Defence Science Journal, Vol. 60, No. 4, July 2010, pp. 392-398
© 2010, DESIDOC

# Boosting Principal Component Analysis by Genetic Algorithm

Divya Somvanshi and R.D.S. Yadava*

*Banaras Hindu University, Varanasi*–221 005
*E-mail: *ardius@ gmail.com, ardius@bhu.ac.in*

**ABSTRACT**

This paper presents a new method of feature extraction by combining principal component analysis and genetic algorithm. Use of multiple pre-processors in combination with principal component analysis generates alternate feature spaces for data representation. The present method works out the fusion of these multiple spaces to create higher dimensionality feature vectors. The fused feature vectors are given chromosome representation by taking feature components to be genes. Then these feature vectors are allowed to undergo genetic evolution individually. For genetic algorithm, initial population is created by calculating probability distance matrix, and by applying a probability distance metric such that all the genes which lie farther than a defined threshold are tripped to zero. The genetic evolution of fused feature vector brings out most significant feature components (genes) as survivours. A measure of significance is adapted on the basis of frequency of occurrence of the surviving genes in the current population. Finally, the feature vector is obtained by weighting the original feature components in proportion to their significance. The present algorithm is validated in combination with a neural network classifier based on error backpropagation algorithm, and by analysing a number of benchmark datasets available in the open sources.

**Keywords:** Feature extraction, genetic algorithm, feature fusion, principal component analysis, pattern recognition

## 1. INTRODUCTION

Genetic algorithms are computational models of evolution based on the natural selection, reproduction, and survival-of-the-fittest principles. Originally invented by Holland,[1] the genetic algorithms (GAs) have now grown into varied forms to solve numerous stochastic optimisation problems in domains ranging from the artificial intelligence and machine learning to biology, economics, sociology, and physics[2-7]. In pattern recognition domain, the genetic algorithms are mostly used as methods of feature subset selection within filter or wrapper approaches.[8,9] The goal in a feature selection process is to remove those features which are not relevant to the pattern recognition task. The resulting benefits are two fold: enhanced classification accuracy, and reduced computational cost in high dimensionality problems.

The filter methods evaluate relevance of features using some intrinsic characteristics of the data and do not involve induction algorithms (e.g., elimination of smallest eigenvalue principal components, removal of noisy and outlier components). The wrapper methods, on the other hand, directly use induction algorithms, and the feature subsets are evaluated by their influence on the classification accuracy[9,10]. Numerous publications and references therein[11-18] report most of the major developments on the genetic approach to feature subset selection. It is noteworthy that in these studies, the GAs have been used mainly for feature selection (often also referred to as feature extraction), and not for the feature generation from raw data. The feature generation is usually done by the methods like principal component analysis (PCA), linear discriminant analysis (LDA) or their variants. The primary target in most of the earlier works on genetic algorithm has been to tackle the curse of dimensionality and to select features that bring out better discrimination among classes in high dimensionality problems such as microarray data analysis[17] biometric identification[18], data mining[19] and so on[11-19]. In the development of classifiers, the GAs have mostly been used as a wrapper method for features/ parameters optimisation[20,21], and recently for classifier fusion[22] as well.

In this paper, GA has been used in a novel way to create feature vectors. The method is based on PCA-generated features to define initial population for genetic evolution. This is followed by the genetic evolution and finding significance of individual feature components. The feature vectors are then weighted according to their significance. Additional generality to the method is brought out by using multiple pre-processors to generate alternate principal component representation, and combining them to produce increased dimensionality feature representation. The new method appears most appropriate for giving low dimensionality problems accurate feature representation. Though the present method increases problem dimensionality, accruing benefits in terms of enhanced classification rate might offset computational costs in many situations.

Received 28 September 2009, Revised 23 March 2010

## 2. ALGORITHM

### 2.1 Creation of Initial Population

Let $m$ denote instances and $n$ attributes in a pattern recognition task. The raw data is given as a m × n data matrix. Each row of the data matrix represents an instance realised through $n$ observations. Mathematically, an instance (or a sample) is represented as a vector in $n$-dimensional data space whose directions are defined by the attributes. The feature extraction is to map the data space into a feature space where an instance is represented by a feature vector. The goal of this transformation is to separate the samples of different classes in feature space. The features are the intrinsic variables of the problem latent in the data matrix. The set of feature components must represent a sample in unique manner to denote its identity (or class). Let the data matrix be denoted as $[x_{ij}]$ where an element $x_{ij}$ denotes $j^{\text{th}}$ observation of $i^{\text{th}}$ sample, $i = 1,2,...,m$ and $j$=1, 2, ...$n$. The algorithm is presented here in two parts, shown in Figs 1 and 2, respectively. The first part is for creating the initial population for the genetic algorithm, which makes the second part. In Fig. 1 the creation of initial population is shown using two pre-processors and a common PCA algorithm. The PCA generated feature sets for a sample through both the pre-processors are combined to make a bigger feature set representation, and their variances are normalised (shown as 'feature fusion' block). From the fused feature vector $[z_{ik}]$ corresponding to a sample, the initial population is created using the probability distance measure, as described by Zohdy[23,24], *et al.*, and assign a fitness value to each individual. In brief, the probability distance is defined by assuming that an individual component (say, $k^{\text{th}}$) is accurate (or reliable), then how accurate (or reliable) the $l^{\text{th}}$ component is, is given by

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

where index $i$ denotes the sample, $z_{ik}$ and $z_{il}$ are $k^{\text{th}}$ and $l^{\text{th}}$ feature components and $\sigma_k$ is the component variance (eigenvalue). Thus, corresponding to each feature component assumed accurate, there will be $P = nN$ probability distances of all components of the fused feature vector. Here, $N$ denotes the number of pre-processors employed. Assuming

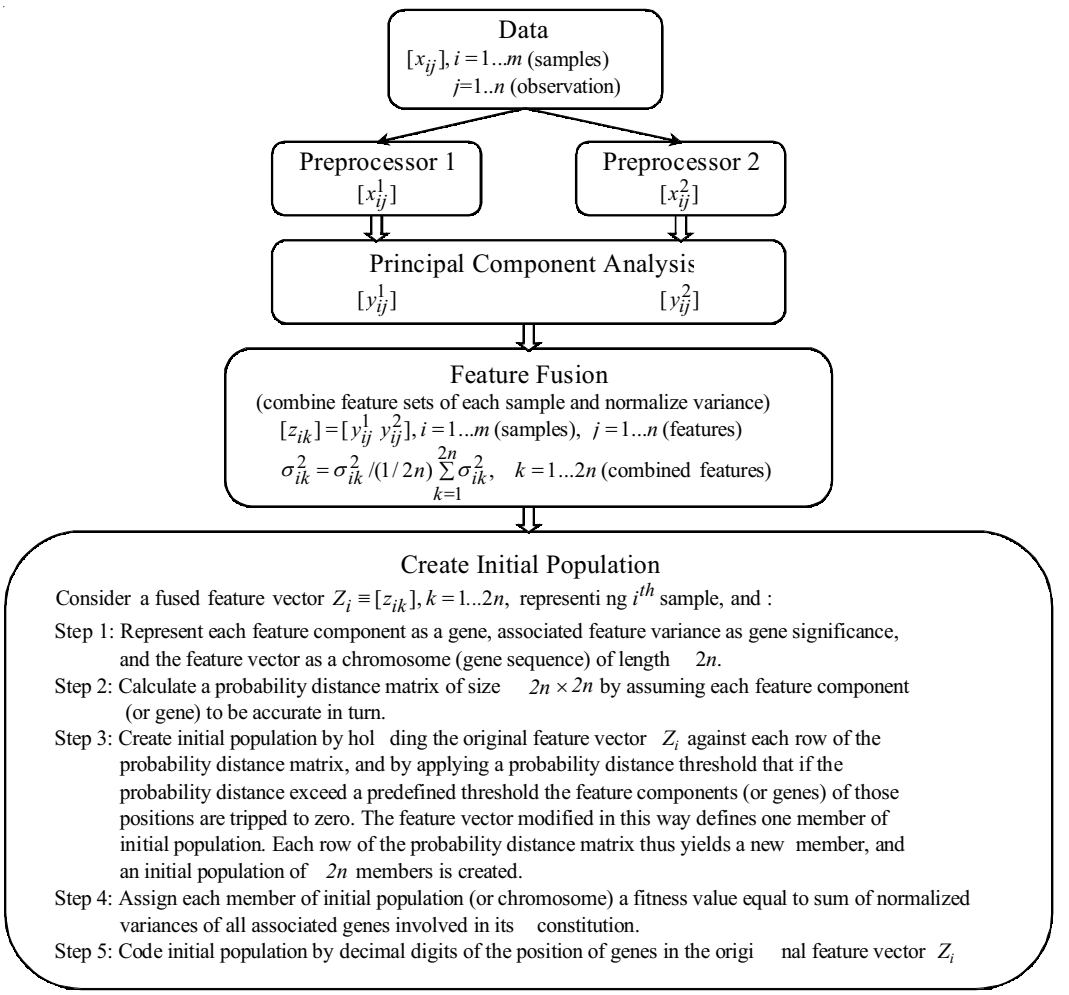


**Figure 1. Flow chart of algorithm to create initial population from two pre-processors and principal component analysis.**

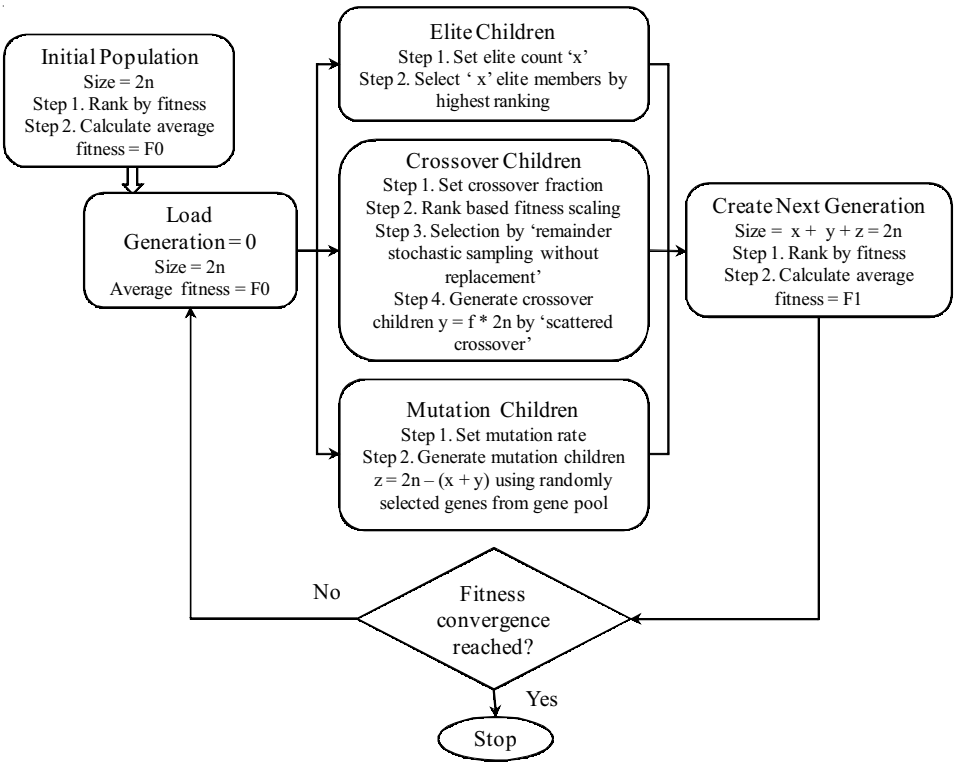


**Figure 2. Flow chart of genetic algorithm for evolution through generations till a population of stable average fitness is reached.**

feature component as being accurate in turn, and representing the corresponding distances in a row generates a $P \times P$ probability distance matrix

$$D_i = \begin{bmatrix} d_{i,11} & d_{i,12} & \ldots & d_{i,1P} \\ d_{i,21} & d_{i,22} & \ldots & d_{i,2P} \\ \ldots & \ldots & \ldots & \ldots \\ d_{i,P1} & d_{i,P2} & \ldots & d_{i,PP} \end{bmatrix} \quad (2)$$

It is obvious that all the diagonal elements in Eqn (2) will be zero because $erf(0) = 0$. These are the distances with self. This gives meaning to the probability distance– closer a feature componant, more reliable it is. Therefore, a distance threshold (say, $d_c$) can be defined beyond which the feature components are not important in a row. A chromosome is created by holding the fused feature vector $Z_i \equiv [z_{ik}]$ against a row, and setting all those components to be zero for which $d > d_c$. The initial population of chromosomes is thus created by repeating this procedure with each row in $D_i$. Thus, after distance thresholding, each row in $D_i$ represents a chromosome, and there are $P$ chromosomes in the initial population. The variances $\sigma_k$ of fused feature components are normalised as

$$\sigma_k^2 = \sigma_k^2 \Big/ P^{-1} \sum_{l=1}^{P} \sigma_l^2 \quad (3)$$

These steps are elaborated in the 'create initial population' block in Fig. 1.

The distance threshold is defined empirically by examining the probability distance matrix. If the probability distance of a feature component is greater than this threshold, then that feature component is eliminated by resetting its value to zero. This creates a member of initial population; e.g., $[z_{i1}, z_{i2}, 0, z_{i4}, 0, z_{i6}]$ represents a member whose 3rd and 5th feature components have been reset to zero. Each row of the probability distance matrix creates one member. The resulting initial population will be of the size equal to the product of number of pre-processors and the number of original attributes in the data matrix. It is equal to $2n$ in Fig. 1. The fitness of a member is taken to be equal to the sum of normalised variances of all the feature components present in its constitution. The population is then ranked according to fitness value of the members.

Finally, the individual members are given digital code representation for implementing the genetic operations. In most applications, binary coding scheme is used where presence or absence of genes is denoted by a binary 1 or 0 bit. In the present implementation, however, the chromosomes were coded by strings of integers in decimal format such that these denote the position of features in the fused vector $Z_i$. For example, if $P = 10$, and a chromosome in the initial population is created by 1st, 3rd, 4th, 7th and 10th feature components, then its coded representation is $[01.00.03.04.00.00.07.00.00.10]$.

### 2.2 Genetic Evolution

The second part implements the standard genetic algorithm to create a new population using genetic operators: selection,

crossover, and mutation. Figure 2 shows the details. Starting with the initial population, the next generation members are created in three categories. The latter are elite, crossover and mutation. The elite children are the highest ranking individuals in the initial population that are taken to the next generation without any alteration. This is done because while creating the next generation members by crossover and mutation, some members with best fitness values may be lost, which otherwise would have been very helpful in pattern discrimination. The number of elite members that will go to the next generation is chosen empirically by monitoring the overall performance of the system with training data. The crossover children are generated by pairwise selection of chromosomes and crossover operation on genes about a crossover point. The total number of crossover children in a population is limited by the amount of crossover fraction. The crossover steps shown in Fig. 2 are implemented using the functions in the Matlab GA tool box. The mutation operator randomly modifies the genes of population with a probability equal to their mutation rate. The number of mutation children is kept to be such that after summing up with the elite and crossover children the new population size becomes equal to the initial population size. This is implemented using the Matlab mutation function 'mutation uniform'. In this method, the individual members from the initial population were selected one by one, and a random number between (0, 1) was generated for each gene position. If this number was less than the mutation rate then that particular gene was replaced by a gene randomly selected from the original gene pool. The next generation consisted of conglomerate of these types of children ranked by fitness.

The current generation was evaluated by its average fitness. The average fitness is defined as the sum of individual's fitness averaged over population size. If the current generation average fitness was larger than the preceding generation, the current generation was loaded for further evolution.

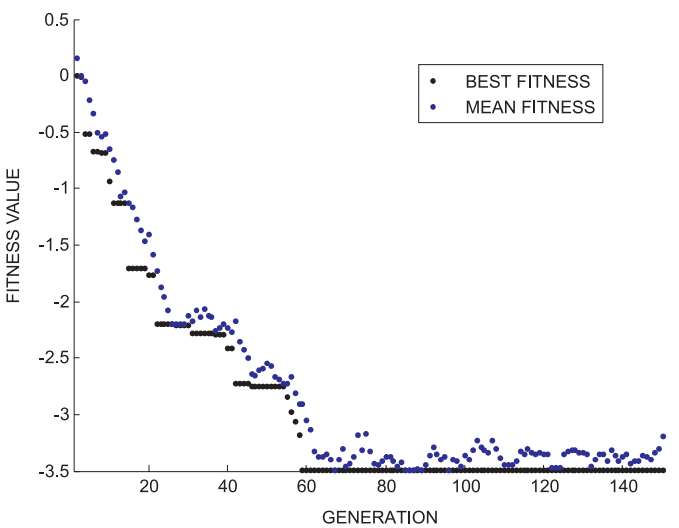


**Figure 3. Convergence of the mean fitness value through successive generations during GA evolution.**

The process was stopped when the fitness value in the successive generations do not change, that is, the population is stabilised. The convergence of GA in a typical run is shown in Fig. 3.

### 2.3 Boosting Principal Component Analysis

The stabilised population obtained after stopping the process of genetic evolution was used to boost the values of feature components (genes) in the gene pool. Recall that the features in gene pool are the PCA-generated features, and the fused feature vector represents sample. On the basis of GA-generated population, the values of these feature components were calculated as

$$z_{ij} = z_{ij}(1 + n_j / P^2) \quad (4)$$

where, $n_j$ denotes the frequency (or the number of times $j^{th}$ gene occurs in GA population of a sample) and $P$ denotes the feature vector length (number of genes). The latter is equal to $2n$ for the two pre-processor algorithm. The $P^2$ therefore represents the number feature elements in the final population.

## 3. VALIDATION

The proposed method was validated by analysing some benchmark data available from open sources. The data used are briefly summarised in Table 1. All the data sets except the coffee data are taken from UCI Repository[26]. The coffee data was provided by Pardo and Sberveglieri[27]. The two pre-processors that were used in this analysis are the 'vector autoscaling' and the 'dimensional autoscaling' as defined by Osuna and Nagle[25], and also available in Matlab. The classifier used was a neural network with error-backpropagation algorithm ing Matlab NN tools. The network architecture consisted of input nodes equal to the number of features and output nodes equal to the number of classes. Only one hidden layer was used in which the number of nodes were empirically adjusted by monitoring the classification results. In most cases, the number of hidden layer neurons a few less than the number of input nodes yields best results. In cases where data was not divided into training and test sets, the division was made in the ratio of 3:2 of the total samples. The entire program was developed in Matlab.

Table 2 shows the average classification results obtained after several runs. The last column of the table also shows the best results reported in other publications. These results show the efficiency of the present algorithm. It can be seen that in most cases, the results were at least as good as the best reported by some other more complicated advanced methods. Only in cases of protein location sites data of *E-coli* and yeast, the results were inferior. However, even in these cases, the results were as good as or better than many earlier reports. For example, in case of yeast data, the classification rate reported by the data donors Horton and Nakai is 55 per cent and by Fan and Poh, it is (53±3) per cent. Moreover, selection and design of classifier itself could play a substantial role in determining the

**Table 1. Benchmark datasets used in the present study**

| Data set | No. of samples | No. of attributes | No. of classes | Remark |
|---|---|---|---|---|
| Iris | 150 | 4 | 3 | The Iris data is a record of 150 samples measured for 4 variables. The dataset contains 50 samples each from 3 different classes of Iris plants. |
| Yeast | 1484 | 8 | 10 | Originally this data is given as 9-variable and 10-class dataset. The different classes are protein-localisation sites (non-numeric) in eukaryotic cells of yeast. One of the variables is the name of sequence. The dataset for the present analysis was therefore taken to be an 8-variable/10-class problem by dropping the sequence name from the variables list. |
| Wine | 178 | 13 | 3 | These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. |
| Pima Indian (diabetes) | 768 | 8 | 2 | This dataset contains total 768 samples (468 training, 300 test) of diabetes test. All patients here were females at least 21 years old of Pima Indian heritage. The classes represent whether patients tested positive or negative. |
| Haberman (breast-cancer) | 306 | 3 | 2 | The dataset contains cases from a study conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer. Of the total 306 samples, 225 survived beyond 5 years and 81 died within 5 years. |
| Glass Identification | 214 | 9 | 6 | This dataset is a 7 class problem but samples of 4 class are zero so it was dropped the data was treated as a 6 class problem. The types of glasses were of forensic interest defined in terms of their oxide content. |
| E-coli | 336 | 7 | 8 | The data contains protein-localisation sites. One of the attributes was the name of sequence, and dropped from variable list. |
| Auto-mpg (automobile) | 336 | 7 | 3 | The data concerns city-cycle fuel consumption in miles per gallon, to be predicted in terms of 3 multivalued discrete and 5 continuous attributes. The original dataset contains total 398 samples. For the present analysis, missing variable data were removed. |
| Coffee (blend) | 249 | 5 | 7 | This data is aroma record of two groups (blend and monovariety) of coffees measured by tin-oxide gas sensor array. Each group contained 7 types of coffees. Measurements were made on two groups of coffees referred to as the monovariety and the blend. In each group there were 7 types of coffees. |
| Coffee (monovariety) | 210 | 4 | 7 | |

classification rate. At present, only a simple neural network model has been used. The full implications of the present algorithm will probably be known only after rigorous testing with other classifiers and with more data.

## 4. CONCLUSIONS

The fusion of feature spaces created by alternate pre-processor/PCA combinations produces extended dispersion of intrinsic dimensionalities (or information content) in the data. The genetic evolution of the fused set of features (taken as chromosome) facilitates survival of only the most important feature components. The strongest genes (most significant features) are passed more frequently in the successive generations. This brings out the identity and the relative importance of different feature components. By taking the frequency of occurrence in current generation as some measure of importance, the final feature vector is calculated by giving additional weight to those features in proportion to their frequency in the final stabilised population. This algorithm of feature extraction in combination with a neural network classifier produces as good classification results as best reported for most of the benchmark data analysed in this paper.

## ACKNOWLEDGMENTS

This work was supported by the Defence Research & Development Organisation Grant No. ERIP-ER-0703643-01-1025 and Department of Science and Technology Grant No. DST-TSG-PT-2007-06. The authors are thankful to all those who provided data utilised in this work. We acknowledge

**Table 2. Neural network classification rates achieved on the basis of present method of GA-assisted feature extraction**

| Dataset | Class | Training samples | Test samples | Predicted classification | | | Best reported rates (per cent) | Reference No. |
|---|---|---|---|---|---|---|---|---|
| | | | | True | False | Rate (per cent) | | |
| Iris | 3 | 90 | 60 | 59 | 1 | 98.3 | 98.1 | ref. 28 |
| Yeast | 10 | 904 | 574 | 319 | 255 | 55.6 | 95.9<br>55<br>53±3 | ref. 29<br>ref. 32<br>ref. 33 |
| Wine | 3 | 105 | 73 | 72 | 1 | 98.6 | 97.8<br>97.2 | ref. 28<br>ref. 29 |
| Pima Indian (diabetes) | 2 | 468 | 300 | 243 | 57 | 81 | 76 | ref. 30 |
| Haberman (breast-cancer) | 2 | 186 | 120 | 90 | 30 | 75 | 74.8 | ref. 29 |
| Glass | 6 | 127 | 87 | 62 | 25 | 71.3 | 67.1<br>68.6 | ref. 28<br>ref. 29 |
| *E-coli* | 8 | 224 | 112 | 84 | 28 | 75 | 87.9 | ref. 29 |
| Auto-mpg (automobile) | 3 | 236 | 100 | 82 | 18 | 82 | 79.2 | ref. 31 |
| Coffee (blend) | 7 | 186 | 63 | 55 | 8 | 87.3 | 87 | ref. 27 |
| Coffee (monovariety) | 7 | 154 | 56 | 42 | 14 | 75 | 82 | ref. 27 |

the UCI Machine Learning Repository particularly for this. The authors would like to thank Mr Prashant Singh, Mr Shashank S. Jha, Mr Sunil K. Jha, Mr Manoj K. Vishwakarma and Mr Dinesh Kumar for their understanding and cooperation.

## REFERENCES

1. Holland, J. H. Adaptation in natural and artificial systems. MIT Press, Cambridge MA, 1975.
2. Davis, L. D. Handbook of genetic algorithms. Van Nostrand Reinhold, New York, 1991.
3. Man, K.F.; Tang, K.S. & Kwong, S. Genetic algorithms: concepts and applications. *IEEE Trans. Indust. Electro.*, 1996, **43**(5), 519-34.
4. Zalzala, A.M.S. & Fleming, P.J. Genetic algorithms in engineering systems. The Institution of Electrical Engineers, UK, London, 1997.
5. Coley, D.A. Genetic algorithms. *Contemporary Physics*, 1996, **37**(2), 145-54.
6. Hartmann, A. H. & Rieger, H. Optimisation algorithms in physics. Wiley-VCH, Weinheim, 2002.
7. Zimmerman, D. C. & Jorgensen, S. F. Parallel multispecies genetic algorithm for physics and parameter estimation in structural dynamics. *Amer. Inst. Aero. Astro. (AIAA)*, 2005, **43**(10), 2224-231.
8. Dash, M. & Liu, H. Feature selection for classification. *Intell. Data Anal.*, 1997, **1**(3), 131-56.
9. Kohavi, R. & John, G. H. Wrappers for feature subset selection, *Artificial Intelligence*, 1997, **97**(1-2), 273-24.
10. Zhu, Z.; Ong, Y. S. & Dash, M. Wrapper-filter feature selection algorithm using a memetic framework. *IEEE Trans. Syst. Man Cybern., B*, 2007, **37**(1), 70-76.
11. Raymer, M. L.; Punch, W.F.; Goodman, E.D.; Kuhn, L.A. & Jain, A.K. Dimensionality reduction using genetic algorithms. *IEEE Trans. Evolut. Comput.*, 2000, **4**(2), 164-71.
12. Vafaie, H. & Jong, K. D. Genetic algorithms as a tool for feature selection in machine learning. *In* IEEE Conference Proceedings on Tools with AI, 1992. http://www.cs.bilkent.edu.tr/~guvenir/courses/cs55.
13. Perez-Jimmenez, A.J. & Perez-Cortes, J.C. Genetic algorithms for linear feature extraction. *Pattern Recog. Lett.*, 2006, **27**, 1508-514.
14. Cantu-Paz, E. Feature subset selection, class separability, and genetic algorithms. *In* Genetic and Evolutionary Computation (Lecture Notes in Computer Science, Vol. **3102**/2004), Springer, Berlin/Heidelberg, 2004. pp. 959-70.
15. Yang. J. & Honavar, V. Feature subset selection using a genetic algorithm. *IEEE Trans. Intelli. Syst.*, 1998, **13**, 44-49.

16. Jain, A. K. & Zongker, D. Feature selection: Evaluation, application, and small sample performance. *IEEE Trans. Pattern Analy. Mach. Intelli.*, 1997, **19**(2), 153-58.
17. Umpai, T.J. & Aitken, S. Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes, *BMC Bioinformatics*, 2005, **6:**148. http://www.biomedcentral.com/1471-2105/6/148.
18. Jeong, M.Y.; Choi, J.Y. & Kim, J. Using genetic algorithms to improve matching performance of changeable biometrics from combining PCA and ICA methods. *In* IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, June 2007. cvpr, pp.1-5.
19. Marmelstein, R.E. Application of genetic algorithms to data mining. *In* Proceedings of 8th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS-97), edited by E. Santos Jr., AAAI Press, 1997, pp. 58-65.
20. Goldberg, D.E. Genetic algorithms in search, optimisation and machine learning. Addison-Wesley, Massachusetts, 1989.
21. Youssif, R.S. & Purdy, C.N. Combining genetic algorithms and neural networks to build a signal pattern classifier, *Neurocomputing*, 2004, **61**, 39-56.
22. Gabrys, B. & Ruta, D. Genetic algorithms in classifier fusion. *Appl. Soft Comput.,* 2006, **6**, 337-347.
23. Zohdy, M.A.; Loh, N. & Liu, J. Application of maximum likelihood identification with multisensor fusion to stochastic systems. *In* Proceedings of the American Control Conference, Pittsburgh, Pennsylvania, June 1989. pp. 411-16.
24. Khan, A.A. & Zohdy, M.A. A genetic algorithm for selection of noisy sensor data in multisensor data fusion. Proc. *In* Proceedings of the American Control Conference, Albuquerque, New Mexico, June 1997. pp. 2256-262.
25. Osuna, R.G. & Nagle, H.T.A method for evaluating data pre-processing techniques for odour classification with an array of gas sensors. *IEEE Trans. Syst. Man Cybern. B,* 1999, **29**(5), 626-32.
26. Asuncion, A. & Newman, D.J. UCI machine learning repository. Irvine, CA: University of California, School of Information and Computer Science, 2007. http://archive.ics.uci.edu/ml/datasets.html.
27. Pardo, M. & Sberveglieri, G. Coffee analysis with an electronic nose. *IEEE Trans-Instrum. Meas.*, 2002, **51**(6), 1334-1339. http://sensor.ing.unibs.it/_people/pardo/dataset.html.
28. Chen, L. Stream data classification using improved Fisher discriminate analysis. *Journal of Computers*, 2009, **4**(3), 208-214.
29. Moradian, M. & Baraani, A. KNNBA: K-nearest-neighbour based-association algorithm. *J. Theor. Appl. Info. Techno.*, 2009, **6**(1), 123-29.
30. Lassi, A.; Juhola, M. & Laurikkala, J. On the neural network classification of medical data and an endeavour to balance non-uniform data sets with artificial data extension. *Comput. Bio. Med.*, 2007, **37**(3), 388-97.
31. Frank, E. & Hall, M. A simple approach to ordinal classification. *In* Machine Learning (Lecture Notes in Computer Science) edited by L. De Raedt & P. Flach Springer-Verlag, Berlin, Heidelberg, 2001, **2167,** 145-56.
32. Horton, P. & Nakai, K. A probabilistic classification system for predicting the cellular localisation sites of proteins. *In* Intelligent Systems in Molecular Biology (ISMB-96 Proceedings), St. Louis, USA, 1996. pp. 109-115.
33. Fan, L. & Poh, K.L. A comparative study of PCA, ICA and class-conditional ICA for naïve Bayes classifier, edited by F. Sandoval *et al.*, *In* Lecture Notes in Computer Sciences, LNCS Vol. **4507**. Springer-Verlag, Berlin, 2007. pp. 16-22.

**Contributors**

**Ms Divya Somvanshi** received her BSc from the Government Degree College, Hardoi, and MSc(Physics) from the DAV College, Kanpur, in 2005 and 2007, respectively. She is working as a Research fellow in a DRDO-sponsored research project on development of data fusion models and algorithms for electronic nose systems at the Department of Physics, Banaras Hindu University, Varanasi, India. Her present research interests include: Genetic algorithms and information fusion methods for building and boosting electronic nose intelligence.

**Dr R.D.S. Yadava** (see page 376)