Throughout most of the history of pattern recognition, the number which the designed classifier possesses the lowest estimated error have been highlighted: (i) applying a classification rule to numerous et al. the difficulty in establishing performance advantages for proposed selection (Molinaro et al., 2005b) and lack of correlation with the true error (Hanczar et al., 2010), which puts mass 1 on zero are the resubstitution and zero-bootstrap error estimators. The original bolstering paper (Braga-Neto and Dougherty, 2004a) proposed a non-parametric estimator for the kernel variance, which was found empirically to perform well in low-dimensional spaces; (ii) applying multiple classification rules to a dataset and comparing the classification rules according to the estimated errors of the designed classifiers (Boulesteix and Strobl, 2009). In both cases, optimism is a result of inaccurate error estimation. A good error estimator ideally would have small bias and small variance. This is a difficult trade-off in small-sample settings. In small-sample cases, resubstitution generally has small variance but tends to be quite optimistically biased. Cross-validation has small bias, but tends to display high variance. Bolstered error estimation (Braga-Neto and Dougherty, 2004a) attempts to achieve a compromise to this bias-variance dilemma in such a way that it reduces bias, while at the same time reducing variance. Bolstered error estimation has shown good performance when compared with popular error estimators in small-sample settings, in particular, for feature-set ranking and when used internally within a feature-selection algorithm (Sima et al., 2005a) and for ranking feature sets (Sima et al., 2005b). Its good performance, including the latter applications, has been demonstrated in the context a small number of features, including feature selection via sequential forward selection (SFS), where it is applied to small potential feature sets in the SFS algorithm. A critical aspect of the method is selecting the right amount of bolstering, which is given by the variance of the bolstering kernels. The original bolstering paper (Braga-Neto and Dougherty, 2004a) showed that a non-parametric estimator for the kernel variance, which was found empirically to perform well in low-dimensional spaces; however, estimation was found to degrade in high dimensions, so that a correction factor can be required (Vu and Braga-Neto, 2008). In fact, it was demonstrated in a preliminary study that a correction factor can also be beneficial for low-dimensional bolstering (Huynh et al., 2007). This leads us to consider optimal bolstering, specifically, finding an optimal variance for the bolstering kernels. Error estimators like resubstitution and cross-validation (assuming the number of folds is preset) are non-parametric. They contain no free parameters. This is not the case for bootstrap. In general, bootstrap has the form of a convex error estimator, namely,

\[ \hat{\epsilon}_{\text{boot}} = (1-a)\hat{\epsilon}_{\text{resub}} + a\hat{\epsilon}_{\text{zero}}. \]  

(1)

where \( \hat{\epsilon}_{\text{resub}} \) and \( \hat{\epsilon}_{\text{zero}} \) are the resubstitution and zero-bootstrap estimators and \( 0 \leq a \leq 1 \). The zero-bootstrap utilizes the empirical distribution \( P^0 \), which puts mass \( \frac{1}{n} \) on each of the \( n \) available
data points. A bootstrap sample \( S'_n \) from \( F^n \) consists of \( n \) equally-likely draws with replacement from the original data \( S_n \). The basic bootstrap zero estimator (Efron, 1983) is written in terms of the empirical distribution as

\[
\hat{g}_0 = \mathbb{E}_{S_n} \left( Y - g(Y, X) \mid (X, Y) \in S_n \right). \tag{2}
\]

In practice, the expectation \( \mathbb{E}_{S_n} \) can be approximated by a Monte-Carlo estimate based on independent replicates \( S^n_{bh} \), for \( b = 1, \ldots, B \), in which case the classifier is designed on the bootstrap sample and tested on the original data points left out. An optimal bootstrap estimator results from a value of \( b \) that minimizes the mean-square error between \( \hat{g}_b \) and the true error for a given feature-label distribution (Simà and Dougherty, 2006b). Setting \( a = 0.632 \), as is commonly done (Efron, 1983), can lead to a far from optimal estimator (optimal weights).

The present article considers optimal bolstering relative to its one free parameter, kernel variance and the manner in which optimal bolstering can be used to arrive at practical implementation of bolstering in high-dimensional feature space. The end product is an implementation protocol in which optimal kernel variances across different models are combined to produce a suitable kernel variance for the problem at hand. Throughout, we will assume feature selection because that would be the standard way to approach classification in the high-dimensional setting we are considering, although this is not a mandatory requirement of the approach.

### 2 SYSTEMS AND METHODS

This section will be broken into subsections, with the aim of arriving at the implementation protocol for real-world data. Section 2.1 briefly reviews the necessary essentials of error estimation, mainly bolstering. Section 2.2 defines the scaling factor by which to adjust the bolstering kernel to high dimensions. Section 2.3 discusses optimization of the scaling factor and illustrates the construction of a set of optimal scaling factors across a family of models varying in both structure and classification difficulty. Section 2.4 provides the implementation of high-dimensional bolstered resubstitution based on a family of optimal scaling factors.

#### 2.1 Error estimation

In two-group statistical pattern recognition, there is a feature vector \( X \in \mathbb{R}^d \) and a label \( Y \in \{0, 1\} \). The pair \((X, Y)\) has a joint probability distribution \( F \), which is unknown in practice. Hence, a classifier is designed from training data, which is a set of \( n \) independent observations, \( S_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\} \), drawn from \( F \). A classification rule is a mapping \( \phi : \mathbb{R}^d \times \{0, 1\} \rightarrow \{0, 1\} \). The classification error \( \varepsilon_s \) is the probability of an erroneous classification:

\[
\varepsilon_s = P(\phi_s(X) \neq Y | (X, Y) \in S_n) = \mathbb{E}_{S_n} \left| Y - \psi(X) \right|. \tag{3}
\]

where \( \mathbb{E}_{S_n} \) denotes expectation with respect to \( F \). Were \( F \) known, then the error could be found via Equation (3). In practice, one must use an error estimator \( \hat{\varepsilon}_s \). An error estimator can suffer from bias, \( \text{Bias} = E[\hat{\varepsilon}_s] - \varepsilon_s \), and deviation variance, \( \text{Var}_{\hat{\varepsilon}_s} = \text{Var}[\hat{\varepsilon}_s] \). These combine to contribute to the most common measure (used herein) for evaluating the accuracy of an error estimator, the root-mean-square (RMS): 

\[
\text{RMS} = \sqrt{\mathbb{E}[\hat{\varepsilon}_s^2]} = \sqrt{\text{Var}_{\hat{\varepsilon}_s} + \text{Bias}^2}. \tag{4}
\]

#### 2.1.1 Classical error estimation

The simplest way to estimate the error in the absence of independent test data is to compute its error directly on the sample data itself. This resubstitution estimator, \( \hat{\varepsilon}_{\text{resub}} \), is usually optimistic (i.e. biased low), sometimes very much so.

In \( k \)-fold cross-validation, the dataset \( S_n \) is partitioned into \( k \) folds \( S^i_n \), for \( i = 1, \ldots, k \) (for simplicity, we assume that \( k \) divides \( n \) each fold is left out of the design process and used as a test set, and the estimate, \( \hat{\varepsilon}_i \), is the overall proportion of error on all folds. A \( k \)-fold cross-validation estimator is unbiased as an estimator of \( \varepsilon_0 \). Cross-validation estimators are pessimistic, since they use smaller training sets to design the classifier; however, their bias tends to be small. Their main drawback is their large variance (Braga-Neto and Dougherty, 2004b; Devroye et al., 1996). Sometimes cross-validation is repeated some number of times with different fold partitions and the results averaged. In this article, we use 10-fold cross-validation without repetition.

A recently developed estimation method, called adjusted bootstrap \( \hat{\varepsilon}_{\text{boot}} \), which carries out further bootstrap resampling in each fold, has been found to have good RMS performance (Jiang and Simon, 2007). Specifically, \( S_n \) is partitioned into \( n \) folds and, for each sample left out for testing, \( B \) bootstrap sample sets of size \( n_i \) are drawn from the remaining \( n - 1 \) points, \( i = 1, 2, \ldots, k \). For each \( l \), the error \( \varepsilon_l \) is the proportion of misclassified samples across \( n \) folds and \( B \) bootstrap sample sets. Finally, the adjusted bootstrap error \( \hat{\varepsilon}_{\text{boot}} \) is computed in the form 

\[
\hat{\varepsilon}_{\text{boot}} = \bar{\hat{\varepsilon}}_l + \hat{\varepsilon}_b,
\]

where \( \bar{\hat{\varepsilon}}_l \) and \( \hat{\varepsilon}_b \) are least squares estimates for the function 

\[
\varepsilon_l = a(n_i)\varepsilon_l + \hat{\varepsilon}_b,
\]

and \( n_i \) is the proportion of the expected number of non-repeated samples in a size \( n_i \) bootstrap sample set.

#### 2.1.2 Bolstered error estimation

The empirical feature-label distribution \( F^n \) is a discrete distribution that puts mass \( \frac{1}{n} \) on each of the \( n \) available data points. The resubstitution estimator can be written in terms of the empirical feature-label distribution as 

\[
\hat{\varepsilon}_{\text{resub}} = \mathbb{E}_{F^n} \left| Y - \psi(X) \right|. \tag{5}
\]

Relative to \( F^n \), no distinction is made between points near or far from the decision boundary. If one spreads the probability mass of the empirical distribution at each point, then variation is reduced because points near the decision boundary will have more mass on the other side of the boundary than will points far from the decision boundary. Consider a probability density function \( f^n_i \), for \( i = 1, \ldots, n \), called a bolstering kernel, and define the bolstered empirical distribution \( F^b \), with probability density function given by 

\[
f^b(X) = \frac{1}{n} \sum_{i=1}^{n} f^n_i(X - X_i). \tag{6}
\]

The bolstered resubstitution estimator (Braga-Neto and Dougherty, 2004a) is obtained by replacing \( F^n \) by \( F^b \) in Equation (5) to obtain 

\[
\hat{\varepsilon}_{\text{boot}} = \mathbb{E}_{F^b} \left| Y - \psi(X) \right|. \tag{7}
\]

Bolstering can be applied to other error estimators; however, we only use bolstered resubstitution, the bolstering method used the most to date. The bolstered resubstitution estimator is given by 

\[
\hat{\varepsilon}_{\text{boot}} = \frac{1}{n} \sum_{i=1}^{n} \left( \int_{-\infty}^{\infty} f^n_i(x - x_i)dx \right) + \int_{-\infty}^{\infty} \left( \sum_{i=1}^{n} A_i(x) f^n_i(x - x_i)dx \right), \tag{8}
\]

where \( A_i(x) = \{\psi(x) = i\} \). The integrals are the error contributions made by the data points, according to whether \( y_i = 0 \) or \( y_i = 1 \). If the classifier is linear, then the decision boundary is a hyperplane and it is usually possible to find...
The situation is akin to 0.632 bootstrap as opposed to optimal bootstrap. A zero-mean, spherical Gaussian bolstering kernel $f_i$ with covariance matrix of the form $\Sigma_i$, where $I$ is the identity matrix, has been proposed (Braga-Neto and Dougherty, 2004a), and has been shown to work well in low-dimensional feature spaces. Since bolstered estimators span the test points, the task is to find the amount of spreading that makes the test points to be as close as possible to the true mean distance to the training data points. The true mean distance can be estimated by its sample-based estimate:

$$d_i = \frac{\sum_{h=1}^n \min_{j \neq i} \|x_i - x_h\|}{\sum_{h=1}^n d_{ij}}$$  \hspace{1cm} (9)

The estimate $d_i$ is the mean minimum distance between points belonging to class $y_i$. Next, let $f_i$ be a unit-variance bolstering kernel, $R_i$ be the random variable equal to the distance of a point randomly selected from $f_i$ to the origin and $F_i(r)$ be the cumulative distribution function of $R_i$. In the case of the bolstering kernel $f_i$ with covariance matrix $\Sigma_i$, all distances get multiplied by $\kappa_i$. In Braga-Neto and Dougherty (2004a), a single variance $\kappa_i$ is estimated for all points from class $y_i$, such that the median distance of a test point to the origin is equal to the estimated true mean distance $d_i$. This implies that half of the mass (i.e. the test points) of the bolstering kernel will be farther from the center than $d_i$, and the other half will be nearer. Hence, $\kappa_i$ is the solution of the equation $\kappa_i F_i(d_i) = 0.5$. Letting $a_{ij} = F_i(d_i/2)$, and recognizing that the $R_i$ are identically distributed, the estimated SDs for the bolstering kernels are given by

$$\kappa_i = \frac{d_i}{a_{ij}}$$  \hspace{1cm} (10)

For $i = 1, 2, ..., n$.

### 2.2 High-dimensional bolstered resubstitution

In high-dimensional settings, its commonplace to perform feature selection and, when performed, feature selection is part of the classification rule, with the entire set of potential features constituting the feature set relative to the classification rule. Feature selection constrains the space of functions from which a classifier might be chosen, but it does not reduce the number of features in the design process. This is why when using cross-validation error estimation, feature selection has to be carried out in each partitioned fold. If we perform feature selection on a D-dimensional dataset $S^D$ and arrive at a D-dimensional set $S'_D$ (d < D), then the bolstered error estimator can use the previously defined kernel size $\kappa_i$, computed on $S'_D$, not $S^D$. Specifically, the mean minimum distance $d_i$ is estimated on $S'_D$ and $a_{ij} = a_{ij}'$. For high dimensions, we replace $\kappa_i$ by

$$\kappa_i' = k_i \times \frac{d_i'}{a_{ij}'}$$  \hspace{1cm} (11)

where $k_i$ is an additional scaling factor determined by the dimension and where we have indicated the dimension in the mean minimum distance estimate. The idea is to adjust the kernel size by choosing $k_i$ so the bolstered error estimator will be optimal (minimum RMS). $k_i = 1$ yields the previously proposed kernel variance. In essence, $\kappa_i$ is a parameter for the bolstered estimator and Equation (11) sets it free, thereby allowing for optimization. The situation is akin to 0.632 bootstrap as opposed to optimal bootstrap.

Given the kernel sizes, the bolstered resubstitution error estimate is given by Equation (8) in $D$ dimensions. For Gaussian kernels with independent variances, this integral reduces. Let $f_{i}(x - x_i)$ and $f'_{i}(x - x_i)$ denote the Gaussian kernels in d- and (D-d)-dimensional spaces, respectively, so that the D-dimensional Gaussian kernel decomposes as

$$f_D(x - x_i) = f_{D-d}(x - x_i) f'_{d}(x - x_i).$$  \hspace{1cm} (12)

Denoting $x - x_i = \Delta x$, then Equation (8) can be rewritten as

$$\hat{\epsilon}^2_{\text{bolst}} = \sum_{i=1}^n \frac{1}{k_i} \int f_{D-d}^{2}(\Delta x) \left( f'_{d}(\Delta x) dx + \int f_{D-d}^{2}(\Delta x) dx \right)$$

$$\hat{\epsilon}^2_{\text{bolst}} = \sum_{i=1}^n \frac{1}{k_i} \int f_{D-d}^{2}(\Delta x) \left( f'_{d}(\Delta x) dx + \int f_{D-d}^{2}(\Delta x) dx \right)$$

$$\hat{\epsilon}^2_{\text{bolst}} = \sum_{i=1}^n \frac{1}{k_i} \int f_{D-d}^{2}(\Delta x) \left( f'_{d}(\Delta x) dx + \int f_{D-d}^{2}(\Delta x) dx \right).$$

where $d_A^j = 0, 1$, is the projection of the classifier decision region $A_j$ into $d_A$-dimensional space, and we added a superscript ‘D’ to the bolstered error estimator to indicate it refers to the error in D-dimensional space. The previous result indicates that the integrals necessary to find the bolstered error estimate in D-dimensional space can be equivalently carried out in d-dimensional space. This is akin to resubstitution, where the error count is the same whether it is done in D- or d-dimensional space. For performance comparison purposes, we will also estimate the kernel size using only the low-dimensional data $S^D$ resulting in a bolstered error estimator $\hat{\epsilon}^2_{\text{bolst}}$, which uses the originally proposed kernel variance (no correction, or $k_i = 1$). For feature selection, we will use sequential forward floating search (SFFS) (Pudil et al., 1994).

### 2.3 Optimization method

To find the optimal kernel scaling factor $k_i$, we utilize the following procedure:

**Protocol 1**

1. Generate a sample set $S^D_n$ of size n and a total of D features from a specified synthetic model.
2. Select a size-d feature set A using a feature-selection method F on $S^D_n$, resulting in a reduced dimension sample set $S'_{D}$ for the feature set A.
3. Design a classifier $\psi_k$ for $S'_{D}$ according to the given classification rule $\psi_k$.
4. Compute the true error $\epsilon_k$ using the underlying distribution of the model.
5. Compute the 10-fold cross-validation error $\hat{\epsilon}_C$ (keeping in mind that feature selection must be repeated for each fold).
6. Compute the bolstered error $\hat{\epsilon}^2_{\text{bolst}}$.
7. Compute the bolstered errors $\hat{\epsilon}^2_{\text{bolst}}$ for a list of kernel scaling factors $k_1, k_2, ..., k_m$.
8. Calculate RMS for each error estimator by repeating Steps 1 through 7 a number N of times.
9. Repeat Steps 1 through 8 for different models $M$, different levels of model complexities and different classification rules $\psi_k$.

We consider four data models, each a two-class Gaussian model with equally likely classes and class-conditional densities having covariance matrices $\Sigma_1$ and $\Sigma_2$. One class mean is located at $\mu = \mathbf{0}$ and the other at $\mu = \mathbf{0}$, depending on the model. The parameter $\delta$ is chosen to achieve prescribed values for the expected classification error $E[\epsilon_k]$; different values of $E[\epsilon_k]$ represent different levels of difficulty at sample size n.

3058
Table 1. Summary of simulation experiments

<table>
<thead>
<tr>
<th>Data models</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification rules</td>
<td>$E_{{3}}$</td>
<td>0.05, 0.10, 0.15</td>
<td>LDA, kNN, SVM, NN, CART</td>
<td></td>
</tr>
</tbody>
</table>

**Feature-selection methods**
- M1: This is a selection of classifier that is used. The classifier is selected based on the type of data. The classifiers include LDA, kNN, SVM, NN, and CART. These classifiers are typically used in high-dimensional data analysis.

**Validation error estimate**
- M2: The validation error estimate is used to assess the performance of the classifier. The validation error estimate is calculated using a method such as cross-validation. The validation error estimate is used to compare the performance of different classifiers.

**K-fold cross-validation**
- M3: The K-fold cross-validation method is used to estimate the performance of the classifier. The K-fold cross-validation method is a method of estimating the performance of a classifier. The K-fold cross-validation method is used to calculate the validation error estimate.

**Robustness**
- M4: The robustness is used to assess the performance of the classifier. The robustness is calculated using a method such as the K-fold cross-validation method. The robustness is used to compare the performance of different classifiers.

**Number of total features**
- M5: The number of total features is used to assess the performance of the classifier. The number of total features is calculated using a method such as the K-fold cross-validation method. The number of total features is used to compare the performance of different classifiers.

**Classification rules**
- M6: The classification rules are used to assess the performance of the classifier. The classification rules are calculated using a method such as the K-fold cross-validation method. The classification rules are used to compare the performance of different classifiers.

**Kernel scaling factors**
- M7: The kernel scaling factors are used to assess the performance of the classifier. The kernel scaling factors are calculated using a method such as the K-fold cross-validation method. The kernel scaling factors are used to compare the performance of different classifiers.

**Number of sample size**
- M8: The number of sample size is used to assess the performance of the classifier. The number of sample size is calculated using a method such as the K-fold cross-validation method. The number of sample size is used to compare the performance of different classifiers.

**No. of total features**
- M9: The number of total features is used to assess the performance of the classifier. The number of total features is calculated using a method such as the K-fold cross-validation method. The number of total features is used to compare the performance of different classifiers.

**No. of total features**
- M10: The number of total features is used to assess the performance of the classifier. The number of total features is calculated using a method such as the K-fold cross-validation method. The number of total features is used to compare the performance of different classifiers.

**Kernel scaling factors**
- M11: The kernel scaling factors are used to assess the performance of the classifier. The kernel scaling factors are calculated using a method such as the K-fold cross-validation method. The kernel scaling factors are used to compare the performance of different classifiers.

**Number of sample size**
- M12: The number of sample size is used to assess the performance of the classifier. The number of sample size is calculated using a method such as the K-fold cross-validation method. The number of sample size is used to compare the performance of different classifiers.

**No. of total features**
- M13: The number of total features is used to assess the performance of the classifier. The number of total features is calculated using a method such as the K-fold cross-validation method. The number of total features is used to compare the performance of different classifiers.

**Kernel scaling factors**
- M14: The kernel scaling factors are used to assess the performance of the classifier. The kernel scaling factors are calculated using a method such as the K-fold cross-validation method. The kernel scaling factors are used to compare the performance of different classifiers.

**Number of sample size**
- M15: The number of sample size is used to assess the performance of the classifier. The number of sample size is calculated using a method such as the K-fold cross-validation method. The number of sample size is used to compare the performance of different classifiers.

**No. of total features**
- M16: The number of total features is used to assess the performance of the classifier. The number of total features is calculated using a method such as the K-fold cross-validation method. The number of total features is used to compare the performance of different classifiers.

**Kernel scaling factors**
- M17: The kernel scaling factors are used to assess the performance of the classifier. The kernel scaling factors are calculated using a method such as the K-fold cross-validation method. The kernel scaling factors are used to compare the performance of different classifiers.

**Number of sample size**
- M18: The number of sample size is used to assess the performance of the classifier. The number of sample size is calculated using a method such as the K-fold cross-validation method. The number of sample size is used to compare the performance of different classifiers.

**No. of total features**
- M19: The number of total features is used to assess the performance of the classifier. The number of total features is calculated using a method such as the K-fold cross-validation method. The number of total features is used to compare the performance of different classifiers.

**Kernel scaling factors**
- M20: The kernel scaling factors are used to assess the performance of the classifier. The kernel scaling factors are calculated using a method such as the K-fold cross-validation method. The kernel scaling factors are used to compare the performance of different classifiers.

**Number of sample size**
- M21: The number of sample size is used to assess the performance of the classifier. The number of sample size is calculated using a method such as the K-fold cross-validation method. The number of sample size is used to compare the performance of different classifiers.

**No. of total features**
- M22: The number of total features is used to assess the performance of the classifier. The number of total features is calculated using a method such as the K-fold cross-validation method. The number of total features is used to compare the performance of different classifiers.

**Kernel scaling factors**
- M23: The kernel scaling factors are used to assess the performance of the classifier. The kernel scaling factors are calculated using a method such as the K-fold cross-validation method. The kernel scaling factors are used to compare the performance of different classifiers.

**Number of sample size**
- M24: The number of sample size is used to assess the performance of the classifier. The number of sample size is calculated using a method such as the K-fold cross-validation method. The number of sample size is used to compare the performance of different classifiers.

**No. of total features**
- M25: The number of total features is used to assess the performance of the classifier. The number of total features is calculated using a method such as the K-fold cross-validation method. The number of total features is used to compare the performance of different classifiers.

**Kernel scaling factors**
- M26: The kernel scaling factors are used to assess the performance of the classifier. The kernel scaling factors are calculated using a method such as the K-fold cross-validation method. The kernel scaling factors are used to compare the performance of different classifiers.
Fig. 1. RMS versus scaling factor $k_{n}$ for LDA with sample size $n = 50$, total feature size $D = 200$ and selected feature size $d = 3$.

selection of expected errors and interpolation can be obtained. One might also use a coarser interpolation for computational purposes, with some loss of performance. In fact, that is precisely what we do here because we will subsequently perform a computationally intensive robustness analysis. Here we use: $k_{min}^{D} = 0.8$ for $\epsilon_{0} < 0.125$; $k_{min}^{D} = 1$ for $0.125 \leq \epsilon_{0} < 0.225$; $k_{min}^{D} = 1.2$ for $0.225 \leq \epsilon_{0} \leq 0.275$; $k_{min}^{D} = 1.4$ for $0.275 \leq \epsilon_{0} \leq 0.325$; and $k_{min}^{D} = 1.6$ for $\epsilon_{0} > 0.325$.

The final bolstered error estimate is computed from the data using this scaling factor. The success of the procedure depends on robustness in choosing a scaling factor because (i) the estimated model will be inaccurate
Owing to small sample size, (ii) cross-validation has significant variance for small samples, (iii) the estimated model will differ to some extent from the models involved in creating the look-up table and (iv) the method of moments is not optimal. The following protocol is used to obtain the bolstered resubstitution error estimate:

**Protocol 2**

1. Given a sample set $S^D_n$ with size $n$ and dimension $D$, select a size-$d$ feature set $A$ using a feature-selection method $F$ on $S^D_n$, resulting in a reduced dimension sample set $S^d_n$ for the feature set $A$.
2. Design a classifier $\psi_n$ for $S^d_n$ according to the given classification rule $\Psi_1$, and compute the 10-fold cross-validation error estimate $\varepsilon_0$.
3. From the look-up table $(E[\varepsilon_n], k_{\min}^D)$ choose the kernel scaling factor $k_{\min}^D$ by setting $E[\varepsilon_n] = \varepsilon_0$.
4. Compute the bolstered error estimate $\hat{\varepsilon}_n^{D,\text{bolst}}$ using the selected scaling factor.

### 3 RESULTS AND DISCUSSION

To illustrate application, we have applied the method to two gene expression datasets:

- **Myeloma dataset**: data are downloaded from the NIH Gene Expression Omnibus (GEO) under accession numbers GSE5900 and GSE2658, which contain 54,613 probe sets and 559 multiple myeloma (MM) samples, as well as 3 other subtypes [monoclonal gammopathy of undetermined significance (MGUS)], 44 samples; smoldering MM (SMM), 12 samples; healthy donors with normal plasma cell (NPC), 22 samples (Zhan et al., 2006). Samples are labeled into two classes, one for MGUS/SMM/NPC and the other for MM. Due to the significant unbalance of the samples between the two classes, only 156 samples are randomly selected from the 559 MM samples. The number 156 has been chosen as...
The total number of repetitions is 200. The average true error and computing the RMS (datasets, respectively, and LDA for classification. We repeatedly two patient datasets. In both cases, 0

For the breast cancer dataset, the average true error and SD are 0

– average scaling factor 1.10; M1,

SD for the myeloma dataset are 0

scaling factors for the four models are given by: M1,

of expected errors and interpolation has proven sufficient. To further demonstrate the effectiveness of Protocol 2, we

cautionary comment regarding the multiple myeloma data, we

interest are the scaling factors produced by the protocol. The average scaling factors for the four models are given by: M1, E[ε|a]|d = 0.20 – average scaling factor 1.10; M1, E[ε|a]|d = 0.35 – average scaling factor 1.39; M2 E[ε|a]|d = 0.20 – average scaling factor 1.09; and M2,

To further investigate this issue, we take the model M2 in Section 2.3, but perturb the skewness and kurtosis of the class at the origin to obtain a Pearson system (Elderton and Johnson, 1969). Figure 6 shows the eight different distributions in the Pearson system with varying skewness and kurtosis. For the resulting model A0 and each skewness and kurtosis combination, where valid, we do the following:

1. Generate a sample set $S^P_{d}$ of size $n = 50$ and a total of $D = 200$ features from the model $A^P$
2. Feature select a size-$d$ = 3 feature set $A_r$, resulting in a reduced dimension sample set $S^R_{d}$
3. Design a classifier $\psi^d_0$ for $S^R_{d}$ using LDA.

Fig. 4. RMS using LDA and protocol 2 for (a) myeloma dataset, total feature size $D = 200$ and (b) breast cancer dataset, total feature size $D = 70$. For both datasets: sample size $n = 50$ and selected feature size $d = 3$.

Fig. 5. RMS using LDA and protocol 2 for (a) myeloma dataset, total feature size $D = 200$ and (b) breast cancer dataset, total feature size $D = 70$. For both datasets: sample size $n = 50$ and selected feature size $d = 3$.

3.1 Robustness to non-Gaussian data

Although $\epsilon^{un}$ is derived with Gaussian models, it is robust enough for models where this assumption is violated, as with the patient data, where the underlying distribution is almost certainly not Gaussian. To further investigate this issue, we take the model M2 in Section 2.3, but perturb the skewness and kurtosis of the class at the origin to obtain a Pearson system (Elderton and Johnson, 1969). Figure 6 shows the eight different distributions in the Pearson system with varying skewness and kurtosis. For the resulting model $A^P_{min}$ and each skewness and kurtosis combination, where valid, we do the following:

1. Generate a sample set $S^P_{d}$ of size $n = 50$ and a total of $D = 200$ features from the model $A^P$
2. Feature select a size-$d$ = 3 feature set $A_r$, resulting in a reduced dimension sample set $S^R_{d}$
3. Design a classifier $\psi^d_0$ for $S^R_{d}$ using LDA.
We have derived an optimal kernel scaling factor that can be used for symmetry, only positive skewness is shown. In all cases, classification of genomic data when samples are small. This bolstered error estimator achieves a significant RMS improvement over cross-validation when samples are small, with continued, albeit smaller, error estimator achieves a significant RMS improvement over cross-validation. Hence, we have been able to incorporate optimality criteria from across a collection of families to arrive at suitable bolstering kernels for practical situations, thereby facilitating its use in applications like classification of genomic data when samples are small.

**3.2 Concluding remarks**

We have derived an optimal kernel scaling factor that can be used for bolstered error estimation in high feature dimensions. This bolstered error estimator achieves a significant RMS improvement over cross-validation when samples are small, with continued, albeit smaller, performance improvement over the adjusted bootstrap. This superior performance is robust over a wide range of models. Hence, we have been able to incorporate optimality criteria from across a collection of families to arrive at suitable bolstering kernels for practical situations, thereby facilitating its use in applications like classification of genomic data when samples are small.

**ACKNOWLEDGEMENTS**

We would also like to thank the High-Performance Biocomputing Center of TGen for providing the clustered computing resources used in this study; this includes the Saguaro-2 cluster supercomputer, a collaborative effort between TGen and the ASU Fulton High Performance Computing Initiative.

**Funding:** National Science Foundation (CCF-0634794 and CCF-0845407); National Institutes of Health grant (1510KR025056-01) to Saguaro-2 cluster, in part.

**Conflict of Interest:** none declared.

**REFERENCES**


