Ames Laboratory Technical Report IS-5147

A STATISTICAL MODEL AND COMPUTER PROGRAM FOR PRELIMINARY CALCULATIONS RELATED TO THE SCALING OF SENSOR ARRAYS

Max D. Morris
Department of Statistics
Department of Industrial and Manufacturing Systems Engineering
Iowa State University

Xiaohu Liu
Department of Statistics
Iowa State University

April, 2001

The research leading to this report was sponsored by the Microtechology Program, Office of Nonproliferation and National Security, Research and Engineering (NN-20) of the U.S. Department of Energy, through a subcontract with Lockheed Martin Energy Systems, Oak Ridge, Tennessee. This report has been authored by the Iowa State University of Science and Technology under Contract No. W-7405-ENG-82 with the U.S. Department of Energy. The U. S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U. S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for U. S. Government purposes.

CONTENTS

ABSTRACT

Recent advances in sensor technology and engineering have made it possible to assemble many related sensors in a common array, often of small physical size. Sensor arrays may report an entire vector of measured values in each data collection cycle, typically one value per sensor per sampling time. The larger quantities of data provided by larger arrays certainly contain more information, however in some cases experience suggests that dramatic increases in array size do not always lead to corresponding improvements in the practical value of the data.

The work leading to this report was motivated by the need to develop computational planning tools to approximate the relative effectiveness of arrays of different size (or "scale") in a wide variety of contexts. The basis of the work is a statistical model of a generic sensor array. It includes features representing measurement error, both common to all sensors and independent from sensor to sensor, and the stochastic relationships between the quantities to be measured by the sensors. The model can be used to assess the effectiveness of hypothetical arrays in classifying objects or events from two classes. A computer program is presented for evaluating the misclassification rates which can be expected when arrays are calibrated using a given number of training samples, or the number of training samples required to attain a given level of classification accuracy. The program is also available via email from the first author (mmorris@iastate.edu) for a limited time.

1. INTRODUCTION

Recent advances in sensor technology have led to the development of devices which have dramatically improved sensitivity, and are small enough to permit the construction of compact multi-sensor arrays. An example of broad interest is the development of coated surface acoustic wave (SAW) chemical sensors. Arrays comprised of a few carefully selected sensors can offer substantially improved detection and classification power relative to single sensors, e.g. Osbourn et al (1997). However, as the technology advances further and it becomes possible to build arrays containing even larger numbers of individual sensors, the question of optimal array size, or *scaling*, becomes critical. Realized classification performance can actually be degraded as arrays are enlarged to include additional sensors unless the effort invested in array calibration is substantially increased. In many applications, more intensive calibration is not practically possible or even physically achievable.

This report is focused on the development of a model and analysis of the functional details and operation of a generic sensor array for the purpose of optimal array scaling. Such issues as measurement error (including both components associated with individual sensors independently, and common to the entire array), the design of calibration experiments and analysis of the resulting data, and the variable and confounded signals due to environmental intensity and contamination are considered. While the specific characteristics of particular sensor systems and classification problems vary greatly, the research aims are toward developing a general methodology/tool which takes its fundamental structure from the properties of real sensors, but is applicable well beyond the context of particular present-day systems. The focus here is on the setting in which a sensor array is used to *classify* a measured item or condition as being associated with one of two distinct kinds, e.g. an organic or inorganic substance, or a 'normal' or 'unusual' activity within a monitored work area. The model is developed within the context of statistical discriminant analysis, used to identify items from two classes of objects or events which can be adequately modeled as multivariate normal (or "Gaussian") distributions. This context is a useful compromise between reality and simplicity. Linear discriminant analysis, in particular, is probably the most often used calibration technique for multivariate measurement systems, and the normal model provides a reasonably rich collection of class structures relative to the number of parameters needed for specification. It allows the assessment of trade-offs between the gain information associated with the addition of new sensors to the system, and the loss of predictive precision associated with the required expansion of the associated model. Calculations lead to guidelines which are useful in assessing the joint impact of the factors noted above on the performance of class predictions.

The trade-offs between array size and calibration effort exist to some degree for any noisy classification problem incorporating any analytical method. However, the optimal solutions differ with the specific characteristics of the classification problem and the approach used in calibration. When underlying population parameters are known, the classification rule based on linear discriminant analysis is well-known to be optimal when the underlying distributions are normal and variance structures of the two classes are equal. Known-parameter quadratic discriminant analysis is an optimal classification procedure for normal populations with unequal variance structures. In this study, we focus on the performance of these two methods, along with a nonparametric nearest-neighbor rule, in the more practical setting where model parameters must be estimated from calibration data, rather than being "known" as assumed in the small-sample optimality proofs.

In some settings, the feature profiles of classes to be discriminated are too complex to be expressed with the framework of normal distributions. In such cases, more flexible discriminant models such as artificial neural networks can result in superior performance if large quantities of calibration data are available. Examples of other flexible calibration rules which can be useful under these conditions are kernel-based approximations to Bayes rules. The present research does not extend to such non-normal classes and non-linear classification rules, but such extensions would be reasonable next steps in continuing this work.

2. GENERAL MODELS

2.1. Physical Model

The idealized situation we shall address is a simple but widely applicable one. We suppose that a sensor array is used to characterize an *object*. In this sense, an object may be a physical sample, a radiation field, an environmental condition, or any other item, substance, assembly, or event of interest in the context of the application. We assume that when "exposed" to this object, the sensor array responds by producing a set of measurements or data with one numerical value associated with each sensor in the array. The sensors are envisioned as being physically similar in nature, but not identical, and are intended to produce separate physically meaningful bits of information about the object, rather than simple replicate measurements of the same physical characteristic.

We let $y^{(i)}$ denote the (scalar) numerical value recorded by the ith element of a sensor array under a given situation, i.e. the *measurement* supplied by that sensor. This measurement is conceptually a *noisy* version of some underlying *true* quantity $x^{(i)}$, and so is actually a function of $x^{(i)}$ and some measurement error $e^{(i)}$, which in this sense represents *any* difference between the value sought and the value obtained:

$$y^{(i)} = f(x^{(i)}, e^{(i)}).$$

The error may likewise be separated into components. Here we consider the specific possibility that it is comprised of two components, one of which is specific to the ith sensor and one of which is common to all sensors in the array at the time of this measurement:

$$y^{(i)} = f(x^{(i)}, e^{(i)}, e).$$

For example, e might reflect error arising from variability in a power supply common to all sensors in the system, while $e^{(i)}$ would include error resulting from any physical contamination of just the ith sensor.

We let $\mathbf{y}$ represent the s-element vector of measurements simultaneously collected from a s-sensor array, in which the ith element is $y^{(i)}$, as defined above. The vector $\mathbf{y}$, then, is the information we have available to us about the object of interest.

2.2. General Classification Problem

Our view of a sensor array is that the measurement vector produced somehow characterizes the unit being examined. The specific form of characterization we shall consider is that of *classification.* In our context, a classification problem arises when every possible unit is a member of exactly one class. For example, samples of homogeneous material might be categorized as one of "organic", "inorganic", or "mixed". A *discriminant rule* is a rule for using the information available about a specific unit, in our case the vector $\mathbf{y}$, to classify the unit into one of these categories. For two classes labeled "1" and "2", this may be expressed as a rule or function $d(\mathbf{y})$ for which the value is either "1" or "2" for any possible $\mathbf{y}$. In this study, we shall focus on two-category systems.

If the system of interest is simple and the "physics" perfectly understood, it may be possible to specify a discriminant rule d from first principled. However, this is generally not the case with complex systems of practical interest. In these cases, a *calibration experiment* is needed to gather data from which a discriminant rule may be empirically constructed. In such cases, the experiment consists of collecting data (values of $\mathbf{y}$) from a collection of objects in each class, and deriving a discriminant rule via some method. A key difference between use of the array in a calibration experiment and in the intended application is that, the class identities of the objects examined in the experiment are *known*, whereas the goal when using the calibrated array is the correct identification of class membership of objects for which this is not known. Clearly, the effectiveness of the calibration exercise depends heavily upon how the "known" objects are selected, and the data analytic method used to construct the discriminant rule, among other considerations.

2.3. Probabilistic Model

In order to develop classification methods which can reasonably be expected to effectively classify objects/data associated with future measurements, based only on the data acquired during calibration, a common *probability model* is generally assumed to characterize the origin of both calibration and field measurements. This can be interpreted as an assumption that, over hypothetical infinite selection and measurement of objects, a distribution of relative frequencies of possible values of $\mathbf{y}$ exists. In the two-class problem, denote the two probability density functions as $p_1(\mathbf{y})$ and $p_2(\mathbf{y})$; that is, $p_1(\mathbf{y})$ is the probability density of measurement vectors associated with objects from class 1, and $p_2(\mathbf{y})$ is the probability density of measurement vectors associated with objects from class 2. Suppose also that, the relative frequency of the appearance of objects from class 1, in actual field use, is π_1, and the similar relative frequency for class 2 is π_2, π_1 , $\pi_2 > 0$, $\pi_1 + \pi_2 = 1$. In the somewhat unusual situation in which π_1, π_2, p_1, and p_2 are precisely known, the Bayes classification rule associated with equal costs for the two possible misclassification errors is of form:

$$d(\mathbf{y}) = 1 \text{ if } p_1(\mathbf{y})\pi_2 > p_2(\mathbf{y})\pi_1$$
$$d(\mathbf{y}) = 2 \text{ if } p_1(\mathbf{y})\pi_2 < p_2(\mathbf{y})\pi_1$$

(We ignore precise equality of $p_1(\mathbf{y})\pi_2$ and $p_2(\mathbf{y})\pi_1$ here since the probability of such an outcome under the models we shall examine is zero.) This rule minimizes the expected (e.g. long-run average) proportion of misclassifications under these circumstances. The logic of the Bayes rule is apparent in the structure of the above equations, which simply says that an object is more likely to be from class 1 if the odds of this event based on the data are greater than the corresponding odds based on no data. The proof is simple, and is presented in most books on discriminant and classification procedures, e.g. Devroye et al (1996).

However, in most practical problems, the premise on which this result is based, i.e. full knowledge of all probabilities in the problem, is not satisfied. In particular, densities p_1 and p_2 must generally be estimated from calibration data, and classification rules of form:

$$d(\mathbf{y}) = 1 \text{ if } \hat{p}_1(\mathbf{y})\pi_2 > \hat{p}_2(\mathbf{y})\pi_1$$
$$d(\mathbf{y}) = 2 \text{ if } \hat{p}_1(\mathbf{y})\pi_2 < \hat{p}_2(\mathbf{y})\pi_1$$

where $\hat{p}_1$ and $\hat{p}_2$, statistical estimates of p_1 and p_2, respectively, are used in place of the unknown density functions.

Generally, “empirical” Bayes rules of this form are not necessarily optimal except asymptotically, i.e. as the sample sizes in the calibration study become large, if this results in appropriate convergence of $\hat{p}_1$ and $\hat{p}_2$ to p_1 and p_2, respectively. So, for example, for small-to-moderate calibration sample sizes, empirical linear discriminant analysis often outperforms empirical quadratic discriminant analysis even in settings where the latter would be optimal if p_1 and p_2 were known.

A second practical issue, and one that is not often addressed in the evaluation of classification rules, is the possibility that the sample measurements used in calibration are not entirely representative of those which will be seen in field work. One way to express this is to say that the distributions from which measurement vectors are drawn in calibration, p_1^c and p_2^c, are different than those encountered in the field, p_1^f and p_2^f. Hence $\hat{p}_1$ and $\hat{p}_2$ used in the empirical Bayes rule are, in effect, estimates of the wrong distributions. The degree to which this affects performance of the classification rule is largely determined by the nature and degree of the differences between p_1^c and p_1^f, and between p_2^c and p_2^f.

3. A SPECIFIC PROBABILITY MODEL

3.1. The Normal Model

In this study, we shall limit attention to probability models which specify multivariate normal, or "Gaussian", distributions (MVN) for the vector of measurements produced by an array:

$$p(\mathbf{y}) = (2\pi)^{-s/2}|\mathbf{\Sigma}|^{-s/2}exp\{-\tfrac{1}{2}(\mathbf{y}-\mu)'\mathbf{\Sigma}^{-1}(\mathbf{y}-\mu)\}$$

where

s = the dimension of $\mathbf{y}$, i.e. the number of sensors,

μ = the s-element vector for which the ith element is the expectation of $y^{(i)}$,

$\mathbf{\Sigma}$ = the s-by-s element positive semi-definite matrix for which the (i, j) element is the covariance of $y^{(i)}$ and $y^{(j)}$,

$|\mathbf{\Sigma}|$ = the matrix determinant of $\mathbf{\Sigma}$,

$\mathbf{\Sigma}^{-1}$ = the matrix inverse of $\mathbf{\Sigma}$.

Hence, the MVN distribution is entirely specified by its probability moments of order 1 and 2. In calibration experiments, these moment parameters are estimated as $\hat{\mu}$ and $\hat{\mathbf{\Sigma}}$, and substituting these estimates into the above expression yields the estimated density $\hat{p}$ used in the empirical Bayes rule. Depending on the explicit or tacit assumptions made in the analysis of the calibration data, some or all of the elements of $\hat{\mu}$ and $\hat{\mathbf{\Sigma}}$ will be different for the two class models, leading to different $\hat{p}_1$ and $\hat{p}_2$.

While not always entirely physically realistic, MVN models have a number of convenient properties in analyses of the sort undertaken here:

1. They are the basis of the theory establishing optimality of the known-parameter linear and quadratic discriminant rules.

2. They reduce characterization of p to moments of order 1 and 2.

3. They have a convenient reproductive property; sums of vectors which have independent MVN distributions are also MVN random vectors.

The third property is convenient in our context, as it allows us to think about MVN measurement vectors ($\mathbf{y}$) as being comprised of MVN "truth" vectors ($\mathbf{x}$) and additive MVN "error" vectors ($\mathbf{e}$),

$$\mathbf{y} = \mathbf{x} + \mathbf{e}$$

where some or all of the characteristics of these random variables are different for different classes. More precisely, with reference to the ideas and notation used in Section 2, we suppose that for the objects in class 1, the associated true values are drawn from a distribution characterized by $MVN(\mathbf{m}_1, \mathbf{V}_1)$, that measurement errors are characterized by $MVN(\mathbf{b}, \mathbf{P})$, and that $\mathbf{x}$ and $\mathbf{e}$ are stochastically independent.

Here the symbols $\mathbf{b}$ and $\mathbf{P}$ are selected because of their relationship to *bias* and *precision* in the measurement system. Hence, for systems of physically similar sensors (e.g. SAWs), it may be very reasonable to conclude that all elements of $\mathbf{b}$ are equal; i.e. that any structural bias affecting the measurements made by one sensor is likely present in the measurements made by all sensors. Similarly, it may be reasonable to consider variance matrices $\mathbf{P}$ for which all diagonal elements are equal (reflecting a common precision for measurements taken from each sensor), and equal but somewhat smaller positive off-diagonal values. These off-diagonal elements are covariances between the errors in associated with different sensors in the array. A relatively large common off-diagonal value reflects a situation in which the errors are highly correlated, as would be the case when instability of a common power supply, or a contaminating agent is the sample to which all sensors are exposed, tends to produce similar distortion in all elements of $\mathbf{y}$. In contrast, a relatively small common off-diagonal value reflects a situation in which the errors are not highly correlated, as would be the case when sensors are more self-contained, and/or are exposed to different subsamples of the object of interest.

The distribution of $\mathbf{y}$ is then also MVN, as suggested by the third property above, and specifically, for the measurement vectors associated with objects from class 1:

$$\mathbf{y} \sim MVN(\mu_1 = \mathbf{m}_1 + \mathbf{b}, \mathbf{\Sigma}_1 = \mathbf{V}_1 + \mathbf{P})$$

Likewise for measurements associated with class 2:

$$\mathbf{y} \sim MVN(\mu_2 = \mathbf{m}_2 + \mathbf{b}, \mathbf{\Sigma}_2 = \mathbf{V}_2 + \mathbf{P})$$

if measurement error characteristics are the same for both classes.

While it seems reasonable that, at least in many situations, $\mathbf{b}$ and $\mathbf{P}$ should not be affected by the class identity of the objects being examined, these quantities may well be affected by the conditions under which the measurements are made. Hence, for example, the variability of repeated measurements of the same object may well be expected to be more variable under field conditions than under the relatively more tightly controlled conditions of laboratory calibration, suggesting that a matrix $\mathbf{P}$ containing larger elements may be more appropriate when modeling field conditions. Following this idea, our general probability model for measurement vectors is:

$\mathbf{y} \sim MVN(\mu_1^c = \mathbf{m}_1 + \mathbf{b}^c, \mathbf{\Sigma}_1^c = \mathbf{V}_1 + \mathbf{P}^c)$ for class 1 objects in a calibration environment,

$\mathbf{y} \sim MVN(\mu_2^c = \mathbf{m}_2 + \mathbf{b}^c, \mathbf{\Sigma}_2^c = \mathbf{V}_2 + \mathbf{P}^c)$ for class 2 objects in a calibration environment,

$\mathbf{y} \sim MVN(\mu_1^f = \mathbf{m}_1 + \mathbf{b}^f, \mathbf{\Sigma}_1^f = \mathbf{V}_1 + \mathbf{P}^f)$ for class 1 objects in a field environment, and

$\mathbf{y} \sim MVN(\mu_2^f = \mathbf{m}_2 + \mathbf{b}^f, \mathbf{\Sigma}_2^f = \mathbf{V}_2 + \mathbf{P}^f)$ for class 2 objects in a fleld environment.

3.2. Reduction of Problem Size

While we have already invoked a number of restrictive assumptions (exclusive use of the *MVN* distribution, and additive and independent error structures), the number of "free" parameters remaining in our probability model is still quite large, especially given the relative lack of information which may generally be available at the initial planning phase of sensor array design. Hence, we shall adopt further restrictions at this point so as to focus attention on a relative few quantities which still allow considerable variety in the kinds of situations which can be modeled.

1.) *Assume that measurement bias properties are the same under calibration and field conditions.* This implies that $\mathbf{b}^c = \mathbf{b}^f$. We effectively say here that any *systematic* sources of error are present in both operating conditions. This assumption is most questionable when instruments may be subject to drift which cannot be corrected over time.

2.) *Assume that calibration(field) precision variances are equal for all sensors, that calibration(field) precision covariances are equal for every pair of sensors, and that each precision variance(covariance) differs by the same proportion between calibration and field conditions.* This implies that the precision matrices can be written as:

$$\mathbf{P}^c = \delta^2[(1-\eta)\mathbf{I} + \eta\mathbf{J}]$$
$$\mathbf{P}^f = \phi^2\mathbf{P}^c$$

Here, δ^2 is the variance associated with error for each sensor under calibration conditions, η is the (unitless) *correlation* between errors associated with any two sensors under calibration conditions, and ϕ (generally one or greater) is the proportional inflation in standard deviations associated with field operation (relative to calibration conditions). Hence, $\phi = 1.2$ represents an increase of 20% in all precision-associated standard deviations under field conditions, and $\phi = 1.0$ represents the situation in which precision is not degraded in the field. This assumption is most questionable when field conditions influence sensor-specific error sources more or oless than sensor-common error sources.

3.) *Assume that the difference between the class-specific means of true values associated with any given sensor has the same absolute value for all sensors.* This implies that $\mathbf{m_1} - \mathbf{m_2} = \Delta\mathbf{u}$, where Δ is nonnegative and $\mathbf{u}$ is a s-element vector consisting only of +1's and -1's. This restriction is admittedly somewhat arbitrary, since class "separation" is apt to be larger in some sensor dimensions than in others. However, these differences may not be large (or at least *known* to be large) for groups of sensors being considered together in array construction.

4.) *Assume that within each class, the variances of true values associated with each sensor are equal, and that the covariances of true values associated with each pair of sensors are equal.* This implies that we may write:

$$\mathbf{V}_1 = \sigma_1^2[(1-\rho_1)\mathbf{I} + \rho_1\mathbf{J}]$$
$$\mathbf{V}_2 = \sigma_2^2[(1-\rho_2)\mathbf{I} + \rho_2\mathbf{J}]$$

While this proposes exchangeable variance properties for sets of true values within a class, it does not imply a relationship between the two classes.

As a postscript to the third assumption listed above, it should be noted that even when the differences between the class-specific means associated with each sensor are equal, the interaction between the *number* of positive and negative differences and the correlation structure can have a major affect on the difficulty of the classification problem. As an illustration of this, Figure 1 depicts two simple situations in which arrays contain two sensors and for which the measurements associated with these two sensors are positively correlated within each class. The ellipses can be thought of as the regions within which the bivariate measurements for most objects within a class would be located. Figure 1a depicts two classes for which the mean difference is of the same sign for each of the two sensors, i.e. the "shift" from one ellipse to the other is along a line of slope +1. Figure 1b displays two classes which have the same internal structure, but for which the intra-class difference in means is of opposite sign for the two sensors. Even though the two-dimensional "shift" from one class to the other has the same *magnitude* in each figure, the classification problem is much more difficult in Figure 1a because of the greater degree of "overlap" between classes.

Taken together, these assumptions/restrictions lead to

$\mathbf{y} \sim MVN(\mu_1 = \mathbf{m}_1 + \mathbf{b}, \mathbf{\Sigma}_1^c = \sigma_1^2[(1-\rho_1)\mathbf{I} + \rho_1\mathbf{J}] + \delta^2[(1-\eta)\mathbf{I} + \eta\mathbf{J}])$ for class 1 objects in a calibration environment,

$\mathbf{y} \sim MVN(\mu_2 = \mathbf{m}_2 + \mathbf{b}, \mathbf{\Sigma}_2^c = \sigma_2^2[(1-\rho_2)\mathbf{I} + \rho_2\mathbf{J}] + \delta^2[(1-\eta)\mathbf{I} + \eta\mathbf{J}])$ for class 2 objects in a calibration environment,

$\mathbf{y} \sim MVN(\mu_1 = \mathbf{m}_1 + \mathbf{b}, \mathbf{\Sigma}_1^f = \sigma_1^2[(1-\rho_1)\mathbf{I} + \rho_1\mathbf{J}] + \phi^2\delta^2[(1-\eta)\mathbf{I} + \eta\mathbf{J}])$ for class 1 objects in a field environment,

$\mathbf{y} \sim MVN(\mu_2 = \mathbf{m}_2 + \mathbf{b}, \mathbf{\Sigma}_2^f = \sigma_2^2[(1-\rho_1)\mathbf{I} + \rho_2\mathbf{J}] + \phi^2\delta^2[(1-\eta)\mathbf{I} + \eta\mathbf{J}])$ for class 2 objects in a environment environment.

As a result, a final reduction in the dimension of this model is possible by effectively shifting the first class mean to a vector of zeros, and rescaling all measurements so that shifts between the population means are +1 or -1 in each direction. Hence the model above is equivalent to:

$\mathbf{y} \sim MVN(\mu_1 = \mathbf{0}, \mathbf{\Sigma}_1^c = (\sigma_1^2/\Delta^2)[(1-\rho_1)\mathbf{I} + \rho_1\mathbf{J}] + (\delta^2/\Delta^2)[(1-\eta)\mathbf{I} + \eta\mathbf{J}])$ for class 1 objects in a calibration environment,

$\mathbf{y} \sim MVN(\mu_2 = \mathbf{u}, \mathbf{\Sigma}_2^c = (\sigma_2^2/\Delta^2)[(1-\rho_2)\mathbf{I} + \rho_2\mathbf{J}] + (\delta^2/\Delta^2)[(1-\eta)\mathbf{I} + \eta\mathbf{J}])$ for class 2 objects in a calibration environment,

$\mathbf{y} \sim MVN(\mu_1 = \mathbf{0}, \mathbf{\Sigma}_1^f = (\sigma_1^2/\Delta^2)[(1-\rho_1)\mathbf{I} + \rho_1\mathbf{J}] + \phi^({\delta^2/\Delta^2})[(1-\eta)\mathbf{I} + \eta\mathbf{J}])$ for class 1 objects in a field environment,

$\mathbf{y} \sim MVN(\mu_2 = \mathbf{u}, \mathbf{\Sigma}_2^f = (\sigma_2^2/\Delta^2)[(1-\rho_1)\mathbf{I} + \rho_2\mathbf{J}] + \phi^({\delta^2/\Delta^2})[(1-\eta)\mathbf{I} + \eta\mathbf{J}])$ for class 2 objects in a environment environment.

Hence nine free parameter values must be specified to characterize the probability model for a system of s sensors:

1. s^+: number of +1 elements in $\mathbf{u}$ (integer, $\{0,s\}$)

2. σ_1/Δ: standard deviation of true values in class 1, relative to the common absolute difference between class means (positive)

3. ρ_1: correlation between each pair of true values in class 1 (0,1),

4. σ_2/Δ: standard deviation of true values in class 2, relative to the common absolute difference between class means (positive)

5. ρ_2: correlation between each pair of true values in class 2 (0,1),

6. δ/Δ: standard deviation of measurement errors in the calibration environment, relative to the common absolute difference between class means (positive),

7. ϕ: inflation in measurement error standard deviations in the field environment (positive, no less than 1),

8. η: correlation between measurement errors for any pair of sensors (0,1),

9. π_1: probability that any given future observation presented for classification is actually a member of class 1.

4. EMPIRICAL DISCRIMINANT RULES

As noted earlier, in most practical classification problems the characteristics of the underlying probability models are not known from first principles, and a calibration experiment is required to allow construction of an empirical discriminant rule. In this study, we shall focus on three popular discriminant rules: linear discriminant (LD), quadratic discriminant (QD) and k-nearest neighbor discriminant (NND). (These methods are discussed in more detail in popular textbooks on classification, e.g. Devroye et al, 1996.) These can each be interpreted as an estimate of the known-parameter Bayes rule, based on varying degrees of assumptions about the underlying probability model. Roughly speaking, LD is based on the strongest assumptions and requires estimation of the fewest underlying parameters, NND is based on the weakest assumptions and requires estimation of the largest (effective) number of underlying parameters, and QD is intermediate on each scale. *Note: None of the procedures investigated here takes advantage of the specific model developed in Section 3. That model is intended to be a sensible test-bed for the representation of physical sensors and classification problems. Here we shall use it only to generate scenarios under which the performance of each of these general analysis methods can be evaluated and compared.*

For each empirical rule, an experiment is required to collect data from which relevant population characteristics acn be estimated. We assume that the experiment consists of one (vector) measurement of each of n objects randomly drawn from class 1, and n objects randomly drawn from class 2. We let:

$$\mathbf{y}_{1,1}, \mathbf{y}_{1,2}, \mathbf{y}_{1,3}, \cdots, \mathbf{y}_{1,n}$$

represent the n vectors associated with the objects from class 1, and

$$\mathbf{y}_{2,1}, \mathbf{y}_{2,2}, \mathbf{y}_{2,3}, \cdots, \mathbf{y}_{2,n}$$

represent the n vectors associated with the objects from class 2.

4.1 Linear Discriminant

Linear discriminant (LD) analysis is an empirical version of the Bayes known-parameter rule, under the assumption that p_1 and p_2 are MVN distributions with $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2$. Let

$$\begin{aligned}
\bar{\mathbf{y}}_i &= \textstyle\sum_{j=1}^{n} \mathbf{y}_{i,j} \qquad i = 1,2 \\
\mathbf{S}_i^2 &= \tfrac{n}{n-1}\left[\tfrac{1}{n}\textstyle\sum_{j=1}^{n} \mathbf{y}_{i,j}\mathbf{y}_{i,j}' - \bar{\mathbf{y}}_i\bar{\mathbf{y}}_i'\right] \\
\mathbf{S}^2 &= \tfrac{1}{2}[\mathbf{S}_1^2 + \mathbf{S}_2^2]
\end{aligned}$$

Then, given a new measurement vector $\mathbf{y}$, the LD rule is:

$$\begin{aligned}
&d(\mathbf{y}) = 1 \text{ if } (\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2)'[\mathbf{S}^2]^{-1}\mathbf{y} + 2log\tfrac{\pi_1}{\pi_2} + \bar{\mathbf{y}}_{\mathbf{2}}'[\mathbf{S}^2]^{-1}\bar{\mathbf{y}}_{\mathbf{2}} - \bar{\mathbf{y}}_{\mathbf{1}}'[\mathbf{S}^2]^{-1}\bar{\mathbf{y}}_{\mathbf{1}} > 0, \\
&d(\mathbf{y}) = 2 \text{ otherwise.}
\end{aligned}$$

4.2 Quadratic Discriminant

Quadratic discriminant (QD) analysis is an empirical version of the Bayes known-parameter rule, under the assumption that p_1 and p_2 are MVN distributions, but without the assumption of equal covariance matrices. Using the notation defined in Section 4.1 along with

$$r_i^2 = (\mathbf{y} - \bar{\mathbf{y}}_i)'[\mathbf{S}^2]^{-1}(\mathbf{y} - \bar{\mathbf{y}}_i), \quad i = 1,2$$

and given a new measurement vector $\mathbf{y}$, the QD rule is:

$$\begin{aligned}
&d(\mathbf{y}) = 1 \text{ if } r_2^2 - r_1^2 + 2log\tfrac{\pi_1}{\pi_2} - log\tfrac{|\mathbf{S}_1^2|}{|\mathbf{S}_2^2|} > 0, \\
&d(\mathbf{y}) = 2 \text{ otherwise.}
\end{aligned}$$

4.3 k-Nearest Neighbor Discriminant

Nearest Neighbor (NND) analysis does not reduce the data collected during calibration, but uses the distinct data values collected in the discriminant rule. An integer-valued parameter k, generally chosen to be odd and small relative to $2n$ is selected, and a distance measure

$||...||$ is chosen. Given a new measurement vector $\mathbf{y}$, the kNND rule is as follows: Identify the k vectors from the pooled set of $2n$ calibration vectors $\mathbf{y}_{i,j}$ which are *closest* to $\mathbf{y}$ in the sense of minimizing $||\mathbf{y} - \mathbf{y}_{i,j}||$. Let the number of these nearest calibration vectors from class 1 be k_1 and the number from class 2 be k_2, $k_1 + k_2 = k$. Then

$$d(\mathbf{y}) = 1 \text{ if } k_1 > k_2,$$
$$d(\mathbf{y}) = 2 \text{ if otherwise.}$$

5. EFFECTIVENESS OF DISCRIMINANT RULES

The practical effectiveness of any discriminant rule is its reliability in practice. A more specific measure of effectiveness is the frequency (or probability) with which it results in erroneous classifications when used in the field. As in statistical hypothesis testing, two kinds of errors are possible when a selection must be made between two possible classes. The probability of incorrectly classifying an object as a member of class 2, given that it is in fact a member of class 1, may be written as:

$$\alpha_1 = Prob\{d(\mathbf{y}) = 2 | \text{ object is in class 1 } \}.$$

The symmetric probability of incorrectly classifying an object as a member of class 1, given that it is in fact a member of class 2, may be written as:

$$\alpha_2 = Prob\{d(\mathbf{y}) = 1 | \text{ object is in class 2 } \}.$$

An overall error rate or probability can be calculated as:

$$\alpha = \alpha_1 \pi_1 + \alpha_2 \pi_2$$

where π_1 and π_2 are the relative frequencies with which objects of class 1 and 2, respectively, will be classified. In many important cases, this overall error probability is not the most meaningful measure of performance because the two kinds of errors have different degrees of impact. For example, if the two classes to be discriminated are groups of chemicals which are poison and harmless to exposed humans, respectively, the potential consequences of making one kind of error may be much greater than those of the other. In such cases, cost factors can be introduced which effectively weight one error more heavily than than the other, combining the two probabilities into an overall expected cost. The unweighted overall error α treats the two specific kinds of error symmetrically, and so is strictly appropriate only when they are of equal significance. In this study, we shall restrict attention to the performance measure α, primarily as an expedient. Modification of both the arguments and programs to be presented in the next section is simple for the case of unequal known costs for each error.

For known-parameter rules, such as the general Bayes rule based on known class distributions, α_i, $i = 1, 2$ are integrals of the probability function indicated by the subscript, over the region for which the decision function yields the incorrect classification, e.g.

$$\alpha_1 = \int_{d(\mathbf{y})=2} p_1^f(\mathbf{y})d\mathbf{y}.$$

α_2 can be similarly defined, and the overall error rate α determined as indicated. However, when the decision rule is formulated based on data from a calibration experiment, d becomes a random variable itself, with distribution determined by p_1^c and p_2^c, the distributions corresponding to the two classes as observed in the calibration experiment. In order to distinguish calibration data from data to be observed in the field, let:

$$\mathbf{y}_{1,1}, \mathbf{y}_{1,2}, \mathbf{y}_{1,3}, ..., \mathbf{y}_{1,n}$$

represent the n vectors associated with the objects from class 1,

$$\mathbf{y}_{2,1}, \mathbf{y}_{2,2}, \mathbf{y}_{2,3}, ..., \mathbf{y}_{2,n}$$

represent the n vectors associated with the objects from class 2, and let $\mathbf{Y}$ represent the collection of all these data from the calibration experiment. Note that the decision rule, d, is actually defined by these data, so that it might be written as $d_{\mathbf{Y}}(\mathbf{y})$ for an arbitrary field observation $\mathbf{y}$. For such empirical discriminant rules, the *expected* (over the distribution of the calibration data set) error rate for objects in class 1 can at least in principle be calculated, e.g.:

$$\alpha_1 = \int_{\mathbf{Y}} [\int_{d(\mathbf{y})=2} p_1^f(\mathbf{y})d\mathbf{y}] \prod_i p_1^c(\mathbf{y_{1,i}}) \prod_j p_2^c(\mathbf{y_{2,j}})d\mathbf{Y}$$

Again, α_2 can be symmetrically defined, and if π_1 and π_2 are known, the expected error rate α calculated as:

$$\alpha = \alpha_1\pi_1 + \alpha_2\pi_2.$$

Discriminant rules for which α is relatively small for class distributions of interest are preferred to rules for which this error rate is larger.

For empirical decision rules, α is generally not easy to calculate. If calibration and field class distributions are the same for each class (i.e. $p_1^c = p_1^f$ and $p_2^c = p_2^f$), the misclassification probability for most sensible rules decreases as the calibration sample size increases. This is because the rule implicitly depends on the samples $\mathbf{Y}$ to provide estimates of the unknown distributions p_1^c and p_2^c, and so will be a more effective classifier when these estimates are more precise. Performance can also improve with increased sample sizes in some cases where the calibration and field class distributions are different, but this situation is much more complex. Even for fairly simple cases of interest, explicit expressions which allow simple evaluate of α are usually unavailable. The following section describes software (included in the appendix of this report) which can be used to evaluate the expected misclassification rates for the models discussed in this report.

By way of a historical perspective, one earlier paper which reported numerical results comparing the effectiveness of classification procedures is Van Ness and Simpson (1976). The authors consider a more limited model than that discussed here, and are not motivated so specifically by considerations arising from sensor arrays. However they do deal with the fundamental question of how the dimensionality of the problem (here, number of sensors) affects the misclassification rates that can be expected from empirical rules based on calibration studies.

6. PROGRAM ARRAYSIZE.sas

The appendix contains SAS (Statistical Analysis System, 2001) routines which can be used to calculate error rates for the three discriminant rules discussed in Section 4, for the model described in Section 3. Both "forward" calculations (e.g. the error rate for a given calibration sample size) and "inverse" calculations (e.g. estimates of the calibration sample size required to obtain a given error rate) are included.

The SAS code is contained in 3 files, entitled ARRAYSIZE.sas, mainproc.sas, and modules.sas. The first of these, ARRAYSIZE.sas, is essentially a specifications file in which the user defines the quantities which set up the problem. The variable names used in this file correspond, at least loosely, to the notation in this report. The other two files, mainproc.sas and modules.sas, contain the programming and subroutines used to execute the calculations, and would typically not need to be modified unless the user wishes to change characteristics of the problem not discussed in this report. The files are set up to be run as batch SAS jobs, rather than interactively, because the calculations – particularly the inverse calculations, and calculations involving the nearest-neighbor rule – can be fairly time-consuming.

Table 1 contains a listing of the variables for which values are set in ARRAYSIZE.sas, along with definitions of these quantities. (Brief definitions are also included as comments in the file.)

Upon execution, the program performs three sets of calculations. The first is a set of "theoretical" error rates for the linear and quadratic discriminant rules. These are the error rates which would apply if the underlying parameters defining the class distributions in the calibration environment were actually known, instead of estimated. They represent a "best case" or lower bound that can be achieved with an empirical rule. These error rates are calculated based on the model parameter values specified in ARRAYSIZE.sas, for values of s (number of sensors) and s^+ from 1 through half the maximum number specified. (It is not necessary to compute performance results for s^+ greater than s because they are identical for $s^+ = \frac{s}{2} + t$ and $s^+ = \frac{s}{2} - t$ for any t resulting in non-negative integer values of s^+. Hence, for example, results for $s = 10$ and $s^+ = 8$ are the same as for $s = 10$ and $s^+ = 2$, as are results for $s = 7$ and $s^+ = 2$ or 5.)

The second set of calculations performed produces expected error rates for the empirical versions of linear, quadratic, and nearest-neighbor classification, for the maximum calibration sample size specified in ARRAYSIZE.sas. These calculations are carried out via stochastic simulation; simulated calibration data are generated, the discriminant rule calculated based on these calibration data, additional simulated field data are generated, and the error rates estimated as the averages of the proportion of errors that would be made in classifying these field data. The number of simulation cycles, and number of simulated field classifications made in each cycle, are set in ARRAYSIZE.sas.

The inverse calculation is made at the end of the run. This is also accomplished via a stochastic simulation using a sequential "up-and-down" rule which varies the calibration sample size used in an effort to match the target error rate specified in ARRAYSIZE.sas. Estimates of the required sample sizes are printed in the final set of output tables.

Table 1. Input Parameters Specified in ARRAYSIZE.sas, and Values Used in the Demonstration Example

parameter	example value	description
s_max	9	maximum number of sensors considered
n_max	100	maximum training sample size considered
err	0.05	targeted error rate for inverse calculation
u_bigdelta	2.0	common absolute difference between class means
u_sigma1	1.0	common standard deviation of true values in class 1
u_rho1	0.5	common correlation between each pair of true values in class 1
u_sigma2	1.0	common standard deviation of true values in class 2
u_rho2	0.5	common correlation between each pair of true values in class 2
u_delta	1.0	common standard deviation of measurement errors in calibration environment
u_eta	0.5	common correlation between measurement errors for any pair of sensors
u_phi	1.1	inflation in measurement error std. dev. for field environment
u_pi1	0.5	probability that any given field measurement is for an object from class 1
linear	1	indicator variable for the linear rule (1=include in calculation)
quad	1	indicator variable for the quadratic rule (1=include in calculation)
nearest	1	indicator variable for the k-nearest-neighbor rule (1=include in calculation)
k	1	number of nearest neighbors used in the k-nearest-neighbor rule
n_updown	500	number of simulated experiments used in sample size calculation
nvalid1	50	validation sample size for class 1 used in each simulated experiment
nvalid2	50	validation sample size for class 2 used in each simulated experiment
nrep_sim	50	number of simulated experiments used in empirical error rate calculations
nrep_th	10000	validation sample size used to calculate “theoretical” error rate for quadratic rule

6.1 Example

Display 1 shows output generated by the SAS program as a result of a calculation made using the parameter values given in Table 1. The first part of the output repeats parameter values (and functions of them) that have been supplied as input. The first line gives the difference in mean between classes (in each dimension), and standard deviations and correlations for measurements made for each sensor. Note that standard deviation values are given as $\sqrt{2}$ for each class, reflecting the fact that the standard deviations of both true values and measurement errors are specified as 1 in the input file. The second line gives similar values of mean differences, standard deviations, and correlations for field measurements. These standard deviations are slightly larger than in the previous line, due to the specification of $\phi = 1$, or a 10% increase in standard deviations in the field relative to calibration conditions. All correlations between measurements are 0.5 because all correlations set between true values and measurement errors were also set to this value. The remaining values printed in the first lines of the output contain other parameter and control values specified by the user in ARRAYSIZE.sas.

The next section of the output is a listing of lines summarizing calculations performed to construct the tables at the end of the output. The values from left to right are the observation (or line) number in the SAS data set, values of s and s^+ for the line, the number of calibration samples required to attain the target error rate, and "theoretical" (known parameter) and simulated (estimated parameter) error rates. The column labeled "err_std" is a standard deviation representing the precision with which the simulated error rate is calculated, and the final column gives a code of the discriminant rule used (L for linear, Q for quadratic, and N for nearest-neighbor). This section of the output is primarily of use for post-processing; most of this information is summarized in more convenient form in the tables which make up the remainder of the output.

The first two tables report "theoretical" error rates for linear and quadratic discriminant classification, for values of s and s^+ specified. These can be though of as best-case values, i.e. the error rates which could be expected if there were no uncertainty in the calibration process. The next three tables report "empirical" error rates calculated via simulation, when calibration sample sizes are as specified by the value "n_max" in the input; in this example, 100 training samples from each of the two classes are assumed. In this case, it can be seen that more sensors (larger s) leads to more accurate discrimination, and that more balanced values of s^+ and $s^- = s - s^+$ are also relatively superior. The upper-right triangle of these tables is omitted since, as explained above, these would be identical to their counterparts in the lower-left triangle. The final three tables report calibration sample sizes which would be required to attain the target error rate (0.05 in this example) for each combination of s and s^+. Again, the upper right triangle of each table is omitted. In these tables, other cells are also omitted (i.e. contain a period rather than a number) if the target rate cannot be attained using calibration samples smaller than the maximum specified (100 in this case).

By comparing the entries of these tables, the user can get a general idea of the degree of classification accuracy which can be expected in a given situation (as expressed by parameters in the model) for arrays of different sizes, or the size of calibration experiments that would be required to attain a set level of accuracy.

Display 1: SAS Output for the Example Problem

```
                                 The SAS System

                          M   SIGMA1C     RHO1C   SIGMA2C     RHO2C
                          2 1.4142136       0.5 1.4142136       0.5

                          M   SIGMA1F     RHO1F   SIGMA2F     RHO2F
                          2 1.4866069       0.5 1.4866069       0.5

                                      PI1       PI2
                                      0.5       0.5

                    ERR     S_MAX     N_MAX      NSIM   NVALID1   NVALID2
                   0.05         9       100       500        50        50

                                 NREP_SIM   NREP_TH
                                       50     10000


                                  The SAS System

     Obs    svec    splus    n_size      err_th     err_simu      err_std    method
       1      1       0         .       0.25058      0.2464      0.040545      L
       2      2       0         .       0.21866      0.2286      0.044995      L
       3      2       1         .       0.08926      0.0898      0.031975      L
       4      3       0         .       0.20501      0.2170      0.042630      L
       5      3       1         .       0.05733      0.0578      0.026748      L
       6      4       0         .       0.19742      0.1948      0.035984      L
       7      4       1         .       0.04440      0.0506      0.023854      L
       8      4       2        11       0.02855      0.0300      0.017728      L
       9      5       0         .       0.19258      0.1976      0.043357      L
      10      5       1        24       0.03756      0.0420      0.021571      L
      11      5       2         8       0.01825      0.0166      0.014086      L
      12      6       0         .       0.18923      0.1916      0.048838      L
      13      6       1        19       0.03337      0.0364      0.017468      L
      14      6       2         9       0.01333      0.0178      0.012664      L
      15      6       3         9       0.00990      0.0094      0.009775      L
      16      7       0         .       0.18677      0.2008      0.035848      L
      17      7       1        19       0.03056      0.0288      0.016738      L
      18      7       2         9       0.01056      0.0104      0.010294      L
      19      7       3         8       0.00631      0.0062      0.006966      L
      20      8       0         .       0.18489      0.1940      0.042185      L
      21      8       1        22       0.02855      0.0328      0.017733      L
      22      8       2         9       0.00882      0.0110      0.010926      L
      23      8       3         8       0.00446      0.0054      0.007343      L
      24      8       4         8       0.00357      0.0036      0.004849      L
      25      9       0         .       0.18340      0.1830      0.038611      L
      26      9       1        25       0.02704      0.0372      0.017733      L
      27      9       2        10       0.00765      0.0074      0.008283      L
      28      9       3         9       0.00339      0.0038      0.006667      L
      29      9       4         8       0.00227      0.0024      0.004314      L
      30      1       0         .       0.24700      0.2452      0.047905      Q
      31      2       0         .       0.21685      0.2232      0.045823      Q
      32      2       1         .       0.08700      0.0922      0.025974      Q
      33      3       0         .       0.20295      0.2040      0.034582      Q
      34      3       1         .       0.05680      0.0608      0.022663      Q
      35      4       0         .       0.19265      0.2172      0.041504      Q
      36      4       1        67       0.04545      0.0496      0.023816      Q
      37      4       2        18       0.02950      0.0364      0.018926      Q
      38      5       0         .       0.19415      0.2120      0.039383      Q
      39      5       1        60       0.03965      0.0390      0.019717      Q
```

Display 1, continued:

```
   40      5       2       16     0.01700      0.0226      0.016011       Q
   41      6       0        .     0.18835      0.2060      0.039898       Q
   42      6       1       31     0.03505      0.0414      0.020801       Q
   43      6       2       16     0.01305      0.0178      0.013596       Q
   44      6       3       15     0.00995      0.0128      0.008816       Q
   45      7       0        .     0.18325      0.2068      0.051802       Q
   46      7       1       38     0.03050      0.0302      0.014356       Q
   47      7       2       17     0.01060      0.0118      0.011373       Q
   48      7       3       15     0.00580      0.0076      0.008935       Q
   49      8       0        .     0.18340      0.2286      0.043237       Q
   50      8       1       43     0.02870      0.0352      0.016567       Q
   51      8       2       19     0.00950      0.0106      0.009348       Q
   52      8       3       16     0.00365      0.0056      0.007602       Q
   53      8       4       16     0.00395      0.0050      0.006776       Q
   54      9       0        .     0.18040      0.2260      0.042137       Q
   55      9       1       45     0.02530      0.0356      0.019604       Q
   56      9       2       21     0.00755      0.0124      0.009596       Q
   57      9       3       18     0.00295      0.0044      0.006749       Q
   58      9       4       16     0.00240      0.0016      0.003703       Q
   59      1       0        .        .         0.3230      0.054220       N
   60      2       0        .        .         0.2886      0.047639       N
   61      2       1        .        .         0.1240      0.036027       N
   62      3       0        .        .         0.2704      0.054882       N
   63      3       1        .        .         0.0874      0.030560       N
   64      4       0        .        .         0.2832      0.045014       N
   65      4       1        .        .         0.0796      0.032385       N
   66      4       2       72        .         0.0482      0.019345       N
   67      5       0        .        .         0.2722      0.042248       N
   68      5       1        .        .         0.0624      0.025759       N
   69      5       2       13        .         0.0352      0.016066       N
   70      6       0        .        .         0.2592      0.041983       N
   71      6       1        .        .         0.0624      0.026385       N
   72      6       2        8        .         0.0282      0.019451       N
   73      6       3        6        .         0.0212      0.013192       N
   74      7       0        .        .         0.2596      0.043045       N
   75      7       1        .        .         0.0600      0.026419       N
   76      7       2       10        .         0.0248      0.016689       N
   77      7       3        4        .         0.0168      0.013161       N
   78      8       0        .        .         0.2520      0.047337       N
   79      8       1        .        .         0.0582      0.024964       N
   80      8       2        8        .         0.0230      0.016444       N
   81      8       3        4        .         0.0158      0.010897       N
   82      8       4        4        .         0.0072      0.009267       N
   83      9       0        .        .         0.2590      0.044869       N
   84      9       1        .        .         0.0630      0.023669       N
   85      9       2        8        .         0.0164      0.012081       N
   86      9       3        4        .         0.0088      0.010428       N
   87      9       4        3        .         0.0048      0.006465       N
```

Display 1, continued:

```
     THEORETICAL (KNOWN-PARAMETER) ERROR RATE
method linear
--------------------------------------------------
|                      |           S+            |
|                      |-------------------------|
|                      | 0  | 1  | 2  | 3  | 4  |
|----------------------+----+----+----+----+----|
|S                     |    |    |    |    |    |
|----------------------|    |    |    |    |    |
|1                     |.251|   .|   .|   .|   .|
|----------------------+----+----+----+----+----|
|2                     |.219|.089|   .|   .|   .|
|----------------------+----+----+----+----+----|
|3                     |.205|.057|   .|   .|   .|
|----------------------+----+----+----+----+----|
|4                     |.197|.044|.029|   .|   .|
|----------------------+----+----+----+----+----|
|5                     |.193|.038|.018|   .|   .|
|----------------------+----+----+----+----+----|
|6                     |.189|.033|.013|.010|   .|
|----------------------+----+----+----+----+----|
|7                     |.187|.031|.011|.006|   .|
|----------------------+----+----+----+----+----|
|8                     |.185|.029|.009|.004|.004|
|----------------------+----+----+----+----+----|
|9                     |.183|.027|.008|.003|.002|
--------------------------------------------------
```

```
    THEORETICAL (KNOWN-PARAMETER) ERROR RATE
method quadratic
--------------------------------------------------
|                      |           S+            |
|                      |-------------------------|
|                      | 0  | 1  | 2  | 3  | 4  |
|----------------------+----+----+----+----+----|
|S                     |    |    |    |    |    |
|----------------------|    |    |    |    |    |
|1                     |.247|   .|   .|   .|   .|
|----------------------+----+----+----+----+----|
|2                     |.217|.087|   .|   .|   .|
|----------------------+----+----+----+----+----|
|3                     |.203|.057|   .|   .|   .|
|----------------------+----+----+----+----+----|
|4                     |.193|.045|.030|   .|   .|
|----------------------+----+----+----+----+----|
|5                     |.194|.040|.017|   .|   .|
|----------------------+----+----+----+----+----|
|6                     |.188|.035|.013|.010|   .|
|----------------------+----+----+----+----+----|
|7                     |.183|.031|.011|.006|   .|
|----------------------+----+----+----+----+----|
|8                     |.183|.029|.010|.004|.004|
|----------------------+----+----+----+----+----|
|9                     |.180|.025|.008|.003|.002|
--------------------------------------------------
```

Display 1, continued:

```
EMPIRICAL (ESTIMATED-PARAMETER) ERROR RATE FOR N = 100
     method linear
     ----------------------------------------------
     |                   |          S+            |
     |                   |------------------------|
     |                   | 0  | 1  | 2  | 3  | 4  |
     |-------------------+----+----+----+----+----|
     |S                  |    |    |    |    |    |
     |-------------------|    |    |    |    |    |
     |1                  |.246|   .|   .|   .|   .|
     |-------------------+----+----+----+----+----|
     |2                  |.229|.090|   .|   .|   .|
     |-------------------+----+----+----+----+----|
     |3                  |.217|.058|   .|   .|   .|
     |-------------------+----+----+----+----+----|
     |4                  |.195|.051|.030|   .|   .|
     |-------------------+----+----+----+----+----|
     |5                  |.198|.042|.017|   .|   .|
     |-------------------+----+----+----+----+----|
     |6                  |.192|.036|.018|.009|   .|
     |-------------------+----+----+----+----+----|
     |7                  |.201|.029|.010|.006|   .|
     |-------------------+----+----+----+----+----|
     |8                  |.194|.033|.011|.005|.004|
     |-------------------+----+----+----+----+----|
     |9                  |.183|.037|.007|.004|.002|
     ----------------------------------------------
```

```
EMPIRICAL (ESTIMATED-PARAMETER) ERROR RATE FOR N = 100
     method nearest neighbors
     ----------------------------------------------
     |                   |          S+            |
     |                   |------------------------|
     |                   | 0  | 1  | 2  | 3  | 4  |
     |-------------------+----+----+----+----+----|
     |S                  |    |    |    |    |    |
     |-------------------|    |    |    |    |    |
     |1                  |.323|   .|   .|   .|   .|
     |-------------------+----+----+----+----+----|
     |2                  |.289|.124|   .|   .|   .|
     |-------------------+----+----+----+----+----|
     |3                  |.270|.087|   .|   .|   .|
     |-------------------+----+----+----+----+----|
     |4                  |.283|.080|.048|   .|   .|
     |-------------------+----+----+----+----+----|
     |5                  |.272|.062|.035|   .|   .|
     |-------------------+----+----+----+----+----|
     |6                  |.259|.062|.028|.021|   .|
     |-------------------+----+----+----+----+----|
     |7                  |.260|.060|.025|.017|   .|
     |-------------------+----+----+----+----+----|
     |8                  |.252|.058|.023|.016|.007|
     |-------------------+----+----+----+----+----|
     |9                  |.259|.063|.016|.009|.005|
     ----------------------------------------------
```

Display 1, continued:

```
EMPIRICAL (ESTIMATED-PARAMETER) ERROR RATE FOR N = 100
     method quadratic
     ---------------------------------------------
     |                  |           S+           |
     |                  |------------------------|
     |                  | 0  | 1  | 2  | 3  | 4  |
     |------------------+----+----+----+----+----|
     |S                 |    |    |    |    |    |
     |------------------|    |    |    |    |    |
     |1                 |.245|   .|   .|   .|   .|
     |------------------+----+----+----+----+----|
     |2                 |.223|.092|   .|   .|   .|
     |------------------+----+----+----+----+----|
     |3                 |.204|.061|   .|   .|   .|
     |------------------+----+----+----+----+----|
     |4                 |.217|.050|.036|   .|   .|
     |------------------+----+----+----+----+----|
     |5                 |.212|.039|.023|   .|   .|
     |------------------+----+----+----+----+----|
     |6                 |.206|.041|.018|.013|   .|
     |------------------+----+----+----+----+----|
     |7                 |.207|.030|.012|.008|   .|
     |------------------+----+----+----+----+----|
     |8                 |.229|.035|.011|.006|.005|
     |------------------+----+----+----+----+----|
     |9                 |.226|.036|.012|.004|.002|
     ---------------------------------------------


CALIBRATION SAMPLE SIZE REQUIRED FOR ERROR RATE < 0.05
     method linear
     ---------------------------------------------
     |                  |           S+           |
     |                  |------------------------|
     |                  | 0  | 1  | 2  | 3  | 4  |
     |------------------+----+----+----+----+----|
     |S                 |    |    |    |    |    |
     |------------------|    |    |    |    |    |
     |1                 |   .|   .|   .|   .|   .|
     |------------------+----+----+----+----+----|
     |2                 |   .|   .|   .|   .|   .|
     |------------------+----+----+----+----+----|
     |3                 |   .|   .|   .|   .|   .|
     |------------------+----+----+----+----+----|
     |4                 |   .|   .|  11|   .|   .|
     |------------------+----+----+----+----+----|
     |5                 |   .|  24|   8|   .|   .|
     |------------------+----+----+----+----+----|
     |6                 |   .|  19|   9|   9|   .|
     |------------------+----+----+----+----+----|
     |7                 |   .|  19|   9|   8|   .|
     |------------------+----+----+----+----+----|
     |8                 |   .|  22|   9|   8|   8|
     |------------------+----+----+----+----+----|
     |9                 |   .|  25|  10|   9|   8|
     ---------------------------------------------
```

Display 1, continued:

```
CALIBRATION SAMPLE SIZE REQUIRED FOR ERROR RATE < 0.05
     method nearest neighbors
     --------------------------------------------
     |                  |          S+           |
     |                  |-----------------------|
     |                  | 0  | 1  | 2  | 3  | 4  |
     |------------------+----+----+----+----+----|
     |S                 |    |    |    |    |    |
     |------------------|    |    |    |    |    |
     |1                 |   .|   .|   .|   .|   .|
     |------------------+----+----+----+----+----|
     |2                 |   .|   .|   .|   .|   .|
     |------------------+----+----+----+----+----|
     |3                 |   .|   .|   .|   .|   .|
     |------------------+----+----+----+----+----|
     |4                 |   .|   .|  72|   .|   .|
     |------------------+----+----+----+----+----|
     |5                 |   .|   .|  13|   .|   .|
     |------------------+----+----+----+----+----|
     |6                 |   .|   .|   8|   6|   .|
     |------------------+----+----+----+----+----|
     |7                 |   .|   .|  10|   4|   .|
     |------------------+----+----+----+----+----|
     |8                 |   .|   .|   8|   4|   4|
     |------------------+----+----+----+----+----|
     |9                 |   .|   .|   8|   4|   3|
     --------------------------------------------
```

```
CALIBRATION SAMPLE SIZE REQUIRED FOR ERROR RATE < 0.05
     method quadratic
     --------------------------------------------
     |                  |          S+           |
     |                  |-----------------------|
     |                  | 0  | 1  | 2  | 3  | 4  |
     |------------------+----+----+----+----+----|
     |S                 |    |    |    |    |    |
     |------------------|    |    |    |    |    |
     |1                 |   .|   .|   .|   .|   .|
     |------------------+----+----+----+----+----|
     |2                 |   .|   .|   .|   .|   .|
     |------------------+----+----+----+----+----|
     |3                 |   .|   .|   .|   .|   .|
     |------------------+----+----+----+----+----|
     |4                 |   .|  67|  18|   .|   .|
     |------------------+----+----+----+----+----|
     |5                 |   .|  60|  16|   .|   .|
     |------------------+----+----+----+----+----|
     |6                 |   .|  31|  16|  15|   .|
     |------------------+----+----+----+----+----|
     |7                 |   .|  38|  17|  15|   .|
     |------------------+----+----+----+----+----|
     |8                 |   .|  43|  19|  16|  16|
     |------------------+----+----+----+----+----|
     |9                 |   .|  45|  21|  18|  16|
     --------------------------------------------
```

7. SUMMARY

Before actual construction and testing of a sensor array, it is difficult to assess how effective it may be in classifying objects of interest. While larger arrays obviously produce more data than smaller arrays, much of the additional information is redundant in many cases. Further, larger arrays depend on calibration models which contain more parameters for larger arrays, and so may be estimated with less precision unless the effort spent in calibration is also increased. The effects of these factors also depend on the nature of the classes of objects to be classified in field operation.

The goal of the work leading to this report has been the development of a statistical model and computer program to aid in planning the design of sensor arrays. The model is not intended to reflect the specific details of any one type of physical sensor system, but is motivated by general characteristics shared by many such systems. This report contains a description of the model, three popular discriminant rules used in classification, and a description and listing of a SAS program which can be used to calculate expected misclassification rates for array configurations specified by the user. The program will also be available via email from the first author (mmorris@iastate.edu) for a limited time.

REFERENCES

Devroye, L, L. Gyorfi, and G. Lugosi (1996). *A Probabilistic Theory of Pattern Recognition*, Springer, New York.

Osbourn, G.C., J.W. Bartholomew, A.J. Ricco, and G.C. Frye (1997). “A New Approach to Sensor Array Analysis Applied to Volatile Organic Compound Detection: Visual Empirical Region of Influence (VERI) Pattern Recognition,”
http://www.sandia.gov/1100/1155Web/chempap.htm.

Statistical Analysis System (2001), SAS Institute, Carey, North Carolina.

Van Ness, John W. and C. Simpson (1976), “On the Effects of Dimension in Discriminant Analysis,” *Technometrics* **18**, 175-187.

APPENDIX

The three files containing the SAS program described in this report are listed below. These files will also be available via email from the first author (mmorris@iastate.edu) for a limited time.

ARRAYSIZE.sas

```
*file name: "ARRAYSIZE.sas";

/*
  This file sets the parameters for the simulation study.  To
  execute a simulation, this file, "main.sas" and "modules.sas"
  must be placed in the same directory, along with a subdirectory
  named "LIU".
*/

libname mylib 'SCRATCH';
options center  LS = 100;

* ARRAY AND EXPERIMENT SIZE PARAMETERS;
* maximum number of sensors;
%let s_max = 5;
* maximum training sample size;
%let n_max=50;
* targeted errpr rate;
%let err = 0.05;

* USER PARAMETERS DEFINING THE TWO CLASS DISTRIBUTIONS;
* common absolute difference between class means;
%let u_bigdelta = 2.5;
* common standard deviation of true values in class 1;
%let u_sigma1 = 1;
* common correlation between each pair of true values in class 1;
%let u_rho1 = .5;
* common standard deviation of true values in class 2;
%let u_sigma2 = 1;
* common correlation between each pair of true values in class 2;
%let u_rho2 = .5;
* common standard deviation of measurement errors in calibration environment;
%let u_delta =1;
* common correlation between measurement errors for any pair of sensors;
%let u_eta = .5;
* inflation in measurement error std. dev. for field environment;
%let u_phi = 1;
* probability that any given field measurement is for an object from class 1;
%let u_pi1 = .5;

* CHOICE OF DISCRIMINATION RULES;
* indicator variable to include (1) the linear rule;
%let linear=1;
* indicator variable to include (1) the quadratic rule;
%let quad=1;
```

```
* indicator variable to include (1) the k-nearest-neighbor rule;
%let nearest=1;
* number of nearest neighbors used in the k-n-n rule;
%let k=1;

* SIMULATION PARAMETERS;
* number of up-and-down searches;
%let  n_updown=500;
* validation sample size for class 1;
%let  nvalid1 = 50;
* validation sample size for class 2;
%let  nvalid2 = 50;
* repetitions to calculate plug-in rule's error rate;
%let  nrep_sim=50;
* sample size used to calculate theoretical error rate for quadratic rule;
%let  nrep_th=10000;

*Include the main program;
%include "mainproc.sas";
```

mainproc.sas

```
*file name: "mainproc.sas";

proc sql;
  create table mylib.result
    (svec num, splus num, n_size num, err_th num, err_simu num,
     err_std num, method char(8));
quit;

proc iml;

%include "modules.sas";

* Set parameters defining the two classes;
* means;
m = &u_bigdelta;
* sd's and correlations for calibration;
sigma1c = sqrt(&u_sigma1**2+&u_delta**2);
sigma2c = sqrt(&u_sigma2**2+&u_delta**2);
rho1c = (&u_sigma1**2*&u_rho1+&u_delta**2*&u_eta)/sigma1c**2;
rho2c = (&u_sigma2**2*&u_rho2+&u_delta**2*&u_eta)/sigma2c**2;
* sd's and correlations for field;
sigma1f = sqrt(&u_sigma1**2+&u_phi**2*&u_delta**2);
sigma2f = sqrt(&u_sigma2**2+&u_phi**2*&u_delta**2);
rho1f = (&u_sigma1**2*&u_rho1+&u_phi**2*&u_delta**2*&u_eta)/sigma1f**2;
rho2f = (&u_sigma2**2*&u_rho2+&u_phi**2*&u_delta**2*&u_eta)/sigma2f**2;
* probabilities for each class;
pi1 = &u_pi1;
pi2 = 1-&u_pi1;

print m sigma1c rho1c sigma2c rho2c;
print m sigma1f rho1f sigma2f rho2f;
print pi1 pi2;

err = &err;
s_max = &s_max;
```

```
n_max=&n_max;
nsim=&n_updown;
nvalid1 = &nvalid1;
nvalid2 = &nvalid2;
nrep_sim=&nrep_sim;
seed=0;
nrep_th=&nrep_th;
k=&k;
linear=&linear;
quad=&quad;
nearest=&nearest;

print err s_max n_max nsim nvalid1 nvalid2;
print nrep_sim nrep_th;

* Permanent codes begin here;

nvalid = nvalid1+nvalid2;
ptemp=int(s_max/2);
if mod(s_max, 2)=0 then nrow=ptemp*(ptemp+2);
  else nrow=ptemp*(ptemp+3)+1;

* LINEAR RULE;
if (linear = 1) then do;
  svec = J(nrow, 1, .);
  splus = J(nrow, 1, .);
  err_th = J(nrow, 1, .);
  err_simu = J(nrow, 1, .);
  err_std = J(nrow, 1, .);
  n_size = J(nrow, 1, .);
  method = J(nrow, 1, 'L');

* Calculate theoretic error rate and the sample_plug_in error rate
  for given n_max;
  irow=1;
  do p = 1 to s_max;
    do p_plus = 0 to int(p/2);
      p_minus = p-p_plus;
      svec[irow]=p;
      splus[irow]=p_plus;
      run muV(mu1,mu2,V1c,V1f,V2c,V2f,V1chalf,V1fhalf,V2chalf,V2fhalf,
              p_plus,p_minus,m,sigma1c,sigma1f,sigma2c,sigma2f,rho1c,
              rho1f,rho2c,rho2f);
      run Lerr_th(pmis,pi1,pi2,mu1,mu2,V1c,V1f,V2c,V2f);
      err_th[irow] = pmis;
      run Le_rsim(pmis,std,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,
                  V2fhalf,p,n_max,nvalid1,nvalid2,nrep_sim);
      err_simu[irow] = pmis;
      err_std[irow] = std;
      irow=irow+1;
    end;
  end;

* Calculate n for targeted error rate err;
* Note: This calculation is by-passed if err cannot be attained for n_max;
  irow=1;
  do p = 1 to s_max;
    do p_plus = 0 to int(p/2);
      p_minus = p-p_plus;
      run muV(mu1,mu2,V1c,V1f,V2c,V2f,V1chalf,V1fhalf,V2chalf,V2fhalf,
              p_plus,p_minus,m,sigma1c,sigma1f,sigma2c,sigma2f,rho1c,
              rho1f,rho2c,rho2f);
      if (err_simu[irow] < err) then do;
```

```
        run Lup_down(n,pmis,pi1,pi2,mu1,mu2,V1chalf,V1fhalf,V2chalf,
        V2fhalf,nvalid,nsim,err);
        run summary(nstar, mean, weight, n, pmis);
        run p_adj_v(n0, mean_hat, nstar, mean, weight, err);
        n_size[irow] = n0;
      end;
      irow=irow+1;
    end;
  end;

  edit mylib.result var{svec splus n_size err_th err_simu err_std method};
  append;
  close mylib.result;
end;

* QUADRATIC RULE;
if (quad = 1) then do;
  svec = J(nrow, 1, .);
  splus = J(nrow, 1, .);
  err_th = J(nrow, 1, .);
  err_simu = J(nrow, 1, .);
  err_std = J(nrow, 1, .);
  n_size = J(nrow, 1, .);
  method = J(nrow, 1, 'Q');

* Calculate theoretic error rate and the sample_plug_in error rate
  for given n_max;
  irow=1;
  do p = 1 to s_max;
    do p_plus = 0 to int(p/2);
      p_minus = p-p_plus;
      svec[irow]=p;
      splus[irow]=p_plus;
      run muV(mu1,mu2,V1c,V1f,V2c,V2f,V1chalf,V1fhalf,V2chalf,V2fhalf,
              p_plus,p_minus,m,sigma1c,sigma1f,sigma2c,sigma2f,rho1c,
              rho1f,rho2c,rho2f);
      run Qerr_th(pmis,pi1,pi2,mu1,mu2,V1c,V1f,V1chalf,V1fhalf,V2c,V2f,
                  V2chalf,V2fhalf,seed,nrep_th);
      err_th[irow] = pmis;
      run Qe_rsim(pmis,std,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,
                  V2fhalf,p,n_max,nvalid1,nvalid2,nrep_sim);
      err_simu[irow] = pmis;
      err_std[irow] = std;
      irow=irow+1;
    end;
  end;

* Calculate n for targeted error rate err;
* Note: This calculation is by-passed if err cannot be attained for n_max;
  irow=1;
  do p = 1 to s_max;
    do p_plus = 0 to int(p/2);
      p_minus = p-p_plus;
      run muV(mu1,mu2,V1c,V1f,V2c,V2h,V1chalf,V1fhalf,V2chalf,V2fhalf,
              p_plus,p_minus,m,sigma1c,sigma1f,sigma2c,sigma2f,rho1c,
              rho1f,rho2c,rho2f);
      if (err_simu[irow] < err) then do;
        run Qup_down(n,pmis,pi1,pi2,mu1,mu2,V1chalf,V1fhalf,V2chalf,
                     V2fhalf,nvalid,nsim,err);
        run summary(nstar, mean, weight, n, pmis);
        run p_adj_v(n0, mean_hat, nstar, mean, weight, err);
        n_size[irow] = n0;
      end;
      irow=irow+1;
    end;
  end;

  edit mylib.result var{svec splus n_size err_th err_simu err_std method};
  append;
  close mylib.result;
end;


* K-NEAREST-NEIGHBOR RULE;
if (nearest = 1) then do;
  svec = J(nrow, 1, .);
  splus = J(nrow, 1, .);
  err_th = J(nrow, 1, .);
  err_simu = J(nrow, 1, .);
  err_std = J(nrow, 1, .);
  n_size = J(nrow, 1, .);
  method = J(nrow, 1, 'N');

* Calculate theoretic error rate and the sample_plug_in error rate
  for given n_max;
  irow=1;
  do p = 1 to s_max;
    do p_plus = 0 to int(p/2);
      p_minus = p-p_plus;
      svec[irow]=p;
      splus[irow]=p_plus;
      run muV(mu1,mu2,V1c,V1f,V2c,V2f,V1chalf,V1fhalf,V2chalf,V2fhalf,
              p_plus,p_minus,m,sigma1c,sigma1f,sigma2c,sigma2f,rho1c,
              rho1f,rho2c,rho2f);
      run KNe_rsim(pmis,std,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,
                   V2fhalf,p,n_max,nvalid1,nvalid2,k,nrep_sim);
      err_simu[irow] = pmis;
      err_std[irow] = std;
      irow=irow+1;
    end;
  end;

* Calculate n for targeted error rate err;
* Note: This calculation is by-passed if err cannot be attained for n_max;
  irow=1;
  do p = 1 to s_max;
    do p_plus = 0 to int(p/2);
      p_minus = p-p_plus;
      run muV(mu1,mu2,V1c,V1f,V2c,V2f,V1chalf,V1fhalf,V2chalf,V2fhalf,
              p_plus,p_minus,m,sigma1c,sigma1f,sigma2c,sigma2f,rho1c,
              rho1f,rho2c,rho2f);
      if (err_simu[irow] < err) then do;
        run Nup_down(n,pmis,pi1,pi2,mu1,mu2,V1chalf,V1fhalf,V2chalf,
                     V2fhalf,nvalid,nsim,n_max,err,k);
        run summary(nstar, mean, weight, n, pmis);
        run p_adj_v(n0, mean_hat, nstar, mean, weight, err);
        n_size[irow] = n0;
      end;
      irow=irow+1;
    end;
  end;

  edit mylib.result var{svec splus n_size err_th err_simu err_std method};
  append;
  close mylib.result;
end;
```

```
quit;

proc print data = mylib.result;
run;

proc format;
  value $method L = 'linear'
                Q = 'quadratic'
                N = 'nearest neighbors';

proc tabulate data=mylib.result(where = (method NE 'N'));
  class svec splus method;
  var err_th;
  format method $method.;
  table  method,
         svec,
         err_th = ' '* SUM =' '*splus*F = 4.3 /
         condense RTS = 25;
  label  svec = 'S'
         splus = 'S+';
  title "THEORETICAL (KNOWN-PARAMETER) ERROR RATE ";
run;

proc tabulate data=mylib.result  missing;
  class svec splus method;
  var err_simu;
  format method $method.;
  table  method,
         svec,
         err_simu = ' '* SUM =' '*splus*F = 4.3 /
         condense RTS = 20;
  label  svec = 'S'
         splus = 'S+';
  title "EMPIRICAL (ESTIMATED-PARAMETER) ERROR RATE FOR N = &n_max ";
run;

proc tabulate data=mylib.result  missing;
  class svec splus method;
  var n_size;
  format method $method.;
  table  method,
         svec,
         n_size = ' '* SUM =' '*splus*F = 4.0 /
         condense RTS = 20;
  label  svec = 'S'
         splus = 'S+';
  title "CALIBRATION SAMPLE SIZE REQUIRED FOR ERROR RATE < &err ";
run;
```

modules.sas

```
*file name: "modules.sas";


/*---------------------------------------------------------------------#
 Modules to support discriminant analysis calculations:
    Two classes are N(mu1, V1), N(mu2, V2).
    Probabilities of the classes, pi1, pi2, are known.
    The training samples of the two classes have equal size n.
    The validation sample size is 'nvalid'.
#--------------------------------------------------------------------------*/



START myhalf(X, sigma, rho, p);
/*-------------------------------------------------------------------------#
 For V=sigma*sigma*[(1-rho)*Ip+rho*1*1'], find X such that V=X*X.
 NOTE: rho>-1/(p-1) must hold such that V is positive definite.
#--------------------------------------------------------------------------*/
    alpha=sqrt(1-rho);
    beta=(-alpha+sqrt(alpha*alpha+p*rho))/p;
    X=sigma*(alpha*I(p)+beta*J(p,p,1));
FINISH;



START  muV(mu1,mu2,V1c,V1f,V2c,V2f,V1chalf,V1fhalf,V2chalf,V2fhalf,p_plus,
           p_minus,m,sigma1c,sigma1f,sigma2c,sigma2f,rho1c,rho1f,rho2c,
           rho2f);
/*-------------------------------------------------------------------------#
 INPUT: p_plus, p_minus, m, sigma1, sigma2, rho1, and rho2,
 OUTPUT: mu1, mu2, V1, V2 V1half, V2half.

          V1=sigma1*sigma1*[(1-rho1)*Ip+rho1*1*1'];
          V2=sigma2*sigma2*[(1-rho2)*Ip+rho2*1*1'];
          V1half**2=V1;
          V2half**2=V2;
          mu1=J(p,1,0);
          mu2={m,...,m, -m, ..., -m};
               -------  -----------
               p_plus    p_minus
               terms      terms
#--------------------------------------------------------------------------*/
  p = p_plus+p_minus;
  mu1 = J(p, 1, 0);
  if (p_plus=0) then mu2=J(p_minus,1,-m);
  else if (p_minus=0) then mu2=J(p_plus,1,m);
  else mu2=J(p_plus,1,m)//J(p_minus,1,-m);
  V1c = sigma1c**2*((1-rho1c)*I(p)+rho1c*J(p,p,1));
  V1f = sigma1f**2*((1-rho1f)*I(p)+rho1f*J(p,p,1));
  V2c = sigma2c**2*((1-rho2c)*I(p)+rho2c*J(p,p,1));
  V2f = sigma2f**2*((1-rho2f)*I(p)+rho2f*J(p,p,1));
  run myhalf(V1chalf, sigma1c, rho1c, p);
  run myhalf(V1fhalf, sigma1f, rho1f, p);
  run myhalf(V2chalf, sigma2c, rho2c, p);
  run myhalf(V2fhalf, sigma2f, rho2f, p);
FINISH;



START smv1(xbar, S, n, p);
/*-------------------------------------------------------------------------#
   Given a sample of size n from N(0, I(pxp)), calculate the sample mean
   vector xbar  and the sample covariance matrix S.
   *INPUT: n, p.
   *OUTPUT: xbar, S.
#--------------------------------------------------------------------------*/
   temp=J(p, 1, 0);
   mtemp=J(p, 1, 0);
   Stemp=J(p, p, 0);
   do i=1 to n;
```

```
      x=normal(temp);
      mtemp=mtemp+x;
      Stemp=Stemp+x*x`;
    end;
    xbar=mtemp/n;
    S=(Stemp-n*xbar*xbar`)/(n-1);
FINISH;



START smv2(xbar, S, n, mu, Vhalf);
/*-----------------------------------------------------------------------#
   Given a sample of size n from N(mu, V), where V=Vhalf*Vhalf, calculate
   the sample mean vector xbar and the sample covariance matrix S
   (assuming we don't know the structure of V).
   INPUT: n, mu, Vhalf.
   OUTPUT: xbar, S.
#-----------------------------------------------------------------------*/
   p=nrow(mu);
   run smv1(xbar0, S0, n, p);
   xbar=mu+Vhalf*xbar0;
   S=Vhalf*S0*Vhalf;
FINISH;



START Lrule(b, k, pi1, pi2, mu1, mu2, V1, V2);
/*-----------------------------------------------------------------------#
  Given two classes: N(mu1, V1)---class 1,  N(mu2, V2)--class 2,
    and the  priors:  pi1, pi2,
  find the LINEAR rule: if b'x<=k then class 1.
  *INPUT: pi1, pi2, mu1, mu2, V1, V2;
  *OUTPUT: b, k.
#-----------------------------------------------------------------------*/
  V = 0.5#V1+0.5#V2; *BECAUSE OF EQUAL TRAINING SAMPLE SIZE for 2 CLASSES;
  Vinv = inv(V);
  b1 = Vinv*mu1;
  b2 = Vinv*mu2;
  b = b2-b1;
  k = 0.5*sum(mu2#b2-mu1#b1)+log(pi1/pi2);
FINISH;



START Qrule(A, b, k, pi1, pi2, mu1, mu2, V1, V2);
/*-----------------------------------------------------------------------#
  Given two classes: N(mu1, V1)---class 1,  N(mu2, V2)--class 2,
    and the  priors:  pi1, pi2,
  find the QUADRATIC rule: if x'Ax+b'x<=k then class 1.
  *INPUT: pi1, pi2, mu1, mu2, V1, V2;
  *OUTPUT: A, b, k.
#-----------------------------------------------------------------------*/
  Vinv1 = inv(V1);
  Vinv2 = inv(V2);
  b1 = Vinv1*mu1;
  b2 = Vinv2*mu2;
  A = Vinv1-Vinv2;
  b = -2*(b1-b2);
  k = 2*log(pi1/pi2)-log(det(V1)/det(V2))+sum(b2#mu2-b1#mu1);
FINISH;
```

```
START Lerr_th(pmis,pi1,pi2,mu1,mu2,V1c,V1f,V2c,V2f);
/*-------------------------------------------------------------------------#
  Given two classes: N(mu1, V1)---class 1,  N(mu2, V2)--class 2,
    and the  priors:      pi1, pi2,
  find the THEORETIC misclassification probability of LINEAR rule: "pmis".
  *INPUT: pi1, pi2, mu1, mu2, V1, V2;
  *OUTPUT: pmis.
#-------------------------------------------------------------------------*/
  run Lrule(b, k, pi1, pi2, mu1, mu2, V1c, V2c);
  d1 = sqrt(sum(b#(V1f*b)));
  d2 = sqrt(sum(b#(V2f*b)));
  pmis = pi1*(1-PROBNORM((k-sum(b#mu1))/d1))+
         pi2*PROBNORM((k-sum(b#mu2))/d2);
FINISH;



START Qerr_th(pmis,pi1,pi2,mu1,mu2,V1c,V1f,V1chalf,V1fhalf,V2c,V2f,
              V2chalf,V2fhalf,seed,nrep);
/*-------------------------------------------------------------------------#
  Given two classes: N(mu1, V1)---class 1,  N(mu2, V2)--class 2,
    and the  priors:      pi1, pi2,
  find the THEORETIC misclassification probability of QUADRATIC rule:
  "pmis".

  METHOD:  We generate a random sample of size "nrep" from each class.
  Then calculate the error rates "pmis1" and "pmis2". The error rate
  is pi1*pmis1+pi2*pmis2.

  *INPUT: pi1, pi2, mu1, mu2, V1, V2;
          seed -- random seed;
          nrep -- the number of simulations;
  *OUTPUT: pmis.
#-------------------------------------------------------------------------*/
  run Qrule(A, b, k, pi1, pi2, mu1, mu2, V1c, V2c);
  p = nrow(mu1);
  count = 0;
  do i = 1 to nrep;
    x = mu1+V1fhalf*normal(J(p, 1, seed));
    if sum(x#(A*x))+sum(b#x) >=  k then count = count+1;
  end;
  pmis1 = count/nrep;
  count = 0;
  do i = 1 to nrep;
    x = mu2+V2fhalf*normal(J(p, 1, seed));
    if sum(x#(A*x))+sum(b#x) < k then count = count+1;
  end;
  pmis2 = count/nrep;
  pmis = pi1*pmis1+pi2*pmis2;
FINISH;



START Lerr_sim(pmis,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,V2fhalf,p,n,
               nvalid1,nvalid2);
/*-------------------------------------------------------------------------#
  Given two classes: N(mu1, V1)---class 1,  N(mu2, V2)--class 2,
  and the  priors:      pi1, pi2,
  find LINEAR rule's real misclassification probability "pmis" by
  simulation.

  *INPUT: pi1 and pi2 are priors,
          mu1, mu2, V1half, V2half,
          p -- the number of sensors;
          n -- training sample size for each class;
          nvalid1/nvailid2 -- validation sample size for class 1/2;
  *OUTPUT: pmis.
#-----------------------------------------------------------------------*/
     *Calculate sample mean vectors mu1_h and mu2_h, and sample
      covariance matrices S1 and S2;
     DO until(det(0.5*S1+0.5*S2)^=0);
       RUN smv2(mu1_h, S1, n, mu1, V1chalf);
       RUN smv2(mu2_h, S2, n, mu2, V2chalf);
     END;
     *generate Validation sample;
     Xvalid1 = normal(J(nvalid1, p, 0))*V1fhalf + J(nvalid1,1,1)*mu1`;
     Xvalid2 = normal(J(nvalid2, p, 0))*V2fhalf + J(nvalid2,1,1)*mu2`;
     RUN  Lrule(b, k, pi1, pi2, mu1_h, mu2_h, S1, S2);
     nmis = sum(Xvalid1*b > k)+sum(Xvalid2*b <=  k);
     pmis = nmis/(nvalid1+nvalid2);
FINISH;



START Qerr_sim(pmis,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,V2fhalf,
               p,n,nvalid1,nvalid2);
/*-----------------------------------------------------------------------#
  Given two classes: N(mu1, V1)---class 1,  N(mu2, V2)--class 2,
    and the  priors:      pi1, pi2,
  find Quadratic rule's real misclassification probability "pmis" by
  simulation.

  *INPUT: pi1 and pi2 are priors,
          mu1, mu2, V1half, V2half,
          p -- the number of sensors;
          n -- training sample size for each class;
          nvalid1/nvailid2 -- validation sample size for class 1/2;
  *OUTPUT: pmis.
#-----------------------------------------------------------------------*/
     *Calculate sample mean vectors mu1_h and mu2_h, and sample covariance
      matrices S1 and S2;
     DO until(det(S1)*det(S2)^=0);
       RUN smv2(mu1_h, S1, n, mu1, V1chalf);
       RUN smv2(mu2_h, S2, n, mu2, V2chalf);
     END;
     *generate Validation sample;
     Xvalid1 = normal(J(nvalid1, p, 0))*V1fhalf + J(nvalid1,1,1)*mu1`;
     Xvalid2 = normal(J(nvalid2, p, 0))*V2fhalf + J(nvalid2,1,1)*mu2`;
     RUN  Qrule(A, b, k, pi1, pi2, mu1_h, mu2_h, S1, S2);
     nmis = sum(((Xvalid1*A)#Xvalid1)[,+]+Xvalid1*b > k)+
            sum(((Xvalid2*A)#Xvalid2)[,+]+Xvalid2*b <= k);
     pmis = nmis/(nvalid1+nvalid2);
FINISH;



START kmin(i_out, x, n, k);
/*-----------------------------------------------------------------------#
  Given a (n by 1) vector x, find the indices(positions) of its k
  smallest elements
#-----------------------------------------------------------------------*/
   i_in = 1: n;
   i_out = J(1,k,0);
   do i = 1 to k;
     temp = x[>:<];
     i_out[i] = i_in[temp];
     x = remove(x, temp);
     i_in = remove(i_in, temp);
   end;
FINISH;



START KNearest(pmis,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,V2fhalf,p,n,
               nvalid1,nvalid2,k);
/*---------------------------------------------------------------------#
  Given two classes: N(mu1, V1)---class 1,  N(mu2, V2)--class 2,
    and the  priors:      pi1, pi2,
  find K-NEAREST NEIGHBORS rule's misclassification probability "pmis" by
    simulation.

  *INPUT: k -- the number of neighbors considered,
          pi1 and pi2 are priors,
          mu1, mu2, V1half, V2half,
          p -- the number of sensors;
          n -- training sample size for each class;
          nvalid1/nvailid2 -- validation sample size for class 1/2;
  *OUTPUT: pmis.
#---------------------------------------------------------------------*/
     *generate training and Validation samples;
     nvalid=nvalid1+nvalid2;

     X1c = normal(J(n,p,0))*V1chalf+J(n,1,1)*mu1`;
     X2c = normal(J(n,p,0))*V2chalf+J(n,1,1)*mu2`;
     X1f = normal(J(nvalid1,p,0))*V1fhalf+J(nvalid1,1,1)*mu1`;
     X2f = normal(J(nvalid2,p,0))*V2fhalf+J(nvalid2,1,1)*mu2`;
     Xtrain=X1c//X2c;
     Xvalid=X1f//X2f;
     Ipred = J(nvalid, 1, 0);
     do i=1 to nvalid;
        dtemp = (Xtrain-J(2*n,1,1)*Xvalid[i,])##2;
        dsqr = (dtemp[, +])`;
        run kmin(temp, dsqr, 2*n, k);
        Ipred[i] = (sum(temp <= n)/k >= 0.5);
     end;
     pmis = pi1*(1-sum(Ipred[1:nvalid1])/nvalid1)+
            pi2*sum(Ipred[(1+nvalid1):nvalid])/nvalid2;
FINISH;



START Le_rsim(pmis,std,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,V2fhalf,
              p,n,nvalid1,nvalid2,nrep);
/*---------------------------------------------------------------------#
 Use a number of simulations to estimate LINEAR rule's misclassification
 error probability.
  *INPUT: pi1 and pi2 are priors,
          mu1, mu2, V1half, V2half,
          p -- the number of sensors;
          n -- training sample size for each class;
          nvalid1/nvailid2 -- validation sample size for class 1/2;
          nrep -- the number of simulations.
  *OUTPUT: pmis -- the estimate of the misclassification probability;
           std -- the standard error of the above estimator.
#---------------------------------------------------------------------*/
    pmis_vec=J(nrep,1,0);
    do i=1 to nrep;
      run Lerr_sim(pmis_t,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,V2fhalf,
                   p,n,nvalid1,nvalid2);
      pmis_vec[i]=pmis_t;
    end;
    pmis=pmis_vec[+]/nrep;
    v=(sum(pmis_vec#pmis_vec)-nrep*pmis**2)/(nrep-1);
    std=sqrt(v);
FINISH;



START Qe_rsim(pmis,std,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,V2fhalf,
             p,n,nvalid1,nvalid2,nrep);
/*---------------------------------------------------------------------------#
 Use a number of simulations to estimate QUADRATIC rule's misclassification
 error probability.
  *INPUT: pi1 and pi2 are priors,
          mu1, mu2, V1half, V2half,
          p -- the number of sensors;
          n -- training sample size for each class;
          nvalid1/nvailid2 -- validation sample size for class 1/2;
          nrep -- the number of simulations.
  *OUTPUT: pmis -- the estimate of the misclassification probability;
           std -- the standard error of the above estimator.
#---------------------------------------------------------------------------*/
    pmis_vec=J(nrep,1,0);
    do i=1 to nrep;
      run Qerr_sim(pmis_t,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,V2fhalf,
                   p,n,nvalid1,nvalid2);
      pmis_vec[i]=pmis_t;
    end;
    pmis=pmis_vec[+]/nrep;
    v=(sum(pmis_vec#pmis_vec)-nrep*pmis**2)/(nrep-1);
    std=sqrt(v);
FINISH;



START KNe_rsim(pmis,std,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,V2fhalf,
              p,n,nvalid1,nvalid2,k,nrep);
/*---------------------------------------------------------------------------#
 Use a number of simulations to estimate K-NN rule's misclassification
 error probability.
  *INPUT: k--the number of neighbors considered,
          pi1 and pi2 are priors,
          mu1, mu2, V1half, V2half,
          p -- the number of sensors;
          n -- training sample size for each class;
          nvalid1/nvailid2 -- validation sample size for class 1/2;
          nrep -- the number of simulations.
  *OUTPUT: pmis -- the estimate of the misclassification probability;
           std -- the standard error of the above estimator.
#---------------------------------------------------------------------------*/
    pmis_vec=J(nrep,1,0);
    do i=1 to nrep;
      run KNearest(pmis_t,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,V2fhalf,
                   p,n,nvalid1,nvalid2,k);
      pmis_vec[i]=pmis_t;
    end;
    pmis=pmis_vec[+]/nrep;
    v=(sum(pmis_vec#pmis_vec)-nrep*pmis**2)/(nrep-1);
    std=sqrt(v);
FINISH;
```

```
START Lup_down(n,pmis,pi1,pi2,mu1,mu2,V1chalf,V1fhalf,V2chalf,V2fhalf,
               nvalid,nsim,err);
/*-----------------------------------------------------------------------#
 For LINEAR rule, suppose err is the targeted error rate, which is
 achievable theoretically.
 We do a random walk ("up and down" search) to figure out the needed
 training sample size.

 *INPUT: pi1, pi2 -- priors;
         mu1/mu2 -- mean vector of class 1/2;
         V1half,V2half -- square root of the cov matrix of class 1/2;
         nsim -- the number of times of "up and down" search,
         nvalid -- validation sample size,
         err -- targeted error rate,
 *OUTPUT:
         n -- nsim-dimensional vector, each element representing a
              training sample size;
         pmis -- the error rate from validation sample.
#-----------------------------------------------------------------------*/
  pmis=J(nsim, 1, 0);
  n=J(nsim, 1, 0);
  p = nrow(mu1);
  nvalid1=pi1*nvalid;
  nvalid2=nvalid-nvalid1;
  ntemp=int(p/2)+1;
  do until(pmis_t < err);
      ntemp = ntemp+20;
      run Lerr_sim(pmis_t,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,V2fhalf,
                   p,ntemp,nvalid1,nvalid2);
  end;
  pmis[1]=pmis_t;
  n[1]=ntemp;
  ntemp=ntemp-1;
  do i=2 to nsim;
       n[i]=ntemp;
       run Lerr_sim(pmis_t,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,V2fhalf,
                    p,ntemp,nvalid1,nvalid2);
       pmis[i]=pmis_t;
       if pmis_t<err then ntemp=max(ntemp-1,int(p/2)+1,2);
       else               ntemp=ntemp+1;
  end;
FINISH;


START Qup_down(n,pmis,pi1,pi2,mu1,mu2,V1chalf,V1fhalf,V2chalf,V2fhalf,
               nvalid,nsim,err);
/*-----------------------------------------------------------------------#
 For QUADRATIC rule, suppose err is the targeted error rate,
 which is achievable theoretically.
 We do a random walk ("up and down" search) to determine the needed
 training sample size.

 *INPUT: pi1, pi2 -- priors;
         mu1/mu2 -- mean vector of class 1/2;
         V1half,V2half -- square root of the cov matrix of class 1/2;
         nsim -- the number of times of "up and down" search,
         nvalid -- validation sample size,
         err -- targeted error rate,
 *OUTPUT:
         n -- nsim-dimensional vector, each element representing a
              training sample size;
         pmis -- the error rate from validation sample.
#------------------------------------------------------------------------*/
  pmis=J(nsim, 1, 0);
  n=J(nsim, 1, 0);
  p = nrow(mu1);
  nvalid1=pi1*nvalid;
  nvalid2=nvalid-nvalid1;
  ntemp=int(p/2)+1;
  do until(pmis_t < err);
      ntemp = ntemp+20;
      run Qerr_sim(pmis_t,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,V2fhalf,
                  p,ntemp,nvalid1,nvalid2);
  end;
  pmis[1]=pmis_t;
  n[1]=ntemp;
  ntemp=ntemp-1;
  do i=2 to nsim;
       n[i]=ntemp;
       run Qerr_sim(pmis_t,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,V2fhalf,
                  p,ntemp,nvalid1,nvalid2);
       pmis[i]=pmis_t;
       if pmis_t<err then ntemp=max(ntemp-1,p+1);
       else               ntemp=ntemp+1;
  end;
FINISH;



START Nup_down(n,pmis,pi1,pi2,mu1,mu2,V1chalf,V1fhalf,V2chalf,V2fhalf,
              nvalid,nsim,n_max,err,k);
/*-------------------------------------------------------------------------#
 For k-NN rule, suppose err is the targeted error rate, which is
 achievable as n=nmax.
 We do a random walk ("up and down" search) to determine the needed
 training sample size.

 *INPUT: pi1, pi2 -- priors;
         mu1/mu2 -- mean vector of class 1/2;
         V1half,V2half -- square root of the cov matrix of class 1/2;
         nsim -- the number of times of "up and down" search,
         nvalid -- validation sample size,
         n_max -- the maximum training sample size that is practical,
         err -- targeted error rate,
         k -- the number of neighbors considered.
 *OUTPUT:
         n -- nsim-dimensional vector, each element representing a
              training sample size;
         pmis -- the error rate from validation sample.
#------------------------------------------------------------------------*/
  pmis=J(nsim, 1, 0);
  n=J(nsim, 1, 0);
  p = nrow(mu1);
  nvalid1=pi1*nvalid;
  nvalid2=nvalid-nvalid1;
  ntemp=2*k;
  do until(pmis_t < err);
      ntemp = ntemp+5;
      run KNearest(pmis_t,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,V2fhalf,
                  p,ntemp,nvalid1,nvalid2,k);
  end;
  pmis[1]=pmis_t;
  n[1]=ntemp;
  ntemp=ntemp-1;
  do i=2 to nsim;
       n[i]=ntemp;
       run KNearest(pmis_t,pi1,pi2,mu1,V1chalf,V1fhalf,mu2,V2chalf,V2fhalf,
                    p,ntemp,nvalid1,nvalid2,k);
       pmis[i]=pmis_t;
       if pmis_t<err then ntemp=max(ntemp-1, k+1);
       else               ntemp=min(ntemp+1, n_max+5);
  end;
FINISH;



START summary(nstar, mean, weight, n, pmis);
/*---------------------------------------------------------------------#
 INPUT:    n -- vector, training sample size;
        pmis -- vector, error rate;
 OUTPUT:
        nstar -- vector with distinct elements, containing all possible
                 values in the input vector "n";
         mean -- the mean of pmis corresponding to each possible value
                 in input vector "n";
       weight -- the frequency of each possible value  in input vector "n";

#---------------------------------------------------------------------*/
     nstar = n;
     pmisstar = pmis;
     create mydata var{nstar pmisstar}; append; close mydata;
     use mydata;
     summary var {pmisstar} class {nstar} stat{mean N} opt{noprint save};
     call delete(work, mydata);
     mean = pmisstar[,1];
     weight =pmisstar[,2];
FINISH;



START p_adj_v(x0, y_hat, x, y, weight, y0);
/*---------------------------------------------------------------------#
"pool_adjacent_violators" algorithm

*INPUT: x, value, weight, value0;
*OUTPUT: x0, v_hat;
#---------------------------------------------------------------------*/
* data structure: linked list;
n=nrow(y);
id=do(1, n, 1)`;
last=do(1, n, 1)`;
prev=do(0, n-1, 1)`;
W=weight;
yy=y;
Ivec=J(n,1,0);
L=J(n,1,0);
V=J(n,1,0);
y_hat=J(n,1,0);
ptr=1;
stop_y=0;
do until(stop_y=1);
  if (ptr=1) then
     do;
       if last[ptr]=n then do; stop_y=1; goto jump; end;
       else ptr=last[ptr]+1;
     end;
  if (yy[ptr]< yy[prev[ptr]]) then
```

```
      do;
        if last[ptr]=n then stop_y=1;
        else ptr=last[ptr]+1;
      end;
   else
      do;
        ptr_new=prev[ptr];
        wgt=W[ptr_new]+W[ptr];
        val=(W[ptr_new]*yy[ptr_new]+W[ptr]*yy[ptr])/wgt;
        last[ptr_new]=last[ptr];
        W[ptr_new]=wgt;
        yy[ptr_new]=val;
        if (last[ptr]^=n) then prev[last[ptr]+1]=id[ptr_new];
        ptr=ptr_new;
      end;
     jump:
         *********;
end;
  ii = 1;
  pt = 1;
  stop2_y = 0;
  do until(stop2_y = 1);
     Ivec[ii]=pt;
     L[ii]=last[pt];
     V[ii]=yy[pt];
     if (last[pt]=n) then do; stop2_y = 1; goto jump2; end;
     pt= L[ii]+1;
     ii=ii+1;
     jump2: ******;
  end;
  Vmin=min(V[1:ii]);
  if (y0<Vmin) then x0=.;
                      *************;
  else
       do;
         ii0=min(loc(V[1:ii]<=y0));
         ind0=Ivec[ii0];
         x0=x[ind0];
       end;
  k = 0;
  do i = 1 to ii;
    do j = Ivec[i] to L[i];
       k = k+1;
       y_hat[k] = V[i];
    end;
  end;
FINISH;
```

ORIGINAL DISTRIBUTION

Dr. David Baldwin
Environmental and Protection Sciences Program
Ames Laboratory
9 Spedding Hall
Ames, IA 50011

Dr. Vivian Baylor
Manager, Nonproliferation R&D
National Security Program Office
1099 Commerce Park, Rm. E-11, MS-8260
Oak Ridge, TN 37830

Dr. Richard Cernosek
Microsensor R&D Dept.
Sandia National Laboratories
P.O. Box 5800, M/S 1425
Albuquerque, NM 87185-1425

Dr. Panos Datskos
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, TN 37831-8039

Dr. Terry Donaldson
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, TN 37831-6044

Dr. Gordon Dudder
Pacific Northwest National Laboratory
PO Box 999, MS K9-55
Richland, WA 99352

Ms. Alison Easter
Ames Laboratory
125 Spedding Hall
Ames, IA 50011

Prof. Sherwood Ebey
Department of Mathematics and Computer Science
The University of the South
735 University Avenue
Sewanee, Tennessee 37383-1000

Dr. Bill Flounders
Biosystems Research Department
Sandia National Laboratories
PO Box 969, MS9951
Livermore, CA 94551-0969

Dr. Carl Friesen
Idaho National Engineering and Environmental Laboratory
P.O. Box 1625, Mail Stop 1222
Idaho Falls, ID 83415

Dr. Barbara Hoffheins
Department of Energy NN-20
Forrestal Building
1000 Independence Avenue S.W.
Washington, DC 20585-0420

Dr. Bob Hughes
P.O. Box 5800, Mail Stop 1425
Sandia National Laboratories
Albuquerque, NM 87185-1425

Prof. Karen Kafadar
Department of Mathematics
University of Colorado at Denver
P.O. Box 173364, Campus Box 170
Denver, CO 80217-3364

Dr. Sallie Keller-McNulty
Los Alamos National Laboratory
Mail Stop F600
Los Alamos, NM 87545

Dr. Fred Milanovich
Lawrence Livermore National Laboratory
P.O. Box 808, MS L-524
Livermore, CA 94550

Dr. Michael O'Connell
Department of Energy NN-20
Forrestal Building
1000 Independence Avenue S.W.
Washington, DC 20585-0420

Prof. Michael Pishko
Department of Chemical Engineering
Texas A&M University
3122 TAMU
College Station, TX 77843-3122

Dr. Richard Pollina
Nevada Test Site
P.O. Box 98521
Las Vegas, NV 89193-8521

Dr. Steve Schubert
Department of Energy NN-20
Forrestal Building
1000 Independence Avenue S.W.
Washington, DC 20585-0420

Dr. Robert Waldron
Department of Energy NN-20
Forrestal Building
1000 Independence Avenue S.W.
Washington, DC 20585-0420

Prof. Denise Wilson
Department of Electrical Engineering
M222 EE/CSE Bldg.
Box 352500
University of Washington
Seattle, WA 98195

Dr. Brian Worley
Oak Ridge National Laboratory
Building 6012
P.O. Box 2008
Oak Ridge, TN 37831

Dr. Robert Young
946 Torrey Pine Drive
Winter Springs, FL 32708-4346