

Fads and Fallacies In the Name of Small-Sample Microarray Classification

Ulisses Braga-Neto

Virology and Experimental Therapy Laboratory,
Aggeu Magalhães Research Center - CPqAM/FIOCRUZ,
Recife, PE 50.670-420 Brazil

E-mail address: ub@ieee.org

July 21, 2006

REVISED VERSION

INTRODUCTION

In his landmark book *Fads and Fallacies In The Name of Science*, Martin Gardner wrote the following words on pseudoscience [10, p. 15]: “The amount of lost energy that has been wasted on these lost causes is almost unbelievable. It will be amusing — at times frightening — to witness the grotesque extremes to which deluded scientists can be misled, and the extremes to which they in turn can mislead others.” As has been already cautioned elsewhere [3, 8, 13, 19], we are unfortunately facing a challenge in genomic signal processing that, though perhaps not on the same scale, is not unlike what has been going on with pseudoscience. Here, we are facing the careless, unsound application of classification methods to small-sample microarray data, which has generated a large volume of publications and an equally large amount of unsubstantiated scientific hypotheses. For example, it is reported in [18] that reanalysis of data from the seven largest published microarray-based studies that have attempted to predict prognosis of cancer patients reveals that five of those seven did not classify patients better than chance!

Since the mid-1990’s, the field of genomic signal processing has exploded due to the development of DNA microarray technology, which made possible the measurement of mRNA expression of thousands of genes in parallel, in a single assay. By that time, researchers had developed a vast body of knowledge in classification methods. Those tools are suited in principle to the analysis of DNA microarray data; naturally, they were immediately brought to bear on this problem. However, microarray data is characterized by extremely high dimensionality and comparatively small number of data points. This makes microarray data analysis quite unique. While data matrices in most traditional applications, such as clinical studies, are “wide” matrices, with the number of cases exceeding the number of variables, in microarray studies, data matrices are “tall,” with the number of variables far exceeding the number of cases. This situation is commonly referred to as a *small-sample* scenario. It means that application of traditional pattern recognition methods must be carried out with judgment, in order to avoid pitfalls. In particular, “popular” methods that have succeeded in other application areas may fail in genomic signal processing applications, if they are carelessly applied.

There are excellent reviews on classification-related methodology [11, 14, 15]. In this paper, our purpose is not to provide another general review of the subject, but rather to highlight those topics that we believe have displayed a record of misunderstanding and erroneous usage

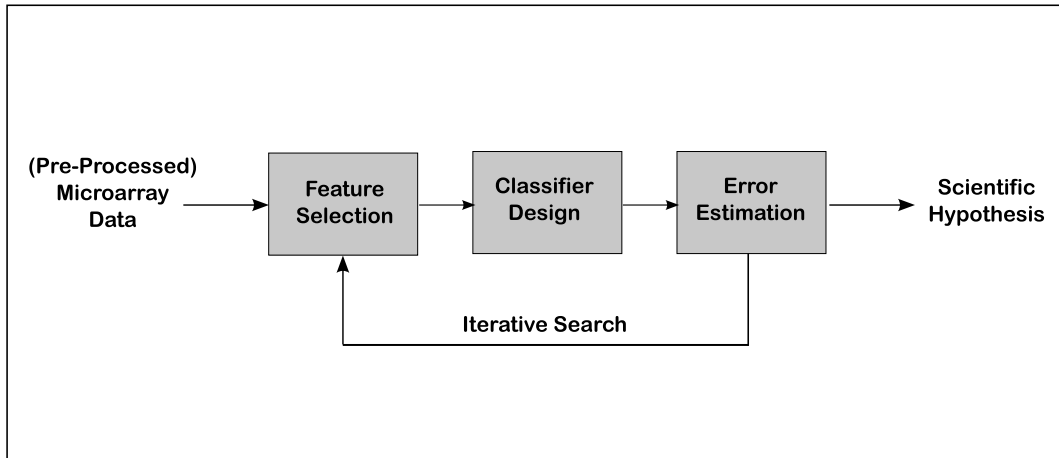


Figure 1: Microarray Classification Pipeline.

in applications of genomic signal processing. We will focus on the related subjects of feature selection, classifier design, and error estimation, which together form a microarray classification pipeline (see Figure 1), discussing along the way some of the most common “fads” and “fallacies” regarding classification methods that are routinely applied in analysis of small-sample microarray data.

FADS AND FALLACIES IN CLASSIFIER DESIGN

Pitfalls in small-sample classifier design arise mostly from the related issues of overfitting, high-dimensionality, and asymptotic performance guarantees.

Fad/Fallacy 1

Complex classification rules are better than simple ones.

The use of complex classification rules, such as neural networks with a large number of nodes, or classification trees with a large number of branches, seems enticing for two reasons: one is that the optimal Bayes classifier can be better approximated by such rules, and the other is that the classifiers that are produced seem to account better for the training data; that is, the *apparent error*, or the rate of misclassified training samples, is close to zero. Rumor has it that one practitioner in the field used to recommend the use of 1000-node neural networks fine-tuned to display zero apparent error!

In fact, this is an insidious mistake. To see this, consider the class \mathcal{C} of all classifiers that can be designed with a given classification rule, and let $\psi_{\mathcal{C}} \in \mathcal{C}$ be the optimal classifier in \mathcal{C} , with

optimal error rate $\varepsilon_{\mathcal{C}}$. The *approximation error*, given by $\Delta_{\mathcal{C}} = \varepsilon_{\mathcal{C}} - \varepsilon_d$, where ε_d is the optimal (unconstrained) Bayes error rate, reflects how well the classification rule can approximate the optimal (unconstrained) Bayes classifier. One can in principle make the rule ever more complex, and the class \mathcal{C} ever larger, so that eventually the Bayes classifier is contained in it, and $\Delta_{\mathcal{C}} = 0$. However, the other side of the coin is how far from $\psi_{\mathcal{C}}$ the actual classifier designed from the data is. Let us denote this classifier by $\psi_{n,\mathcal{C}} \in \mathcal{C}$, with error rate $\varepsilon_{n,\mathcal{C}}$. The *design error* $\Delta_{n,\mathcal{C}} = \varepsilon_{n,\mathcal{C}} - \varepsilon_{\mathcal{C}}$ reflects how good a job we can do with the available data to design the best possible classifier in \mathcal{C} . We would like both the approximation error $\Delta_{\mathcal{C}}$ and the design error $\Delta_{n,\mathcal{C}}$ to be small. Unfortunately, this is not generally possible. This can be viewed as a manifestation of the *bias-variance dilemma*, a general phenomenon in Statistics. One can usually control bias or variance, but not both. The dilemma in our case is between having a complex classification rule and a small $\Delta_{\mathcal{C}}$, controlling bias, or having a simple classification rule and small $\Delta_{n,\mathcal{C}}$, controlling variance.

For large samples, the design problem is minimized ($\Delta_{n,\mathcal{C}}$ tends to be small), and the most important objective is to have a small approximation error $\Delta_{\mathcal{C}}$, so in this case the use of complex classification rules is warranted (perhaps even the 1000-node neural network, if one has a truly *big* number of samples). However, in small-sample settings, which are prevalent in Genomic Signal Processing applications, there is plenty of evidence that the trade-off is tilted in the other direction. The design error can be large and classifier variance is definitely the most important issue to control.

A bound on the design error can be obtained as a function of the size of class \mathcal{C} , i.e., the complexity of the classification rule. The latter can be measured by the *VC (Vapnik-Chervonenkis) dimension* [7]. Briefly, the VC dimension corresponds to the largest n such that a set of n points can be *shattered*, i.e., completely partitioned (in 2^n ways), by the classification rule. Therefore, the larger the VC dimension is, the more finely the classification rule can “cut up” the feature space. Suppose the classification rule works by picking the classifier with minimum apparent error in the class \mathcal{C} (which is in line with one’s intuition of a classifier that “accounts for” the data). A fundamental theorem by Vapnik and Chervonenkis [7] states that the expected design error satisfies the following bound:

$$E[\Delta_{n,\mathcal{C}}] \leq 8 \sqrt{\frac{V_{\mathcal{C}} \log n + 4}{2n}} \quad (1)$$

where $V_{\mathcal{C}}$ is the VC Dimension of \mathcal{C} . It can be seen that the bound will be tight if $V_{\mathcal{C}}$ is small, or if the sample size n greatly exceeds $V_{\mathcal{C}}$. Simple classification rules tend to have small VC dimension, obtaining tight bounds in (1) for modest sample sizes n . For example, it can be shown that the VC dimension of a linear classifier is $d+1$ [7], where d is the number of variables, whereas the VC dimension of a neural network with k nodes in one hidden layer has the lower bound $V_{\mathcal{C}} \geq 2 \lfloor \frac{k}{2} \rfloor d$, where $\lfloor x \rfloor$ denotes the greatest integer less or equal than x [1]. If k is even, this bound simplifies to $V_{\mathcal{C}} \geq dk$. This means that the VC dimension of a neural network is at least around k times larger than the VC dimension of a linear classifier (with $k = 1000$ in the case of our 1000-node neural network). While it can be argued that this result holds for a neural network tuned by the apparent error, it is shown in [7, Chap. 14] that for a guaranteed small error, one must have $n \gg V_{\mathcal{C}}$, regardless of the way of tuning the parameters. It is also worth mentioning that Vapnik estimates a problem to be in a small-sample situation if the ratio of number of samples to the VC dimension is smaller than 20 [22, p. 219]. Therefore, increasing complexity is counterproductive unless there is a large number of training samples available. Otherwise, one could easily end up with a very bad classifier, with large design error, whose apparent error looks very small!

The problem of large design error with small samples can be understood in terms of *overfitting*. A complex, high-VC classification rule will typically have too many tunable parameters for the available number of data points, resulting in a classifier that “learns” the training data, and has therefore bad performance on future data. A way of assessing overfitting and classifier variance is by employing a simple “jackknife” approach: removing one data point from the training data at a time and observing the impact on the designed classifiers. There should be of course larger impact on a classifier produced by an overfitting classification rule than one produced by a non-overfitting rule. Figure 2 displays classifiers produced by three different classification rules: Linear Discriminant Analysis (LDA), 3-Nearest Neighbors, and Classification Trees (CART) with a stopping rule to reduce overfitting [4]. The original classifier is obtained from 20 training samples, and the other classifiers are obtained from the same data, but with one sample point removed. The “deleted” classifiers displayed in Figure 2 correspond to those cases where change of the decision regions was most accentuated. We can see that the LDA classifiers change slightly, the 3NN classifiers change more, while for the CART classification rule, even with the stopping rule, the changes are radical. This “jackknife” procedure ranks LDA, 3NN, and CART, in this

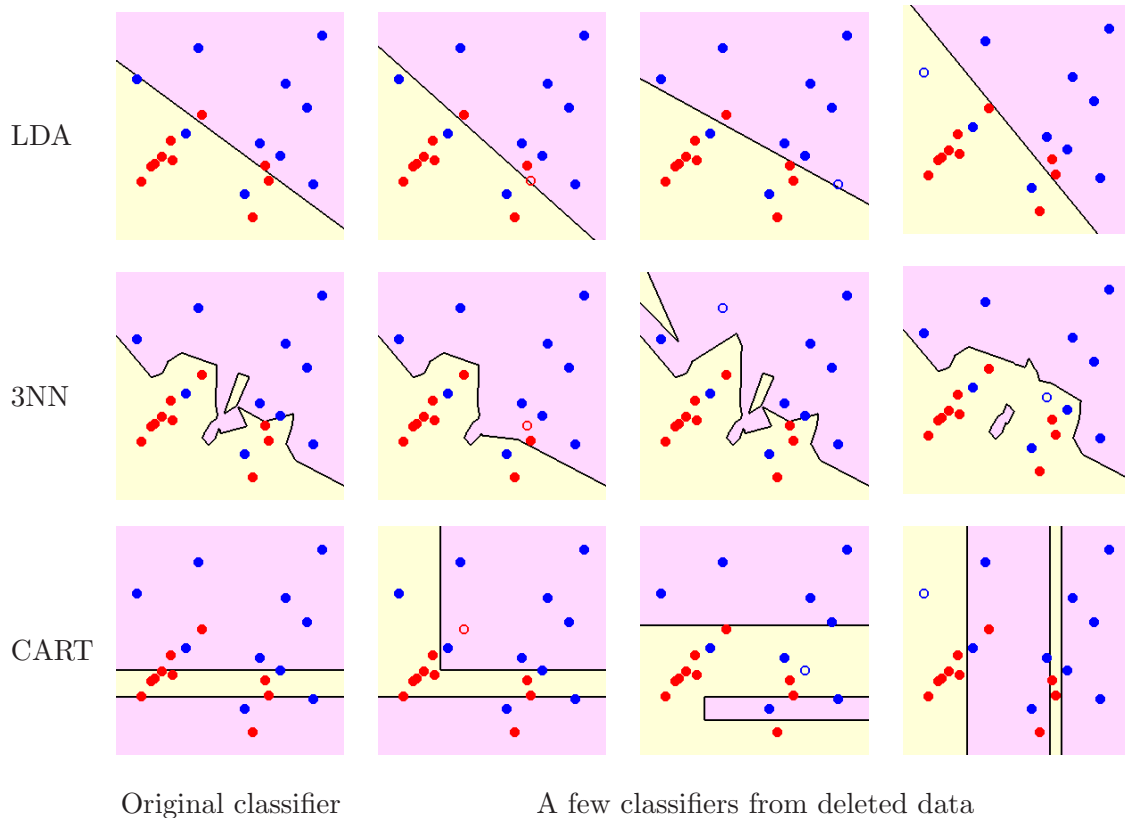


Figure 2: Original classifier and a few classifiers obtained after one sample point is deleted. Different colors depict the samples and the decision regions according to the respective classes. Deleted sample points are represented by unfilled circles.

order, with respect to sensitivity of the classification rule to the training data, which is indicative of the potential of overfitting.

Fad/Fallacy 2

Adding more variables leads to better classifier performance.

This is unfortunately a widespread misconception. A larger number of variables means a larger-dimensional feature space, which accentuates the small-sample problem. There are more ways to shatter points in a higher-dimensional space, as there is more “room” to work with, which effectively increases the VC dimension of classification rules, and the data requirement. For example, as we saw previously, the VC dimension of linear classifiers and the lower bound of the VC dimension of neural networks increase with d . This generally increases the expected design error $E[\Delta_{n,C}]$ as well, affecting classifier performance.

As the number of variables increases, the typical observed behavior is an initial decrease

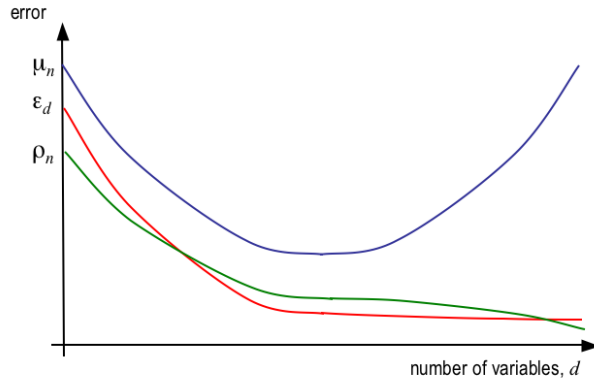


Figure 3: Peaking phenomenon.

in expected classification error, followed by an increase after a certain point. And with small-samples, it is a fact that this optimum is achieved for a small number of variables. See Figure 3 for an illustration (here, μ_n , ϵ_d , and ρ_n denote respectively the expected true error, the Bayes error, and the expected apparent error). This “peaking phenomenon” is also known as the “curse of dimensionality,” or the “Hughes phenomenon” (after G.F. Hughes, who first demonstrated this phenomenon, for discrete classifiers [12]).

The attractiveness of using classifiers based on many variables is the same as of using complex classification rules: they apparently produce small error. An example of improperly using too many variables in classification is the microarray-based predictor of metastasis in [21], a well-known study in the microarray literature (see also the follow-up study [20]). The authors used 78 microarray samples to obtain a classifier based on 70 genes! That is barely one sample point per variable. We did the following experiment: using the 70-gene data publicly available from the journal website, and using the 295 microarrays from the follow-up study [20], we did feature selection based on a Bayes-error approximation score (see [4, Section 2.5] for more details), arriving at the top 2 discriminating genes out of the 70. Those are not the top two genes for the entire data, but only the estimated top two genes among the 70 that were published. We designed a simple LDA classifier based on these two genes, and the result is displayed in Figure 4. Since this is not a small-sample case (there are 295 samples to 2 variables, with a low-VC-dimension classification rule), a reasonable estimate for classification error is the apparent error itself (more on this will be discussed in the error estimation Section below). This comes out to be $52/295 =$

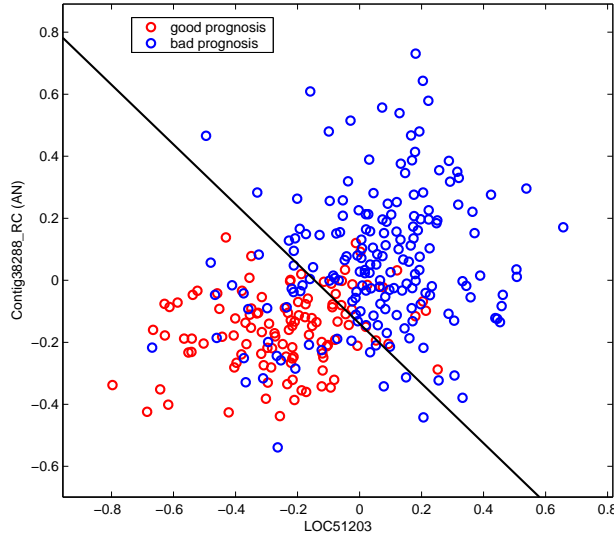


Figure 4: Two-gene classifier for the data in [20, 21]. The two genes were selected from the 70-gene published signature.

17.6 %, which is not bad. Now, estimating the apparent error rate based on Table 2 in [20], which reports classification results for 180 out of the 295 patients based on their 70-gene signature, gives a value of $68/180 = 37.7\%$ (this is also the error rate reported in [18]). This is a reliable estimate, since the 180 samples are new data points not involved in design of the 70-gene classifier. Thus, we have produced a 2-gene classifier at 17% error rate, compared to the proposed 70-gene classifier at 37% error rate, using 2 among the 70 genes.

Fad/Fallacy 3

Consistent classification rules are better than non-consistent ones.

A classification rule is said to be *consistent* if classifier error converges in the mean to the optimal Bayes error, that is, $E[\varepsilon_n] \rightarrow \varepsilon_d$, as $n \rightarrow \infty$. A *universally consistent* rule is consistent regardless of the particular feature-label distribution of the problem. This fits our intuition about increasing sample sizes, and so this property appears to be mandatory. However, what happens at $n \rightarrow \infty$ does not constrain what happens at small n . Examples of universally consistent rules include complex classification rules such as neural networks and classification trees. In contrast, simple classification rules such as LDA or k -nearest neighbor (with fixed k) are not universally consistent. The simple non-consistent rules often produce better classifier errors at small n than the more complex consistent rules. See Figure 5 for an illustration.

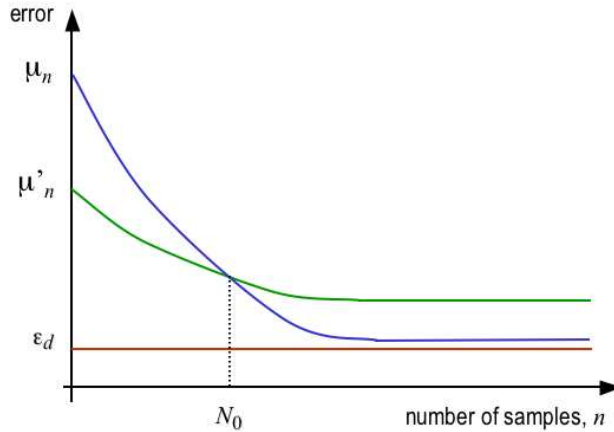


Figure 5: The expected error of a consistent rule (blue curve) tends to the Bayes error, while that of a non-consistent rule (green curve) does not. However, in the small sample region, the non-consistent rule performs better.

We can demonstrate this fact by means of simulation (this simulation also illustrates points made previously regarding complexity and number of variables). For this we use the data from [20]. We consider four basic cases, corresponding to $d = 2, 3, 4, 5$ among the top genes reported in [20]. In each case, 1000 data sets of size ranging from $n = 20$ to $n = 120$, in steps of 5, were drawn independently from the pool of 295 microarrays. For each data set, a classifier was designed to predict bad vs. good prognosis, according to the labelling given in [20]. We considered seven classification rules: linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), nearest-mean classification (NMC), 1-nearest neighbor (1NN), 3-nearest neighbor (3NN), CART with a stopping rule of a minimum of six samples in a leaf, and a neural network (NNET) with 4 nodes in the hidden layer. The classification error was approximated in each case by means of a holdout estimator, whereby the $295 - n$ sample points not drawn are used as an independent test set (this is a good approximation to the true error, given the large test sample). The errors for the 1000 independent sample sets were averaged to provide a Monte-Carlo estimate of the expected error for the classification rule. Fig. 6 displays the four plots, one for each dimensionality considered.

We can see that the non-consistent, simple rules LDA and 3NN do a very good job. QDA (also non-consistent, and more complex than LDA) does a very good job for the largest sample

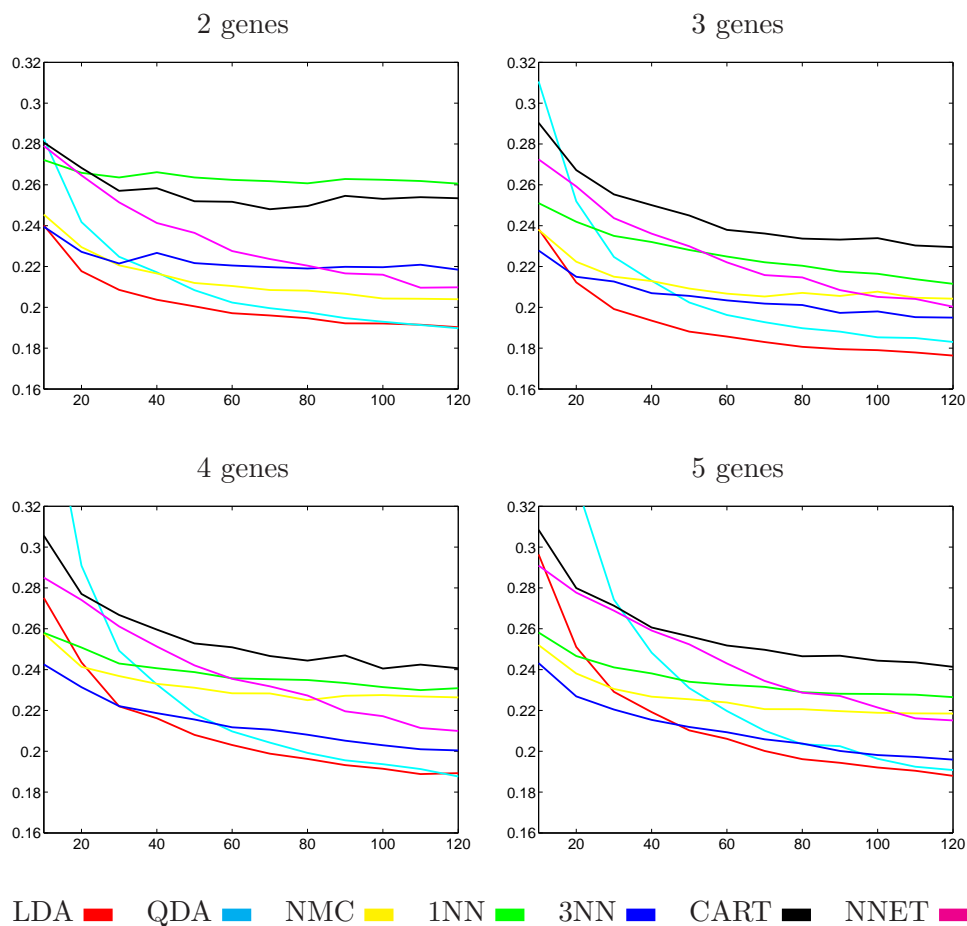


Figure 6: Expected error versus sample size for several classification rules and number of genes.

sizes, but its performance degrades quickly for the smallest sample sizes. NMC is a very simple, non-consistent rule, simpler than LDA, but does a very credible job, given its simplicity, and it can actually do quite well for very small sample sizes, as compared to the other classification rules. The neural network performed well for 2 variables, but its performance quickly degrades as the number of genes increases, due to the high complexity of this classification rule, which leads to overfitting. CART and 1NN do not perform well with this data set, due to severe overfitting (even with the regularizing stopping criterion used for CART). Note that the performance of some rules, especially QDA, degrades as the number of variables increases.

FADS AND FALLACIES IN FEATURE SELECTION

The feature selection problem is to select a subset of features from the overall set of features (genes) that provides an optimal classifier with minimum error. We discuss below pitfalls in

feature selection that arise from avoiding the exhaustive search of all feature sets.

Fad/Fallacy 4

All features of interest can be found by doing univariate statistical tests.

The majority of papers in the GSP literature dealing with functional genomics are based on using statistical tests, or even “fold-changes” (which is definitely not recommended), to find “differentially-expressed genes.” Classical statistical tests, such as t-tests, and fold changes consider one gene at a time and select those that display a pattern of difference between the conditions. However, there are situations in which an interesting gene displays a pattern of difference in conjunction with other genes, but does not by itself. These interesting genes cannot be picked out by univariate statistical tests. For example, in the microarray-based study of [16], four classes of glioma were considered: Glioblastoma Multiforme (GM), Anaplastic Astrocytoma (AA), Anaplastic Oligodendroglioma (AO), and Low-Grade Oligodendroglioma (OL). It so happens that in the 598 quality-filtered genes in their study, there were plenty of genes that displayed univariate patterns of differential expression for GM and OL, but not for AA and AO (which are incidentally the two most heterogenous classes). Nevertheless, the researchers were able to find many interesting genes characteristic of AA and AO expression by considering linear discrimination with subsets of two and three genes. See Figure 7 for an illustration. Here, the combination of three genes creates a distinctive multivariate signature for Anaplastic Oligodendroglioma that none of the genes alone could provide.

Fad/Fallacy 5

The optimal feature set can be found by non-exhaustive search.

Unfortunately, that is not true. As shown in [6], any ordering of feature sets in terms of the Bayes error is possible (provided that the monotonicity property of the Bayes error with respect to feature size is satisfied), so that selection of the optimal feature set must be done by exhaustive search. Actually, the monotonicity of the Bayes error with respect to feature size can be cleverly taken advantage of, avoiding the exhaustive search, resulting in the so-called *branch-and-bound* algorithm [17]. However, there are two caveats: firstly, in the case of selection of the best feature set of a given fixed size k (a common application), monotonicity cannot be taken advantage of, and there is no way of avoiding the exhaustive search of all feature sets of size k . Secondly, in practice, the Bayes error is not known, and an estimate of the error of a designed classifier

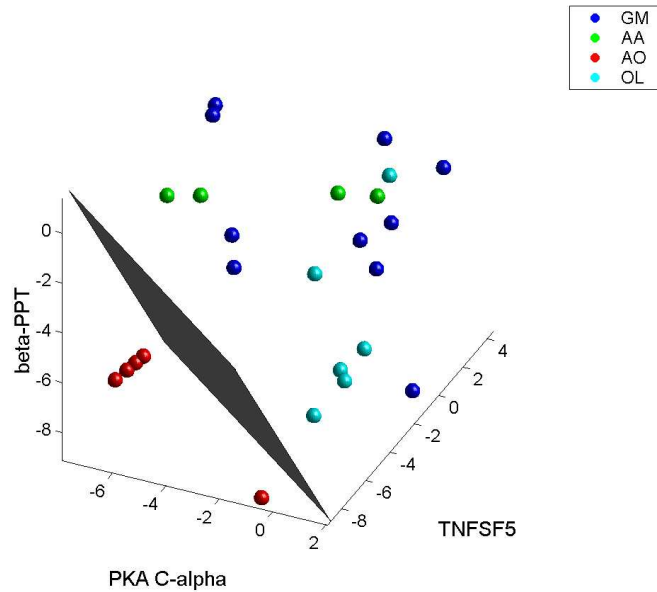


Figure 7: This three-dimensional signature for Anaplastic Oligodendroglioma cannot be derived by considering the individual genes separately; a true multivariate approach is needed (reprinted from [16], with permission).

based on the feature set in question is used instead, so there is no monotonicity property, and the Branch-and-Bound algorithm and its variations *cease to be optimal*. In practice, all feature selection algorithms are sub-optimal, and even exhaustive search cannot guarantee the discovery of the true optimal feature set — even though it is superior to algorithms based on non-exhaustive searches — due to the fact that only estimates of true classification error are available.

FADS AND FALLACIES IN ERROR ESTIMATION

Error estimation is a key aspect of microarray classification, as it impacts both classifier design and feature selection. As the true error rate of a classifier depends on the feature-label distribution of the data, it is usually unknown, so that accurate and fast error estimation methods are essential. Unfortunately, error estimation is a subject beset by misunderstanding and “rules of thumb,” which create pitfalls in small-sample settings [3]. Most popular statistical pattern recognition textbooks bring a handful of pages, or none at all, on this subject, which helps add to the confusion.

Fad/Fallacy 6

Using an independent test set is always the best approach.

This is true only if one has an abundance of data, which is very rare in GSP applications. Let the holdout error estimator be denoted by $\hat{\varepsilon}_{n,m}$, where n and m are the number of sample points in the training and test data, respectively. The classifier is designed on the training data, and the estimated error rate is the proportion of errors observed on the test data. This estimator is unbiased in the sense that, given the training data S_n , $E[\hat{\varepsilon}_{n,m}|S_n] = \varepsilon_n$, and thus $E[\hat{\varepsilon}_{n,m}] = E[\varepsilon_n]$. The *internal variance* of this estimator (i.e., the variance given the data) can be bounded as follows [7]:

$$V_{\text{int}} = E[(\hat{\varepsilon}_{n,m} - \varepsilon_n)^2|S_n] \leq \frac{1}{4m} \quad (2)$$

which tends to zero as $m \rightarrow \infty$. Moreover, it can be shown by using the conditional variance formula $\text{Var}(X) = E[\text{Var}(X|Y)] + \text{Var}(E[X|Y])$ that the full variance of the holdout estimator is given by $\text{Var}(\hat{\varepsilon}_{n,m}) = E[V_{\text{int}}] + \text{Var}[\varepsilon_n]$. Thus, provided that m is large, so that V_{int} is small (this is guaranteed by (2) for large enough m), the variance of the holdout estimator is approximately equal to the variance of the true error itself, which is typically small, provided n is not too small.

The problem here is that, in practice, one often does not have a large enough data set to be able to make both n and m large enough. For example, in order to get the standard-deviation bound in Eq. (2) down to an acceptable level, say 0.05, it would be necessary to use 100 test samples, which is not available in most GSP applications. For another perspective, consider Fig. 5 at small sample sizes: in this region, any change in the value of n creates a significant change in the expected error rate (one might even *define* the small-sample region in terms of this property). If one happens to be in the small-sample region, any amount of data to be held out for testing will produce an unacceptable increase in classification error. Therefore, for a large class of applications where samples are at a premium, holdout error estimation is effectively ruled out. In such cases, one must use the same data for training and testing.

Fad/Fallacy 7

Cross-validation is unbiased, and therefore the estimator of choice in GSP applications.

Cross-validation is perhaps the biggest “fad” in the practice of Genomic Signal Processing. Everyone seems to be ready to apply it in any situation and is completely confident about it. We should first note that cross-validation is not really unbiased. A k -fold cross-validation estimator

$\hat{\varepsilon}_{n,k}^{CV}$ is unbiased as an estimator of $E[\varepsilon_{n-n/k}]$, i.e., $E[\hat{\varepsilon}_{n,k}^{CV}] = E[\varepsilon_{n-n/k}]$. Therefore, it is *not* in general unbiased as an estimator of $E[\varepsilon_n]$. In fact, it will have a positive bias, since typically $E[\varepsilon_{n-n/k}] > E[\varepsilon_n]$. This effect can be observed in Fig. 5; leaving data out corresponds to moving left on the plot, and in the small-sample region this may incur in sizeable positive bias, especially for large values of k .

Another fad involved here is the excessive importance placed on unbiasedness itself. Unbiasedness of an error estimator $\hat{\varepsilon}_n$ on its own does not guarantee accuracy; actually, it only guarantees accuracy in a mean sense: $E[\hat{\varepsilon}_n] = E[\varepsilon_n]$. However, the expectation is taken over all possible training data sets, and the ‘‘point estimate’’ $\hat{\varepsilon}_n$ may be far from the true conditional error rate ε_n . In fact, this may happen with high probability and the estimator still be unbiased. There is plenty of evidence that in small-sample scenarios, estimator variance $\text{Var}(\hat{\varepsilon}_n)$ is the salient issue regarding error estimation. Unbiasedness is of limited use if the estimate corresponding to a given sample can be often far from the actual error value, due to high variance, which is usually what happens with cross-validation estimators in small-sample settings [3].

These points about cross-validation can be illustrated with the help of simulation experiments based on synthetic data (see also [3]). We consider seven error estimators: resubstitution (resub), leave-one-out (loo), stratified 10-fold cross-validation with 10 repetitions (cv10r), the .632 bootstrap (b632) [9], Gaussian bolstered resubstitution (bresub), Gaussian semi-bolstered resubstitution (sresub) and Gaussian bolstered leave-one-out (bloo) [2]. For each bootstrap estimator, $T = 100$ balanced bootstrap samples were employed. We employ three classification rules (in order of complexity, as we saw previously): Linear Discriminant Analysis (LDA), 3-Nearest Neighbor (3NN), and Classification and Regression Trees (CART). The experiments assess the empirical distribution of $\hat{\varepsilon}_n - \varepsilon_n$ for each error estimator $\hat{\varepsilon}_n$. This distribution measures the difference between the estimated error and the true error of the designed classifier. Deviation distributions are from 1000 independent data sets drawn from Gaussian-mixture models with equal variances for each class. Plots of beta-density fits of the empirical deviation distribution for sample size $n = 20$ and dimensionality $d = 2$ are displayed in Fig. 8. The narrower and taller the distribution, the smaller the variance of the deviation, whereas the closer its mean is to the vertical axis, the less biased the error estimator is. Note the positive bias and extremely large variance displayed by cross-validation estimators, especially in the case of the more complex clas-

sification rules, 3NN and CART. Resubstitution, leave-one-out, and even 10-fold cross-validation are generally outperformed by the bootstrap and bolstered estimators. Bolstered resubstitution is competitive with the bootstrap, in some cases outperforming it; this despite the fact that bolstered resubstitution is typically much faster to compute than bootstrap error estimators [2]. The bolstered leave-one-out is generally much more variable than the bolstered resubstitution estimator, but it displays less bias.

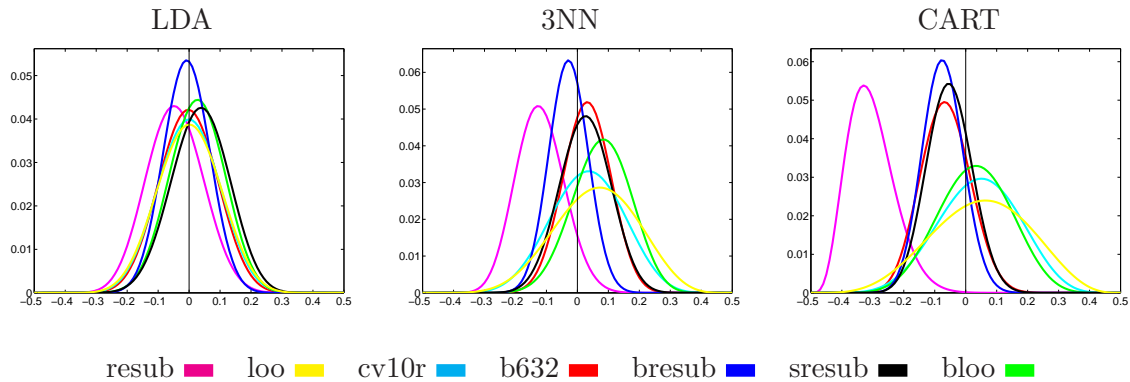


Figure 8: Beta fits to empirical deviation distribution in error estimation simulation.

Fad/Fallacy 8

Resubstitution is useless, except in large-sample and asymptotic studies.

Resubstitution has been often dismissed as a non-option in error estimation due to its notorious optimistic bias. However, in small-sample problems, it is sometimes advantageous to tolerate some bias in exchange for small variance. There is evidence that resubstitution is a small-variance estimator (compare its variance to that of cross-validation in Figure 8). Resubstitution is also a lightning-fast estimator, which is advantageous in feature selection. Could not there be practical situations where the use of resubstitution is actually recommended? The answer is yes. The optimistic bias of resubstitution is a function of classifier complexity, as we have mentioned in connection with the apparent error of complex rules. For tight, simple classification rules, the bias of resubstitution is small. This can be seen in Figure 8 for the case of LDA, but we give in the following an even more remarkable example.

Consider a discrete classification problem, where the predictor variables X_1, \dots, X_d each take on finite number b_i of values. This is the case of classification with quantized gene expression

values, for example. The predictors as a group take on values in a finite space of $b = \prod_{i=1}^p b_i$ possible states. A bijection exists between this finite state-space and the sequence of integers $1, \dots, b$. Therefore, we may assume a single predictor variable X taking on values in the set $X \in \{1, \dots, b\}$. The value b is the number of “bins” into which the data is categorized — it provides a direct measure of the complexity of the discrete classification rule; in fact, it can be shown that the VC dimension in this case is equal to b [7]. Figure 9 displays RMS values (which combine estimator bias and variance) for a simulation using the simple discrete histogram (majority-voting) rule and a Zipf power-law model (for more details, see [5]). The results displayed correspond to complexity (number of bins) ranging between $b = 4$ and $b = 16$, sample sizes of $n = 20, 40, 60$, and moderate classification difficulty, $E[\varepsilon_n] = 0.2$. We consider resubstitution (resub), leave-one-out (loo), 4-fold cross-validation (cv4), 10-repeated 4-fold cross-validation (cv4r), the .632 bootstrap (b632), and the bias-correct bootstrap (bbc) [9]. For each bootstrap estimator, $T = 100$ balanced bootstrap samples were employed. RMS values for resubstitution and leave-one-out are exact; they are computed using the analytical expressions developed in [5]. For the other error estimators, RMS values are obtained by a Monte-Carlo computation using 20,000 samples from the synthetic distribution.

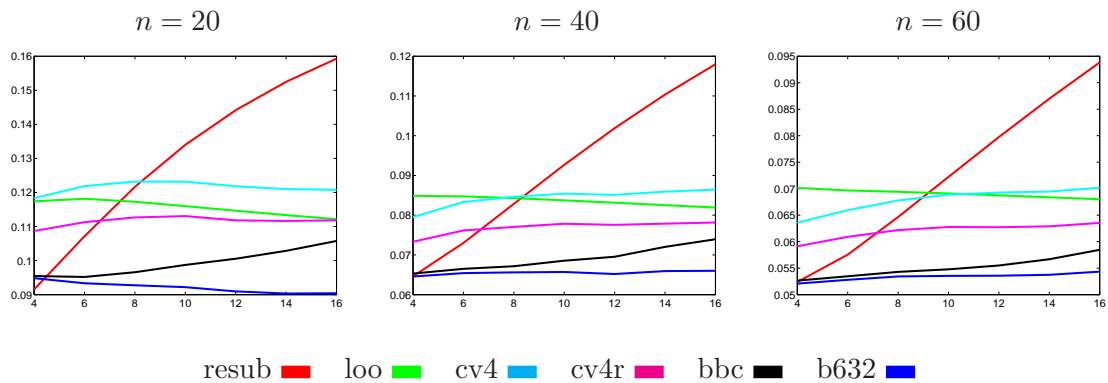


Figure 9: RMS values for several error estimators vs. number of bins, for $E[\varepsilon] = 0.2$ and varying number of samples.

Figure 9 shows that the bootstrap error estimators have similar performance to each other, and are generally the best-performing estimators. What is truly remarkable is that for $b = 4$, resubstitution is equivalent in performance (as measured by the RMS) to the bootstrap esti-

mators, and for $b \leq 6$, it is superior to the repeated cross-validation estimator (in functional genomics applications, gene expression is often binarized, e.g., a promoter is either on or off, so that $b = 4$ corresponds to prediction using 2 genes). These are very computationally-intensive estimators, with execution timings typically hundreds of times larger than those of resubstitution [3]. We can see that, as the number of samples increases (which alleviates the bias problem of resubstitution), then the classification complexity cutoff at which resubstitution beats some or all the cross-validation estimators increases.

CONCLUSIONS

Critical scientific issues are raised by using false or imprecise microarray classification methods in genomic signal processing applications. It can lead to false discovery, which wastes the efforts and resources of scientists, or it can lead to missing important discoveries, which may be even more damaging. The overall theme that permeates all the discussion in this paper is that choosing complex classification rules and computer-intensive error estimators does not necessarily help, and in fact can be, and often is, counter-productive in the small-sample case. We have shown evidence that, in the small-sample case, the trade-off in the bias-variance dilemma moves towards controlling variance more than bias, both in the case of classifier design and error estimation. In small-sample scenarios, we recommend the use of simple classification rules — we prefer linear classifiers, which even helps to guide feature selection towards linearly separable classes, and genes with more biological meaning. But 3NN has also shown to be a good alternative, when the classes are indeed non-linearly separable. Exhaustive feature selection should be done whenever possible, by reducing the number of initial genes if necessary. Thus, time-consuming error estimators such as repeated cross-validation and bootstrap may not be an option. For classification rules with small degree of overfitting, resubstitution can be adequate with a small number of variables and large number of cases, otherwise the use of bolstered resubstitution has shown to be a good choice.

Acknowledgments

The author would like to acknowledge the support of the Brazilian National Research Council (CNPq), under Grant DCR 35.0382/2004.2, and the support of the National Institute of Allergy and Infectious Diseases (NIAID/NIH), under Grants U19 AI56541 and N01 AI 40085.

References

- [1] E. Baum. On the capabilities of multilayer perceptrons. *Complexity*, 4:193–215, 1988.
- [2] U.M. Braga-Neto and E.R. Dougherty. Bolstered error estimation. *Pattern Recognition*, 37(6):1267–1281, 2004.
- [3] U.M. Braga-Neto and E.R. Dougherty. Is cross-validation valid for microarray classification? *Bioinformatics*, 20(3):374–380, 2004.
- [4] U.M. Braga-Neto and E.R. Dougherty. Classification. In E. Dougherty, I. Shmulevich, J. Chen, and Z. J. Wang, editors, *Genomic Signal Processing and Statistics*, EURASIP Book Series on Signal Processing and Communication. Hindawi Publishing Corporation, 2005.
- [5] U.M. Braga-Neto and E.R. Dougherty. Exact performance of error estimators for discrete classifiers. *Pattern Recognition*, 2005. In press.
- [6] T. Cover and J. van Campenhout. On the possible orderings in the measurement selection problem. *IEEE Trans. on Systems, Man, and Cybernetics*, 7:657–661, 1977.
- [7] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, 1996.
- [8] E.R. Dougherty. Small sample issues for microarray-based classification. *Comparative and Functional Genomics*, 2:28–34, 2001.
- [9] B. Efron. Estimating the error rate of a prediction rule: Improvement on cross-validation. *Journal of the American Statistical Association*, 78(382):316–331, 1983.
- [10] M. Gardner. *Fads and Fallacies In The Name of Science*. Dover, New York, NY, 1957.
- [11] D.J. Hand. Recent advances in error rate estimation. *Pattern Recognition Letters*, 4:335–346, 1986.
- [12] G.F. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63, 1968.

- [13] P. Ioannidis. Microarrays and molecular research: noise discovery? *The Lancet*, 365:454–455, 2005.
- [14] A.K. Jain, R.P.W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.
- [15] A.K. Jain and D. Zongker. Feature selection: Evaluation, application, and small sample performance. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997.
- [16] S. Kim, E.R. Dougherty, I. Shmulevich, K.R. Hess, S.R. Hamilton, J.M. Trent, G.N. Fuller, and W. Zhang. Identification of combination gene sets for glioma classification. *Molecular Cancer Therapy*, 1:1229–1236, 2002.
- [17] P.M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Trans. on Computers*, 26(9):917–922, 1977.
- [18] C. Hill S. Michiels, S. Koscielny. Prediction of cancer outcome with microarrays: a multiple random validation strategy. *The Lancet*, 365:488–492, 2005.
- [19] R. Simon, M.D. Radmacher, K. Dobbin, and L.M. McShane. Pitfalls in the use of dna microarray data for diagnostic and prognostic classification. *Journal of the National Cancer Institute*, 95(1):14–18, 2003.
- [20] M.J. van de Vijver, Y.D. He, L.J. van’t Veer, H. Dai, A.A.M. Hart, D.W. Voskuil, G.J. Schreiber, J.L. Peterse, C. Roberts, M.J. Marton, M. Parrish, D. Astma, A. Witteveen, A. Glas, L. Delahaye, T. van der Velde, H. Bartelink, S. Rodenhuis, E.T. Rutgers, S.H. Friend, and R. Bernards. A gene-expression signature as a predictor of survival in breast cancer. *The New England Journal of Medicine*, 347(25):1999–2009, Dec 2002.
- [21] L.J. van’t Veer, H. Dai, M.J. van de Vijver, Y.D. He, A.A.M. Hart, M. Mao, H.L. Peterse, K. van der Kooy, M.J. Marton, A.T. Witteveen, G.J. Schreiber, R.M. Kerkhoven, C. Roberts, P.S. Linsley, R. Bernards, and S.H. Friend. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415:530–536, Jan 2002.
- [22] V.N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.