Correspondence

A Method for Evaluating Data-Preprocessing Techniques for Odor Classification with an Array of Gas Sensors

Ricardo Gutierrez-Osuna and H. Troy Nagle

***Abstract*—The performance of a pattern recognition system is dependent on, among other things, an appropriate data-preprocessing technique. In this paper, we describe a method to evaluate the performance of a variety of these techniques for the problem of odor classification using an array of gas sensors, also referred to as an electronic nose. Four experimental odor databases with different complexities are used to score the data-preprocessing techniques. The performance measure used is the cross-validation estimate of the classification rate of a K nearest neighbor voting rule operating on Fisher's linear discriminant projection subspace.**

I. Introduction

An electronic nose is "*an instrument which combines an array of electronic chemical sensors with partial specificity and an appropriate pattern-recognition system, capable of recognizing simple or complex odours*" [3]. Our research focuses on the pattern recognition component of such a device. In particular, this article addresses the data-preprocessing module of the pattern recognition architecture depicted in Fig. 1. The inputs to the data-preprocessing module are raw voltage measurements (± 5 V) in the form of a bundle of exponential-like curves (see Fig. 2), which represent the transient response of the sensor array. The data-preprocessing module is responsible for shifting, compressing, and normalizing the raw data into a format that improves the performance of the subsequent modules. Numerous ad-hoc techniques to preprocess the steady state has been proposed in the literature [3], [4]. Our approach to data preprocessing differs from the references in that we are interested in preprocessing the complete transient response, which can be shown to convey useful classification information [6], [9].

The issue of data preprocessing has received very little attention in the related literature and, to the best of our knowledge, only [4] presents an evaluation of these different techniques. In [4], the authors evaluate 36 steady-state preprocessing techniques using a feedforward neural network trained to classify two different classes of microorganisms. They score each technique by the average classification rate after 10 learning trials. In this paper, we extend their work to the processing of signal dynamics and widen the evaluation for a wider variety of odor analysis applications.

A. Data Preprocessing of Gas Sensor-Array Transient Responses

The transient response of an array of gas sensors presents an exponential-like shape, as is shown in the typical response waveforms of Fig. 2. In Fig. 2, an array of 15 Capteur and Figaro metal-oxide gas sensors was exposed to the headspace of a sample of fruit juice. The data-preprocessing module operates on these responses in a way that helps the overall pattern recognition performance. For ease of notation, we will assume a data set with $e = 1 \ldots N_E$ examples. Each example (like the one in Fig. 2) consists of $s = 1 \ldots N_S$ sensor transients, with $k = 1 \ldots N_T$ samples per transient and will be denoted by $R_{e,s}(t_k)$.

There is a myriad of ad-hoc preprocessing transforms that can potentially be performed on the array transient response $R_{e,s}(t_k)$. For clarity, we have divided them into three major categories: *baseline manipulation, compression,* and *normalization*. However, the order in which these transforms can be performed is not unique. The normalization transform could be performed at different stages in the preprocessing module. For instance, Kermani [6] performs the normalization transform prior to compression. More stages could be cascaded but then the number of combinations to be evaluated grows exponentially. Our choices for the type, number and ordering of the transforms will be detailed in the following subsections.

B. Baseline Manipulation

This set of operations refers to transformations based on the initial value of the transient (the baseline) and is an attempt to reduce the effects of sensor drift. Three basic transformations are common practice.

1) Differential: Subtract the baseline of each sensor

$$R^B_{e,s}(t_k) = R_{e,s}(t_k) - R_{e,s}(t_1) \quad \forall e, s, k. \tag{1}$$

2) Relative: Divide by the baseline of each sensor

$$R^B_{e,s}(t_k) = \frac{R_{e,s}(t_k)}{R_{e,s}(t_1)} \quad \forall e, s, k. \tag{2}$$

3) Fractional: A combination of the previous two

$$R^B_{e,s}(t_k) = \frac{R_{e,s}(t_k) - R_{e,s}(t_1)}{R_{e,s}(t_1)} \quad \forall e, s, k. \tag{3}$$

A fourth transformation is commonly cited, which is extracting the *logarithm* of the previous measures. However, it is not considered here since it is not related to the baseline value and the logarithm, a monotonically increasing function, will not modify the relative ordering of the array responses.

C. Compression

Data preprocessing compression operations are used to reduce the number of measurements per example to a reasonable number. For an array with 15 sensors, a sampling frequency of 1 Hz and 60 s of transient response, the number of measurements grows to 900. The common procedure used in many commercial instruments is to keep the steady-state or final value of each sensor (15 measurements in the numerical example above) and disregard the transient. However, the transient information conveys useful information. For this reason we evaluate a number of measures that attempt to capture some of the transient information.

1) Steady-state: The default procedure; the transient is reduced to its final or steady-state value

$$R^{BC}_{e,s} = R^B_{e,s}\left(t_{N_T}\right) \quad \forall e, s. \tag{4}$$

Manuscript received August 30, 1998; revised April 2, 1999. This paper was recommended by Associate Editor J. Oommen.

R. Gutierrez-Osuna is with the Computer Science and Engineering Department, Wright State University, Dayton, OH 45435-0001 (e-mail: rgutier@cs.wright.edu).

H. T. Nagle is with the Electrical and Computer Engineering Department, North Carolina State University, Raleigh, NC 27695-7911 USA.

Publisher Item Identifier S 1083-4419(99)07951-0.

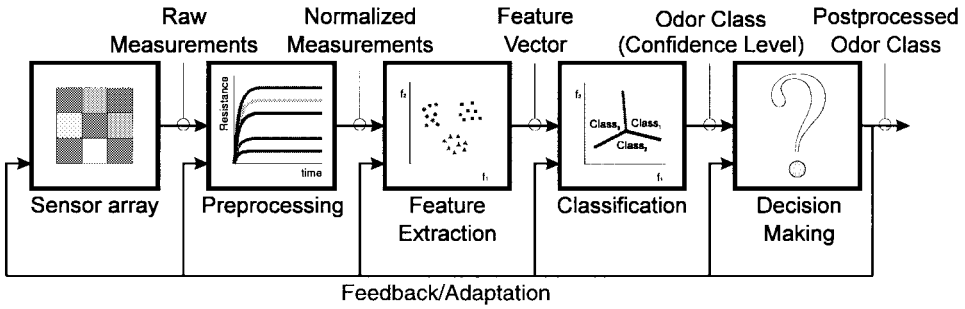

Fig. 1. Electronic nose pattern recognition architecture.

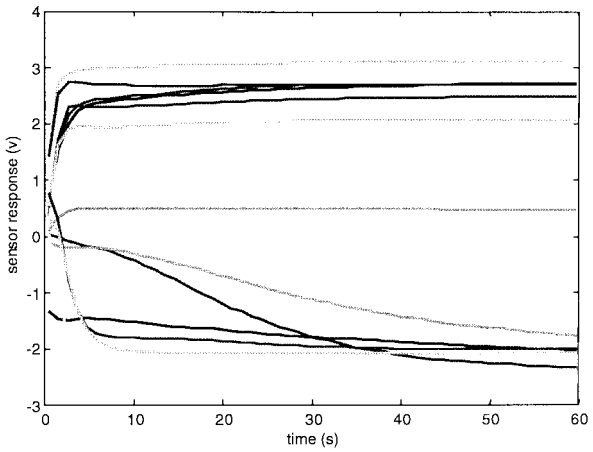

Fig. 2. Raw transient response of an array of metal-oxide gas sensors.

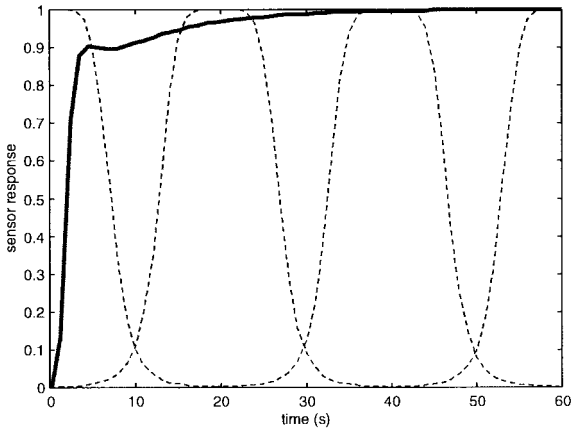

Fig. 3. Bell-shaped kernels for windowed time slicing.

2) Transient integral: A variation of the previous transform; it can be interpreted as the integral or scaled average of the transient response curve

$$R_{e,s}^{BC} = \sum_{k=1}^{N_T} R_{e,s}^{B}(t_k)\Delta t \quad \forall e, s. \tag{5}$$

3) Windowed time slicing: The transient response is multiplied by N_W smooth, bell-shaped windowing functions like the ones in Fig. 3, and integrated with respect to time. The idea behind this compression technique is to capture some information about the dynamic characteristics of the response. The number of windowing functions is arbitrary, but we choose $N_W = 4$ for consistency with [6]. The parameters (a_i, b_i, c_i) define the width, shape and center of the different windowing functions $K_i(t_k)$

$$R_{e,s}^{BC} = [W^1 \quad W^2 \quad \dots \quad W^{N_W}] \quad \forall e, s$$

$$\text{with } W^i = \sum_{k=1}^{N_T} R_{e,s}^{B}(t_k) K_i(t_k)\Delta t \quad \text{and}$$

$$K_i(t_k) = \frac{1}{1 + \left(\frac{t_k - c_i}{a_i}\right)^{2b_i}}. \tag{6}$$

After compression, each example has been reduced to a vector $\vec{R}_e^{BC} = \left[R_e^{BC}(1), R_e^{BC}(2), \dots, R_e^{BC}(N_M)\right]^T$ with $N_M = N_S$ dimensions for the first two compression techniques and $N_M = N_W N_S$ dimensions for the last compression technique.

D. Normalization

This set of operations works either across the $m = 1 \dots N_M$ dimensions for each example, or across the $e = 1 \dots N_E$ examples for each dimension. They are aimed at smoothing example to example variations. The following operations are commonly performed.

1) Vector normalization: Each example vector is divided by its Euclidean norm so that it lies in a hypersphere of unit radius

$$R_e^{BCN}(m) = \frac{R_e^{BC}(m)}{\sqrt{\sum_{m=1}^{N_M} R_e^{BC}(m)^2}} \quad \forall e, m. \tag{7}$$

2) Vector auto-scaling: Each example vector is normalized so its coordinates have zero mean and unit variance. Note that mean and variance are computed for each example across the different dimensions

$$R_e^{BCN}(m) = \frac{R_e^{BC}(m) - \mu_e}{\sigma_e} \quad \forall e, m$$

$$\text{with } \mu_e = \frac{1}{N_M}\sum_{m=1}^{N_M} R_e^{BC}(m) \quad \text{and}$$

$$\sigma_e = \sqrt{\frac{1}{N_M}\sum_{m=1}^{N_M}\left(R_e^{BC}(m) - \mu_e\right)^2}. \tag{8}$$

3) Dimension auto-scaling: Each dimension is normalized so it has zero mean and unit variance. Note that mean and variance are computed for each dimension along all the examples in the database

$$R_e^{BCN}(m) = \frac{R_e^{BC}(m) - \mu_m}{\sigma_m} \quad \forall e, m$$

$$\text{with } \mu_m = \frac{1}{N_E}\sum_{e=1}^{N_E} R_e^{BC}(m) \quad \text{and}$$

$$\sigma_m = \sqrt{\frac{1}{N_E}\sum_{e=1}^{N_E}\left(R_e^{BC}(m) - \mu_m\right)^2}. \tag{9}$$

The first two techniques are aimed at reducing the effects of different concentrations in the odor samples. The third technique prevents the subsequent pattern recognition modules from being biased by sensors with intrinsic larger magnitudes.

II. Measures for Performance Evaluation

There are two basic alternatives for evaluating the performance of the different data-preprocessing techniques: *information content* and *predictive accuracy*. The first alternative makes use of all the training data to rank the feature vector, and determines a score based on a measure of the information content, consistency of the data or class separability. The second alternative computes a score by estimating the error rate of a classifier via statistical re-sampling or cross-validation. Information content measures have the advantage that they are not influenced by the bias of a particular classifier, but can produce overly optimistic measures since they do not account for over-fitting the training data. Predictive accuracy measures, on the other hand, achieve better generalization, but are much slower to run since they require resampling and retraining [5].

We decide to use predictive accuracy as a performance measure. Our choice is guided by the characteristics of the problem space. First, our pattern recognition system frequently operates with a small number of examples, due to the high cost of data collection. As an example, a one day long experiment usually yields 70 examples, with an average of 5 min per example plus initial warm-up and final cleansing periods. Second, the device suffers from high levels of experimental noise and sensor drift. For these two reasons, we are forced to consider measures that take into account the generalization capabilities of the interaction between the feature set and the classifier.

Our choice for the classifier is guided by computational and analytical considerations. We favor the combination of linear discriminant analysis (LDA) [1] as a feature extraction tool and the K nearest neighbor voting rule (KNN) [2] as a classifier for the following reasons. LDA finds a linear projection W with $N_C - 1$ dimensions that maximizes the objective function $J(W) = \frac{|W^T S_B W|}{|W^T S_W W|}$, where S_B is the between-class scatter matrix, S_W is the within-class scatter matrix and N_C is the number of classes. It can be shown that the optimal projection can be found as the solution of the generalized eigenvalue problem $S_B w_i = \lambda_i S_W w_i$, where (λ_i, w_i) is the ith eigenvalue-eigenvector pair. In short, LDA finds a projection where examples from the same class are clustered together and different classes are spread apart. LDA is fast, computes the optimal feature set under the unimodal Gaussian assumption for the class-conditional densities, and provides useful scatter plots for visual inspection. The KNN voting rule classifies an unlabeled test example by finding the K nearest neighbors (i.e., using Eucledian distance) and assigning the label of that class represented by a majority among the K neighbors. The KNN rule is computationally simple and has nice asymptotic properties. Finally, the error rate of this LDA–KNN classifier is estimated by N-fold cross-validation (NFCV) [7], which is well-known for its reliable estimates of prediction error. This resampling technique estimates the prediction error by performing N classification runs, assigning randomly $\frac{N-1}{N} N_E$ examples for training and $\frac{1}{N} N_E$ examples for testing.

III. Odor Databases

We evaluated the performance of the proposed data-preprocessing techniques on four databases of odors collected on the electronic nose prototype that has been designed and built at the Biomedical Instrumentation Laboratory, North Carolina State University, Raleigh. The device consists of a 15-element array of Capteur and Figaro metal-oxide gas sensors. The databases have different number of classes and various complexities, but all belong to the application domain of food and beverage products.

- Cola: Ten different commercial colas were sampled over five days. Several brand-name colas and store-name colas were chosen together with diet and cherry versions. Each cola was poured into a glass bubbling vial and sampled for 90 s.
- Fruit juice: Seven fruit juices were sampled over four days. The juices had different nominal flavors but all of them were from the same brand name. They were sampled for 60 s from glass bubbling vials.
- Peanut butter: Seven different commercial peanut butters were sampled over four days. The peanut butter samples came from several brand and store names together with their reduced-fat versions. Peanut butter samples were spread into Petri dishes whose covers formed a headspace that was sampled for 60 s.
- Bacteria: Three typical food-borne bacteria cultures were grown in Petri dishes overnight and sampled the following day using the protocol of [8]. Each example was sampled for 60 s.

The characteristics of these databases are summarized in Table I and Table II.

TABLE I
Database Characteristics

Database	Type of odor	Number of classes	Number of experiments	Number of Examples
1	Cola	10	5	200
2	Fruit juice	7	4	280
3	Peanut butter	7	4	280
4	Bacteria	3	1	54

TABLE II
Database Class Identifiers

Class label	Cola	Fruit juice	Peanut butter	Bacteria
1	Coke	Orange	Smucker's	Salmonella
2	Diet Coke	Apple	Jif	Listeria
3	Pepsi	Cherry	Skippy	E-coli
4	Dr. Pepper	Grape	Better Value	---
5	Cherry Coke	Fruit punch	Smucker's Red. Fat	---
6	Cherry Pepsi	Berry punch	Jif Red. Fat	---
7	Cheerwine	Tropical punch	Best Yet Red. Fat	---
8	RC-Cola	---	---	---
9	Eckerd Cola	---	---	---
10	Eckerd Dr. Riffic	---	---	---

IV. Experimental Procedure

To study the interactions between the different data-preprocessing transformation, compression and normalization techniques we must evaluate every possible combination of them. As a reference we will consider no-transformation and no-normalization as additional techniques. This yields four baseline manipulations, three compression, and four normalization (BCN) techniques, for a total of 48

TABLE III
DATA-PREPROCESSING TECHNIQUES UNDER EVALUATION

Baseline Manipulation	Compression	Normalization
None (NOBM)	Steady-state (SS)	None (NONR)
Differential (DIFF)	Transient integral (TI)	Vector norm (VNORM)
Relative (REL)	Win. Time Slicing (WTS)	Vector auto-scaling (VAS)
Fractional (FRACT)		Dim. auto-scaling (DAS)

TABLE IV
N-FOLD CROSS-VALIDATION OUTLINE

- *For each of N runs*
  - *Split data set into* $\frac{N-1}{N}N_E$ *training examples and* $\frac{1}{N}N_E$ *test examples*
  - *Compute LDA eigenvectors and eigenvalues on the training data*
    - *Sort eigenvectors by decreasing eigenvalues*
    - *Keep only the first* N_C-1 *eigenvectors*
  - *Project the entire data set using the training data LDA eigenvectors*
  - *Classify the test data using the KNN rule on the LDA projection*
  - *Compute the ratio of correctly classified test data*
- *Compute the average classification rate over all N runs*

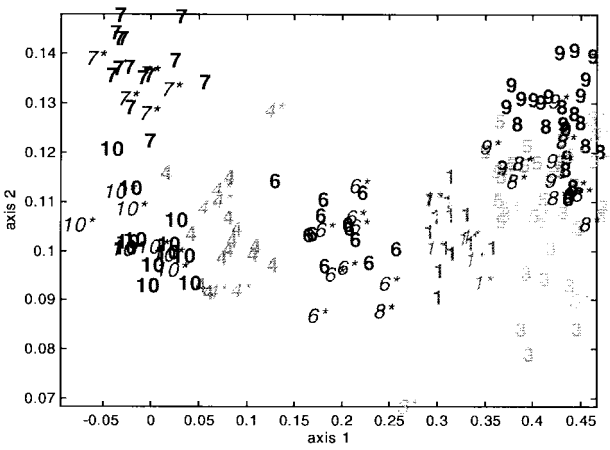

Fig. 4. LDA projections for the cola database.

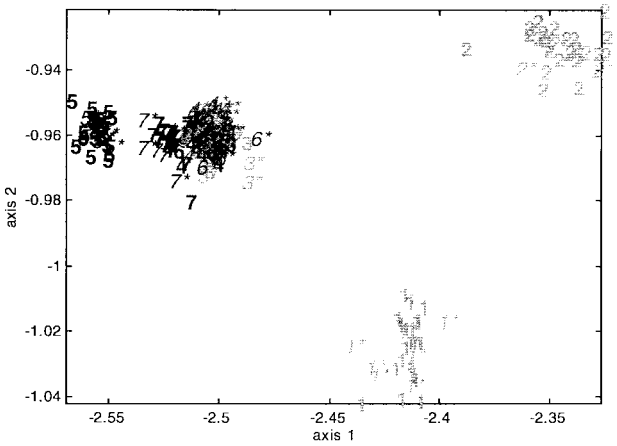

Fig. 5. LDA projections for the fruit juice database.

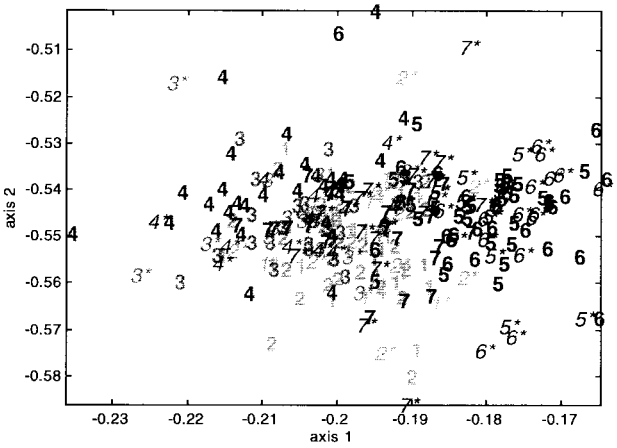

Fig. 6. LDA projections for the peanut butter database.

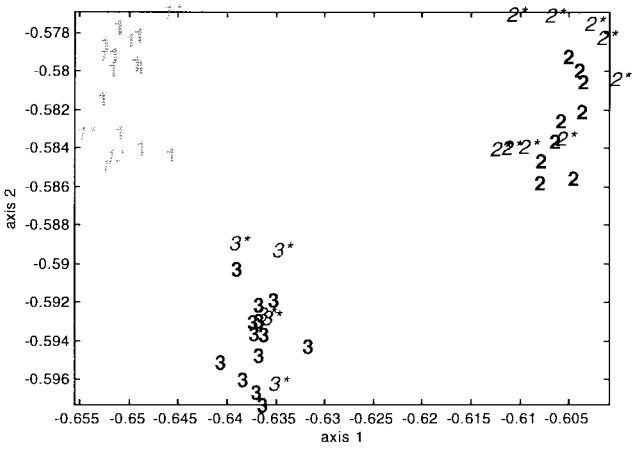

Fig. 7. LDA projections for the bacteria database.

$(4 \times 3 \times 4)$ combinations. These data-preprocessing techniques are summarized in Table III. The N-fold cross-validation procedure that will be used for each BCN technique and each database is detailed in Table IV. For the experiments in this article we have used five-fold cross-validation and $K = 5$ neighbors for the KNN voting rule.

Figs. 4–7 present some typical two-dimensional scatter plots for each of the databases. Training examples are labeled in bold fonts, test examples are in Italics and have been marked with an asterisk $(^*)$. All of the plots were obtained with the NOBM-SS-NONR data-preprocessing combination so that the reader can judge the relative complexity of the different databases. It is important to notice that the KNN voting rule computes the Euclidean distance not on the two-dimensional space shown in the figures but on all the $N_C - 1$ dimensions of the LDA projection, although the two dimensions shown in the scatter plots are the most discriminatory projections.

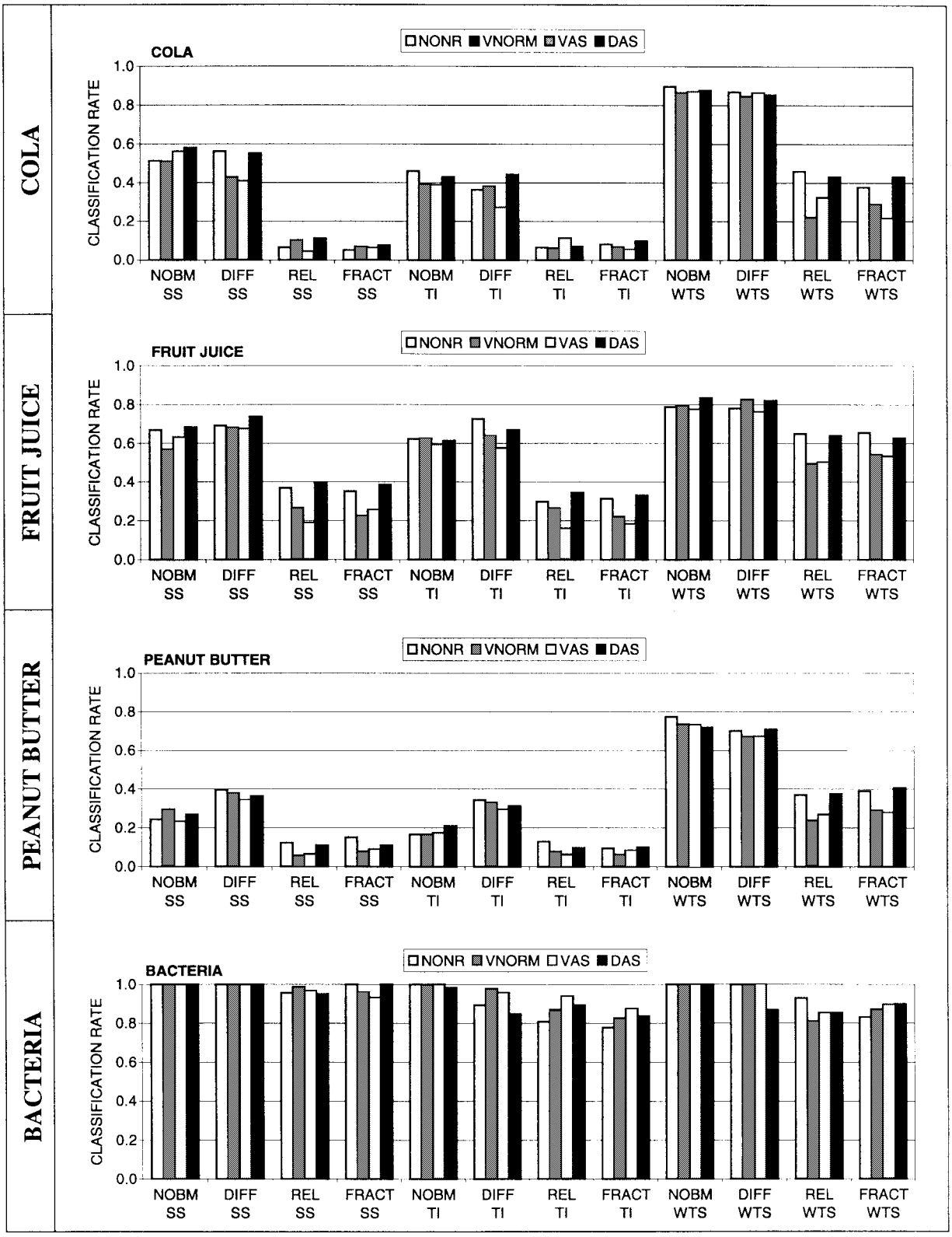

Fig. 8. Predictive accuracy of all the BCN preprocessing combinations.

V. Results

The results of the five-fold cross-validation simulation are shown in Figs. 8 and 9. Fig. 8 presents the prediction accuracy scores for each BCN preprocessing combination and each database. Fig. 9 presents a score for each individual technique, which was obtained by averaging the scores of all the combinations in which it was employed on each database. The results show substantial differences in the transformation and compression techniques, whereas the different normalization techniques end up with similar scores. The following observations are made.

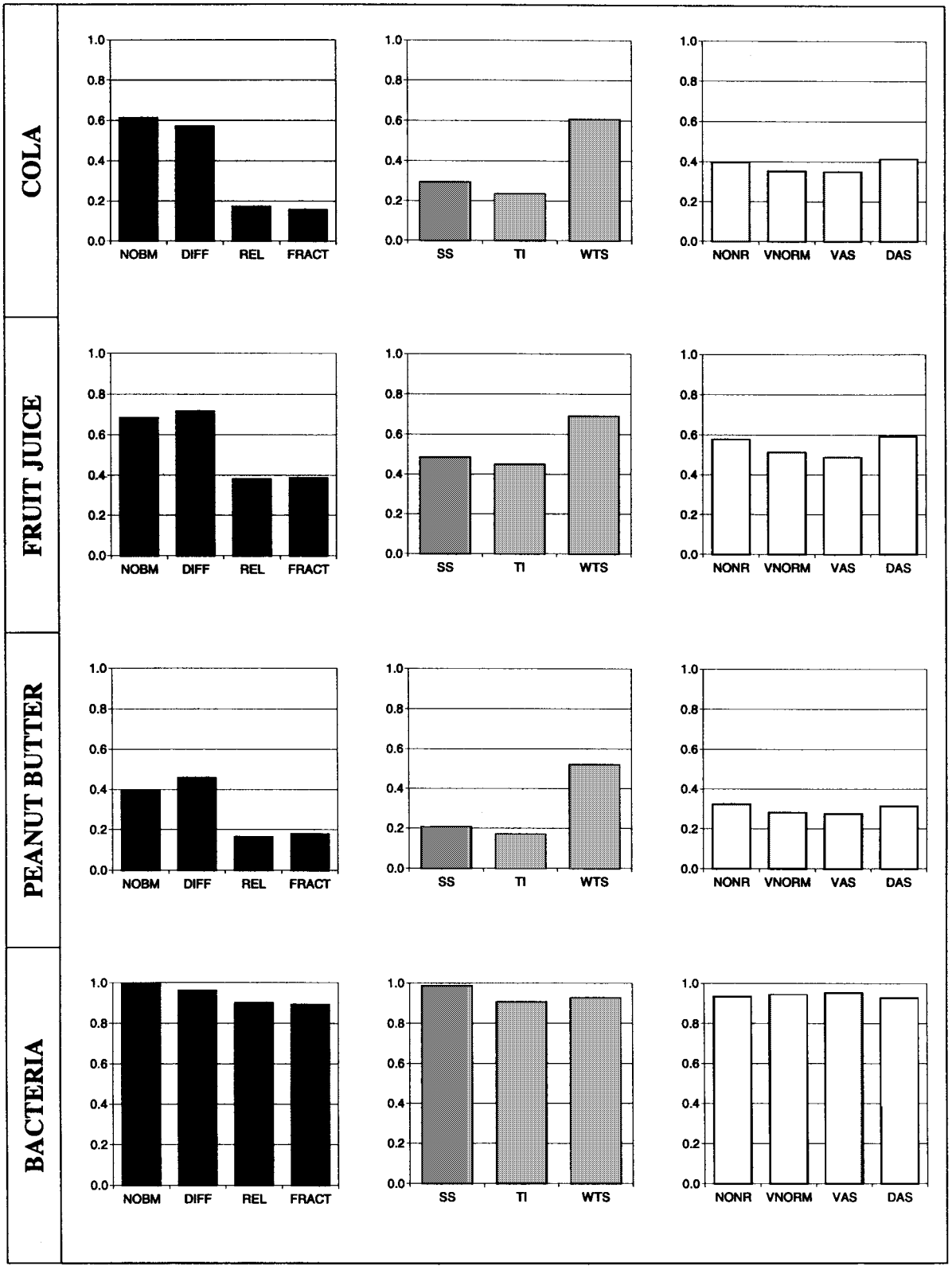

Fig. 9. Average predictive accuracy score of each BCN technique.

- Of the transformation techniques, REL and FRACT consistently give much lower scores than any of the other two. DIFF and NOBM are practically identical, DIFF being beneficial in some cases when only steady-state information is used.
- Of the compression techniques, the use of sensor transient response (WTS) results in improved prediction accuracy in all but the bacteria databases. The improvements are more dramatic for the peanut butter and cola databases, which have

more complex LDA scatter plots. The fact that the transient information does not improve but rather hurt the prediction accuracy of the bacteria database is an example of over-fitting. The three classes are very clearly separated using steady-state (refer to Fig. 7) and adding more dimensions makes the LDA projection more unstable for the unseen test cases. TI invariably scores lower than SS (or WTS), a reasonable result since it is a scaled average of the transient response and tends to smooth out the differences between odors.

- On the normalization techniques, VNORM and VAS generally result in lower scores than any of the other two. This is a reasonable result since the databases do not contain substantial concentration differences. DAS results in slight improvements over NONR on the average.

VI. Discussion

This article addresses the issue of data preprocessing for the transient response of gas sensors for the task of odor classification. In our pattern recognition architecture data preprocessing is split into three steps: baseline manipulation, compression, and normalization (BCN). Several techniques for each of the BCN steps were reviewed and evaluated on a number of odor databases of different complexities. We chose to score the performance of the data-preprocessing module by estimating the classification rate of the overall pattern recognition system. We use two efficient and well-understood statistical pattern recognition techniques: LDA and the KNN voting rule, which allowed us to evaluate a large number of combinations in relatively few computation cycles. As an example, the computation of the scores for the 48 BCN combinations on each database using five-fold cross-validation (a total of 240 classifiers) takes less than five minutes on a Pentium 200 MHz. Our numerical results present some general trends. However, we want to emphasize that the objective of this article has not been to determine the best technique or set of techniques for any particular odor application or gas sensor array. Rather, our goal has been to present a methodology that allows the evaluation of different techniques to be performed on a case by case basis.

References

[1] R. O. Duda and P. E. Hart, *Pattern Recognition and Scene Analysis*. New York: Wiley, 1973.

[2] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. New York: Academic, 1990.

[3] J. W. Gardner and P. N. Bartlett, "A brief history of electronic noses," in *Sens. Actuators B*, pp. 18–19, 211–220, 1994.

[4] J. W. Gardner, M. Craven, C. Dow, and E. L. Hines, "The prediction of bacteria type and culture growth phase by an electronic nose with a multi-layer perceptron network," in *Meas. Sci. Technol.*, vol. 9, pp. 120–127, 1998.

[5] M. A. Hall and L. A. Smith, "Feature subset selection: A correlation based filter approach," in *Proc. 4th Int. Conf. Neural Information Processing Intelligent Information Systems*, 1997, pp. 855–858.

[6] B. G. Kermani, "On using neural networks and genetic algorithms to optimize the performance of an electronic nose," Ph.D. dissertation, Elect. Comput. Eng. Dept., North Carolina State Univ., Raleigh, 1996.

[7] M. Stone, "Cross-validatory choice and assessment of statistical predictions," in *J. R. Stat. Soc. B*, vol. 36, pp. 111–147, 1974.

[8] V. Vernat-Rossi, C. Garcia, R. Talon, C. Denoyer, and J. L. Berdagué, "Rapid discrimination of meat products and bacterial strains using semiconductor gas sensors," *Sens. Actuators B*, vol. 37, pp. 43–48, 1996.

[9] D. M. Wilson and S. P. DeWeerth, "Odor discrimination using steady-state and transient characteristics of tin-oxide sensors," *Sens. Actuators B*, vol. 28, pp. 123–128, 1995.

On the Analysis of Neural Networks with Asymmetric Connection Weights or Noninvertible Transfer Functions

Ping Liang and Kaiqi Xiong

***Abstract*— This paper extends the energy function to the analysis of the stability of neural networks with asymmetric interconnections and noninvertible transfer functions. Based on the new energy function, stability theorems and convergent criteria are derived which improve the available results in the literature. A simpler proof of a previous result for complete stability is given. Theorems on complete stability of neural networks with noninvertible output functions are presented.**

***Index Terms*— Energy function, Hopfield network, neural networks, stability.**

I. Introduction

Recurrent neural networks including Hopfield neural networks and cellular neural networks are described by the system of differential equations of the form

$$\dot{u} = -Au + TS(u) + I \tag{1}$$

where $u = (u_1, \cdots, u_n)^T \in R^n$ (n is the number of neurons) whose components represent the average membrane potentials. Matrix $A = \text{diag}(R_1^{-1}, R_2^{-1}, \cdots, R_n^{-1}) \in R^{n\times n}$ stands for self-feedback terms with $R_j > 0$ $(j = 1, \cdots, n)$, $T = (T_1, \cdots, T_n) = [T_{ij}]_{n\times n}$ is an $n \times n$ real-valued constant matrix of the synaptic connection weights, each component of I is the exogenous input signal to its corresponding neuron [5], and $S = (s_1(u_1), s_2(u_2), \cdots, s_n(u_n))^T$ where $s_j(u_j)$ $(j = 1, 2, \cdots, n)$ are nonlinear neuronal transfer functions [1], [2].

The energy function plays an important role in studying the dynamical behavior and convergent property of recurrent neural networks described by system (1). As is well known, the definition of energy function depends on the symmetry of the system interconnection matrix. That is, the energy function is not applicable to any asymmetric interconnection matrix. The aim of this paper is to extend the energy function method for the analysis of the stability of neural networks from symmetric connections and invertible transfer functions to asymmetric interconnections and noninvertible transfer functions.

There are two reasons that asymmetric interconnection matrices may be of interest. First, it is unrealistic to assume that the interconnection matrix is symmetric because this requires that two physical quantities (such as resistances or the gains of operational amplifiers) be *exactly* equal [20]. The consequences of even *slight* asymmetries in the interconnection matrix are disastrous to the theory of [7] and [18] which showed that neural networks are completely stable for symmetric interactions. Liu and Michel [14] considered perturbations to the parameters in system (1) in the context of robust associate memory. They showed that if the perturbations ΔA, ΔT, and ΔI satisfy

$$\|A^{-1}\Delta A\|_\infty + \|A^{-1}\Delta T\|_\infty + \|A^{-1}\Delta I\|_\infty < v - 1 \tag{2}$$

then the asymptotically stable equilibrium points of the perturbed system will be sufficiently close to that of the unperturbed system.

Manuscript received October 2, 1997; revised July 20, 1998. This paper was recommended by Associate Editor L. O. Hall.

The authors are with the Department of Electrical Engineering, University of California, Riverside, CA 92521 USA (e-mail: liang@ee.ucr.edu; kxiong@ee.ucr.edu).

Publisher Item Identifier S 1083-4419(99)07948-0.