Available online at www.sciencedirect.com

Pattern Recognition 38 (2005) 1799 – 1814

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Exact performance of error estimators for discrete classifiers

Ulisses Braga-Neto[a], Edward Dougherty[b, c, d, *]

[a]Virology and Experimental Therapy Laboratory, Aggeu Magalhães Research Center, CPqAM/FIOCRUZ, Recife, PE 50.670-420, Brazil
[b]Department of Electrical Engineering, Texas A&M University, College Station, TX 77843, USA
[c]Division of Computational Biology, Translational Genomics Research Institute, Phoenix, AZ 85004, USA
[d]Department of Pathology, University of Texas MD Anderson Cancer Center, Houston, TX 77030, USA

Received 6 August 2004; accepted 25 February 2005

Abstract

Discrete classification problems abound in pattern recognition and data mining applications. One of the most common discrete rules is the discrete histogram rule. This paper presents exact formulas for the computation of bias, variance, and RMS of the resubstitution and leave-one-out error estimators, for the discrete histogram rule. We also describe an algorithm to compute the exact probability distribution of resubstitution and leave-one-out, as well as their deviations from the true error rate. Using a parametric Zipf model, we compute the exact performance of resubstitution and leave-one-out, for varying expected true error, number of samples, and classifier complexity (number of bins). We compare this to approximate performance measures-computed by Monte-Carlo sampling—of 10-repeated 4-fold cross-validation and the 0.632 bootstrap error estimator. Our results show that resubstitution is low-biased but much less variable than leave-one-out, and is effectively the superior error estimator between the two, provided classifier complexity is low. In addition, our results indicate that the overall performance of resubstitution, as measured by the RMS, can be substantially better than the 10-repeated 4-fold cross-validation estimator, and even comparable to the 0.632 bootstrap estimator, provided that classifier complexity is low and the expected error rates are moderate. In addition to the results discussed in the paper, we provide an extensive set of plots that can be accessed on a companion website, at the URL `http://ee.tamu.edu/~edward/exact_discrete`.

Keywords: Error estimation; Discrete classification; Histogram rule; Resubstitution; Leave-one-out; Cross-validation; Bootstrap

1. Introduction

Discrete classification, also called categorical classification, or multinomial discrimination [1–4] is very important in several applications, particularly in biology, economics, psychology and social science [3]. In the field of Data mining, discrete classification algorithms and applications are particularly prevalent—in fact, Data mining practitioners often advocate discretizing continuous attributes to achieve a purely discrete problem [5] (even though this certainly introduces a loss of information). Discrete classification also applies to the case of fixed-partition classification in Euclidean space [1]—however, this is of less widespread interest than purely categorical problems.

In practical real-world problems, especially in the small-sample settings prevalent in many applications, a fundamental issue is how to estimate the error of a classifier, since the underlying probability structure (here referred to as the *probability model*), and therefore the true error of the designed classifier, is unknown. There are many error estimation techniques in use, but the performance analysis of these techniques has been based on ad hoc methods or

* Corresponding author. Tel.: +1 979 862 8896; fax: +1 979 845 6259.
E-mail address: e-dougherty@tamu.edu (E. Dougherty).

approximate asymptotic performance bounds, which are often useless in small-sample settings. We show in this paper that it is possible to write simple "closed-formula" expressions for performance measures, such as bias, variance and RMS, of non-randomized error estimators (e.g., resubstitution and leave-one-out); furthermore, we present an algorithm to calculate the exact probability distribution of non-randomized error estimators, with the help of high-performance computers (the use of computers for exact calculation has in fact been considered in several areas of statistics; e.g. see [6–10]).

Using a parametric Zipf model, we compute the exact performance of resubstitution and leave-one-out, for varying expected true error, number of samples, and classifier complexity (number of bins). We compare this to approximate performance measures—computed by Monte-Carlo sampling—of more complex, randomized error estimators, namely, 10-repeated 4-fold cross-validation and the 0.632 bootstrap error estimator. The exact performance results prove that resubstitution is low-biased but much less variable than leave-one-out, and is effectively the superior error estimator provided classifier complexity is low. Comparisons with the Monte-Carlo performance measures of the randomized error estimators indicate that the 0.632 bootstrap error estimator generally displays the best performance. Surprisingly, the overall performance of resubstitution, as measured by the RMS, can be substantially better than 10-repeated 4-fold cross-validation, and even comparable to the 0.632 bootstrap estimator, provided that classifier complexity is low and the expected error rates are moderate (conditions often found in practice). Considering that resubstitution is a very inexpensive error estimator computationally, as compared to cross-validation, and particularly bootstrap estimators, this provides an argument for avoiding complex resampling-based error estimators in favor of resubstitution in applications where a very large number of error estimates have to be computed and classifier complexity is low—for instance, feature selection for gene regulation (Kim et al., Shmulevich et al.). In addition to the results discussed in the paper, we provide an extensive set of plots that can be accessed on a companion website, at the URL `http://ee.tamu.edu/~edward/exact_discrete`.

We mention here the work of G.F. Hughes, who calculated exact results on distribution-free mean performance of discrete classifiers [9,10]. Hughes did not consider the error estimation problem, being concerned only with the true error of the designed classifiers. Hughes gives an expression for the mean expected true error over the probability model space, assuming equally likely models. Such a mean performance measure is distribution-free, being dependent only on sample size n and complexity b, and allows the author to study classification performance as a function of these parameters. However, the usefulness of such a measure in a practical setting is questionable. By assuming equally likely models and computing the mean over all possible probability models, one is giving equal importance to models with small and large errors. In practice, we know that sound experimental design and effective feature selection algorithms will ensure that errors are moderate to small. As a result, the results and recommendations made in [9] tend to be overly pessimistic (as pointed out in [11]).

Even though our work applies specifically to discrete classification (a relevant and worthy problem in itself), we hope that our results can also illuminate issues related to error estimation for classification in general. Hughes' work is an example of this kind of extension—his 1968 paper [9] used discrete classifiers to demonstrate rigorously the "peaking" of the mean accuracy, whereby performance at first improves as the number of variables increase, and then eventually deteriorates (for a fixed number of training samples). But this is a phenomenon that affects all classifiers, including continuous-variable ones. In fact, the peaking phenomenon is referred to by some authors as the "Hughes phenomenon" [12].

2. Discrete classification

Formally, in the discrete classification problem there are p predictor variables $X_1, \ldots, X_p$, such that each X_i takes on a finite number b_i of values, and a binary target variable $Y \in \{0, 1\}$ (we adopt in this paper the usual notation, whereby capital letters denote random variables and small letters denote deterministic realizations of those variables). The predictors often correspond to nominal attributes (i.e., values without explicit numerical meaning or ordering, such as yes/no, gender, marital status, and so on), so that there is little hope in the use of traditional numerical discrimination procedures. The predictors as a group take on values in a finite space of $b = \prod_{i=1}^{p} b_i$ possible states. A bijection can be established between this finite state-space and the sequence of integers $1, \ldots, b$. Therefore, we may assume a single predictor variable X taking on values in the set $X \in \{1, \ldots, b\}$. The value b can be viewed as the number of "bins" into which the data is categorized—it provides a direct measure of the *complexity* of the classification rule [1].

The complete probability structure of the discrete classification problem is specified by $2b+2$ real numbers: the class prior probabilities $c_0 = P(Y = 0)$ and $c_1 = P(Y = 1)$, and the class-conditional probabilities: $p_i = P(X = i \mid Y = 0)$ and $q_i = P(X = i \mid Y = 1)$, for $i = 1, \ldots, b$.

Since we have the identities

$$c_1 = 1 - c_0, \tag{1}$$

$$p_b = 1 - \sum_{i=1}^{b-1} p_i, \tag{2}$$

$$q_b = 1 - \sum_{i=1}^{b-1} q_i, \tag{3}$$

the problem is in fact determined by a $(2b-1)$-dimensional vector

$$\pi = (c_0, p_1, \ldots, p_{b-1}, q_1, \ldots, q_{b-1}) \in \mathbb{R}^{2b-1}. \tag{4}$$

Furthermore, we have the constraints:

$$0 \leqslant c_0 \leqslant 1, \tag{5}$$

$$\sum_{i=1}^{b-1} p_i \leqslant 1 \quad \text{and} \quad p_i \geqslant 0, \quad i = 1, \ldots, b-1, \tag{6}$$

$$\sum_{i=1}^{b-1} q_i \leqslant 1 \quad \text{and} \quad q_i \geqslant 0, \quad i = 1, \ldots, b-1. \tag{7}$$

Hence, $c_0 \in I$ (the unit closed interval) and $p_i, q_i \in S_{b-1}$ (the $(b-1)$-dimensional simplex), and the probability vector π that defines the discrete classification problem is a member of the set $\Pi = I \times S_{b-1} \times S_{b-1} \subset \mathbb{R}^{2b-1}$. We call Π the *probability model space*, and each $\pi \in \Pi$ a *probability model*. Note that the probability model space is a relatively small subset of a finite-dimensional space.

For a given a probability model, the *error rate* of a discrete classifier $g : \{1, \ldots, b\} \to \{0, 1\}$ is the probability of misclassification: $\varepsilon = P(Y \neq g(X)) = E(|Y - g(X)|)$. Clearly,

$$\begin{aligned}\varepsilon &= \sum_{i=1}^{b} P(X = i,\ Y = 1 - g(i)) \\ &= \sum_{i=1}^{b} P(X = i | Y = 1 - g(i)) P(Y = 1 - g(i)) \\ &= \sum_{i=1}^{b} [p_i c_0 I_{g(i)=1} + q_i c_1 I_{g(i)=0}]. \end{aligned} \tag{8}$$

From this, it is clear that the optimal minimum-error Bayes classifier is given by

$$g_{\text{BAYES}}(i) = \begin{cases} 1 & \text{if } p_i c_0 < q_i c_1 \\ 0 & \text{otherwise} \end{cases}, \quad i = 1, \ldots, b, \tag{9}$$

with corresponding optimal error rate

$$\varepsilon_{\text{BAYES}} = \sum_{i=1}^{b} \min\{p_i c_0, q_i c_1\}. \tag{10}$$

3. The histogram rule

This histogram rule [1,2,9] is by no means the only discrete classification rule in use, but it is certainly the most intuitive for categorical problems (another noteworthy example of a discrete rule is the "maximum-mean-accuracy" rule, used in [10]—there is also a myriad of discrete rules used in Data mining [5]). The histogram rule corresponds to the "plug-in" rule for approximating the Bayes classifier, as we discuss below. In this paper, we will assume the histogram rule, but the methods described here are general and can be applied in principle to any discrete classification rule.

Let $S_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ be an i.i.d. sample taken from the probability model distribution F_π, that is, $S_n \sim F_\pi^n$. Let us define the random variables:

$$U_i = \#\{X_j = i \mid Y_j = 0\}, \quad i = 1, \ldots, b, \tag{11}$$

$$V_i = \#\{X_j = i \mid Y_j = 1\}, \quad i = 1, \ldots, b. \tag{12}$$

If we assume symmetric classification rules, i.e., rules for which the order of the samples does not matter (which is the case of the histogram rule and virtually all useful rules), then the sample is completely determined by the random variables $\mathbf{U} = \{U_1, \ldots, U_b\}$, $\mathbf{V} = \{V_1, \ldots, V_b\}$. Let $N_0 = \sum_i U_i$ and $N_1 = \sum_i V_i$. Note that $N_0 + N_1 = n$, the total number of samples. Two possibilities present themselves. One may assume that N_0 and N_1 are random variables, where N_0 is drawn from a binomial distribution with parameters (n, c_0) and $N_1 = n - N_0$. This corresponds to a *full sampling* setting, wherein the samples are taken from the mixture of populations, and one knows a priori only the total number of samples, but not the number of samples that belong to each class (e.g., until the samples are labeled by an expert). Another possibility is to fix the values $N_0 = n_0$ and $N_1 = n_1 = n - n_0$ during experimental design, and to sample the populations separately. This is the setting assumed, for example, in [9]. We assume that the values of n_0 and n_1 are chosen to reflect the a priori probabilities c_0 and c_1 of each class: $n_0 = [c_0 n]$ and $n_1 = [c_1 n] = n - n_0$, where $[x]$ denotes the nearest integer to x. This is referred to as *stratified sampling*. Both full sampling and stratified sampling are relevant from a practical perspective, and both will be considered in this paper.

The classifier designed by the histogram rule given the sample S_n is given by

$$g(i) = I_{v_i > u_i} = \begin{cases} 1 & \text{if } v_i > u_i \\ 0 & \text{otherwise} \end{cases}, \quad i = 1, \ldots, b. \tag{13}$$

For this reason, the histogram rule is also known as the "majority" rule.

The maximum-likelihood (ML) estimates for the model parameters c_0, c_1 and $\{p_i\}, \{q_i\}$ are

$$\widehat{c}_0 = \frac{n_0}{n}, \quad \widehat{c}_1 = \frac{n_1}{n} \quad \text{and} \quad \widehat{p}_i = \frac{u_i}{n_0}, \quad \widehat{q}_i = \frac{v_i}{n_0} \quad \text{for } i = 1, \ldots, b. \tag{14}$$

Plugging these back in the expression for the Bayes classifier in (9) leads to the histogram classifier in (13). In other words, the histogram rule is the plug-in rule for discrete classification.

4. True error rate

From (8), it is clear that the true error of the designed histogram classifier is given by

$$\varepsilon = \sum_{i=1}^{b} [c_0 p_i \, I_{V_i > U_i} + c_1 q_i \, I_{U_i \geqslant V_i}]. \tag{15}$$

Being a function of the random sample, the true error ε is a random variable. The mean $E[\varepsilon]$ has an important meaning in the context of classification rules: it gives the expected true error over the random sample; hence, it does not depend on the sample and is an intrinsic performance measure of the histogram rule. The mean true error $E[\varepsilon]$ measures the difficulty of classification, if one uses the histogram rule, given a probability model, the sample size *n*, and complexity *b*. The expected error can be calculated exactly using the following expression:

$$\begin{aligned} E[\varepsilon] &= \sum_{i=1}^{b} [c_0 p_i \, E[I_{V_i > U_i}] + c_1 q_i \, E[I_{U_i \geqslant V_i}]] \\ &= \sum_{i=1}^{b} [c_0 p_i P(V_i > U_i) + c_1 q_i P(U_i \geqslant V_i)] \\ &= \sum_{i=1}^{b} \sum_{\substack{k,l=0 \\ k+l \leqslant n}}^{n} \alpha^i_{k,l} [c_0 p_i \, I_{l>k} + c_1 q_i \, I_{k \geqslant l}], \end{aligned} \tag{16}$$

where $\alpha^i_{k,l}$ is the "first-order" joint probability distribution $P(U_i = k, V_i = l)$. In the stratified sampling case, U_i is clearly independent of V_i, and both are binomially distributed with parameters (n_0, p_i) and (n_1, q_i), respectively, so that

$$\begin{aligned} \alpha^{i\,(\text{strat})}_{k,l} &= P(U_i = k) P(V_i = l) \\ &= \binom{n_0}{k} p_i^k (1 - p_i)^{n_0 - k} \binom{n_1}{l} q_i^l (1 - q_i)^{n_1 - l}, \end{aligned} \tag{17}$$

for $k = 0, \ldots, n_0$ and $l = 0, \ldots, n_1$, with $\alpha^{i\,(\text{strat})}_{k,l} = 0$, otherwise. In the full sampling case, we condition on N_0, and use the fact that, given $N_0 = n_0$, U_i and V_i are conditionally independent and binomially distributed with parameters (n_0, p_i) and $(n - n_0, q_i)$. Hence,

$$\begin{aligned} \alpha^{i\,(\text{full})}_{k,l} &= \sum_{n_0=0}^{n} P(U_i = k, V_i = l | N_0 = n_0) \; P(N_0 = n_0) \\ &= \sum_{n_0=0}^{n} P(U_i = k | N_0 = n_0) P(V_i = l | N_0 = n_0) \\ &\quad \times P(N_0 = n_0) \\ &= \sum_{n_0=k}^{n-l} \binom{n_0}{k} p_i^k (1 - p_i)^{n_0 - k} \binom{n - n_0}{l} \\ &\quad \times q_i^l (1 - q_i)^{n - n_0 - l} \binom{n}{n_0} c_0^{n_0} c_1^{n - n_0} \\ &= \sum_{n_0=k}^{n-l} \binom{n}{k,\ l,\ n_0 - k,\ n - n_0 - l} p_i^k q_i^l c_0^{n_0} c_1^{n-n_0} \\ &\quad \times (1 - p_i)^{n_0 - k} (1 - q_i)^{n - n_0 - l}, \end{aligned} \tag{18}$$

for $k, l = 0, \ldots, n$, such that $k + l \leqslant n$.

The variance of the true error can be computed by finding the second moment $E[\varepsilon^2]$. This can be calculated exactly as follows:

$$\begin{aligned} E[\varepsilon^2] &= \sum_{i=1}^{b} \left\{ c_0^2 p_i^2 \, E[I_{V_i > U_i}] + c_1^2 q_i^2 \, E\left[I_{U_i \geqslant V_i}\right] \right\} \\ &\quad + \sum_{\substack{i,j=1 \\ i \neq j}}^{b} \left\{ c_0^2 p_i p_j \, E\left[I_{V_i > U_i} I_{V_j > U_j} \right] \right. \\ &\quad + c_0 c_1 \left(p_i q_j E\left[I_{V_i > U_i} I_{U_j \geqslant V_j} \right] \right. \\ &\quad \left. + \; p_j q_i E\left[I_{U_i \geqslant V_i} I_{V_j > U_j} \right] \right) \\ &\quad \left. + c_1^2 q_i q_j \, E\left[I_{U_i \geqslant V_i} I_{U_j \geqslant V_j} \right] \right\} \\ &= \sum_{i=1}^{b} \left\{ c_0^2 p_i^2 \, P(V_i > U_i) + c_1^2 q_i^2 \, P(U_i \geqslant V_i) \right\} \\ &\quad + \sum_{\substack{i,j=1 \\ i \neq j}}^{b} \left\{ c_0^2 p_i p_j \, P(V_i > U_i, V_j > U_j) \right. \\ &\quad + c_0 c_1 [p_i q_j P(V_i > U_i, U_j \geqslant V_j) \\ &\quad + p_j q_i P(U_i \geqslant V_i, V_j > U_j)] \\ &\quad \left. + c_1^2 q_i q_j \, P(U_i \geqslant V_i, U_j \geqslant V_j) \right\} \\ &= \sum_{i=1}^{b} \sum_{\substack{k,l=0 \\ k+l \leqslant n}}^{n} \alpha^i_{k,l} \left[c_0^2 p_i^2 \, I_{l>k} + c_1^2 q_i^2 \, I_{k \geqslant l} \right] \\ &\quad + \sum_{\substack{i,j=1 \\ i \neq j}}^{b} \sum_{\substack{k,l,r,s=0 \\ k+l+r+s \leqslant n}}^{n} \beta^{i,j}_{k,l,r,s} \left\{ c_0^2 p_i p_j \, I_{l>k} I_{s>r} \right. \\ &\quad + c_0 c_1 [p_i q_j I_{l>k} I_{r \geqslant s} + p_j q_i I_{k \geqslant l} I_{s>r}] \\ &\quad \left. + \; c_1^2 q_i q_j \, I_{k \geqslant l} I_{r \geqslant s} \right\}, \end{aligned} \tag{19}$$

where $\alpha^i_{k,l}$ is as defined previously, and $\beta^{i,j}_{k,l,r,s}$ is the "second-order" joint probability distribution $P(U_i = k, V_i = l, U_j = r, V_j = s)$. In the stratified sampling case, U_i, U_j

are independent, as a group, of V_i, V_j, and each group is trinomially distributed with parameters (n_0, p_i, p_j) and (n_1, q_i, q_j), respectively, so that

$$\begin{aligned}\beta^{i,j\,(\mathrm{strat})}_{k,l,r,s} &= P(U_i = k, U_j = r)P(V_i = l, V_j = s)\\ &= \binom{n_0}{k,\ r,\ n_0-r-k} p_i^k p_j^r (1-p_i-p_j)^{n_0-r-k}\\ &\quad\times \binom{n_1}{l,\ s,\ n_1-s-l} q_i^l q_j^s (1-q_i-q_j)^{n_1-s-l},\end{aligned} \tag{20}$$

for $k, r = 0, \ldots, n_0$ such that $k+r \leqslant n_0$, and $l, s = 0, \ldots, n_1$ such that $l + s \leqslant n_1$, with $\beta^{i,j\,(\mathrm{strat})}_{k,l,r,s} = 0$, otherwise. In the full sampling case, we condition on N_0, and use the fact that, given $N_0 = n_0$, U_i, U_j are conditionally independent, as a group, of V_i, V_j, and each group is trinomially distributed with parameters (n_0, p_i, p_j) and $(n - n_0, q_i, q_j)$, respectively, so that we may write

$$\begin{aligned}\beta^{i,j\,(\mathrm{full})}_{k,l,r,s} &= \sum_{n_0=0}^{n} P(U_i = k, V_i=l, U_j=r, V_j=s|N_0=n_0)\\ &\quad\times P(N_0 = n_0)\\ &= \sum_{n_0=0}^{n} P(U_i = k, U_j = r|N_0 = n_0)\\ &\quad\times P(V_i = l, V_j = s|N_0 = n_0)P(N_0 = n_0)\\ &= \sum_{n_0=k+r}^{n-s-l} \binom{n_0}{k,\ r,\ n_0-r-k}\\ &\quad\times p_i^k p_j^r (1-p_i-p_j)^{n_0-r-k}\\ &\quad\times \binom{n-n_0}{l,\ s,\ n-n_0-s-l}\\ &\quad\times q_i^l q_j^s (1-q_i-q_j)^{n-n_0-s-l}\\ &\quad\times \binom{n}{n_0} c_0^{n_0} c_1^{n-n_0}\\ &= \sum_{n_0=k+r}^{n-s-l} \binom{n}{k,\ l,\ r,\ s,\ n_0-r-k,\ n-n_0-s-l}\\ &\quad\times p_i^k q_i^l p_j^r q_j^s c_0^{n_0} c_1^{n-n_0}\\ &\quad\times (1-p_i-p_j)^{n_0-r-k}(1-q_i-q_j)^{n-n_0-s-l},\end{aligned} \tag{21}$$

for $k, l, s, r = 0, \ldots, n$, such that $k + l + r + s \leqslant n$.

The variance of the true error can be computed by using (16) and (19) and the simple expression $\mathrm{Var}[\varepsilon] = E[\varepsilon^2] - E[\varepsilon]^2$.

The formulas given above provide a fast way to compute the expected value, second moment, and variance of the true error. Other statistical summaries, such as skewness, kurtosis, tail probabilities (in fact, *any* statistical summary), of the true error can be found, in a more computationally expensive way, from the probability distribution P_ε of ε. Due to the fact that, in discrete classification, the total number of sample configurations is finite, it is possible to calculate P_ε by computer—for moderate sample size n and complexity b—as we describe next.

First, recall that the random sample is specified by the random vectors $\mathbf{U}$ and $\mathbf{V}$, defined as before, so we write $\varepsilon=\varepsilon(\mathbf{U},\mathbf{V})$. The random vectors $\mathbf{U}$ and $\mathbf{V}$ are discrete, and so the random variable ε is also discrete. Let the *configuration space* D_n be the (finite) set of all possible distinct values that can be taken on by the pair $(\mathbf{U},\mathbf{V})$. The discrete probability distribution P_ε is given by

$$P_\varepsilon(a) = \sum_{(\mathbf{u},\mathbf{v})\in D_n} I_{\{\varepsilon(\mathbf{u},\mathbf{v})=a\}} P(\mathbf{U} = \mathbf{u}, \mathbf{V} = \mathbf{v}). \tag{22}$$

In the stratified sampling case, $\mathbf{U}$ is independent of $\mathbf{V}$, and both are multinomially distributed with parameters $(n_0, p_1, \ldots, p_b)$ and $(n_1, q_1, \ldots, q_b)$, respectively, so that

$$\begin{aligned}P^{(\mathrm{strat})}(\mathbf{U}=\mathbf{u}, \mathbf{V}=\mathbf{v}) &= \binom{n_0}{u_1, \ldots, u_b} \prod_{i=1}^{b} p_i^{u_i}\\ &\quad\times \binom{n_1}{v_1, \ldots, v_b} \prod_{i=1}^{b} q_i^{v_i}.\end{aligned} \tag{23}$$

To visualize the full sampling case, imagine that each bin is split in two halves, one for the first class, and another for the second class. The probability of a sample falling in bin i of the first class is just $P(X = i, Y = 0) = c_0 p_i$. Analogously, the probability of a sample falling in bin i of the second class is $P(X = i, Y = 1) = c_1 q_i$. The pair $(\mathbf{U},\mathbf{V})$ is thus jointly multinomially distributed with parameters $(n, c_0p_1, \ldots, c_0p_b, c_1q_1, \ldots, c_1q_b)$. Thus we can write

$$\begin{aligned}&P^{(\mathrm{full})}(\mathbf{U}=\mathbf{u}, \mathbf{V}=\mathbf{v})\\ &\quad= \binom{n}{u_1, \ldots, u_b, v_1, \ldots, v_b} c_0^{\sum_i u_i} c_1^{\sum_i v_i} \prod_{i=1}^{b} p_i^{u_i} q_i^{v_i}.\end{aligned} \tag{24}$$

Now, let us define C_m as the set of possible configurations that m objects can take over the bins. An algorithm to generate C_m is described in the Appendix. It is clear that, in the case of stratified sampling, the configuration space is given by $D_n^{(\mathrm{strat})} = C_{n_0} \times C_{n_1}$, whereas in the full sampling case, we have that $D_n^{(\mathrm{full})} = \bigcup_{n_0=0}^{n} [C_{n_0} \times C_{n-n_0}]$.

The probability distribution P_ε in (22) can be calculated by computer by using the following simple algorithm.

Algorithm PDF: Find D_n, as described above. For each pair $(\mathbf{u},\mathbf{v}) \in D_n$, calculate its probability $P(\mathbf{U}=\mathbf{u},\mathbf{V}=\mathbf{v})$ and the corresponding error $\varepsilon(\mathbf{u},\mathbf{v})$—if this error value has not been encountered before, create a new entry in the probability distribution vector and initialize it with $P(\mathbf{U}=\mathbf{u}, \mathbf{V}=\mathbf{v})$, otherwise simply add $P(\mathbf{U}=\mathbf{u}, \mathbf{V}=\mathbf{v})$ to the existing entry for $\varepsilon(\mathbf{u},\mathbf{v})$.

5. Error estimators

In practice, the underlying probability model is unknown, and the true error ε has to be estimated from the sample data using an *error estimator* $\hat{\varepsilon}$. An error estimator may be a deterministic function of the sample data S_n, in which case it is a *non-randomized error estimator*. Such an error estimator is random only through the random sample. Among popular non-randomized error estimators, we have resubstitution and leave-one-out. By contrast, *randomized error estimators* have "internal" random factors that affect their outcome. Popular randomized error estimators include (random-fold) cross-validation and all bootstrap error estimators.

The *internal variance* of an error estimator is the variance due only to its internal random factors, $V_{\text{int}} = \text{Var}(\hat{\varepsilon}|S_n)$. This variance is zero for non-randomized error estimators. The full variance $\text{Var}(\hat{\varepsilon})$ of the error estimator, on the other hand, also takes into account the uncertainty introduced by the random sample data. See [13] for a detailed discussion of issues regarding randomized and non-randomized error estimators, internal and full variance, etc.

Of great interest in the analysis of performance of an error estimator $\hat{\varepsilon}$ are its *bias*,

$$\text{Bias}[\hat{\varepsilon}] = E[\hat{\varepsilon}] - E[\varepsilon], \tag{25}$$

the *deviation variance*,

$$\begin{aligned}\text{Var}_{\text{d}}[\hat{\varepsilon}] &= \text{Var}[\hat{\varepsilon} - \varepsilon] \\ &= E[(\hat{\varepsilon} - \varepsilon)^2] - E[(\hat{\varepsilon} - \varepsilon)]^2 \\ &= E[\varepsilon^2] - 2E[\varepsilon\hat{\varepsilon}] + E[\hat{\varepsilon}^2] - \text{Bias}[\hat{\varepsilon}]^2,\end{aligned} \tag{26}$$

and the *root mean-square error*,

$$\begin{aligned}\text{RMS}[\hat{\varepsilon}] &= \sqrt{E[(\hat{\varepsilon} - \varepsilon)^2]} \\ &= \sqrt{\text{Var}_{\text{d}}[\hat{\varepsilon}] + \text{Bias}[\hat{\varepsilon}]^2} \\ &= \sqrt{E[\varepsilon^2] - 2E[\varepsilon\hat{\varepsilon}] + E[\hat{\varepsilon}^2]}.\end{aligned} \tag{27}$$

Note that all three measures require only the first and second moments of ε and $\hat{\varepsilon}$, and the correlation between ε and $\hat{\varepsilon}$.

The bias of an error estimator measures whether, on average, it overestimates the true error, or underestimates it. The deviation variance measures the spread of the deviation distribution. Low bias is not good enough if the deviation variance is large. This means that on average the error estimator is close to the true error, but that in fact the estimate for any particular sample set is likely to be far away from the true error. The RMS combines the two complementary measures of bias and deviation variance into a single figure of merit that provides an error measure for estimating the classifier error by the estimator.

In the next few subsections, we discuss a number of well-known non-randomized and randomized error estimators.

5.1. Resubstitution

The simplest and fastest way to estimate the error of a classifier g is to compute its error directly on the sample data itself:

$$\hat{\varepsilon}_r = \frac{1}{n}\sum_{i=1}^{n}|y_i - g(x_i)|. \tag{28}$$

This *resubstitution estimator* [14] is clearly a non-randomized error estimator. It is very fast; however, it can be shown that $\text{Bias}[\hat{\varepsilon}_r] < 0$ for the histogram rule [1]. The larger b is, the more optimistic is resubstitution, since complex classifiers tend to overfit the data, especially with small samples [15]. Nevertheless, provided that classifier complexity is not too high, resubstitution can be surprisingly accurate relative to more complex error estimation schemes, as we will show in Section 7.

5.2. Cross-validation

Cross-validation removes the optimism from resubstitution by employing test points not used in classifier design [16]. In *k-fold cross-validation*, the data set S_n is partitioned into k *folds* S_i, for $i = 1, \ldots, k$ (for simplicity, we assume that k divides n). A classifier g_i is designed on the sample set $S_n \backslash S_i$, and it is tested on S_i, for $i = 1, \ldots, k$. The estimate is the overall proportion of error committed on all folds:

$$\hat{\varepsilon}_{cvk} = \frac{1}{n}\sum_{i=1}^{k}\sum_{j=1}^{n/k}|y_j^i - g_i(x_j^i)|, \tag{29}$$

where (x_j^i, y_j^i) is a sample in the ith fold. The process may be repeated: a number of r cross-validation estimates are computed using different partitions of the data into folds, and the results are averaged, producing the *r-repeated k-fold cross validation* estimator $\hat{\varepsilon}_{cvkr}$. Clearly, both $\hat{\varepsilon}_{cvk}$ and $\hat{\varepsilon}_{cvkr}$ are randomized error estimators.

Let ε_n denote the error of a classifier designed on a sample of size n. A k-fold cross-validation estimator is unbiased as an estimator of $E[\varepsilon_{n-n/k}]$, which makes it slightly pessimistic (high-biased) as an estimator of $E[\varepsilon_n]$. The most well-known cross-validation method, usually attributed to [17], is the *leave-one-out estimator*, where a single observation is left out each time:

$$\hat{\varepsilon}_l = \frac{1}{n}\sum_{i=1}^{n}|y_i - g_i(x_i)|, \tag{30}$$

This corresponds to n-fold cross-validation and is a non-randomized estimator. The leave-one-out estimator is unbiased as an estimator of $E[\varepsilon_{n-1}]$. The main drawback of cross-validation estimators are their large deviation variance [1,18]. They can also be quite slow to compute when the number of folds or samples is large.

5.3. Bootstrap

The bootstrap error estimation technique [19,20] is based on the notion of a "bootstrap sample" S_n^*, which consists of n equally likely draws with replacement from the original data S_n. Hence, some of the samples will appear multiple times, whereas others will not appear at all. The actual proportion of times a data point (x_i, y_i) appears in S_n^* can be written as $P_i^* = 1/n \sum_{j=1}^{n} I_{(x_j^*, y_j^*)=(x_i,y_i)}$. For the computation of the basic *bootstrap zero estimator* [20], a number of bootstrap samples S_n^{*t}, for $t = 1, \ldots, T$ are drawn—a value of T between 25 and 200 being recommended in [20]. A classifier g_t is designed on the bootstrap sample S_n^{*t}, and it is tested on $S_n \backslash S_n^{*t}$, for $t = 1, \ldots, T$. The estimate is the overall proportion of error committed on all bootstrap samples:

$$\hat{\varepsilon}_0 = \frac{\sum_{t=1}^{T}\sum_{i=1}^{n} |y_i - g_t(x_i)|\, I_{P_i^{*t}=0}}{\sum_{t=1}^{T}\sum_{i=1}^{n} I_{P_i^{*t}=0}}. \tag{31}$$

The bootstrap zero estimator works like cross-validation: the classifier is designed on the bootstrap sample and tested on the original data points that are left out. It tends to be high-biased as an estimator of $E[\varepsilon_n]$, since the amount of samples available for designing the classifier is on average only $(1 - e^{-1})n \approx 0.632n$. The *.632 bootstrap estimator* [20],

$$\hat{\varepsilon}_{b632} = (1 - 0.632)\, \hat{\varepsilon}_r + 0.632\, \hat{\varepsilon}_0, \tag{32}$$

tries to correct this bias by doing a weighted average of the bootstrap zero and resubstitution estimators. This has been perhaps the most popular bootstrap estimator in data mining [5]. It has low variance, but can be extremely slow to compute. In addition, it can fail when resubstitution is too low-biased [18].

6. Exact performance of non-randomized error estimators

As remarked in connection with Eqs. (25)–(27), in order to compute the bias, deviance variation, and RMS of an error estimator, one needs the first and second moments of the true error, $E[\varepsilon]$ and $E[\varepsilon^2]$; the first and second moments of the error estimator $E[\hat{\varepsilon}]$ and $E[\hat{\varepsilon}^2]$; and the correlation between true error and error estimator, $E[\varepsilon\hat{\varepsilon}]$. Exact formulas for computing $E[\varepsilon]$ and $E[\varepsilon^2]$ were given in Section 4. In this section, we give exact formulas for the computation of $E[\hat{\varepsilon}]$, $E[\hat{\varepsilon}^2]$, and $E[\varepsilon\hat{\varepsilon}]$, for two widely used non-randomized error estimators: resubstitution and leave-one-out. This allows one to compute exact values for bias, deviance variation, and RMS for these error estimators, for a given number of samples, number of bins, and probability model.

6.1. Resubstitution

From (13) and (28), it follows that the resubstitution error estimator can be written as

$$\hat{\varepsilon}_r = \frac{1}{n}\sum_{i=1}^{b} \min\{U_i, V_i\} = \frac{1}{n}\sum_{i=1}^{b} [U_i\, I_{V_i > U_i} + V_i\, I_{U_i \geqslant V_i}]. \tag{33}$$

Given the training sample (i.e., the sample values of U_i, V_i, for $i = 1, \ldots, b$), the resubstitution estimate is deterministic; therefore, it is a non-randomized error estimator. An interesting fact about resubstitution, as can be easily verified, is that plugging the ML estimates of the model parameters in (14) into the expression for the Bayes error in (8) leads to (33). In other words, resubstitution is the "plug-in" estimator of the Bayes error for the histogram rule.

The expected resubstitution error over the sample can be calculated exactly using the following expression:

$$\begin{aligned} E[\hat{\varepsilon}_r] &= \frac{1}{n}\sum_{i=1}^{b} [E[U_i I_{V_i > U_i}] + E[V_i I_{U_i \geqslant V_i}]] \\ &= \frac{1}{n}\sum_{i=1}^{b} \sum_{\substack{k,l=1 \\ k+l \leqslant n}}^{n} \alpha_{k,l}^{i} [k\, I_{l>k} + l\, I_{k \geqslant l}], \end{aligned} \tag{34}$$

where $\alpha_{k,l}^{i}$ can be calculated with expressions (17) and (18), in the stratified and full sampling cases, respectively.

From (33), it follows that the second moment $E[\hat{\varepsilon}_r^2]$ can be written as

$$\begin{aligned} E[\hat{\varepsilon}_r^2] &= \frac{1}{n^2}\sum_{i=1}^{b} \left\{E[U_i^2 I_{V_i > U_i}] + E[V_i^2 I_{U_i \geqslant V_i}]\right\} \\ &\quad + \frac{1}{n^2}\sum_{\substack{i,j=1 \\ i \neq j}}^{b} \Big\{E[U_i U_j I_{V_i > U_i} I_{V_j > U_j}] \\ &\quad + E[U_i V_j I_{V_i > U_i} I_{U_j \geqslant V_j}] \\ &\quad + E[U_j V_i I_{U_i \geqslant V_i} I_{V_j > U_j}] \\ &\quad + E[V_i V_j I_{U_i \geqslant V_i} I_{U_j \geqslant V_j}]\Big\} \\ &= \frac{1}{n^2}\sum_{i=1}^{b} \sum_{\substack{k,l=1 \\ k+l \leqslant n}}^{n} \alpha_{k,l}^{i} [k^2\, I_{l>k} + l^2\, I_{k \geqslant l}] \\ &\quad + \frac{1}{n^2}\sum_{\substack{i,j=1 \\ i \neq j}}^{b} \sum_{\substack{k,l,r,s=1 \\ k+l+r+s \leqslant n}}^{n} \\ &\quad \times \beta_{k,l,r,s}^{i,j} [kr\, I_{l>k} I_{s>r} + ks\, I_{l>k} I_{r \geqslant s} \\ &\quad + rl\, I_{k \geqslant l} I_{s>r} + ls\, I_{k \geqslant l} I_{r \geqslant s}], \end{aligned} \tag{35}$$

where $\beta_{k,l,r,s}^{i,j}$ can be calculated with expressions (20) and (21), in the stratified and full sampling cases, respectively.

The expression for the correlation between the true error and the resubstitution estimator can be derived as follows:

$$\begin{aligned}
E[\varepsilon\hat{\varepsilon}_r] &= \frac{1}{n}\sum_{i=1}^{b}\left\{c_0 p_i\, E[U_i I_{V_i>U_i}] + c_1 q_i\, E[V_i I_{U_i \geqslant V_i}]\right\} \\
&\quad + \frac{1}{n}\sum_{\substack{i,j=1\\ i\neq j}}^{b}\Big\{c_0 p_i (E[U_j I_{V_i>U_i} I_{V_j>U_j}] \\
&\quad + E[V_j I_{V_i>U_i} I_{U_j\geqslant V_j}]) \\
&\quad + c_1 q_i (E[U_j I_{U_i\geqslant V_i} I_{V_j>U_j}] \\
&\quad + E[V_j I_{U_i\geqslant V_i} I_{U_j\geqslant V_j}])\Big\} \\
&= \frac{1}{n}\sum_{i=1}^{b}\sum_{\substack{k,l=1\\ k+l\leqslant n}}^{n} \alpha^i_{k,l}[c_0 p_i k I_{l>k} + c_1 q_i l I_{k\geqslant l}] \\
&\quad + \frac{1}{n}\sum_{\substack{i,j=1\\ i\neq j}}^{b}\sum_{\substack{k,l=0\\ r,s=1\\ k+l+r+s\leqslant n}}^{n} \beta^{i,j}_{k,l,r,s} \\
&\quad \times \{c_0 p_i (r\, I_{l>k} I_{s>r} + s\, I_{l>k} I_{r\geqslant s}) \\
&\quad + c_1 q_i (r\, I_{k\geqslant l} I_{s>r} + s\, I_{k\geqslant l} I_{r\geqslant s})\}. \qquad (36)
\end{aligned}$$

One can also compute the PDF of the resubstitution estimator, for a given model, by applying algorithm PDF (see Section 4), with $\hat{\varepsilon}_r(\mathbf{u},\mathbf{v})$ in place of $\varepsilon(\mathbf{u},\mathbf{v})$.

Of more interest is the PDF of the deviation $\hat{\varepsilon}_r - \varepsilon$. The mean and variance of the deviation PDF give the bias and deviation variance of the error estimator. Other statistics, such as skewness, kurtosis, etc. can be easily computed. The deviation PDFs of several error estimator have been approximated in [18] by Monte-Carlo sampling, in a study involving continuous classification rules. In the present discrete classification setting, the deviation PDF is discrete and can be computed exactly by applying algorithm PDF, with $\hat{\varepsilon}_r(\mathbf{u},\mathbf{v}) - \varepsilon(\mathbf{u},\mathbf{v})$ in place of $\varepsilon(\mathbf{u},\mathbf{v})$.

6.2. Leave-one-out

In the case of discrete histogram classification, the leave-one-out error estimator can be written as

$$\hat{\varepsilon}_l = \frac{1}{n}\sum_{i=1}^{b}[U_i\, I_{V_i\geqslant U_i} + V_i\, I_{U_i\geqslant V_i-1}]. \qquad (37)$$

It can be easily seen that this is a non-randomized error estimator. It is interesting to note that equation (37) for leave-one-out is similar to the expression (33) for resubstitution—however, the behaviour of the two estimators is completely different, as we will in Section 7.

The expected leave-one-out error over the sample can be calculated exactly using the following expression:

$$\begin{aligned}
E[\hat{\varepsilon}_l] &= \frac{1}{n}\sum_{i=1}^{b}[E[U_i I_{V_i\geqslant U_i}] + E[V_i I_{U_i\geqslant V_i-1}]] \\
&= \frac{1}{n}\sum_{i=1}^{b}\sum_{\substack{k,l=1\\ k+l\leqslant n}}^{n} \alpha^i_{k,l}[k\, I_{l\geqslant k} + l\, I_{k\geqslant l-1}], \qquad (38)
\end{aligned}$$

where the coefficients $\alpha^i_{k,l}$ are computed with expressions (17) and (18), in the stratified and full sampling cases, respectively.

From (37), it follows that the second moment $E[\hat{\varepsilon}_l^2]$ can be written as

$$\begin{aligned}
E[\hat{\varepsilon}_l^2] &= \frac{1}{n^2}\sum_{i=1}^{b}\Big\{E[U_i^2 I_{V_i\geqslant U_i}] + 2\,E[U_i V_i I_{U_i\leqslant V_i\leqslant U_i+1}] \\
&\quad + E[V_i^2 I_{U_i\geqslant V_i-1}]\Big\} \\
&\quad + \frac{1}{n^2}\sum_{\substack{i,j=1\\ i\neq j}}^{b}\Big\{E[U_i U_j I_{V_i\geqslant U_i} I_{V_j\geqslant U_j}] \\
&\quad + E[U_i V_j I_{V_i\geqslant U_i} I_{U_j\geqslant V_j-1}] \\
&\quad + E[U_j V_i I_{U_i\geqslant V_i-1} I_{V_j\geqslant U_j}] \\
&\quad + E[V_i V_j I_{U_i\geqslant V_i-1} I_{U_j\geqslant V_j-1}]\Big\} \\
&= \frac{1}{n^2}\sum_{i=1}^{b}\sum_{\substack{k=0\\ l=1\\ k+l\leqslant n}}^{n} \alpha^i_{k,l}[k^2\, I_{l\geqslant k} + 2kl I_{k\leqslant l\leqslant k+1} \\
&\quad + l^2\, I_{k\geqslant l-1}] \\
&\quad + \frac{1}{n^2}\sum_{\substack{i,j=1\\ i\neq j}}^{b}\sum_{\substack{k,r=0\\ l,s=1\\ k+l+r+s\leqslant n}}^{n} \beta^{i,j}_{k,l,r,s} \\
&\quad \times [kr\, I_{l\geqslant k} I_{s\geqslant r} + ks\, I_{l\geqslant k} I_{r\geqslant s-1} \\
&\quad + rl\, I_{k\geqslant l-1} I_{s\geqslant r} + ls\, I_{k\geqslant l-1} I_{r\geqslant s-1}], \qquad (39)
\end{aligned}$$

where the coefficients $\beta^{i,j}_{k,l,r,s}$ are calculated via expressions (20) and (21), in the stratified and full sampling cases, respectively.

Next, we derive the correlation between the true error and the leave-one-out estimator:

$$\begin{aligned}
E[\varepsilon\hat{\varepsilon}_l] &= \frac{1}{n}\sum_{i=1}^{b}\big\{c_0 p_i (E[U_i I_{V_i>U_i}] + E[V_i I_{V_i=U_i+1}]) \\
&\quad + c_1 q_i (E[U_i I_{U_i=V_i}] + E[V_i I_{U_i\geqslant V_i}])\big\} \\
&\quad + \frac{1}{n}\sum_{\substack{i,j=1\\ i\neq j}}^{b}\Big\{c_0 p_i (E[U_j I_{V_i>U_i} I_{V_j\geqslant U_j}] \\
&\quad + E[V_j I_{V_i>U_i} I_{U_j\geqslant V_j-1}])
\end{aligned}$$

$$
\begin{aligned}
&+ c_1 q_i (E[U_j I_{U_i \geqslant V_i} I_{V_j \geqslant U_j}] \\
&+ E[V_j I_{U_i \geqslant V_i} I_{U_j \geqslant V_j - 1}]) \Big\} \\
&= \frac{1}{n} \sum_{i=1}^{b} \sum_{\substack{k=0 \\ l=1 \\ k+l \leqslant n}}^{n} \alpha^i_{k,l} [c_0 p_i (k I_{l>k} + l I_{l=k+1}) \\
&+ c_1 q_i (k I_{k=l} + l I_{k \geqslant l})] \\
&+ \frac{1}{n} \sum_{\substack{i,j=1 \\ i \neq j}}^{b} \sum_{\substack{k,l,r=0 \\ s=1 \\ k+l+r+s \leqslant n}}^{n} \beta^{i,j}_{k,l,r,s} \\
&\times \left\{ c_0 p_i (r\, I_{l>k} I_{s \geqslant r} + s\, I_{l>k} I_{r \geqslant s-1}) \right. \\
&\left. + c_1 q_i (r\, I_{k \geqslant l} I_{s \geqslant r} + s\, I_{k \geqslant l} I_{r \geqslant s-1}) \right\}. \qquad (40)
\end{aligned}
$$

As in the case of resubstitution, one can also compute the PDF of the leave-one-out estimator, for a given model, by applying algorithm PDF (see Section 4), with $\hat{\varepsilon}_l(\mathbf{u}, \mathbf{v})$ in place of $\varepsilon(\mathbf{u}, \mathbf{v})$. Similarly, one can compute the PDF of the deviation $\hat{\varepsilon}_l - \varepsilon$ by substituting $\hat{\varepsilon}_l(\mathbf{u}, \mathbf{v}) - \varepsilon(\mathbf{u}, \mathbf{v})$ for $\varepsilon(\mathbf{u}, \mathbf{v})$.

7. Parametric model

In this section, we display exactly computed performance measures for resubstitution and leave-one-out, under varying expected true error, number of samples, and number of bins. We use a parametric Zipf model, where the parameter controls the difficulty of classification. In each case, we also plot the approximate performance measures for cross-validation and bootstrap-based error estimators, computed by Monte-Carlo sampling.

We observed that results for the stratified and full sampling cases display the same general trends, so we will focus here on the stratified sampling case (the difference between the two types of sampling is most noticeable in the bias of the error estimators, and the difference becomes significant only for moderate to high errors). An extensive set of plots for both stratified and full sampling can be found on the companion website.

For simplicity, we assume throughout equally likely classes, i.e., $c_0 = c_1 = 0.5$. We consider only even number of samples n, so that there is the same number $n/2$ of samples in each class.

The Zipf distribution is a well-known power-law discrete distribution, encountered in many applications. It was originally introduced by G.K. Zipf to model the frequency of words in common text [21]. The class-conditional probabilities under the parametric Zipf model are given by:

$$p_i = \frac{K}{i^\alpha}, \qquad (41)$$

$$q_i = p_{b-i+1} \qquad (42)$$

for $i = 1, \ldots, b$. Here $\alpha > 0$, and the normalizing constant K is given by

$$K = \left[\sum_{i=1}^{b} \left(\frac{1}{i^\alpha} \right) \right]^{-1}. \qquad (43)$$

It is clear that, as $\alpha \to 0$, the distributions tend to become uniform—which represents maximum confusion between the classes—whereas, as $\alpha \to \infty$, the distributions become concentrated in single (distinct) bins—which corresponds to maximum discrimination between the classes. In fact, the expected true error of the histogram rule decreases monotonically with α.

As remarked previously, for moderate n and b, we can compute the probability distribution function of the true error, by applying algorithm PDF described in Section 4. Fig. 1 displays the PDFs so computed, for a few values of the parameter α (ranging from easy to hard classification), in the case $n = 20$ and $b = 4$. These plots indicate that the true error is small for large α, moderate near $\alpha = 1$, and near 0.5 for small α. Varying the parameter α therefore traverses the probability model space continuously from easy to difficult models. This is needed because we want to study the performance of error estimators under varying difficulty of classification.

Besides resubstitution and leave-one-out, we consider 10-repeated 4-fold cross-validation and the .632 bootstrap (additionally, simple 4-fold cross-validation and the *bias-corrected bootstrap* error estimator [20] are included on the website). For the 0.632 error estimator, $T = 100$ bootstrap samples are employed. Performance measures for resubstitution and leave-one-out are exact; they are computed using the analytical expressions developed in Section 6. For the other error estimators, which are randomized, performance measures are derived from a Monte-Carlo computation using 20,000 samples from each probability model (parameter value).

First, consider the case $n = 40$, and three representative values for the complexity, $b = 4, 8, 16$; in a setting where each feature is binary, this would correspond to classification using 2,3, and 4 features, respectively. For example, in functional genomics applications, gene expression is often binary (a promoter is either on or off). The cases considered here correspond to prediction using 2,3 or 4 genes.

Fig. 2 displays the bias, deviance variation and RMS of the error estimators considered here, as a function of the expected true error computed for a number of distinct models (i.e., distinct parameters α) of the parametric Zipf model. The parameters are selected such that the corresponding expected true errors are equally spaced. Easy discrimination is thus located on the left of the plots, whereas difficult discrimination is located on the right.

Several expected facts become readily apparent. Resubstitution is low-biased, whereas cross-validation is slightly high-biased, and the bias increases in each case with the

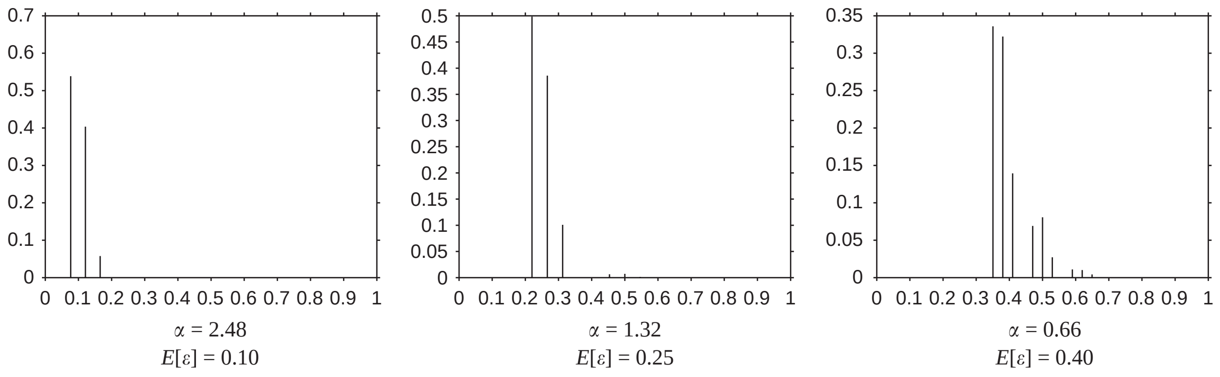

Fig. 1. Exact PDF of true error under the Zipf model, for a few values of the parameter α, in the case $n = 20$ and $b = 4$, with associated expected error rates.

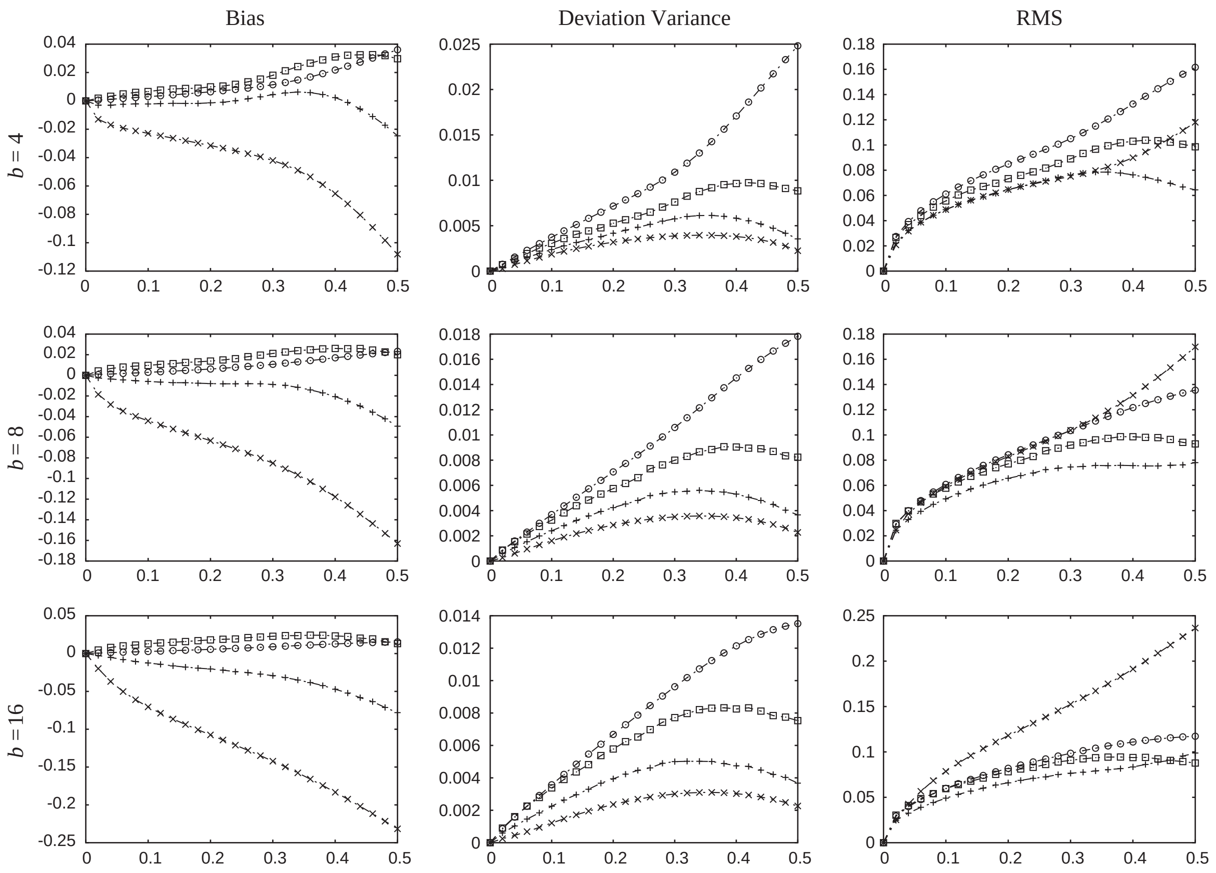

Fig. 2. Bias, deviation variance, and RMS for several error estimators vs. expected true error, for $n = 40$ and varying classifier complexity. Plot key: $\times$= resubstitution, $\circ$= leave-one-out, $\square$= 10-repeated 4-fold cross-validation, $+$= 0.632 bootstrap. The curves for resubstitution and leave-one-out are exact; the curves for the other error estimators are approximations based on Monte-Carlo computation.

difficulty of classification. Resubstitution has quite low variance, whereas cross-validation is highly variable. The bootstrap error estimator is generally the best-performing estimator, whereas leave-one-out is very variable, and often the worst-performing error estimator. What may be perhaps a bit surprising is that resubstitution is equivalent in overall

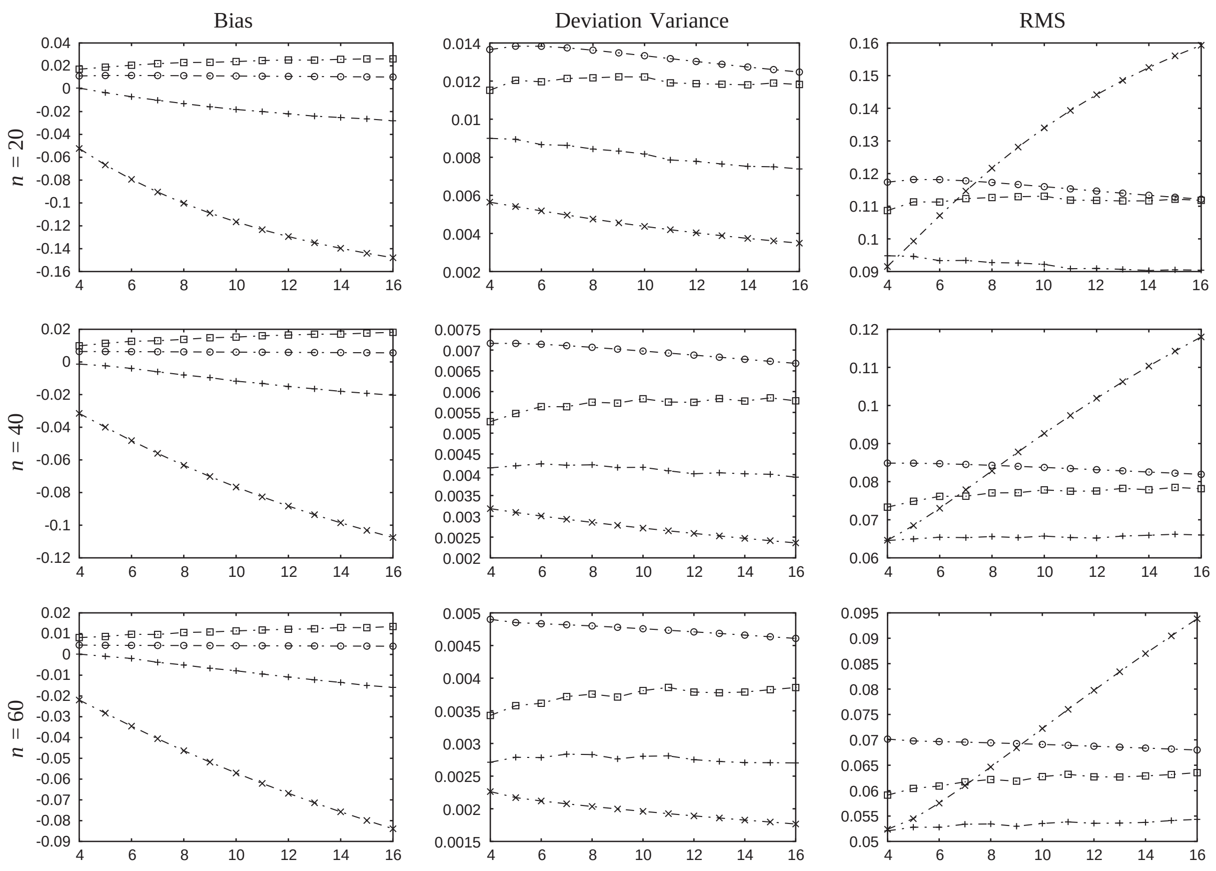

Fig. 3. Bias, deviation variance, and RMS for several error estimators vs. number of bins, for $E[\varepsilon]=0.2$ and varying number of samples. Plot key: $\times$= resubstitution, $\circ$= leave-one-out, $\square$= 10-repeated 4-fold cross-validation, $+$= 0.632 bootstrap. The curves for resubstitution and leave-one-out are exact; the curves for the other error estimators are approximations based on Monte-Carlo computation.

performance (as measured by the RMS) to the bootstrap estimator, which is much more expensive to compute, for the case $b=4$ (low complexity). We will have more to say about this in the next section.

In order to assess the performance of resubstitution and the remaining error estimators with respect to complexity, we display in Fig. 3 performance measures as a function of the number of bins, for $n=20, 40, 60$, and moderate classification difficulty: $E[\varepsilon]=0.2$. The RMS column shows that resubstitution is equivalent in performance to the bootstrap estimator for $b=4$, and is better than the cross-validation error estimator for low enough complexity. We can see that, as the number of samples increases (which alleviates the bias problem of resubstitution), then the classification complexity cut-off at which resubstitution beats the cross-validation estimator increases. It is interesting to note that leave-one-out is less biased than the other more complex cross-validation estimator, across the whole range of complexity displayed in the plot, and for all sample sizes. However, as the number of samples increases, its deviation variance becomes the worst among all error estimators.

To visualize the effect of sample size, we display in Fig. 4 performance measures as a function of n, for $b=4, 8, 16$, and again moderate classification difficulty: $E[\varepsilon]=0.2$. As expected, as sample size increases, there is a decrease in bias (in magnitude), deviance variation and RMS. We again can see that resubstitution is the least variable error estimator, whereas leave-one-out is the most variable one. On the RMS column, we can see that resubstitution is equivalent in performance to the bootstrap estimator for $b=4$, but its performance quickly degrades as the classifier complexity increases.

Some of the observations made above about resubstitution and leave-one-out are confirmed by plotting the PDFs of these error estimators, which are computed using the algorithm PDF described in Section 4 (see remarks made in Section 6 on how to adapt that algorithm for computing the PDFs shown here). As argued previously, we are really interested in the deviation PDFs, i.e., the PDFs of $\hat{\varepsilon}-\varepsilon$, where

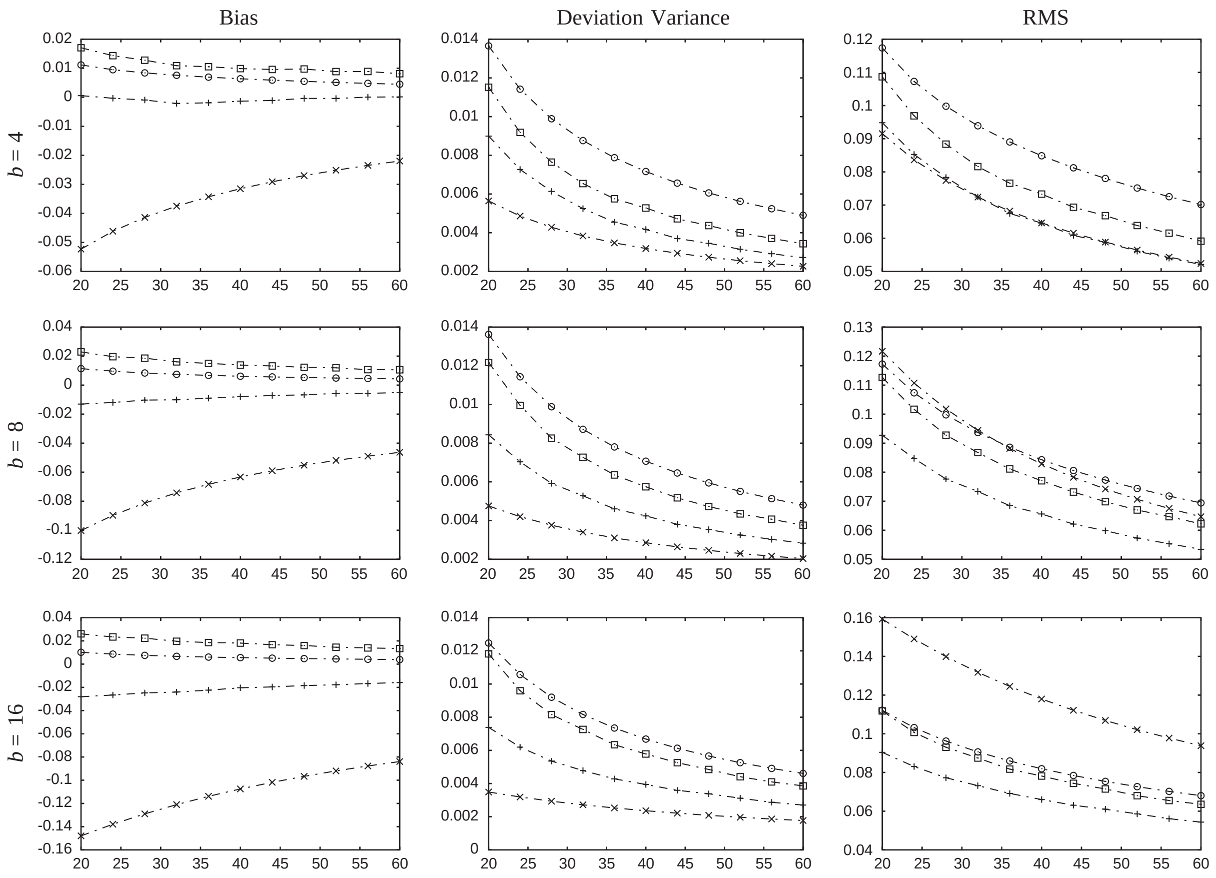

Fig. 4. Bias, deviation variance, and RMS for several error estimators vs. sample size, for $E[\varepsilon] = 0.2$ and varying number of bins. Plot key: $\times$= resubstitution, $\circ$= leave-one-out, $\square$= 10-repeated 4-fold cross-validation, $+$= 0.632 bootstrap. The curves for resubstitution and leave-one-out are exact; the curves for the other error estimators are approximations based on a Monte-Carlo computation.

$\hat{\varepsilon}$ is the error estimator under consideration. Fig. 5 displays the exact deviation PDFs of resubstitution and leave-one-out, for a few values of the expected true error, in the case $n = 20$ and $b = 4$ (the models here correspond to the same choices of α in Fig. 1). Note that the PDFs for resubstitution are skewed to the left (low-bias), whereas the PDFs for leave-one-out are approximately centered, but much more spread out than the corresponding ones for resubstitution. They are also quite skewed to the right, which is noteworthy. In particular, for moderate to hard classification, displayed in the middle and rightmost columns of Fig. 5, there are likely outcomes with high positive deviation, close to 0.5. There are even some outcomes—admittedly, improbable ones—that have positive deviation over 0.5!

8. Discussion

The results presented in the previous section provide a few definite conclusions and also evidence for some general trends. Owing to increasing low bias for an increasing number of bins, the RMS for resubstitution becomes prohibitive for around $b > 10$; however, for low-complexity classifiers, resubstitution becomes competitive with leave-one-out. With 8 bins, resubstitution performs almost as well as leave-one-out for sample size as low as $n = 20$, does slightly better than leave-one-out for $n = 40$, and outperforms leave-one-out for $n = 60$. Factoring in computation speed, this means that for binomial discrimination, such as that used with Boolean networks, resubstitution is preferable to leave-one-out when the transition functions for Boolean networks possess 3 variables or less, which is usually the case for Boolean gene regulatory networks [22,23]. Moreover, although we shall not go quantitatively into the matter here, taking into account complexity considerations with $n = 40$, it is prudent not to use more than 3 variables for binomial discrimination if one wants to be confident that the expected design error (owing to overfitting) is not excessive [1].

Whereas we have come to the preceding conclusions via exact representation of the RMS, it is interesting to note that

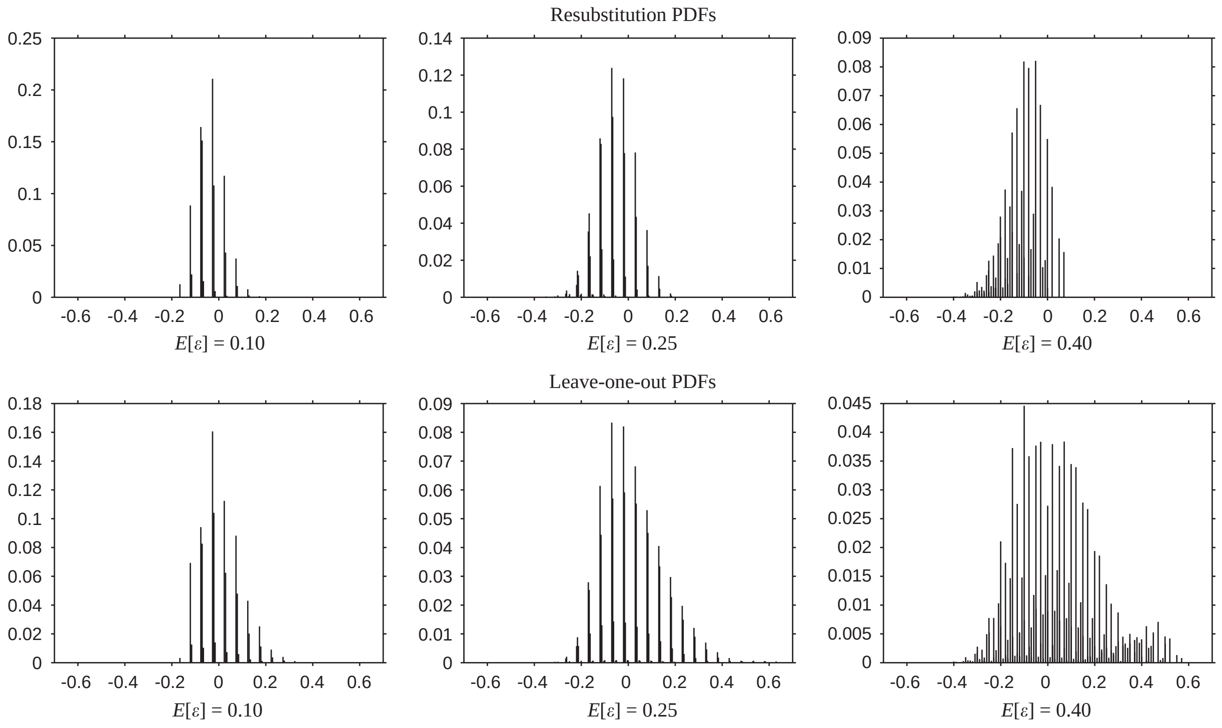

Fig. 5. Exact deviation PDFs of resubstitution and leave-one-out, for a few values of the expected true error, in the case $n = 20$ and $b = 4$.

similar, albeit less precise, conclusions can be gleaned from previously established RMS bounds (Devroye et al., 1996), for resubstitution,

$$\mathrm{RMS}[\hat{\varepsilon}_r] \leqslant \sqrt{\frac{6b}{n}}, \tag{44}$$

and for leave-one-out,

$$\mathrm{RMS}[\hat{\varepsilon}_l] \leqslant \sqrt{\frac{1+6e^{-1}}{n} + \frac{6}{\sqrt{\pi(n+1)}}}. \tag{45}$$

The strong point of these bounds is that they are distribution-free. Hence, it is not surprising that they are quite loose and not helpful for small samples. For instance, for $n = 100$, the leave-one-out bound exceeds 0.435. The bounds contain asymptotic information. For instance, the resubstitution bound goes to 0 much faster than the leave-one-out bound as $n \to \infty$, indicating that resubstitution is better than leave-one-out for large samples. Although the bounds are not practically useful for small samples, they do have the property that the leave-one-out bound exceeds the resubstitution bound for a sufficiently small number of bins, even in the case of small samples. Thus, the relation between the RMS bounds agrees with the relations we have discovered via exact representation of the RMS.

Generally, there are two basic problems with leave-one-out that negatively affect application for small samples. First, it possesses a prohibitively large deviation variance that gives rise to high RMS. Second, its PDF is skewed to the right and with significant probability the error estimator produces high outliers. The latter behavior is critical when predictability is being used to discover potential multivariate regulatory behavior [24]. In such situations, one wishes to avoid false negatives because these will be erroneously excluded from further bio-chemical analysis (as opposed to false positives, which will be thrown out upon wet-bench analysis).

Regarding the relationship between leave-one-out and the 10-repeated 4-fold cross-validation estimator, we note that, as the sample size increases, the deviation variance of leave-one-out decreases slower. This results in a corresponding slower decrease in RMS. The overall performance of 10-repeated 4-fold cross-validation is superior; however, in some cases the difference is so small that it cannot justify the high computational cost of this error estimator.

The 0.632 bootstrap error estimator is affected by the low-bias of resubstitution when complexity is high, since it incorporates the resubstitution estimate in their computation. The 0.632 bootstrap estimator is clearly superior to 10-repeated 4-fold cross-validation, but it is also the most computationally costly error estimator considered in this study.

Perhaps the most remarkable observation is that, for very low complexity classifiers (around $b = 4$), resubstitution becomes as accurate as the 0.632 bootstrap error estimator, despite the fact that resubstitution is typically much faster to compute (in some cases considered in [18], hundreds of times faster). In fact, we observe that for small sample sizes ($n < 30$), resubstitution can actually be more accurate than the 0.632 bootstrap estimator, for low to moderate true errors. The restriction to moderate true errors is not consequential because in real-world applications that employ robust feature selection algorithms, it is highly unlikely that true errors of over 0.3 will be encountered.

9. Conclusion

The main contribution made in this paper is an analytical formulation of performance measures of the resubstitution and leave-one-out error estimators for the discrete histogram rule. We also describe an algorithm to compute the PDFs of these estimators, or their deviation with respect to the true error. The algorithm is computationally intensive, but nevertheless effective for moderate sample size and classifier complexity. We have compared the performance of resubstitution and leave-one-out cross-validation against approximate performance measures of cross-validation and bootstrap error estimators, and the results indicate at least one perhaps surprising fact: resubstitution, a very simple, inexpensive, and sometimes neglected error estimator, can be the best option, even over the bootstrap error estimator, for very-low-complexity classifiers, such as those used in gene prediction. We believe that too scant attention has been paid to non-asymptotic analytical studies of error estimators in the literature, and we hope that this paper will provide motivation for further study.

Appendix

For a given $0 \leqslant m \leqslant n$, the algorithm to find all configurations in C_m (for either class) is described next. Let d_i is the number of samples in bin i, for $i = 0, \ldots, b-1$. We will interpret a bin configuration vector $[d_{b-1} \ldots d_1\, d_0]$ as a b-digit number t represented in base $(m+1)$. To represent a valid bin configuration, t must satisfy: (1) $m \leqslant t \leqslant m * (m+1)^{b-1}$, since these limits correspond to the "minimum" and "maximum" configuration vectors $[0 \ldots 0m]$ and $[m0 \ldots 0]$, respectively; (2) the sum of the digits of t must be m.

An algorithm to find all bin configurations is therefore to go through all numbers between m and $(m+1)^{b-1}$, checking for each number whether the sum of its digits in base $(m+1)$ is m. However, since one would have to go through $(m+1)^{b-1} - m + 1$ numbers, this algorithm can be very expensive computationally when m and b are moderate to large. We can reduce the computationally load significantly by exploiting the properties given in the next two results.

Proposition 1. *The total number of configurations in C_m is given by*

$$|C_m| = \binom{m+b-1}{b-1} \tag{46}$$

Proof. First, let us consider the number of configurations where no bin is empty. Each such configuration corresponds uniquely to a choice of $b-1$ spots among the $m-1$ spaces between the m samples, on which to erect "bin walls". The total number of configurations where no bin is empty is therefore $\binom{m-1}{b-1}$. Now each configuration in C_m corresponds uniquely to a configuration of $m+b$ samples distributed in b non-empty bins, by subtracting from the latter configuration one sample from each bin. Therefore, $|C_m| = \binom{m+b-1}{b-1}$, as required. □

Proposition 2. *The sum of the digits of a number t in base $(m+1)$ is a multiple of m if and only if t is a multiple of m.*

Proof. The digits of t in base $(m+1)$ are given by

$$d_i = \left\lfloor \frac{t}{(m+1)^i} \right\rfloor - (m+1) \left\lfloor \frac{t}{(m+1)^{i+1}} \right\rfloor \tag{47}$$

for $i = 0, \ldots, b-1$, where $\lfloor x \rfloor$ is the largest integer less or equal than the real number x. Direct summation leads to

$$s := \sum_{i=0}^{b-1} d_i = t - m \sum_{i=0}^{b-1} \left\lfloor \frac{t}{(m+1)^i} \right\rfloor . \tag{48}$$

Therefore, $s = t - km \approx t - s = km$, where k is an integer. In other words, $t - s$ is a multiple of m, so that s is a multiple of m if and only if t is. □

Note that Proposition 2 specializes to a well-known arithmetic fact in the case $m = 9$. As a corollary of Proposition 2, for the sum of the digits of a number t in base $(m+1)$ to be equal to m, it is necessary that t be a multiple of m. This means that only $(1/m)$ of numbers need to be checked for valid configurations in our algorithm, which provides considerable savings. Furthermore, the smallest and largest configurations are known, so the search needs to be done inside that interval. This can be done by initializing the list of configurations with the smallest one, proceed with the search and stop when $l - 1$ configurations have been found, where l is the total number of configurations, given by Proposition 1. The algorithm, in pseudo-code, consists of the following:

```
nc := $\binom{m+b-1}{b-1}$
t := $m$;
c := $1$;
```

```
initialize list of configurations with
  [0 ... 0$m$];
repeat
    find representation [$d_{b-1} \dots d_1 d_0$] of t
      in base($m+1$);
    if $\sum_{i=0}^{b-1} d_i = m$
        add [$d_{b-1} \dots d_1 d_0$] to list of
          configurations;
        c := c + 1;
    end if
    t := t+$m$;
until c = nc-1
add [$m$0 ... 0] to list of configurations.
```

We remark that this is not the most efficient possible algorithm to compute the list of configurations—for example, it is possible to compute very efficiently the list of configurations for $m = m_0$ in recursive fashion, based on the configuration lists for $m = 0, 1, \dots, m_0 - 1$. This of course requires that all configuration lists for $m = 0, 1, \dots, m_0 - 1$ have been previously computed and stored. In any event, considering that the configurations need to be computed only once and then can be reused indefinitely, the algorithm presented is simple and fast enough to serve our purposes.

References

[1] L. Devroye, L. Gyorfi, G. Lugosi, A Probabilistic Theory of Pattern Recognition, Springer, New York, 1996.

[2] N. Glick, Sample-based multinomial classification, Biometrics 29 (2) (1973) 241–256.

[3] M. Goldstein, W.R. Dillon, Discrete Discriminant Analysis, Wiley, New York, 1978.

[4] M. Hills, Discrimination and allocation with discrete data, Appl. Statist. 16 (3) (1967) 237–250.

[5] I.H. Witten, E. Frank, Data Mining, Academic Press, San Diego CA, 2000.

[6] K.F. Hirji, C.R. Mehta, N.R. Patel, Computing distributions for exact logistic regression, J. Am. Statist. Assoc. 82 (400) (1987) 1110–1117.

[7] M.A. van de Wiel, A. Di Bucchianico, P. van der Laan, Symbolic computation and exact distributions of nonparametric test statistics, Statistician 48 (4) (1999) 507–516.

[8] J.H. Klotz, The wilcoxon, ties, and the computer, J. Am. Statist. Assoc. 61 (315) (1966) 772–787.

[9] G.F. Hughes, On the mean accuracy of statistical pattern recognizers, IEEE Trans. Inform. Theory 14 (1) (1968) 55–63.

[10] G.F. Hughes, Number of pattern classifier design samples per class, IEEE Trans. Inform. Theory 15 (5) (1969) 615–618.

[11] L. Kanal, B. Chandrasekaran, On dimensionality and sample size in statistical pattern classification, Pattern Recognition 3 (3) (1971) 225–234.

[12] A.K. Jain, B. Chandrasekaran, Dimensionality and sample size considerations in pattern recognition practice, in: P.R. Krishnaiah, L.N. Kanal (Eds.), Classification Pattern Recognition and Reduction of Dimensionality, Handbook of Statistics, vol. 2, North-Holland, Amsterdam, 1982, pp. 835–856, Chapter 39.

[13] U.M. Braga-Neto, E.R. Dougherty, Classification, in: Genomic Signal Processing and Statistics, in: E. Dougherty, I. Shmulevich, J. Chen, Z.J. Wang (Eds.), EURASIP Book Series on Signal Processing and Communication, Hindawi publishing corporation, 2005.

[14] C.A.B. Smith, Some examples of discrimination, Ann. of Eugenics 18 (1947) 272–282.

[15] V.N. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.

[16] R.O. Duda, P.E. Hart, D. Stork, Pattern Classification, second ed., Wiley, New York, 2001.

[17] P.A. Lachenbruch, M.R. Mickey, Estimation of error rates in discriminant analysis, Technometrics 10 (1968) 1–11.

[18] U.M. Braga-Neto, E.R. Dougherty, Is cross-validation valid for microarray classification?, Bioinformatics 20 (3) (2004) 374–380.

[19] B. Efron, Bootstrap methods: another look at the jacknife, Ann. Statist. 7 (1969) 1–26.

[20] B. Efron, Estimating the error rate of a prediction rule: improvement on cross-validation, J. Am. Statist. Assoc. 78 (382) (1983) 316–331.

[21] G.K. Zipf, Psycho-Biology of Languages, Houghton-Mifflin, Boston, 1935.

[22] S. Kauffman, The Origins of Order: Self-Organization and Selection in Evolution, Oxford University Press, Oxford, 1993.

[23] I. Schmulevich, et al., Probabilistic Boolean networks: a rule-based uncertainty model for gene-regulatory networks, Bioinformatics 18 (2002) 261–274.

[24] S. Kim, et al., A general framework for the analysis of multivariate gene interaction via expression arrays, Biomed. Opt. 5 (4) (2000) 411–424.

About the Author—EDWARD DOUGHERTY is a professor in the Department of Electrical Engineering at Texas A&M University in College Station. He holds a Ph.D. in mathematics from Rutgers University and an M.S. in Computer Science from Stevens Institute of Technology. He is author of twelve books, editor of four others, and author of more than one hundred and sixty journal papers. He is an SPIE fellow, is a recipient of the SPIE President's Award, and has served as editor of the Journal of Electronic Imaging for six years. Prof. Dougherty has contributed extensively to the statistical design of nonlinear operators for image processing and the consequent application of pattern recognition theory to nonlinear image processing. His current research is focused in genomic signal processing, with the central goals being to model genomic regulatory mechanisms for the purposes of therapy and to develop small-sample pattern-recognition methods for expression-based diagnoses. He is Director of the Genomic Signal Processing Laboratory at Texas A&M University, Director of the Division of Computational Biology at the Translational Genomics Research Institute, and Adjunct Professor in the Department of Pathology of the University of Texas M. D. Anderson Cancer Center.

About the Author—ULISSES BRAGA-NETO received the Baccalaureate degree in Electrical Engineering from the Universidade Federal de Pernambuco (UFPE), Brazil, in 1992, the Master’s degree in Electrical Engineering from the Universidade Estadual de Campinas, Brazil, in 1994, the M.S.E. degree in Electrical and Computer Engineering and the M.S.E. degree in Mathematical Sciences, both from The Johns Hopkins University, in 1998, and the Ph.D. degree in Electrical and Computer Engineering, from The Johns Hopkins University, in 2001. He was a Post-Doctoral Fellow at the University of Texas MD Anderson Cancer Center and a Visiting Scholar at Texas A&M University, from 2002 to 2004. He is currently a Researcher at the Aggeu Magalhães Research Center of the Osvaldo Cruz Foundation, Brazilian Ministry of Health. His research interests include Computational Biology, Pattern Recognition, and Image Analysis.