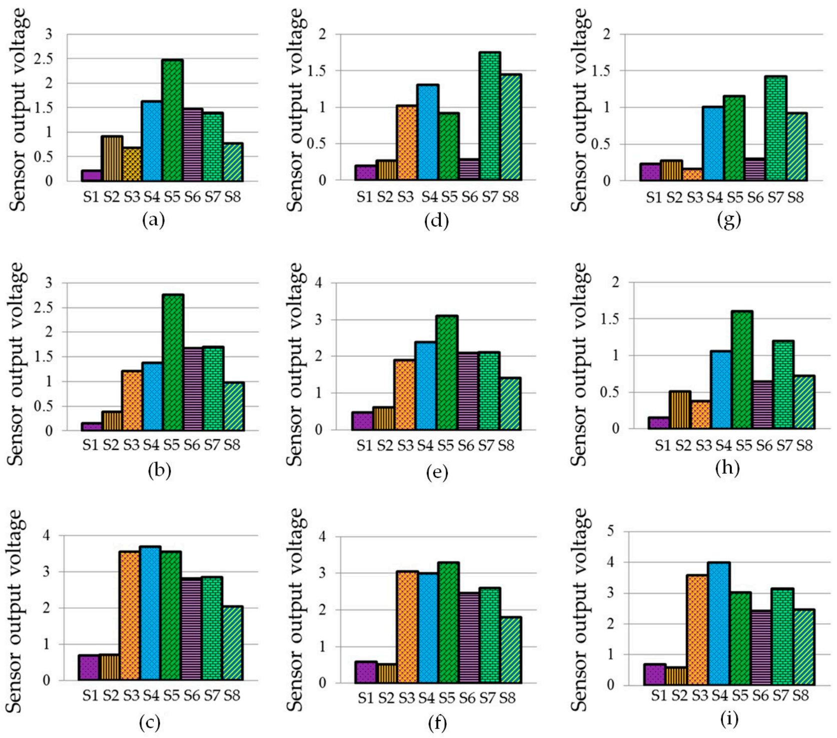

Figure 9. Signature patterns of the means of three types of fruits at three ripeness states: (**a**) green banana; (**b**) ripe banana; (**c**) rotten banana; (**d**) green sapodilla; (**e**) ripe sapodilla; (**f**) rotten sapodilla; (**g**) green pineapple; (**h**) ripe pineapple, and (**i**) rotten pineapple.

For training, validation, and testing, 70%, 15%, and 15% of the data samples were chosen from the trained classes in equal numbers, respectively, to train, validate, and test the MLPNN, RBFNN, GRNN, and MMM algorithms. Whereas, to train the *k*-NN and SVM algorithms, 70% of data samples were used for training, and 30% of data samples were used for testing as training; these algorithms did not need a validation step.

Training and testing time taken by different classification algorithms are shown in Table 2. The proposed MMM algorithm required minimum training time, followed by the *k*-NN, GRNN, SVM, MLPNN, and RBFNN algorithms, respectively. The testing time taken by the MMM classification method was also minimal, followed by the MLPNN, RBFNN, SVM, GRNN, and *k*-NN algorithms, respectively. Thus, training the proposed classification algorithm and classifying any test data was faster compared to the MLPNN, RBFNN, SVM, GRNN, and *k*-NN algorithms. The training and testing were faster with the MMM method, as less calculations were required to find the minimum, maximum, and mean vectors for each class, and the compression or expansion factor of the classification boundaries. The big '*O*' complexities of the algorithms, according to Section 2.6, were $O(LM(N+k))$ for *k*-NN, within $O(NL^2M^2)$ to $O(NL^3M^3)$ for SVM, $O(I^2{}_{\mathrm{MLPNN}})$ for MLPNN, $O(I_{\mathrm{GRNN}})$ for GRNN, within $O(I_{\mathrm{RBFNN}})$ to $O(I^2{}_{\mathrm{RBFNN}})$ for RBFNN, and within $O(M)$ to $O(M + I_{ties}(N+k)$ for the MMM method. This was the case because MLPNN and SVM both had square terms, which implied more computations and thereby more complexity. Although GRNN and RBFNN had low orders of complexity, as they need as many neurons as there were training data, the design complexity was also high. *k*-NN needed to be applied within the MMM method to break ties, and I_{ties} were less likely to occur and were usually much smaller compared to *LM*, keeping the complexity of the MMM method small. Thus, the MMM method took less time compared to other algorithms analyzed in this research.

Table 2. Training and testing time taken by the algorithms to train and test with the data samples of banana, sapodilla, and pineapple, each at three ripeness states.

Algorithm	Big *O* Complexity	Train Time (s)	Test Time (s)
k-NN	$O(LM(N+k))$	0.2175	0.2175
SVM	$O(NL^2M^2)$ to $O(NL^3M^3)$	0.7452	0.0461
GRNN	$O(I_{\mathrm{GRNN}})$	0.3922	0.0946
RBFNN	$O(I_{\mathrm{RBFNN}})$ to $O(I^2{}_{\mathrm{RBFNN}})$	0.9922	0.0230
MLPNN	$O(I^2{}_{\mathrm{MLPNN}})$	0.8652	0.0153
MMM	$O(M)$ to $O(M + I_{ties}(N+k))$	0.1874	0.0047

Test data from the trained classes are classified by the classification algorithms to compare their misclassification and correct classification performances are averaged over 10 simulations and summarized in Table 3. The proposed MMM, GRNN, *k*-NN, and SVM classification methods showed 1.8519% classification errors, while misclassification errors with the RBFNN and MLPNN methods were 24.0740% and 29.6296%, respectively.

In a one-way multiple analysis of variance (MANOVA) test, the maximum *p*-value of 1.1088×10^{-10} (which is very small compared to 5% significance level) occurred for eight dimensions. This indicates that the eight dimensional mean vectors of the twelve classes were obviously distinguishable from each other. This was the case because the MANOVA test assumes the variance-covariance matrix is the same for each population, which is not exactly true for real scenarios. Thus, the means were distinguishable, but the dataset had a few occurrences of minor overlap, as seen in the PCA plot in Figure 8 causing classification errors. In Table 3 it was seen that false negative errors occurred for different classification models.

Table 3. Misclassification error and correct classification rate of the classification algorithms during testing with validation and test data set from trained classes, i.e., banana, sapodilla, and pineapple each at three ripeness states, with nine classes in total. Results are averaged over 10 simulations. In the table *k*-NN, SVM, GRNN, RBFNN, MLPNN, and MMM are abbreaviated forms of *k*-nearest neighbor, support vector machine, generalized regression neural network, radial basis function neural network, multilayer perceptron neural network, and minimum-maximum-mean, respectively.

Algorithm	**False Negative (Misclassification Error)**		**True Positive (Correct Classification)**	
	Number of Test Samples Misclassified/Total Number of Test Samples	**Percent**	**Number of Test Samples Correctly Classified/Total Number of Test Samples**	**Percent**
k-NN	1/54 [1]	1.8519	53/54	98.1481
SVM	1/54	1.8519	53/54	98.1481
GRNN	0.5/27 [1]	1.8519	26.5/27	98.1481
RBFNN	6.5/27	24.0740	20.5/27	75.9260
MLPNN	8/27	29.6296	19/27	70.3704
MMM	0.5/27	1.8519	26.5/27	98.1481

[1] 54 and 27 are the sizes of test dataset for different algorithms performance test.

The number of samples should not be too small compared to the number of features. In this research, the number of features was eight, and for each class, 20 samples were collected. To verify over-fitting of the classification models, the R-square statistics of the target outcomes from the models were evaluated for training, validation, and test datasets are listed in Table 4. The statistics showed that R-squared values were smaller for MLPNN and RBFNN classification models compared to *k*-NN, SVM, GRNN, and MMM classification models. These statistics validate the classification performances presented in Table 3.

Table 4. R-square values of different algorithms for training, validation, and test dataset. In the table *k*-NN, SVM, MLPNN, RBFNN, GRNN, and MMM are abbreaviated forms of *k*-nearest neighbor, support vector machine, multilayer perceptron neural network, radial basis function neural network, generalized regression neural network, and minimum-maximum-mean, respectively.

Dataset	***k*-NN**	**SVM**	**MLPNN**	**RBFNN**	**GRNN**	**MMM**
Training	0.9827	0.9889	0.9349	0.9503	0.9889	0.9867
Validation	0.9801	0.9872	0.9217	0.9314	0.9805	0.9872
Test	0.9819	0.9825	0.9258	0.9361	0.9815	0.9803

False classification and correct rejection performances of the classification algorithms analyzed in this paper are summarized in Table 5. These analyses have not been evaluated in literatures yet, to the best of our knowledge. It was expected that the 60 samples, i.e., 100% odor data from the irrelevant (i.e., mango odor data at three ripeness states for this research) classes should have been correctly rejected by an E-Nose, and thereby would not produce false classification errors. The *k*-NN and SVM algorithms falsely classified all the data from irrelevant classes and produced false alarms. The RBFNN algorithm falsely classified 15% of the mango data to trained classes and misclassified 85% to unknown extraneous classes. The MLPNN algorithm falsely classified 35% of the data samples and misclassified 65% to unknown extraneous classes. It was seen that the proposed MMM method (with each feature value of each class in the maximum matrix increased by 11.11% of the corresponding class's standard deviations) and the GRNN method (with a spreading factor of 0.03) do not show any false classification error, and all irrelevant data samples were correctly rejected.

Table 5. Classification performance of the algorithms with 60 irrelevant data samples (i.e., data from mango ripening stages). In the table *k*-NN, SVM, GRNN, RBFNN, MLPNN, and MMM are abbreaviated forms of *k*-nearest neighbor, support vector machine, generalized regression neural network, radial basis function neural network, multilayer perceptron neural network, and minimum-maximum-mean, respectively.

Classification Method	Misclassification Of Irrelevant Data				True Negative (Irrelevant Data Classified to No Trained or Irrelevant Classes i.e., Correctly Rejected)	
	False Positive (Irrelevant Data Classified to Trained Classes)		Irrelevant Data Classified to Unknown Classes			
	Number of Samples	Percent	Number of Samples	Percent	Number of Samples	Percent
k-NN	60	100	0	0	0	0
SVM	60	100	0	0	0	0
GRNN	0	0	0	0	60	100
RBFNN	9	15	51	85	0	0
MLPNN	21	35	39	65	0	0
MMM	0	0	0	0	60	100

4. Discussion

In this paper classification performances of the *k*-NN, SVM, MLPNN, RBFNN, GRNN, and the proposed MMM classification algorithms were compared in terms of computation speed, correct classification rate, and false classification rate. Correct classification performances experienced by different algorithms in this paper are shown in Table 3. These results were consistent with previous works, with true positive, i.e., correct classification accuracy ranges from 82.4% to 100% with *k*-NN [10–13], 86% to 98.66% with SVM [11–16], 100% with GRNN [17], 88% to 100% with RBFNN [18–22], and 68% to 100% with MLPNN [9,20–24]. With the MMM classification method, 98.1481% classification accuracy has been achieved.

Odor data from an irrelevant class should not be classified to any trained class by an E-Nose, and thereby produce no false alarm. False classification performances of the existing classification methods have not yet been analyzed in the literature. To reduce false classification and misclassification errors and improve correct rejection performance, classification algorithms with a hyperspheric boundary were used. It is seen from Table 5 that the hyperspheric classification algorithm, i.e., GRNN with Gaussian activation function, showed zero false classification errors. In contrast, RBFNN, another hyperspheric classification method with Gaussian activation functions in the hidden layer neurons, showed 15% false classification errors. Compared to this, the proposed MMM classification method was found to be efficient for E-Nose data classification. It is seen from the analysis as tabulated in Table 3 that the MMM method classified the test data of trained classes with only a 1.8519% misclassification error, i.e., false negative error. In addition, and similar to the GRNN, the MMM method correctly rejected data samples of irrelevant classes and thereby did not produce any false alarm. In addition, the implementation complexity, training, and testing duration of the MMM method was less compared to other methods analyzed in this paper. Thus, the MMM method could be a prominent method for E-Nose application, to improve classification rate of the data samples which belong to training classes, reduce false classification rates, and increase correct rejection rates of the data samples from irrelevant classes.

As the design of a GRNN is complex and expensive due to its high level of neuron requirement, a simple hyperspheric classification method based on minimum, maximum, and mean (MMM) values of the training dataset was proposed in this paper. It was seen that the MMM algorithm is simple, fast, and highly accurate for classifying data of trained classes and correctly rejecting data of extraneous odors. Although the MMM and GRNN methods showed similar misclassification and false classification performance, the MMM method is promising for E-Nose applications because of the simplicity and speed of its implementation.

Acknowledgments: The authors thank, (1) the Thailand Office of Higher Education Commission (NRU Project) and (2) the Thammasat University Research Fund (Contract Number 2/22/2557) for financial support.

Author Contributions: Mohammad Mizanur Rahman, Chalie Charoenlarpnopparut, Prapun Suksompong, and Pisanu Toochinda conceived and designed the experiments; Mohammad Mizanur Rahman performed the experiments; Mohammad Mizanur Rahman, Chalie Charoenlarpnopparut, and Prapun Suksompong analyzed the data; Mohammad Mizanur Rahman, Chalie Charoenlarpnopparut, Prapun Suksompong, and Attaphongse Taparugssanagorn contributed analysis tools; Mohammad Mizanur Rahman and Attaphongse Taparugssanagorn wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Persaud, K.; Dodd, G. Analysis of discrimination mechanisms in the mammalian olfactory system using a model nose. *Nature* **1982**, *299*, 352–355. [CrossRef] [PubMed]
2. Berna, A. Metal oxide sensors for electronic noses and their application to food analysis. *Sensors* **2010**, *10*, 3882–3910. [CrossRef] [PubMed]
3. Omatu, S.; Yoshioka, M. Electronic nose for a fire detection system by neural networks. *IFAC Proc. Vol.* **2009**, *42*, 209–214. [CrossRef]
4. Emmanuel, S.; Pisanelli, A.M.; Persaud, K.C. Development of an electronic nose for fire detection. *Sens. Actuator B Chem.* **2006**, *116*, 55–61. [CrossRef]
5. Reimann, P.; Schutze, A. Fire detection in coal mines based on semiconductor gas sensors. *Sens. Rev.* **2012**, *32*, 47–58. [CrossRef]
6. Kwan, C.; Schmera, G.; Smulko, J.M.; Kish, L.B.; Heszler, P.; Granqvist, C.G. Advanced agent identification with fluctuation-enhanced sensing. *IEEE Sens. J.* **2008**, *8*, 706–713. [CrossRef]
7. Gomez, A.H.; Wang, J.; Hu, G.; Pereira, A.G. Discrimination of storage shelf-life for mandarin by electronic nose technique. *LWT Food Sci. Technol.* **2007**, *40*, 681–689. [CrossRef]
8. Gomez, A.H.; Wang, J.; Hu, G.; Pereira, A.G. Monitoring storage shelf life of tomato using electronic nose technique. *J. Food Eng.* **2008**, *85*, 625–631. [CrossRef]
9. Yu, H.; Wang, J. Discrimination of LongJing green-tea grade by electronic nose. *Sens. Actuator B Chem.* **2007**, *122*, 134–140. [CrossRef]
10. Tang, K.T.; Chiu, S.W.; Pan, C.H.; Hsieh, H.Y.; Liang, Y.S.; Liu, S.C. Development of a portable electronic nose system for the detection and classification of fruity odors. *Sensors* **2010**, *10*, 9179–9193. [CrossRef] [PubMed]
11. Guney, S.; Atasoy, A. Multiclass classification of n-butanol concentrations with k-nearest neighbor algorithm and support vector machine in an electronic nose. *Sensor Actuat. B Chem.* **2012**, *166–167*, 721–725. [CrossRef]
12. Shao, X.; Li, H.; Wang, N.; Zhang, Q. Comparison of different classification methods for analyzing electronic nose data to characterize sesame oils and blends. *Sensors* **2015**, *15*, 26726–26742. [CrossRef] [PubMed]
13. Guney, S.; Atasoy, A. Classification of n-butanol concentrations with k-NN algorithm and ANN in electronic nose. In Proceedings of the 2011 International Symposium on Innovations in Intelligent Systems and Applications (INISTA), Istanbul, Turkey, 15–18 June 2011; pp. 138–142.
14. Khalaf, W.; Pace, C.; Gaudioso, M. Gas detection via machine learning. *Int. J. Comput. Electr. Autom. Control Inf. Eng.* **2008**, *2*, 61–65. Available online: http://waset.org/Publication/gas-detection-via-machine-learning/10785 (accessed on 5 August 2016).
15. Amari, A.; Barbri, N.E.; Llobet, E.; Bari, N.E.; Correig, X.; Bouchikhi, B. Monitoring the freshness of Moroccan Sardines with a neural-network based electronic nose. *Sensors* **2006**, *6*, 1209–1223. [CrossRef]
16. Sanaeifar, A.; Mohtasebi, S.S.; Varnamkhasti, M.G.; Ahmadi, H.; Lozano, J. Development and application of a new low cost electronic nose for the ripeness monitoring of banana using computational techniques (PCA, LDA, SIMCA, and SVM). *Czech J. Food Sci.* **2014**, *32*, 538–548. Available online: http://www.agriculturejournals.cz/publicFiles/138030.pdf (accessed on 30 August 2016).
17. Kurup, P.U. An electronic nose for detecting hazardous chemicals and explosives. In Proceedings of the 2008 IEEE Conference on Technologies for Homeland Security, Waltham, MA, USA, 12–13 May 2008; pp. 144–149.

18. Xiong, Y.; Xiao, X.; Yang, X.; Yan, D.; Zhang, C.; Zou, H.; Lin, H.; Peng, L.; Xiao, X.; Yan, Y. Quality control of Lonicera japonica stored for different months by electronic nose. *J. Pharm. Biomed.* **2013**, *91*, 68–72. [CrossRef] [PubMed]
19. Evans, P.; Persaud, K.C.; McNeish, A.S.; Sneath, R.W.; Hobson, N.; Magan, N. Evaluation of a radial basis function neural network for the determination of wheat quality from electronic nose data. *Sens. Actuator B Chem.* **2000**, *69*, 348–358. [CrossRef]
20. Dutta, R.; Hines, E.L.; Gardner, J.W.; Udrea, D.D.; Boilot, P. Non-destructive egg freshness determination: An electronic nose based approach. *Meas. Sci. Technol.* **2003**, *14*, 190–198. [CrossRef]
21. Dutta, R.; Kashwan, K.R.; Bhuyan, M.; Hines, E.L.; Gardner, J.W. Electronic nose based tea quality standardization. *Neural Netw.* **2003**, *16*, 847–853. [CrossRef]
22. Borah, S.; Hines, E.L.; Leeson, M.S.; Iliescu, D.D.; Bhuyan, M.; Gardner, J.W. Neural network based electronic nose for classification of tea aroma. *Sens. Instrum. Food Qual. Saf.* **2008**, *2*, 7–14. [CrossRef]
23. Anjos, O.; Iglesias, C.; Peres, F.; Martinez, J.; Garcia, A.; Taboada, J. Neural networks applied to discriminate botanical origin of honeys. *Food Chem.* **2015**, *175*, 128–136. [CrossRef] [PubMed]
24. Llobet, E.; Hines, E.L.; Gardner, J.W.; Bartlett, P.N.; Mottram, T.T. Fuzzy ARTMAP based electronic nose data analysis. *Sens. Actuator B Chem.* **1999**, *61*, 183–190. [CrossRef]
25. Cooper, P.W. The hypersphere in pattern recognition. *Inform. Control* **1962**, *5*, 324–346. [CrossRef]
26. Hwang, J.N.; Choi, J.J.; Oh, S.; Marks, R.J. Classification boundaries and gradients of trained multilayer perceptrons. In Proceedings of the 1990 IEEE International Symposium on Circuits and Systems, New Orleans, LA, USA, 1–3 May 1990; pp. 3256–3259.
27. Tax, D.M.J.; Duin, R.P.W. Support vector domain description. *Pattern Recogn. Lett.* **1999**, *20*, 1191–1199. [CrossRef]
28. Saevels, S.; Lammertyn, J.; Berna, A.Z.; Veraverbeke, E.A.; Di Natale, C.; Nicolai, B.M. An electronic nose and a mass spectrometry-based electronic nose for assessing apple quality during shelf life. *Postharvest Biol. Technol.* **2004**, *31*, 9–19. [CrossRef]
29. Brezmes, J.; Llobet, E.; Vilanova, X.; Saiz, G.; Correig, X. Fruit ripeness monitoring using an electronic nose. *Sens. Actuator B Chem.* **2000**, *69*, 223–229. [CrossRef]
30. Zhang, H.; Chang, M.; Wang, J.; Ye, S. Evaluation of peach quality indices using an electronic nose by MLR, QPST and BP network. *Sens. Actuator B Chem.* **2008**, *134*, 332–338. [CrossRef]
31. Guohua, H.; Yuling, W.; Dandan, Y.; Wenwen, D.; Linshan, Z.; Lvye, W. Study of peach freshness predictive method based on electronic nose. *Food Control* **2012**, *28*, 25–32. [CrossRef]
32. Benedetti, S.; Buratti, S.; Spinardi, A.; Mannino, S.; Mignani, I. Electronic nose as a non-destructive tool to characterise peach cultivars and to monitor their ripening stage during shelf-life. *Postharvest Biol. Technol.* **2008**, *47*, 181–188. [CrossRef]
33. Li, C.; Krewer, G.W.; Ji, P.; Scherm, H.; Kays, S.J. Gas sensor array for blueberry fruit disease detection and classification. *Postharvest Biol. Technol.* **2010**, *55*, 144–149. [CrossRef]
34. Simon, J.E.; Hetzroni, A.; Bordelon, B.; Miles, G.E.; CHARLES, D.J. Electronic sensing of aromatic volatiles for quality sorting of blueberries. *J. Food Sci.* **1996**, *61*, 967–970. [CrossRef]
35. Brezmes, J.; Llobet, E.; Vilanova, X.; Orts, J.; Saiz, G.; Correig, X. Correlation between electronic nose signals and fruit quality indicators on shelf-life measurements with pinklady apples. *Sens. Actuator B Chem.* **2001**, *80*, 41–50. [CrossRef]
36. Huerta, R.; Mosqueiro, T.; Fonollosa, J.; Rulkov, N.F.; Rodriguez-Lujan, I. Online decorrelation of humidity and temperature in chemical sensors for continuous monitoring. *Chemom. Intell. Lab. Syst.* **2016**, *157*, 169–176. [CrossRef]
37. Peris, M.; Escuder-Gilabert, L. A 21st century technique for food control: Electronic noses. *Anal. Chim. Acta* **2009**, *638*, 1–15. [CrossRef] [PubMed]
38. Kuhn, H.W.; Tucker, A.W. Nonlinear Programming. In Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, USA, 31 July–12 August 1950; University of California Press: Berkeley, CA, USA, 1951; pp. 481–492.