On Genetic Algorithm and Multiple Preprocessors Assisted Feature Boosting for Electronic Nose Signal Processing

Prabha Verma¹, Divya Somvanshi² and R.D.S. Yadava³

Sensors & Signal Processing Laboratory, Department of Physics, Faculty of Science, Banaras Hindu University, Varanasi 221005, India.
E-mail: ¹pverma.bhu@gmail.com, ²somvanshi.divya@gmail.com, ³ardius@gmail.com

Abstract

The paper presents a method for feature extraction that explores data space through different preprocessing strategies in combination with principal component analysis (PCA) and genetic algorithm (GA). A preprocessor/PCA combination transforms data space into feature space; a change in preprocessor results in an alternate feature space. The proposed method first fuses the feature spaces by simple concatenation of the alternate feature vectors then allows them to evolve genetically. The genetic evolution of each fused feature vector is based on treating the feature vector as chromosome and the feature components as genes. The initial population is created on the basis of a probability distance metric. The fitness and ranking is done by using PCA generated variances as measure of significance. In the terminal population the frequency of a gene (principal component) occurrence is interpreted as a measure of its significance in defining the feature vector. Finally, the feature components are given additional weight according to $z_{i,j} = z_i (1 - p_j \log_2 p_j)$ where $z_{i,j}$ denotes $j$-th feature component of $i$-th sample in fused feature space and $p_j$ denotes the probability of $j$-th component occurring in the terminal population. In order to demonstrate the efficacy of this idea we employed only two well known preprocessing methods: vector autoscaling and dimensional autoscaling. The feature vectors defined in this manner were used as input a radial basis neural network classifier for validation. Several benchmark datasets (both chemical and non-chemical) available from open sources were used in the analysis for validation. It is found that the scheme of feature fusion and weighting enhances classification rate in most cases analyzed.

Keywords: Feature extraction, electronic nose, feature fusion, genetic algorithm, pattern recognition.
Introduction

An electronic nose instrument employs an array of microelectronic sensors that records chemical fingerprints of odor samples [1], [2]. Its pattern recognition system processes sensor array data to generate specific odorprints for identification in a manner analogous to the biological processing of odor response in human smell sensing organ [3], [4], [5], [6], [7]. The sensors array measures the odor variables. The sensors represent data variables, and their outputs contain the object identity information encoded in some manner not known a priori. Mathematically, the set of multisensor output represents the object identity by a vector (called pattern vector) in a multidimensional space whose dimensions are the individual sensors in the array. The measurements are usually noisy, span over wide dynamic range, contain outliers and suffer from various instabilities in the sensor system [3], [7], [8]. These limitations of sensor measurements prevent discrimination of objects directly in the measurement space, and the application of some efficient pattern recognition methods becomes necessary.

Pattern recognition from the data generated by multiple sensors proceeds in three successive stages: data preprocessing, feature extraction and pattern classification. Numerous algorithms do data processing at each stage. Specific pattern recognition tasks need to select these algorithms in some combination to obtain most accurate classification results. Often, different combinations work best in different domains. The pattern recognition algorithms, first, seek to transform the measured (raw) multisensor data into an alternate representation in such a way that information overlap among different sensors are minimized, and the individual objects get unique representation denoted by a new set of values (called features) in the transformed space. The latter defines the feature space in which the feature vectors denote the objects identities or signatures. Then, assign each signature its identity label. The identity declaration step needs a classifier, which maps the input feature vectors into class identities [9].

The present work is an extension of the earlier work done by our group [10]. In this, the authors had reported a method for feature boosting based on genetic algorithm (GA). The proposed method operated in three steps. First, several alternate feature space representations were generated by processing the measured sensor array data (input data) through different preprocessor and linear PCA combinations. The feature vectors in alternate spaces corresponding to a data sample were then concatenated to create a new high dimensional feature vector. Second, the feature components in the new fused feature space were used to define a gene pool, and the feature vector as chromosome. The chromosomes represent input data sample in fused feature space. The set of genes in a chromosome (feature set of a feature vector) are then used to create an initial population for genetic evolution of the feature vector. The creation of initial population is based on a probability distance measure introduced earlier in [11]. Then, in the third step, a genetic algorithm augments the feature components according to the evolution statistics. The probability of occurrence of a gene (feature component) was used to put an additional weight on it
in linear relation to the probability value. The new feature vector was defined by the set of weighted components in the fused feature space. Some more details of this method is presented in Section III. The validation in [10] was done by employing an error backpropagation neural network as classifier.

The present analysis differs in two ways. First, the weighting method has been modified in accordance with the concept of information content in information theory. It is defined by the Shannon’s entropy. Second, the classifier used is a radial basis function (RBF) neural network, and comparing its performance by the earlier used linear weighting system. The classifier used in the present analysis is a radial basis neural network. Motivated by the desire to establish an appropriate weighting scheme for the features based on the genetic algorithm, the present analysis compares the performance of earlier weighting method with the present one in combination with RBF network classifier and also with the earlier published result based on backpropagation neural network classifier. Section IV gives the detailed description.

Though primary target application for the present work is to develop data processing method for enhancing performance of electronic systems, the procedure developed here is of generic nature. It has been validated by analyzing data from both the chemical and non-chemical domains. An analysis of 14 data sets (6 from chemical domain and 8 from other domains) is presented in Section V. The paper completes with some discussion in Section VI and conclusion Section VII.

**Background Survey**

The preprocessing stage usually consists of several sub procedures that prepare the data for feature extraction by removal of noise, correction for drift, mean centering and normalization. The goal here is to reduce dependencies on nonidentity making factors and to accentuate dependencies on the identity making factors. At feature extraction stage, the data transformation methods combine the original sensor variables in such a way that correlations among them are eliminated, or reduced, and a new set of independent mathematical descriptors (possibly lower in number) are generated. The new vapor descriptors are sometimes called virtual sensors. The measurement space defined by the real sensors is thus transformed into feature space defined by the virtual sensors where an odor sample is represented by the virtual sensors output (or components of feature vector). Different odor classes occupy separate regions in the feature space [12]. Finally, at pattern classification stage, a classifier assigns the odor identity labels (or classes) by using some mathematical measure of separability and a prior training or a database of likely class identities. The integrated pattern recognition system maps an odor sample in measurement space to its class label. The odor identity declaration in electronic nose systems is invariably done based on training with known odor class labels [13].

The feature extraction is the most crucial stage in sensor array data processing. A proper extraction and selection of the object discriminating descriptors reduces the complexities in classifier designs, and adds to the robustness of the pattern
recognition system. The principal component analysis (PCA) is one of the most commonly used feature extraction method. It is an unsupervised linear feature extractor. The feature vectors are generated by linear combination of the preprocessed data vectors assuming that the features are uncorrelated Gaussian random variables. The feature dimensions are mutually orthogonal, and the projections of the original data onto these directions (called principal components) maximize the variance in the data structure [14]. In PCA, the feature dimensions are arranged in the order of decreasing values of their variances (eigenvalues). The variance represents information content; therefore, the order of principal components also represents their relative importance for object discrimination. For this reason, the PCA is often used for the dimensionality reduction in the large dimensionality problems by eliminating the lowest eigenvalue components. The singular value decomposition (SVD) and independent component analysis (ICA) are the two other unsupervised linear feature extraction methods. In SVD, the data space is transformed to a decorrelated feature space by rank decomposition of the data matrix, and the feature dimensions are arranged in the order of decreasing singular values [15]. The search in ICA is however, for statistically independent directions assuming that the features are non-Gaussian. In doing so, some measure of non-Gaussianity such as negentropy is maximized [16], [17]. The linear discriminant analysis (LDA) is another important linear feature extraction method. However, this is a supervised method, and requires training data. The LDA is based on within-class and between-class scatter matrices, and the feature dimensions are those that maximize separation between class means [18].

In most linear feature extraction methods, the feature space is orthogonalized, and some measure of information is maximized along the feature dimensions (e.g. variance in the PCA). The dimensionality reduction is achieved by discarding the feature components having low information content (e.g. lowest eigenvalue components in PCA, lowest singular value components in SVD, lowest negentropy components in ICA). The success of this approach however depends on the meaning of information content and its relevance to the classification problem in hand. Many researchers pointed out that it is no guarantee that the highest order feature components as per some specific definition of information content necessarily carry the most discriminating information. Prakash and Murty [19] pointed out that the first few principal components will be useful in class discrimination only if intra- and inter-class variations have the same directions of dominance, otherwise by eliminating the lowest eigenvalue components one may throw away the most useful information concerning the class separability. Cantu-Paz [20] made a similar comment that by capturing maximum variance the principal components are not necessarily useful to discriminating objects of different classes. Most researches on the feature extraction or selection algorithms have focused on the large dimensionality problems with goals to reduce dimensionality and computational cost. As a result, numerous feature extraction or selection algorithms have been reported [12]. An important approach is based on application of genetic algorithm (GA).
Standard genetic algorithm (GA) is a search and optimization process to seek solution of a multivariate problem. The GA operations mimic biological process of evolution wherein the genes represent the problem variables and the chromosome (which is a sequence of genes) represent the solution in the form of an optimum set of values of the variables. The genetic algorithm was originally formulated by Holland [21] on principles of natural selection, reproduction and survival-of-the-fittest. Since then, numerous variations of GAs are developed to suit applications in wide range of domains. A comprehensive overview of the genetic algorithms and their applications is available in [22], [23]. The operations in a GA begin by setting up an initial population of chromosomes with different gene structures as potential solutions to the problem. The genetic evolution of chromosomes is then set to start through methods of selection and reproduction analogous to biological parents bearing offspring. The transformation of basic concepts of evolution into a powerful computational tool necessitates mathematical description and representation of various biological factors and laying down the rules for chromosome combination and offspring survival. The installed chromosome population at the start is the first generation. Individual members of this population are assigned a value for their fitness, which makes the basis for their selection as a parent for reproduction. The fitness values are calculated using a fitness function that utilizes either their internal attributes (e.g., gene characteristics) or their performance in accomplishing a designated task (e.g., success rate of a classifier). The members satisfying a predefined fitness criterion are selected as parents to reproduce the population of next generation. From the pool of fit members, the parents are selected in pairs to exchange the genetic material and produce new pairs of chromosomes. The process of reproduction is consists of crossover (gene exchange) and mutation (gene alteration). The fitness of new members is again evaluated using the same fitness criterion, and only those whose fitness is greater than the fitness of their parents are selected to populate the current generation. This process is continued until a new population of the same size, and having average fitness greater than the previous generation, is installed as the second generation. This process is repeated to go through several generations until a termination criterion is met. The termination criterion is defined to yield the target solution. This is equivalent to individuals in the last generation converging to a desired optimum solution. At each generation, a constant population size is maintained. Through successive generations, the population with individuals having greater and greater fitness emerges until some predefined stopping criterion is met. Either the solution to the problem is the fittest individual in the final generation, or some estimate based on the whole population. The GA based feature extraction methods are reported to yield better results than some other methods such as the sequential search in particularly large dimensionality problems [24], [25]. The GA has been used in several ways for achieving dimensionality reduction while maintaining high classification accuracy. The papers by Raymer et al. [26], Perez-Jimenez and Perez-Cortes [27] and Zhao et al. [28], besides proposing new GA based methods, contain summary and references to most of the preceding works. In most common
approach to the GA based feature extraction, the genetic search of features is combined with the performance of some on a test dataset. Most commonly used classifiers are the k-nearest neighbor classifier and the artificial neural network classifier. The objective function for GA optimization is defined in terms of classification accuracy with respect to a chosen subset of features [19], [20], [26], [28], [29], [30], [31]. The selected features are those that yield best classification result. Some variations of this direct selection strategy are also used. Raymer et al. [26] used the classifier output as feedback to optimize the weights of individual features iteratively in combination with a masking vector for the subset selection. Corcoran et al. [32] used a different objective function that is defined by using pair-wise normalized class means.

In electronic nose data processing, a few reports have appeared using GA for feature selection or sensor selection [32], [33], [34], [35], [36]. In all these studies the GA has been used for subset selection either from a conglomerate of sensors as in Gardner et al. [33] or from a set of features generated by methods like the principal component analysis as in Corcoran et al. [32]. The reason for GA not being used as extensively in the sensor array based electronic nose data processing as in some other domains like the image processing or the speech recognition is that the ‘curse of dimensionality’ is perhaps not a major issue here. Typically, only 3 to 6 sensors are involved in the measurement of odor samples by an electronic nose system. It is now understood that the number of sensors should be only as much as there are different kinds of chemical and solvation interactions (hydrogen bonding, polar, dispersive etc.) into action when vapor is exposed to the sensors, and each sensor should have broadly selective dominance for one of these interactions [37], [38]. During early days of electronic nose development, medium to large number of array sensors (nearly 7 to 30) were employed. The detailed experimentation and pattern analysis however illustrated that just by increasing the number of sensors the performance of electronic noses does not improve. Rather, often it deteriorates due to degeneracy of information among different sensors output. A proper selection of up to 5-6 sensors yields best results. A vast literature in this domain can be accessed through some important recent reviews [1], [2], [38], [39]. Some applications of GA in sensor or feature selection in electronic nose systems have been reported in the past [33], [40]. Gardner et al. [33] underlined the utility of GA in selecting an optimum set of sensor materials in electronic nose designs. Kermani et al. [40] reported performance optimization of a 32-element odor sensor array by a GA supervised procedure for data normalization, feature selection and neural network parameter selection.

This method differs from the commonly employed method of dimensionality enhancement by kernel-PCA [41]. In kernel PCA, nonlinear mapping of the input space through a chosen kernel function creates an arbitrarily high dimensionality feature space. The choice of a kernel function (usually a polynomial, a radial basis function or a sigmoidal function) is arbitrary. The effective nonlinear mapping of the data space it produces has no direct bearing on the parametric nonlinearities in the sensor signal generation. The kernel PCA, therefore, is perhaps most suited for
unsupervised machine learning. However, in applications like electronic nose where substantial insight about signal generation processes is available the parametric nonlinearities can be handled (perhaps more efficiently) by proper design of data preprocessors. The present application of GA for feature boosting in multiple preprocessor and linear PCA based feature space can account for remaining nonlinearities.

This algorithm is an approach of information fusion in contrast to decision fusion [42]. The information here comprises feature sets generated by different preprocessor/PCA combinations, and GA completes fusion process by feature boosting. Usually, the information fusion refers to combining attributes collected from several independent sources in a decision taking process, and the decision fusion refers to combining individual decisions based on different attributes in some way (e.g., voting based bagging and boosting methods [43] or Dempster-Shafer evidential fusion [44]) to arrive at the final decision. In information fusion or feature fusion, the decision is based on a single classifier. In decision fusion, the decisions of multiple classifiers are combined. A review by Dietterich [45] gives a comprehensive summary of various bagging and boosting algorithms.

In order to make this approach effective, the preprocessor designs must be prompted largely by the operational physics and chemistry of the sensors (data sources), and specific application situations. The linear PCA constructs statistically uncorrelated set of new variables. Often, either it is not possible to comprehend fully the analytical dependencies of non-identity making parameters in sensors output, or the dependencies are implicit and it is difficult to design a single preprocessor to reduce their influence. The design of preprocessors is sensor specific, and is itself an area of research. The present paper merely outlines the strategy, and demonstrates its efficacy by analyzing certain data sets collected from published sources. The method appears most suitable for low dimensionality problems such as the sensor array based electronic noses where accurate feature extraction from responses of broad selectivity sensors is difficult and where computational cost due to curse of dimensionality is not a major issue.

**Genetic Algorithm Assisted Feature Fusion**

The present analysis procedure is presented here in the context of an odor discrimination problem by an electronic nose system based on sensor array.

**Problem Definition**

Let $M$ denote the number of odor samples that are measured by an array of $N$ sensors. The complete set of measurements can be represented by $M \times N$ data matrix whose elements $x_{ij}$ represent the sensors output with the index $i=1,2,...,M$ varying over the odor samples and the index $j=1,2,...,N$ varying over the array sensors. Mathematically, $X = (x_1, x_2, x_3, ..., x_M)^T = \{x_{ij}\}$ represents the data matrix with the
sample vectors \( X_i = \{x_{i1}, x_{i2}, x_{i3}, \ldots, x_{iN}\} \) in rows and the sensors in columns. Because the sensors of an electronic nose have a broad range of selectivity towards different chemical constituents in an odor sample, there is substantial information overlap between outputs from different sensors in the array. The measured response vectors do not provide adequate discrimination between different odor types (or classes). The feature extraction is to generate new sample vectors (called feature vectors) through transformation or combination of raw data in such a way that different odor classes get unique representation in the feature space. The variables in feature space are intrinsic variables of odors, which could be composition of odorants or dominant chemical interaction mechanisms or some combination of them. The distinct odors assume different realizations of these feature variables, which define their mathematical signature. The feature extraction task is to extract odor signatures latent in the data matrix.

The feature extraction method makes use of multiple preprocessor based PCA generated features to define a gene pool for GA. The gene pool can also be enriched by bringing in genes created by alternate feature generation processes, which emphasize variability in data from different perspectives. For example, feature extraction algorithms such as PCA, ICA and LDA, or a single feature extraction algorithm preceded by alternate methods of data preparation (scaling, normalization, filtering) would focus different aspects of statistical and parametric dependencies embedded in data. The objectives of data preprocessing procedures are to emphasize odor discriminating information and eliminate nonidentity-making factors. The later are noise, outlier, odor concentration, sensor-operating conditions etc. There is no universally accepted method of data preprocessing. A number of subprocesses in different combinations may define alternate preprocessors. By using them in combination with alternate feature generation algorithms would create alternate feature representations of the data space. The proposed algorithm is to combine these feature spaces and optimize feature values through genetic evolution. In effect, an enhanced dimensionality feature space with GA assisted feature weighting is created.

Figure 1 presents flowchart schematic of the present implementation. Its details are described in the following subsections.

**Creation of Initial Population for GA**

Experimentally measured output of each sensor contains contributions from different constituents of the odor sample. Each constituent has several intrinsic chemical interaction dimensionalities. For example, an odor molecule may interact with the sensor via hydrogen bonding, polar interaction, dispersive interaction and/or oxidation-reduction process. The value of parameters representing these interactions will be different for different odor and sensor combination. Therefore, a sensor output is complex interplay of diversities at species level, and at molecular interaction level.
The sensor array for electronic nose is designed such that different sensors combine contributions from these diverse factors in different proportions hoping that by analyzing the set of outputs from the array one can separate individual contributions or create at least a set of parameters that represents the odor identity. This set of parameters is mathematical signature of the odor. Development of an efficient signature extraction procedure is however complicated due to several factors. First, different inherent contributions may not be all independent, that is, they have some correlation among them in unknown manner. Second, the experimental measurements are usually noisy having contributions from some intrinsic (inherent to the sensor signal generation principles and operation condition) and some extrinsic (electrical fluctuation, pick up, interference etc.) sources. Third, the data may have
some spurious (from undesired or unplanned sources, residual effects, outliers etc.) contributions. Besides, long-term instabilities in experimental set up and operating condition may cause drift in sensors output. An accurate and reliable signature extraction method is therefore a difficult task in electronic nose systems.

**Multiple Preprocessors and PCA**

The linear PCA is a widely employed feature extraction method. It assumes that the features at signal generation stage are combined linearly, and are uncorrelated and noise free. If one expects the PCA to produce good result then the measured data should be cleaned for noise and outliers, should be corrected for drift, and should be linearized before doing PCA. More deviated the data are from these conditions more inaccurate the PCA estimates would be. For this reason, a large number of data preprocessing methods are in use to do shifting, scaling, denoising, outlier removal and base line correction before attempting feature extraction. Table 1 contains the used preprocessing methods.

Taking input from each preprocessor the principal component analysis generates principal component scores and eigenvalues. The PC scores are projections of preprocessed sample vectors onto unit-length eigenvectors that define the feature space. The sample vectors \( X_i = [x_{i1}, x_{i2}, x_{i3}, ..., x_iN] \) are transformed to feature vectors \( Y_i = [y_{i1}, y_{i2}, y_{i3}, ..., y_{iN}] \) in \( N \)-dimensional feature space defined by the eigenvector directions. The principal component scores of a sample comprise its feature values arranged in the order of decreasing eigenvalues \( \lambda_1 > \lambda_2 > .... > \lambda_N \). It is known that the eigenvalues are equal to the variances of the data projections onto principal axes estimated over all samples \([14]\). That is, \( \lambda_i = \sigma_i^2 \) with \( \sigma_1^2 \geq \sigma_2^2 \geq \sigma_3^2 \geq .... \geq \sigma_N^2 > 0 \).

**Defining Gene Pool and Chromosome**

PCA generated feature components of a sample are taken as genes, and interpret the associated variances as measures of their importance in some way that influences object discrimination. By combining the entire sets of genes created by all the preprocessor/PCA combinations, we create a gene pool. Chromosomes are constructed by combinations of PCA scores joined in succession to make a bigger set of \( PN \) features where \( P \) denotes the number of preprocessors. The feature vector of a sample in the combined feature space thus becomes

\[
Z_i = \{y_{i1}^0, y_{i1}^1, ..., y_{i1}^N, y_{i2}^0, y_{i2}^1, ..., y_{i2}^N, ..., y_{iK}^0, y_{iK}^1, ..., y_{iK}^N\}
\]

where \( s = 1,2,...,P \) runs over the number of preprocessors. The fused feature vector can thus be alternately rewritten as

\[
Z_i = \{z_{ik}\} = [z_{i1}, z_{i2}, z_{i3}, ..., z_{ik}]
\]

with \( k = 1,2,...,K(=PN) \). The variances of the feature components will also be renormalized according to

\[
\sigma_k^2 = \frac{\sigma_i^2}{K \sum_{l=1}^{K} \sigma_l^2}
\]
Equation (1) and (2) define the gene pool that will be utilized to create initial population for GA as detailed in the following Table.

**Creation of initial population**

From the fused feature vector $Z_i$ corresponding to a sample, we will generate an initial population of chromosomes, and assign a fitness value to each individual member. To create chromosomes we use the probability distance metric introduced by Zohdy et al. [11], [46]. A distance matrix is defined on the basis of the relative closeness of the feature components as follows. If it is assumed that the $k$-th feature component is accurate (or fully reliable), then how accurate (or reliable) the $l$-th component is, is given by the probability distance measure

$$d_{i,k} = \frac{1}{\sigma_k \sqrt{2\pi}} \text{erf} \left( \frac{z_{il} - z_{ik}}{\sigma_k \sqrt{2}} \right)$$

in which the index ‘i’ stands for $i$-th sample, and $z_{ik}, z_{il}$ and $\sigma_k$ are defined by (1) and (2). A $K \times K$ distance matrix is thus generated by assuming each feature component in $Z_i$ to be accurate one by one in turn. That is,

$$D_i = \begin{bmatrix} d_{i,11} & d_{i,12} & \ldots & d_{i,1K} \\ d_{i,21} & d_{i,22} & \ldots & d_{i,2K} \\ \vdots & \vdots & \ddots & \vdots \\ d_{i,K1} & d_{i,K2} & \ldots & d_{i,KK} \end{bmatrix}$$

It is obvious that all the diagonal elements in (4) will be zero because $\text{erf}(0) = 0$. It has to be, being the distance of a component from itself. This only affirms the meaning of the probability distance is more reliable a feature component, closer it is. Therefore, one can define a distance threshold $d_c$ to eliminate those feature components that are farther than this. The initial population of chromosomes is created by holding the fused feature vector $Z_i$ against each row of the distance matrix $D_i$ and comparing $d$-values at each feature position against the set threshold $d_c$. If at certain feature positions, the $d$-values are greater than $d_c$, then those feature components are set to be zero. The resultant feature vector thus defines a member of the initial population. By repeating this procedure with each row of the distance matrix, a new member is created. Thus, a population of $K$ chromosomes represents the odor sample.

**Coding**

The algorithms need chromosome strings coded in a suitable digital format for mathematical implementation of the genetic operators. Though numerous coding schemes are employed which suite specific application situations and objectives [33], [22], [23], the most widely used is the binary coding. In this, the binary digits 1 and 0 denote whether a gene is present or absent. For implementation of the genetic algorithm in this work we found it convenient to code the chromosomes by the strings of integers in decimal format such that the decimal number strings denote the position
of features in the gene pool $Z_i$ defined by (1). The absence of a feature is denoted by 0. The digit length to represent each integer is taken to be equal to that needed for writing the size of the gene pool. For example, if $K = 10$, and a chromosome in the initial population is constructed of 1st, 3rd, 4th, 7th and 10th feature components then its coded representation is $[01.00.03.04.00.00.07.00.00.10]$.

**Ranking by Fitness**

The variance of a variable represents information content inherent in that variable. Therefore, the sum of variances of all feature components (genes) defining a chromosome denotes its information bearing significance relevant to odor discrimination. This sum is used to represent its ‘fitness’. The fittest individual is placed at the top position (rank 1). The remaining chromosomes are ranked successively as rank 2, rank 3… rank K. The rank of a member represents the probability of it being selected as a parent for recombination to produce the next generation population. Figure 1 presents flowchart schematic of the present implementation.

**Genetic Evolution**

The genetic evolution is to create individuals of next generation by applying genetic operators (selection, crossover and mutation) on the current population, and install a new population of the same size. The present implementation is based on the functions available in the Matlab GA toolbox. We created a genetic algorithm to produce three types of children for the next generation – elite, crossover and mutation. The numbers of children in each category are adjustable by choosing the function parameters ‘elite count’, ‘crossover fraction’ and ‘mutation rate’. The next generation population is obtained as, next generation = elite children + crossover children + mutation children, keeping the population size fixed, see Fig. 1. The methods to generate these three types of children are explained below.

**Elite Children**

A predefined number of individuals from the initial population (current generation) of highest ranking are selected to go to the next generation without any genetic alteration. This number is empirically adjusted by monitoring the overall influence of the genetic algorithm on the classifier performance during training and validation phases, described in Section 3.

**Crossover Children**

The number of crossover children is determined by assigning a value of the parameter ‘crossover fraction’. It is obtained as the integer part of the product of initial population size and crossover fraction. The crossover children are produced by adapting the ‘rank based fitness scaling’ and ‘remainder stochastic sampling without replacement’ methods for selection, and the scattered crossover.

Rank based fitness scaling and expectations. Each individual in the initial population is assigned a ‘score’ based on its rank order. The score function is chosen such that the fitness values are scaled across the population without altering the rank
order. This is done to tilt the probability of being selected in favor of members of lower rank. Otherwise, only the members of highest fitness values will be selected more often, and the characteristic features of low rank members (which may be important for pattern recognition) will be lost. A commonly employed score function is \( \text{score} = 1/(\text{rank})^{1/2} \). Using the score value of an individual the probability that it will be selected for recombination is calculated as

\[
pselect(i) = \frac{\text{score}(i)}{\sum_{j=1}^{K} \text{score}(j)}
\]

where summation is over the entire population of size \( K \). A quantity defined as ‘Expectations’ is then calculated according to \( \text{Expectations}(i) = K \times pselect(i) \) for each member of the initial population. This number indicates a desired number of offspring that a chromosome should produce commensurate with its rank position. The sum of expectations rounded to the nearest integer \( \sum \text{Expectations}(i) \) defines the size of an intermediate population that will be created by the selection method.

**Selection by remainder stochastic sampling without replacement**

The expectations values obtained as defined above will in general be fractional. However, the number of selections must be an integer. This is managed in two steps. First, the integer parts of expectations are used to place that many copies of respective chromosomes in the intermediate population with size constraint equal to \( nParents \). In the second step, more chromosomes are added to this intermediate population stochastically according to the remaining fractional parts of expectations. Each remaining fraction is converted into probability by normalizing it with respect to sum of all the fractions as \( \text{prob}(i) = \frac{\text{frac}(i)}{\sum_{j=1}^{P} \text{frac}(j)} \). One by one, these probabilities are compared against a random number \( r \) generated over \([0, 1]\). A copy of that chromosome (say, \( j \)-th) is added to the intermediate population for which \( r \leq \text{prob}(j) \). This is continued until intermediate population size is full (that is, equal to \( nParents \)).

**Scattered crossover**

This is a position independent crossover function. The pairs of chromosomes are selected by picking up consecutive members of the intermediate population starting from the top. That is, by taking (1, 2), (2, 3), (3, 4), and so on as pairs. For each pair selection, a random binary vector of size equal to the chromosome size is generated. This vector defines the child if the positions where 1 appears are replaced with the genes from the first parent and where 0 appears are with the genes from the second parent. In this manner, each selected pair generates one child. As an example,

1st parent: 01.00.03.00.00.06.07.00.09.00
2nd parent: 01.02.00.04.05.00.07.08.00.10
binary vector 1 0 1 1 0 0 0 1 0 0
child: 01.02.03.00.05.00.07.00.00.10
This procedure is repeated with successive pairs until the target number of crossover children is reached.

**Mutation children**
The number of mutation children needed to add up to the target population of the next generation is created by using the Matlab mutation function the ‘mutation uniform’. In this method, an input parameter the ‘mutation rate’ $\mu$ needs to be specified. First, the individual members from the initial population are selected one by one, and a random number is generated in the range $(0, 1)$ for each gene position. If this number is less than the mutation rate ($\text{rand}(0,1) < \mu$) then that particular gene is replaced by a gene randomly selected from the original gene pool.

**Stopping Criteria**
By combining all the chromosomes (elite + crossover + mutation) generated in this way, and ranking them according to the procedure described above the second-generation population is obtained. This will be treated as the initial population for creating the third-generation population, and so on. The evolutionary process is terminated by applying a stopping criterion based on the average fitness of the current population. This average generation fitness is monitored at successive generations. The process is terminated when the average generation fitness stabilizes. In all the validation cases reported here it was found that the average population fitness converges to a stable limit nearly after 80-100 generations of the genetic evolution. An example of the fitness convergence while doing the analysis of the Insect Odorants data (presented in Section 5) is shown in Figure 2.

![Figure 2. Convergence of average population fitness with genetic evolution through](image)
successive generations for insect odorants data.

**Feature Weighting**

The final set of features that would represent an odor sample is obtained by weighting of the components of the fused feature vectors $Z_i = \{z_{ij}\} = [z_{i1} z_{i2} z_{i3} \ldots z_{iK}]$, Eq. (1). In the earlier work a linear weighting was implemented.

The final set of features that would represent an odor sample is obtained by a method of feature weighting. In the final population, every feature component of the gene pool is sorted according to the number of times it has appeared across the entire population. Since there are $K$ individual chromosomes each of length $K$ in the terminal population, the total number of feature elements is equal to $K^2$. Let $n_j$ denote the number of times $j$-th feature component (or gene) has appeared in the terminal population. We can define the probability its occurrence as $p_j = n_j / K^2$. In the earlier work [10] the feature components of fused feature vector were given additional weight in linear relation to the probability of their being in the terminal population. That is, the feature components were weighted according to

$$z_{ij} = z_{ij}(1 + p_j)$$

for $j = 1, 2, \ldots, K$ components of $i$-th sample. This was done by interpreting the prevalence of occurrence of a gene in the final population as a measure of its significance.

However, in the framework of information theory if we interpret the probability of a gene appearance as a measure of its information carrying capacity then a better measure should be the Shannon’s entropy defined as $-p_j \log_2 p_j$ [47]. Therefore, it is reasonable to weight the feature components according to

$$z_{ij} = z_{ij}(1 - p_j \log_2 p_j).$$

**Validation**

**Algorithm Implementation**

All the programs for feature extraction by PCA, feature boosting by GA and pattern classification by RBF neural network were implemented in the Matlab environment by using functions available statistical, neural and GA tool boxes and by developing customized codes. The PCA was implemented by using ‘prestd’ and ‘premnmx’ functions. The genetic algorithm was implemented by using the options structure ‘gaoptimset’. In this implementation the initial population was created through a customized program; and the selection, crossover and mutation steps were implemented by using the functions ‘selectionremainder’, ‘crossoverscattered’ with crossover fraction 0.9 and ‘mutationuniform’ with mutation rate 0.09 to 0.3. The elite count was set to be equal to 2. All the unspecified parameters were set to their default values.

The two preprocessors used in the present analysis are: ‘vector autoscaling’ and ‘dimensional autoscaling’ [8]. These are the most commonly used preprocessors for
electronic nose data processing, and are defined as follows.

**Vector autoscaling**

\[ x_j \to \frac{x_j - \bar{x}_i}{\sigma_j} \text{ with } \bar{x}_i = \frac{1}{N} \sum_{j=1}^{N} x_{ij} \text{ and } \sigma_i = \sqrt{\frac{1}{N} \sum_{j=1}^{N} (x_{ij} - \bar{x}_i)^2} . \]

**Dimensional autoscaling**

\[ x_j \leftarrow \frac{x_j - \bar{x}_j}{\sigma_j} \text{ with } \bar{x}_j = \frac{1}{M} \sum_{i=1}^{M} x_{ij} \text{ and } \sigma_j = \sqrt{\frac{1}{M} \sum_{i=1}^{M} (x_{ij} - \bar{x}_j)^2} . \]

The vector autoscaling implements mean-centering and variance normalization with respect to data attributes. The dimensional weighting implements mean-centering and variance normalization with respect to data samples. An additional power law scaling of data was implemented for the data generated by tin-oxide sensor array prior to the vector or dimensional autoscaling. That is, the raw tin-oxide sensor data were first transformed according to \( x_j \leftarrow (x_{ij})^{\alpha} \) with \( \alpha = 0.84 \). This was done in view of the reported studies on tin-oxide sensors [53, 54]. The tin-oxide sensors exhibit nonlinear dependence on vapor concentration with power law dependency. Therefore, to linearize the data this power law scaling was implemented. The value of exponent however depends on the odor and sensor construction. In the present analysis \( \alpha = 0.84 \) was found to be optimum.

The radial basis function neural network classifier had three layer architecture. The number of input nodes was equal to the number of feature components. The output nodes were as many as the number of classes. The numbers of neurons in the hidden layer were optimized empirically by monitoring the classification results. In all the cases, taking hidden layer neurons less in number than the number of input nodes were found to produce stable performance. The typical network parameters using the ‘newrb’ structure were as follows:

- Hidden layer activation function: **Gaussian**
- Hidden layer neuron: **radbas**
- Output layer neurons: **purelin**
- Mean squared error goal: 0.001
- Spread of radial basis functions: 1 - 33.

**Data**

The data sets used in the analysis for validation are listed in Table 1. It consists of six datasets for odor recognition and eight data sets for non-odor object recognition. For RBF classification the data sets were divided into training and test sets in the ratio of approximately 3:2.

**Results**

Table 2 shows the results on classification for various combinations of data
processing. Both GA assisted weighting schemes defined by Eqs. (6) and (7) were analyzed in combination with the backpropagation and RBF neural networks. In order to assess the impact of proposed processing strategy on classification efficiency, the analyses were carried out without GA boosting also, see column 2 in Table 2. The results in column 3 and column 4 compare the influences of \(-p \log p\) and \(p\) weightings on the RBF classifier performance. The results in column 5 and 6 are included for making a comparison with the earlier analysis reported by the authors in [10] and with those reported in some other publications respectively.

The positive influence of feature fusion based on multiple preprocessors and genetic algorithm on classification rate can be clearly seen by comparing the results in column 2 and 3. The amount of improvement however depends on the data set—smallest improvement occurs for yeast data (0.5%) and largest improvement occurs for toxic vapor data (30%). In majority of the cases, typical improvement in classification can be noticed to occur over 2% to 10%. From a comparison of the results in column 3 and 4 it is quite apparent that the present modification to the feature weighting, Eq. (7) against Eq. (6) used earlier, either maintains or improves the classifier performance. Another point worth noticing is that for the same feature extraction procedure the RBF classifier performs better than the backpropagation algorithm.

Table 1. Summary of the datasets used in the present analysis.

<table>
<thead>
<tr>
<th>Odors</th>
<th>No. of Classes</th>
<th>No. of Variables</th>
<th>No. of Samples</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee (mono)</td>
<td>7</td>
<td>4</td>
<td>210</td>
<td>Coffee aroma of monovariety group [48]</td>
</tr>
<tr>
<td>Coffee (blend)</td>
<td>7</td>
<td>5</td>
<td>249</td>
<td>Coffee aroma of blend group [48]</td>
</tr>
<tr>
<td>Insect Odorants</td>
<td>10</td>
<td>12</td>
<td>110</td>
<td>Types of insect odorants [49]</td>
</tr>
<tr>
<td>Toxic Vapors</td>
<td>9</td>
<td>9</td>
<td>40</td>
<td>Types of toxic vapors [50]</td>
</tr>
<tr>
<td>Nerve Agents</td>
<td>2</td>
<td>3</td>
<td>125</td>
<td>Nerve and non-nerve agent gases [51]</td>
</tr>
<tr>
<td>Wine</td>
<td>3</td>
<td>13</td>
<td>178</td>
<td>Wines derived from three different cultivars [52]</td>
</tr>
<tr>
<td>Other data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auto-mpg</td>
<td>3</td>
<td>8</td>
<td>398</td>
<td>The classes are city-cycle fuel consumption.</td>
</tr>
<tr>
<td>Iris</td>
<td>3</td>
<td>4</td>
<td>150</td>
<td>The classes are different types of iris plant.</td>
</tr>
<tr>
<td>Glass Identifi-</td>
<td>6</td>
<td>9</td>
<td>214</td>
<td>The classes are types of glasses of forensic interest.</td>
</tr>
<tr>
<td>Dataset</td>
<td>Classes</td>
<td>Features</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Haberman</td>
<td>2</td>
<td>3</td>
<td>The classes are survival status of the patients who had undergone surgery for breast cancer.</td>
<td></td>
</tr>
<tr>
<td>(breast cancer)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yeast</td>
<td>10</td>
<td>8</td>
<td>The different classes are protein localization sites (non-numeric) in eukaryotic cells of yeast.</td>
<td></td>
</tr>
<tr>
<td>Pima Indian</td>
<td>2</td>
<td>8</td>
<td>The classes represent whether the patients tested positive or negative.</td>
<td></td>
</tr>
<tr>
<td>(diabetes)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spect Heart</td>
<td>2</td>
<td>22</td>
<td>The dataset describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each of the patients is classified into two categories: normal and abnormal.</td>
<td></td>
</tr>
<tr>
<td>Liver Disorder</td>
<td>2</td>
<td>6</td>
<td>The classes represent whether the patients tested positive or negative.</td>
<td></td>
</tr>
</tbody>
</table>

**Discussion**

The performance of a pattern recognition algorithm depends on all the processing steps beginning from the data preparation to feature extraction to classification. The selection of these methods and their combination greatly influences the final classification rate. There is substantial scope for improving the performance of a given methodology by making alterations at various steps. For example, a neural network classifier may yield different results with different preprocessors even though the feature extraction and network architecture may remain fixed. Therefore, it is difficult to make a qualitative assessment of the present method of feature extraction by comparing the classification rates produced by other authors in different reports as each of them have used varying procedures for data preparation. Nevertheless, by using a radial basis neural network classifier the present method yields high performance for all the data set analyzed here. The data from different domains may have different specifics of the parametric dependencies; hence, the preprocessor designed for one domain may not be optimum for the other. Further, the present performance is based on the fusion of information generated only by two preprocessors (vector-autoscaling and dimensional-autoscaling). There is enough scope for improvised construction of individual preprocessors, and for using several preprocessors.

The linear principal component analysis transforms raw data space to an orthogonal feature space that is created by seeking directions of maximum variance in the data. By preprocessing the measured data differently before inputting to PCA can be expected to generate alternate feature spaces that reveal diversity of the information content hidden in the measured data. A fusion of multiple feature spaces created by multiple preprocessor-PCA combination can be expected to create an
enriched feature space. In the present work, we exploited this idea. The motivation for GA boosting of the fused feature space came from the consideration that certain feature components might be carrying excessively overlapping information; therefore, they might gain undue importance in facilitating the recognition task. The evolutionary weightage procedure may adjust the relative significance of different features. The results of present analysis given in Table 2 sufficiently validate this idea, and the present method of feature extraction. A drawback of the present method is that it enhances the dimensionality of the problem. Therefore, it is not suitable for high dimensionality problems. However, it seems to offer substantial advantage in terms of accuracy in those cases where the data space dimensionalities are small such as sensor array based electronic noses.

Conclusion
The paper concludes that the proposed feature fusion based on multiple preprocessing and GA boosting helps the PCA in creating more accurate representation of the data vectors. Further, for the GA boosting of feature components, the Shannon entropy is more effective weighting function than the probability of gene occurrence in the terminal population. The present analysis based on GA assisted feature fusion and RBF neural network classifications yields the best results for most the data analyzed.

Acknowledgment
This work was supported by the Defence Research & Development Organization (Government of India) Grant No. ERIP-ER-0703643-01-1025. The authors are thankful to all the authors whose experimental data were in this study.

Table 2: Classification results for the test data derived from the data sets mentioned in Table 1.

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>% Classification Rate</th>
<th>Some results reported by others</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RBF without GA boosting</td>
<td>RBF with GA boosting</td>
</tr>
<tr>
<td></td>
<td>z_{ij} = z_{ij}</td>
<td>z_{ij} = z_{ij} (1 - p_j \log_2 p_j)</td>
</tr>
<tr>
<td>Coffee (mono)</td>
<td>76.78</td>
<td>85.71</td>
</tr>
<tr>
<td>Coffee (blend)</td>
<td>90.48</td>
<td>96.83</td>
</tr>
<tr>
<td>Dataset</td>
<td>SVD/ANN</td>
<td>Toxod Vapors</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>--------------</td>
</tr>
<tr>
<td>Odorants</td>
<td>56.10</td>
<td>70.73</td>
</tr>
<tr>
<td>Toxic Vapors</td>
<td>70.01</td>
<td>100</td>
</tr>
<tr>
<td>Nerve Agents</td>
<td>90.32</td>
<td>98.06</td>
</tr>
<tr>
<td>Toxic Vapors</td>
<td>70.01</td>
<td>100</td>
</tr>
<tr>
<td>Nerve Agents</td>
<td>90.32</td>
<td>98.06</td>
</tr>
<tr>
<td>Wine</td>
<td>97.26</td>
<td>98.63</td>
</tr>
<tr>
<td>Auto-mpg (automobile)</td>
<td>74.24</td>
<td>86.00</td>
</tr>
<tr>
<td>Iris</td>
<td>96.71</td>
<td>98.33</td>
</tr>
<tr>
<td>Glass Identification</td>
<td>65.52</td>
<td>75.86</td>
</tr>
<tr>
<td>Haberman (breast cancer survival)</td>
<td>73.33</td>
<td>80.00</td>
</tr>
<tr>
<td>Glass Identification</td>
<td>65.52</td>
<td>75.86</td>
</tr>
<tr>
<td>Yeast</td>
<td>58.45</td>
<td>58.95</td>
</tr>
<tr>
<td>Pima-Indian (diabetes)</td>
<td>66.67</td>
<td>83.33</td>
</tr>
<tr>
<td>Spect Heart</td>
<td>83.42</td>
<td>90.90</td>
</tr>
<tr>
<td>Lemon</td>
<td>75.83</td>
<td>79.23</td>
</tr>
</tbody>
</table>

SVD/ANN: Support vector machine, artificial neural network, FDA: Fisher discriminant analysis, TAN: Tree augmented naïve-Bayes, NNEE: neural network ensemble editing, KNNBA: K-nearest neighbor based association algorithm. This is an improvised KNN algorithm that implements feature weighting based on certain association rules. In their paper the authors [57] have compared its performance with 7 other classification methods, namely, NB, NN, C4.4, NBTREE, VFI, LWL and IBK. The range of classification results represents...
On Genetic Algorithm and Multiple Preprocessors

the performance of all the 8 classifiers.

References


