Design and Analysis of Classifier Learning Experiments in Bioinformatics: Survey and Case Studies

Ozan İrsoy, Olcay Taner Yıldız, Ethem Alpaydın, *Senior Member, IEEE*

***Abstract*—In many bioinformatics applications, it is important to assess and compare the performances of algorithms trained from data, to be able to draw conclusions unaffected by chance and are therefore *significant.* Both the design of such experiments and the analysis of the resulting data using statistical tests should be done carefully for the results to carry significance. In this paper, we first review the performance measures used in classification, the basics of experiment design and statistical tests. We then give the results of our survey over 1500 papers published in the last two years in three bioinformatics journals (including this one). Although the basics of experiment design are well-understood, such as resampling instead of using a single training set and the use of different performance metrics instead of error, only 21 per cent of the papers use any statistical test for comparison. In the third part, we analyze four different scenarios which we encounter frequently in the bioinformatics literature, discussing the proper statistical methodology as well as showing an example case study for each. With the supplementary software, we hope that the guidelines we discuss will play an important role in future studies.**

***Index Terms*—Statistical tests, Classification, Model selection**

I. Introduction

In many bioinformatics applications, there is an underlying process whose details we barely know, but we can collect a sample of examples from the process by doing experiments, and using machine learning techniques, we can make statistical inference about the process from this sample. In *supervised learning*, the sample is composed of pairs of independent and dependent variables and the aim is to learn a mapping from the independent variable to the dependent. In *classification*, the dependent variable is a class code and the aim is to devise a rule that can predict the class labels of instances. For example, a biologist may want to categorize a given protein as binding or non-binding, and this is a two-class problem. The independent variable is represented by a feature set x composed of different properties of a protein, such as the amino acid sequence, the evolutionary information, structural information, and so on. If the *discriminant function* that is used for predicting the class label is denoted by $f(x|\phi)$, different models, e.g., decision trees, support vector machines, neural networks, correspond to different $f(\cdot)$ and learning corresponds to optimizing the model parameters ϕ to minimize some loss measure on a given training sample [1].

Ozan İrsoy and Ethem Alpaydın are with the Department of Computer Engineering, Boğaziçi University, 34342, İstanbul Turkey. Olcay Taner Yıldız is with the Department of Computer Engineering, Işık University, 34398, İstanbul Turkey.

Typically, we have candidate $f_i(\cdot|\phi_i)$, where $i = 1, \ldots, L$ are different learning algorithms, and we want to choose the best according to some performance measure. The aim is to find the algorithm that generalizes best to unseen data and to measure that, we use a validation set on which we test how well our trained $f(\cdot|\phi)$ performs. Because the examples in the training and validation sets are random variables drawn from some unknown joint probability distribution, the discriminant we fit to the sample contains some randomness. Although we use the same classification algorithm, different training samples may induce different classifiers and in making a decision among algorithms, we need to make sure that our decision is not affected by chance, for example, by how the data is split between training and validation sets.

In the statistics literature, there is considerable body of work done on the *design and analysis of experiments* [2]—the aim of this paper is to discuss those principles in the context of classification experiments in bioinformatics and show the proper methodologies using case studies. In experiment design, there is a process which takes an input and generates an output; the output is affected by a number of factors some of which are controllable and some are not. In our case, the process is the classifier which after having been trained on a training set gives the class as output for an input from the validation set. Here, the major controllable factor is the learning algorithm and the major uncontrollable factor is the randomness in the data. The aim is to find the configuration of controllable factors that maximize a response variable measuring quality. In classification, there are different performance metrics that can be calculated from that data, such as, misclassification error, hit rate, precision, and so on. In Section II, we discuss such metrics in detail and also point out how they differ, to be able to point out which one to use in which type of experiment.

The three principles of experimental design are *randomization*, *replication*, and *blocking*—in machine learning, these imply the need for multiple paired runs using resampling. Once a set of experiments are done and we have a set of results, *statistical hypothesis testing* is used to check for differences that are significant, that is, unlikely to have been caused by chance. We discuss the resampling procedures in Section III, statistical tests in Section IV, and give pointers to related work in Section V.

We did a survey on the use of such procedures in the recent bioinformatics literature to check how frequently these different approaches to experiment design and analysis have been employed; the results are given in Section VI.

Then in Section VII, we review four different scenarios we encounter frequently in classification experiments and show the proper methodology for each using a case study. We conclude in Section VIII.

II. Assessing Performance

A. Confusion Matrix and the Measures of Performance

In a two-class problem, we have positive and negative instances, for example, binding vs. nonbinding proteins. Having trained our classifier $f(x|\phi)$ on the training set, typically we predict that x drawn from the validation set is a positive example if $f(x|\phi) \geq \theta$, for some threshold θ. We can assume that $f(x|\phi) \in [0, 1]$ estimates the posterior probability that x is a positive example [1], that is, $\hat{P}(+|x) \equiv f(x|\phi)$. We say that x is a negative example if $f(x|\phi) < \theta$ and $\hat{P}(-|x) \equiv 1-f(x|\phi)$. Then, depending on the true label of x, there are four cases which make up the *confusion matrix* and we count the number of their occurrences over the whole validation set (Table I):

- True positive (*tp*): The number of instances for which both the class label and the predicted class are positive.
- False negative (*fn*): The number of instances for which the class label is positive but the predicted class is negative.
- False positive (*fp*): The number of instances for which the class label is negative but the predicted class is positive.
- True negative (*tn*): The number of instances for which both the class label and the predicted class are negative.

TABLE I
2×2 CONFUSION MATRIX

	Prediction		
Truth	+	-	Total
+	*tp*	*fn*	p
-	*fp*	*tn*	n
Total	p'	n'	N

Different performance measures used in the literature are all calculated from these four values:

$$\begin{aligned} &\textit{error rate} = \frac{\textit{fp}+\textit{fn}}{N} \quad &&\textit{accuracy} = \frac{\textit{tp}+\textit{tn}}{N} \\ &\textit{tp-rate} = \frac{\textit{tp}}{p} &&\textit{fp-rate} = \frac{\textit{fp}}{n} \\ &\textit{recall} = \frac{\textit{tp}}{p} &&\textit{precision} = \frac{\textit{tp}}{p'} \\ &\textit{sensitivity} = \frac{\textit{tp}}{p} &&\textit{specificity} = \frac{\textit{tn}}{n} \\ &\textit{F-measure} = 2\frac{\textit{precision}\cdot\textit{recall}}{\textit{precision}+\textit{recall}} = 2\frac{\textit{tp}}{p+p'} \\ &\textit{Balanced accuracy} = \frac{\textit{sensitivity}+\textit{specificity}}{2} \end{aligned} \tag{1}$$

Tp-rate, also known as the *hit rate*, is the same as recall and sensitivity. Fp-rate is sometimes called *false alarm rate*, and is equal to $1-$specificity. Different names for these related measures are due to historical reasons where they have been proposed in different domains, namely, signal processing, information retrieval, or diagnostics, almost independently.

[1]Possibly after some normalization, if the classifier is a nonprobabilistic classifier such as a support vector machine.

B. Performance Curves and the Area Under the Curves

The threshold θ of decision depends on the relative costs of a false positive and a false negative. We use $\theta = 0.5$ when they have equal cost and for example θ needs to be larger when a false positive has a higher cost than a false negative. In some cases, we do not know the exact costs and we may want to see how the performance measure varies as we vary them, which corresponds to varying θ. Then we can plot the performance as a function of θ to see the overall behavior. A *receiver operating characteristics* (ROC) curve is a plot of tp-rate (hit rate) and fp-rate (false alarm rate); similarly, one can plot a precision-recall curve or a sensitivity-specificity curve [3]. Some people use a "partial curve" when they are interested in the performance of the classifier in a particular subrange for θ (that corresponds to a subrange for costs of misclassification); our discussion holds also for this case where instead of the whole curve, we use a subset of the curve.

Curves are complex and it is difficult to compare two curves. One way to summarize a curve (full or partial) by a single value is by calculating the *area under the curve* (AUC), which can be estimated by summing the trapezoidal areas formed by successive points on the performance curve [3]. The two most popular are the *ROC curve* of tp-rate vs fp-rate and the area under it (AUC-ROC) and the *Precision-Recall* (PR) curve and the area under it (AUC-PR).

PR curve is mostly used in information retrieval [4] where for a query, some of the stored items are relevant (the true label is positive) and some are not (the true label is negative). Given x that are the attributes associated with the item, we retrieve some of them (the predicted label is positive) and some we do not (the predicted label is negative). In this context, *precision* is the proportion of the relevant and retrieved documents to the total number of retrieved documents, and *recall* is the proportion of the relevant and retrieved documents to the total number of relevant documents.

Note that ROC measures the performance of a two-class classifier and checks for good performance on both positives and negative instances, whereas in an information retrieval application (whose performance is measured by PR), we have basically a one-class problem where we care for the positives more. In an application like medical diagnosis, more than the true negatives, i.e., the large proportion of healthy individuals, we care about detecting the sick, and it is better to focus on PR. In an application where we classify face images as male or female, we care about the accuracy on both genders, and need ROC to measure performance.

PR is sensitive to class skewness, whereas ROC is not [5]. When the ratio p/n changes, because precision uses values from both rows of Table I, it changes; however tp-rate and fp-rate may not change since they use values from only one row [3]. PR and ROC make different statistics apparent: In PR, we are basically interested in how well we classify the positive examples, whereas in ROC, in trying to minimize fp-rate, we also want to increase the true negative rate. This makes sense in information retrieval, for a given query, adding a lot more irrelevant documents (which we will not be retrieving anyway) has no effect on our performance assessment for this query.

A one-to-one correspondence between a ROC curve and a PR curve has been shown [5]. It has also been proven that one ROC curve dominates the other ROC curve if and only if the corresponding PR curve dominates the other [3]. Despite the dominance relationship between ROC and PR curves, if AUC-ROC of the first curve is greater than the second one, AUC-PR of the first curve can be less than the second one, therefore optimizing for AUC-ROC does not mean also optimizing for AUC-PR. The corresponding points in curves can dominate each other in parallel in ROC and PR curves; however it is the magnitude of these differences that determines the area differences and consequently, since the metrics are different, the area between the curves may be different.

III. Resampling Procedures

When we are comparing two or more algorithms trained from data, the training algorithm may have some randomness (for example, gradient descent starts from a random initial point), or the way the data is divided between training and validation sets is random. If we do training and validation only once, we can not know if any difference between two results is because of difference in algorithms or because of the split of data.

The three basics of experimental design are *randomization*, *replication*, and *blocking*. To be able to average out the effect of randomness and hence arrive at conclusions deemed *statistically significant*, we do the training and validation multiple times randomly (*randomization*), run the algorithms many times (*replication*), and compare the *distributions* of results rather than single values. This requires that we be able to generate multiple training and validation set pairs from a single data set. Note that when we are comparing a number of algorithms, they should all use the same training and validation splits so that we make sure that any difference is due to the algorithm (the controllable factor) and not due to the split of data (uncontrollable factor); this is the idea behind *paired* tests (*blocking*). We also require *stratification*, that is, the proportion of positive to negative instances is respected in all parts so that the prior class probabilities do not change between folds.

There are various resampling algorithms [1]:

1) In *k-fold cross-validation* (cv), we divide the data randomly into k equal parts. At each *fold*, we leave one of the k parts out as the validation set and use the remaining $k-1$ parts together as the training set. By cycling over all the k parts, we get k training and validation set pairs.
2) *Leave-one-out* is the extreme case of k-fold cv, where k is taken to be equal to N, the number of instances in the training set. That is, at each fold, we use $N-1$ instances for training and one instance for validation, leaving out another one, in a total of N folds. With very small data sets, leave-one-out is used.
3) In $k_1 \times k_2$-fold cross validation, there is an outer loop that replicates k_2-fold cv k_1 times and a statistic is defined over the $k_1 \cdot k_2$ results. Examples are 5×2 cv [6], [7] and 10×10-fold cv.
4) In *bootstrap*, from a sample of N instances, we draw N instances *with replacement*, so some instances may be drawn more than once, and some never. Different training folds hence partially overlap. The whole set is used as the validation set in all folds [8].

We use the following notation: Let y_{ij} denote the performance of classifier $i = 1, \ldots, L$ on validation fold $j = 1, \ldots, k$. The performance value can be the misclassification error rate, precision, area under the ROC curve, and so on. Then, for example, in comparing algorithms 1 and 2, we need to compare the distributions of y_{1j} and $y_{2j}, j = 1, \ldots, k$.

IV. Statistical Tests

In hypothesis testing, we have a null hypothesis H_0 that we want to test on the sample, against an alternative hypothesis H_1. For example

$$H_0 : \mu = 2 \text{ vs. } H_1 : \mu \neq 2$$

What we do is we collect a sample and then calculate a statistic on the sample and check the probability that this statistic takes a particular value or higher under the assumption that the null hypothesis is true. If that probability—the so-called p value—is very small, i.e., smaller than a pre-defined significance value α, e.g., 0.05, we *reject* the null hypothesis in favor of the alternative hypothesis, otherwise we fail to reject it. Note that a failure to reject does not imply the truth of the null hypothesis, nor rejection implies that the alternative hypothesis is correct. If we reject when the null hypothesis holds, this is a *type I error*; the failure to reject when the null hypothesis is wrong is a *type II error*.

Typically, there are four scenarios where hypothesis testing is used in classification experiments (see Table II):

1) We have two algorithms that we want to compare on a single data set in terms of some performance metric. This is typically the most frequently used scenario. For example, we may want to compare two algorithms in terms of error, or AUC-ROC. Or, we may want to test two variants of the same algorithm; for example, we may want to see if having feature selection before our neural network leads to significant improvement.
2) We have $L > 2$ algorithms that we want to compare on a single data set in terms of some metric. These may be different algorithms or different variants of the same algorithm; for example, we may be interested in comparing $L > 2$ different feature extraction algorithms that precede the classifier.
3) We have two algorithms that we want to compare on $M > 1$ data sets in terms of some performance metric. For example, we may have M different cancer data sets but because of different properties of the data sets, we cannot combine them in a single data set to train a single classifier. What we want is to train and test both algorithms separately on these data sets, compare performances on each separately and then combine those comparisons to get an overall result.
4) We have $L > 2$ algorithms that we want to compare on $M > 1$ data sets in terms of some performance metric. This is the most general case.

TABLE II
SCENARIO AND TESTS WE USE.

Number of algorithms	Number of data sets $M=1$	$M>1$
$L=2$	5×2 cv F test	Wilcoxon's signed rank test
$L>2$	ANOVA + 5×2 cv F test	Friedman's and Nemenyi's test

Now let us see the tests for each scenario one by one[2]. Later on, in Section VII, we will see how each one is used in a real-world case study.

A. Comparing Two Algorithms on a Single Data Set

The number of errors (or true positives, precision, and even AUC) is a count of 0/1 events and is hence binomially distributed. Unless the validation set is very small, from the central limit theorem, the binomial converges to the normal distribution and we can use parametric tests based on the normal distribution.

We want to compare the expected performance values of the two algorithms:

$$H_0 : \mu_1 = \mu_2 \text{ vs. } \mu_1 \neq \mu_2 \tag{2}$$

and in a paired setting, we test if their paired difference has zero mean:

$$H_0 : \mu_d \equiv \mu_1 - \mu_2 = 0 \text{ vs. } \mu_d \neq 0. \tag{3}$$

Dietterich [6] has compared various pairwise tests, including McNemar's test which uses a single training/validation pair, and the t test used with k-fold cross-validation. He then proposed the 5×2 cross-validation sampling and an associated t test, which he has shown to have lower type I and type II errors. The 5×2 cross-validation F test [7] is an improved version of this t test and it works as follows:

In 5×2 cross-validation, we perform 2-fold cross-validation five times. Let us say $p_i^{(j)}$ is the difference between the performance values of the two algorithms on fold $j=1,2$ of replication $i=1,\ldots,5$. The average on replication i is $\overline{p}_i = (p_i^{(1)} + p_i^{(2)})/2$ and the estimated variance is $s_i^2 = (p_i^{(1)} - \overline{p}_i)^2 + (p_i^{(2)} - \overline{p}_i)^2$.

Under the null hypothesis that the two algorithms have the same expected performance, $p_i^{(j)}$ is approximately normal with mean 0 and its square divided by the variance is chi-squared and hence

$$f = \frac{\sum_{i=1}^{5}\sum_{j=1}^{2}(p_i^{(j)})^2}{2\sum_{i=1}^{5} s_i^2} \tag{4}$$

is F-distributed with 10 and 5 degrees of freedom [7]. We reject the null hypothesis that two algorithms have the same expected performance if $f > F_{\alpha,10,5}$.

B. Comparing $L>2$ Algorithms on a Single Data Set

Analysis of Variance (ANOVA) tests if all populations have the same mean:

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_L \text{ vs. } \mu_r \neq \mu_s, \text{for any } r \neq s. \tag{5}$$

[2]Matlab functions for these tests are made available as a supplement.

In our case, this corresponds to checking if all algorithms have the same expected performance.

Let us say $y_{ij}, i=1,\ldots,k, j=1,\ldots,L$ is the performance value of algorithm j on fold i. The average performance of algorithm j on all folds and the overall average are defined as

$$m_j = \frac{\sum_{i=1}^{k} y_{ij}}{k} \text{ , } m = \frac{\sum_{j=1}^{L} m_j}{L}$$

ANOVA calculates the between- and within-algorithm sums of squares

$$SS_b = k\sum_j (m_j - m)^2 \text{ , } SS_w = \sum_j \sum_i (X_{ij} - m_j)^2$$

Both are chi-square distributed random variables. Under the null assumption, their ratio after each is divided by its degrees of freedom

$$f = \frac{SS_b/(L-1)}{SS_w/L(k-1)} \tag{6}$$

is F distributed with $L-1, L(K-1)$ degrees of freedom. We reject the null hypothesis that all algorithms perform equally well if $f > F_{\alpha,L-1,L(K-1)}$.

If the test fails to reject, all are equally good. If the test rejects, we know that there is an inequality somewhere. To find where, we do a set of pairwise *posthoc tests* to try to find cliques, that is, subsets of algorithms in which there is no significant difference between any two. To do this, we first sort all L algorithms in terms of average performance and then we compare the first and the last in a pairwise manner for significant difference. If the test rejects, we take the first $L-1$ leaving out the last and compare the first and the $L-1$st; we also compare the second and the Lth leaving out the first. As long as there is a reject, we keep on leaving out the first and the last recursively and on both sides. At any stage if the test fails to reject, we underline that group and we do not examine it any further. This compares all consecutive subsets of algorithms and the underlines (which may partially overlap) indicate cliques of algorithms whose performances are comparable in terms of the metric we use. For example, with algorithms A, B, C, D, E, we may have the result

$$\underline{B\ C\ \underline{A\ E}}\ D$$

Here, $\{B, C, A\}$ form one clique, and $\{A, E\}$ form another clique; for example, there is significant difference between B and E and also between E and D.

C. Comparing Two Algorithms on $M>1$ Data Sets

When we have values calculated over different data sets, we can no longer use any parametric test because the performance over different data sets do not come from a normal or any known distribution (That is why, it does not make sense to

calculate the average performance over different data sets either). In this case, we can only use a nonparametric test that compares which of the two algorithms is better in how many of these different data sets—if we do resampling and have results on multiple folds, we compare the averages over the folds. On some of these data sets, the first one wins, on some the second wins (the first loses) and on the rest, they tie. We then need to check if those number of wins and losses is likely under the null hypothesis that the two algorithms perform equally well, i.e., when the win probability is $1/2$—ties are equally split between wins and losses. This is called the *sign test.* If if the first algorithm wins in 12 data sets out of 20 and loses on 8, the null hypothesis that they are equally good can be claimed; if however the first wins in 19 out of 20 and loses on one, that would be a very rare event if indeed they were equally good, and it makes sense to reject.

The *Wilcoxon's signed rank test* is an extension of the sign test and uses the same idea except that it also takes into account the difference in performance for wins and losses. We calculate the difference at each fold as $d_j = y_{1j} - y_{2j}$ and then sort them in terms of $|d_j|$ and give them ranks between 1 and M. If ties occur, we give them the average of what they would get if they differed slightly. We then calculate w_+ as the sum of all ranks whose signs of difference are positive and w_- as the sum of ranks whose signs of differences are negative. The null hypothesis that $\mu_1 = \mu_2$ can be rejected if either of w_+ and w_i, that is, $\min(w_+, w_-)$ is very small. The critical values for the Wilcoxon's signed rank test are tabulated and for $M > 20$, normal approximation can be used.

D. Comparing $L > 2$ Algorithms on $M > 1$ Data Sets

When we have more than two algorithms, on each data set we do not have a win/loss/tie; instead, each algorithm assumes a rank between 1 and L in terms of its performance (averaged over different folds). We then use nonparametric tests to check for significant difference in average ranks over the M data sets.

Friedman's test is the nonparametric version of ANOVA and uses ranks instead of the absolute performances [9]. On each data set j, the performance values of the algorithms are sorted from the best to the worst so that the best one gets the rank of 1, the second 2, and so on, until we get to L. Let r_{ij} denote the rank of algorithm $i = 1, \ldots, L$ on data set $j = 1, \ldots, M$. The average rank of algorithm i over the M data sets is

$$R_i = \frac{1}{M} \sum_j r_{ij}$$

The test statistic of Friedman's test is

$$\chi_F^2 = \frac{12M}{L(L+1)} \left[\sum_i R_i^2 - \frac{L(L+1)^2}{4} \right] \tag{7}$$

which, under the null hypothesis that all algorithms are equally good, is chi-square distributed with $L-1$ degrees of freedom. An improved statistic

$$F_F^2 = \frac{(M-1)\chi_F^2}{M(L-1) - \chi_F^2} \tag{8}$$

is F distributed with $L-1$ and $(L-1)(M-1)$ degrees of freedom.

If Friedman's test rejects, we use *Nemenyi's test* as the posthoc test to compare neighboring algorithms for significant difference in rank [9]. Two algorithms lead to classifiers with significantly different performance ranks at significance level α if the difference of their average ranks is greater than or equal to the critical difference

$$CD = q_\alpha \sqrt{\frac{L(L+1)}{6M}} \tag{9}$$

where q_α is the Studentized range statistic divided by $\sqrt{2}$. This again allows us to find cliques of equally good subsets which we can represent by underlining them.

V. Related Work

The importance of good experimental design and the use of resampling algorithms and hypothesis testing in learning algorithms was discussed by Cohen [10]. In the first textbook on machine learning, Mitchell dedicates a chapter to hypothesis testing for the assessment and comparison of learning algorithms [11]. In another early work, Salzberg draws attention to the risk of the use of the same, small number of data sets repeatedly by many researchers [12] which may result in algorithms too much finetuned to and hence overfitting those particular data; this risk holds for the domain of bioinformatics where experimentation to collect new data is expensive.

In a seminal study, Dietterich [6] reviews four statistical tests and proposes the 5×2 cross-validation method and an associated paired t test for comparing the error rates of two classification algorithms. Resampling has the risk of high type I error, and this issue has been theoretically investigated by Nadeau and Bengio [13]; they propose variance correction to take into account not only the variability due to test sets, but also the variability due to training examples. Bouckeart [14] shows that the widely used t test has superior performance compared to the Sign test in terms of replicability. On the other hand, he found the 5×2 cv t test dissatisfactory and suggested the corrected resampled t test. Hastie et al. [15] discuss the wrong and right ways of doing k-fold cross-validation.

The use of measures alternative to error/accuracy is old. AUC-ROC has been related to the Wilcoxon statistic and it is possible to calculate the required number of positive and negative examples for comparing two AUC-ROC values for given type I and type II probabilities [16]. Both AUC-ROC and AUC-PR use a single training and testing pair [17], [18], [19]; Hanley and McNeil [20] argue that comparing different ROC curves with a single data set limits their usefulness. One can use a resampling algorithm, such as k-fold cross-validation, to generate k ROC or PR curves hence k AUC-ROC or AUC-PR values. After fitting distributions to AUC-ROC or AUC-PR values, one can test hypotheses on them, as we discuss here.

More recently, Cortes and Mohri [21] have proposed to calculate confidence intervals for AUC-ROC from the confidence interval of error without any parametric assumptions. First, they define the expectation and variance of AUC-ROC in terms of the expected error, the number of negative instances and the number of positive instances by using the Wilcoxon-Mann-Whitney statistic. Using these values, the confidence intervals

are constructed without any assumption on the distribution for AUC-ROC. For large values of the sample size, they make a normal distribution assumption for error. Fitting distribution to AUC-ROC values has also been used by Bravo et al [22], though they do not compare it with the error and just use it to evaluate their results. The effect of class distribution on error and AUC-ROC is experimented in [23].

Hanczar et al. [24] discuss small sample estimation of ROC related samples and the difference of the estimated and true values of the AUC, tp-rate and fp-rate. Through a simulation study and analysis of real microarray data, they show that the difference is considerable. Swamidass et al. [25] propose the *concentrated ROC framework* in which any relevant portion of the ROC curve is magnified smoothly by an appropriate continuous transformation. The area under the ROC curve assesses retrieval performance of the relevant portion. Similar to ROC curves, PR curves are also used for performance evaluation [26], mostly in information retrieval applications [27] and they are preferred to ROC curves when the class distribution is skewed [4], [5], [28], [29].

Bengio et al. [30] argue that reporting statistics from ROC curve such as a break-even point may be misleading, and propose the *expected performance curve* to provide unbiased estimates at various operating points. Drummond et al. [31] introduce *cost curves* for visualizing the error rate or expected cost of two-class classifiers over all possible class distributions and misclassification costs. They argue that cost curves are better than ROC curves for visualization, for example in showing confidence intervals and visualizing the statistical significance of the difference between two classifiers.

When we compare $L > 2$ algorithms, after we apply the pairwise posthoc tests on all pairs, we may find pairs where the test does not reject, and in such a case, we underline such cliques. To break ties and get a full ordering, MultiTest [32] combines the results of the pairwise tests with a cost measure that specify a prior preferrence on algorithms. Various types of cost can be used [33], e.g., the space and/or time complexity during training and/or testing, interpretability, ease of programming, etc. In a bioinformatics application where different algorithms use results of different experimental procedures as inputs, some more costly than others, the cost of extracting the input may be another cost measure. Multi^{2}Test uses the same methodology to order algorithms on multiple data sets [34].

When doing multiple comparisons, there are various methods to adjust the value of α for each comparison. The simple method is Bonferroni correction [35]. If we compare L algorithms, there are $L(L-1)/2$ comparisons, and the Bonferroni correction sets the significance level of each comparison to $\alpha/(L(L-1)/2)$. Nemenyi's test is based on this correction, and that is why it has low power for large L. Garcia and Herrera [36] explain and compare the use of various correction algorithms, such as, Holm correction [37], Shaffer's static procedure [38] and Bergmann-Hommel's dynamic procedure [39]. They show that although it requires intensive computation, Bergmann-Hommel has the highest power.

VI. Survey of Classification Experiments in Bioinformatics Literature

To observe the practice of researchers in bioinformatics applications of machine learning for scenarios related to those discussed in this paper, we did a survey by examining the published papers in three journals (one of which is this one) in the years 2010 and 2011. Table III shows the number of papers surveyed in our work[3]. We include all the papers except software, application notes and proceedings. Among all the papers, we look at the ones related to machine learning and among those, we focus on those that use classification, which is our topic of study in this paper.

TABLE III
Number of papers surveyed

Journal	All Papers	ML Related	Classification
BMC Bioinformatics 2010	466	167	71
Bioinformatics 2010	334	167	65
IEEE/ACM TCBB 2010	69	34	20
Total (2010)	869	368	156
BMC Bioinformatics 2011	266	85	41
Bioinformatics 2011	272	99	28
IEEE/ACM TCBB 2011	125	54	21
Total (2011)	663	238	90
Grand total	1532	606	246

The results show that during these two years, $606/1532 = 40\%$ of the papers are machine learning related, and $246/606 = 41\%$ of these are related to classification tasks; the percentages do not change much from year to year. These high percentages indicate that there is a fair amount of classification done in the bioinformatics community, and these tasks require measures to evaluate the performances of different classifiers in different settings and domains—what we discuss in this paper relates to approximately 16% of the papers published in the last two years in these three journals.

From these papers that use classification, we collect data related to

1) the attributes of the problem (the number of classes and the number of input dimensions),
2) the attributes of the learning method (whether input dimensionality reduction is done or not, and the classification algorithm), and
3) statistical methodology used (resampling strategy, performance metrics, and the statistical test, if any is used).

Table IV shows the attributes we are interested in and their percentages in the years 2010, 2011, and in total. These percentage values are not mutually exclusive, e.g., if there are both two-class and multi-class data sets in a paper, that paper is included in both of the statistics; hence, these values do not always sum up to 100. Based on this data and our observations of these papers, we reach the following conclusions:

- We observe that most of the classification tasks are two-class classification tasks. This shows that the measures based on the confusion matrix (as given in Eq. 1), such as precision, recall, and so on, are applicable in most situations.

[3]A spreadsheet of this data is made available as a supplement.

TABLE IV
PERCENTAGES OF ATTRIBUTES OF CLASSIFICATION PROBLEMS, STATISTICAL METHODOLOGIES, AND THEIR PERCENTAGES IN THE SURVEYED PAPERS

Attribute	Percentage		
	2010	2011	Total
Two-Class	79	54	70
Multi-Class	23	47	32
Performance Metrics			
Accuracy / Error Rate	63	73	67
Precision (Positive Predictive Value)	28	26	27
False Positive Rate	10	13	11
F-Measure	17	14	16
Sensitivity (Recall, True Positive Rate)	49	54	51
Specificity	27	32	29
Receiver Operating Characteristics Curve	28	20	25
Area Under the ROC Curve	44	27	38
Precision-Recall Curve	11	4	9
Data Set Size			
1-9	1	0	0
10-99	25	20	23
100-999	55	55	55
1000-9999	29	33	31
10000+	26	19	23
Input Dimensionality			
1-9	9	9	9
10-99	27	26	26
100-999	23	34	27
1000-9999	22	29	25
10000+	21	21	21
Kernel/Other/Unspecified	24	10	19
Dimensionality Reduction	40	43	41
Algorithm Used			
Decision Tree (DT)	26	24	26
Support Vector Machine (SVM)	51	69	57
Rule Based Learning	4	3	4
Artificial Neural Network (ANN)	10	17	13
Naive Bayes (NB)	16	14	15
k-Nearest Neighbor (KNN)	15	17	15
Resampling Strategy			
k-fold Cross Validation	58	61	59
$k_1 \times k_2$-fold Cross Validation	11	9	10
Leave-One-Out	21	18	20
Bootstrapping	4	4	4
Independent Test Set	33	29	31
k Random Partitions Into Training/Test Sets	7	7	7
Statistical Tests for Comparison			
Parametric Test	13	6	10
Nonparametric Test	10	12	11
Other/Unspecified	1	1	1

- As expected, accuracy/error rate is the most frequently used metric. In cases where one needs to focus on the positives, precision and recall are also used. The use of the area under the ROC curve seems to be established in the community, but of the papers which give AUC values, only 51 per cent show the actual ROC curves. Precision-recall curves are also used though less frequently.
  We check for dependency between the type of classification problem and the performance measure used. Table V shows the percentages with which various performance metrics are used in two-class and multi-class classification problems. As expected, accuracy/error rate is used in multi-class problems more than in two-class problems and in two-class problems, the percentages of the use of precision/recall, sensitivity/specificity, or ROC curve or AUC-ROC are higher.
- Data set sizes indicate that in nearly half of the problems, the data set size is less than 1000 and in such cases, the variance of any statistic calculated from the data can be high. The use of suitable resampling strategies and hypothesis testing is hence apparent.
- Bioinformatics applications generally have high dimensional inputs—almost one-fifth of papers use data that have more than 10000 inputs!—indicating a higher propensity for overfitting with small data. In some papers, input dimensionality is not specified, in some, sequences of different lengths are processed, e.g., using hidden Markov models, and in some (with support vector machines), rather than in a vectorial form, a pairwise kernel matrix is used for inputs. Because many applications have high dimensional data, it is not surprising that some sort of dimensionality reduction is done before classification. As expected, we see in Table VI that the percentage of the use of dimensionality reduction increases as the input dimensionality increases.
- Support vector machines and decision trees (mostly random forests) are currently the best known off-the-shelf learning algorithms and they are also those most frequently used in bioinformatics applications. It has also been noted in a recent editorial [40] that the use of neural networks and hidden Markov models are decreasing whereas support vector machines and random forests are becoming more popular. Since support vector machine works well in small sample settings due to its inherent regularization and random forest works well in high dimensional, noisy data due to its averaging behavior, the use of these algorithms is justified.
  We check if there is a correlation between the algorithms used and data set size, input dimensionality, and whether or not dimensionality reduction is done before. As we see in Table VII, there does not seem to be any strong interaction. We would expect to see k-NN more with smaller data sets (because it needs to store the whole set) and naive Bayes more when input dimensionality

TABLE V
PERCENTAGES OF PERFORMANCE METRICS FOR TWO-CLASS AND MULTI-CLASS PROBLEMS

	Two-Class	Multi-Class
Accuracy / Error Rate	63	75
Precision (PPV)	28	26
False Positive Rate	13	6
F-Measure	16	15
Sensitivity (Recall, TPR)	54	46
Specificity	33	22
ROC Curve	30	14
Area Under the ROC Curve	46	21
Precision-Recall Curve	10	5

TABLE VI
PERCENTAGES OF THE USE OF A DIMENSIONALITY REDUCTION METHOD FOR DIFFERENT INPUT DSSIMENSIONALITIES

Input Dimensionality	Dimensionality Reduction
1-9	23
10-99	31
100-999	39
1000-9999	66
10000+	75

TABLE VII
PERCENTAGES OF DATA SET SIZE, INPUT DIMENSIONALITY AND THE USE OF DIMENSIONALITY REDUCTION FOR DIFFERENT ALGORITHMS

	DT	SVM	ANN	NB	KNN
Dataset Size					
1-9	0	0	0	0	0
10-99	22	26	13	18	34
100-999	57	57	61	58	58
1000-9999	27	32	26	24	24
10000+	24	19	19	26	16
Other / Unspecified	0	1	0	0	0
Dimensionality					
1-9	10	4	16	21	3
10-99	40	26	39	32	21
100-999	33	34	42	34	26
1000-9999	25	26	10	29	47
10000+	16	23	10	24	26
Kernel / Other / Unspecified	6	16	10	8	11
Dimensionality Reduction	43	45	39	47	53

TABLE VIII
PERCENTAGES OF RESAMPLING STRATEGIES FOR DIFFERENT DATA SET SIZES

	10-99	100-999	1000-9999	10000+
k-fold cv	46	58	71	65
$k_1 \times k_2$-fold cv	12	12	7	5
Leave-One-Out	32	21	13	9
Bootstrapping	12	6	4	2
Independent Test Set	37	35	34	35
k Random Partitions	7	7	7	5

is high (because it assumes independent inputs) or less dimensionality reduction with artificial neural networks (because it does its own feature extraction in its hidden units) and to a certain extent the data reflect these, but we do not see a strong domination of one algorithm over another one for a given data set size or input dimensionality.

- With small samples, leave-one-out is used; k-fold or $k_1 \times k_2$-fold cross-validation is used in almost 70 per cent of the cases. This shows that the need for multiple replications is well understood by the community.

  We check for dependency between data set size and resampling strategy. As we see in Table VIII, k-fold cross-validation is the most popular method. As we would expect, $k_1 \times k_2$-fold cv, leave-one-out and bootstrapping are used more frequently with smaller data sets. If the sample size is large, putting aside an independent test set unused for training is the cheapest way, but surprisingly it is used even with smaller data sets.
- Even though k-fold cross validation or other types of resampling strategies are used frequently, the use of statistical tests to compare the performance of different classifiers is rare (in only about 21 per cent). Some papers show standard deviations of the performance metrics without applying any test, and some use only a single performance value to conclude that one algorithm is better than the other. This shows that the use of statistical tests is not well established in the bioinformatics community indicating the need for the approaches we discuss here.

  We check for dependency between the use of a test (and its type) and the data set size. With small data sets, statistics have large variance and are more affected by chance and there is more need for a test to make sure that differences are significant. Indeed as we see in Table IX, as expected, we see tests used more with smaller data sets. Statistical tests should always be used while expecting a small power when the sample size is small. We would have expected to see nonparametric tests more with smaller data sets where central limit theorem may not hold, but the two types of tests seem to be used equally frequently.

  In Table X, we show the percentages of the number of models used in the studies. We see that 93 per cent of the studies use more than a single model, which indicates the need for statistical comparison. Note that we use the word "model" here rather than "algorithm" because when we compare k-NN with SVM, we count them as two models, and also when we compare k-NN with and without dimensionality reduction, we count them as two models too; when we compare 1-NN and 3-NN, we do not count them as two models but one model with different settings of the hyperparameter. We see that the number of models used—and hence needs to be compared—may be as high as tens in some studies, which points out again the need for rigorous experimentation and analysis.

  We also check the measures that the tests use. In Table XI, we show the number of papers that use the tests (divided into two as parametric and nonparametric) and the measures used. We see that tests mostly use either error or AUC-ROC and for these, either the parametric t test or nonparametric Wilcoxon's signed rank test are used most frequently; in very few cases, both are used. This supports well our recommendations in this paper.

TABLE IX
PERCENTAGES OF STATISTICAL TESTS FOR DIFFERENT DATA SET SIZES

	10-99	100-999	1000-9999	10000+
Parametric Test	14	11	11	7
Nonparametric Test	11	12	14	9

TABLE X
PERCENTAGES OF THE NUMBER OF MODELS USED

Models	1	2	3	4	5	6	7–10	≥ 11
Percentage	7	11	16	17	11	14	16	9

VII. CASE STUDIES

We use six well-known learning algorithms [1]:

- Knn: k-nearest neighbor with k between 1 and 10.
- Svm: Support vector machine (SVM) with a linear kernel; we use the LIBSVM 2.82 library [41].
- Rip: Rule learning algorithm Ripper where a rule contains a conjunction of univariate propositions [42].
- Mlp: Multilayer perceptron with 10 hidden units.
- Mdt: Multivariate decision tree algorithm where the decision at a node is a linear combination of all inputs [43].
- RnF: Random forest is an ensemble of decision trees.

In single data set case studies, we use the *acceptors* and *donors* data sets [44]. These are splice site detection data sets

TABLE XI
NUMBER OF PAPERS THAT USE STATISTICAL TESTS AND THE USED PERFORMANCE METRICS

	Error	AUC-ROC	AUC-PR	F Measure	Balanced Accuracy	Recall	Precision	Other	TP	Σ
Paired t Test	5	9		1	2	1	1	1	1	18
One Tailed Z Test	1	1								2
F Test		1								1
Wald Test				1						1
ANOVA	1	2								3
Wilcoxon Signed Rank Test	6	6	2	2	1		1			14
Wilcoxon Rank Sum Test	1	3								4
McNemar's Test	4									4
Sign Test	1	1								2
Kolmogorov-Smirnoff Test	1									1
Permutation / Randomization Test		1		1		1	1			2
Bootstrap Test				1						1
Unspecified / Other	1	1		1				1		4
Σ	18	22	2	7	2	2	3	2	1	

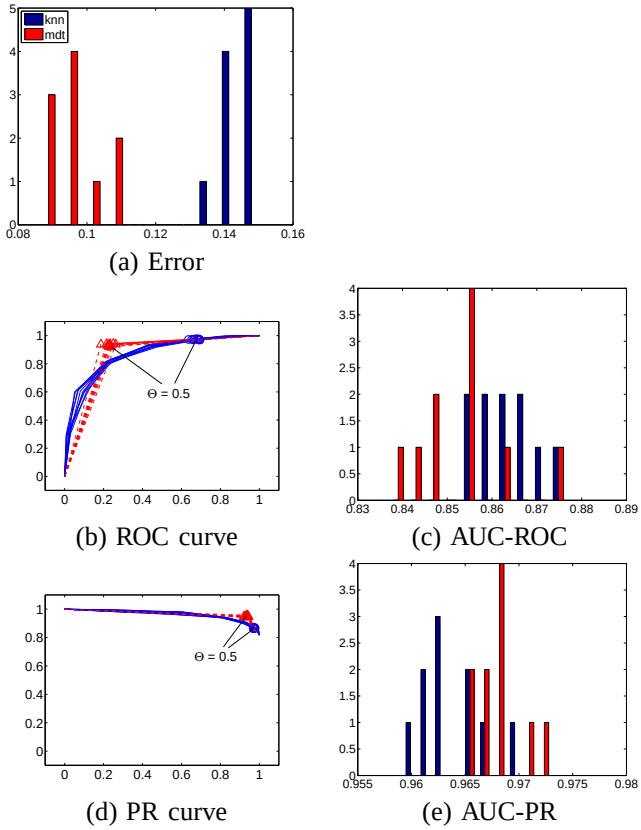
and the trained models should distinguish 'GT' and 'AG' sites occurring in the DNA sequence that function as splice sites and those that do not. A positive example for a donor site is a window of 13 residues of DNA around the 'GT' in an actual human donor splice site, while a negative example is a window of the same size around a 'GT' which is not itself a real splice site. The examples for the acceptor site are similar except that the window size is larger, i.e., a positive example for an acceptor site is a window of 88 residues of DNA around the 'AG' in an actual human acceptor splice site. There are 3889 (708 positive, 3181 negative) and 6246 (1324 positive, 4922 negative) examples in *acceptors* and *donors* respectively.

For multiple data set comparisons, we use the 11 cancer-related gene expression data sets [45]; details are given in Table XII. Nine are multi-class and two are two-class. The data were produced by oligonucleotide-based technology. In all data sets except *srbct*, RNA was hybridized to high-density oligonucleotide Affymetrix arrays and gene expression values were computed with Affymetrix software. In *srbct*, the experimenters used two-color cDNA platform with consecutive image analysis and filtered for a minimum level of gene expression. The genes or oligonucleotides with absent calls in all samples were removed from the analysis to reduce noise.

Our methodology is as follows: A data set is first divided into two parts, with $1/3$ as the test set, and $2/3$ as the training set. The training set is then resampled using 5×2 cross-validation (cv) where 2-fold cv is done five times (with stratification) and the roles swapped at each fold to generate ten training and validation folds. The validation folds are used to tune the hyperparameters of the algorithms, e.g., k of the k-nearest neighbor, C of the SVM, pruning thresholds for rules and trees, and so on. For the best setting, the ten classifiers trained on the ten training folds are tested on the left-out test set and these ten test results are reported and used in the statistical tests.

A. Comparing Two Algorithms on a Single Data Set

We compare the k-nearest neighbors (Knn) and the multivariate decision tree (Mdt) on the *acceptors* data set. We use the 5×2 cv F test for pairwise comparison as per our discussion in Section IV-A.



(a) Error (b) ROC curve (c) AUC-ROC (d) PR curve (e) AUC-PR

Fig. 1. Comparison of Knn with Mdt on *acceptors* data set.

Because we do 5×2 cross-validation, we have ten test results per algorithm. Figure 1 shows the (a) error histograms, (b) ROC curves, (c) histograms of areas under the ROC curves, (d) precision-recall curves, and (e) histograms of areas under the precision-recall curves. In Fig. 1(b) and 1(d), we mark the points that correspond to the threshold of 0.5; these are the values used in error comparison (shown in Fig. 1(a)).

This case is a good example illustrating that different measures make different things explicit. With the 5×2 cv F test in terms of error, the null hypothesis that the algorithms have equal expected error is rejected—Mdt leads to smaller expected error. As we see in the ROC curves, though the two have similar tp values at the threshold of 0.5, Knn has higher

TABLE XII
DETAILS OF THE 11 CANCER-RELATED GENE EXPRESSION DATA SETS USED IN THIS STUDY.

Dataset	Diagnostic Task	# of examples	# of features	# of classes
9tumors	9 various human tumor types	60	5726	9
11tumors	11 various human tumor types	174	12533	11
14tumors	14 various human tumor types and 12 normal tissue types	308	15009	26
braintumor1	5 human brain tumor types	90	5920	5
braintumor2	4 malignant glioma types	50	10367	4
dlbcl	Diffuse large B-cell lymphomas and follicular lymphomas	77	5469	2
leukemia1	3 types of leukemia	72	5327	3
leukemia2	3 types of leukemia	72	11225	3
lungtumor	4 lung cancer types and normal tissues	203	12600	5
prostatetumor	Prostate tumor and normal tissues	102	10509	2
srbct	small, round blue cell tumors of childhood	83	2308	4

fp and hence higher error. When we compare the two over the whole ROC curves, we see that the two algorithms excel in different parts but if we average over all possible losses, in terms of AUC-ROC, the 5×2 cv F test finds no significant difference. In terms of PR curves, the difference seems even less slight and again 5×2 cv F test on AUC-PR fails to reject.

Even though insignificant, ROC curve favors Knn whereas PR curve favors Mdt. We understand why if we compare Fig. 1(b) and (d): To the left of the curve (for high θ), Knn is to the left of Mdt implying less fp and hence overall, Knn seems to be better (In this case, for Knn, $k = 10$ and we have meaningful intermediate thresholds whereas the leaves of Mdt contain examples that highly favor one or the other class and the only meaningful intermediate threshold is at 0.5). AUC-PR does not make use of the fp (or tn) and hence this has no effect; since Mdt has slightly higher precision than Knn overall, it seems to be slightly better overall, though not significantly.

B. Comparing $L > 2$ Algorithms on a Single Data Set

The first case study can easily be generalized to more than two algorithms. We may be (i) proposing a novel learning algorithm and want to compare it against $L - 1$ previous approaches, or (ii) run L off-the-shelf learning algorithms via a data mining tool and decide which algorithm suits best to our data set. We find examples of this during our survey of the literature: Song et al. [46] propose an approach, Casclave, to predict caspase cleavage sites; they use different sequence encodings in their method and compare them over a single data set that they have constructed from multiple sources. Jeong et al. [47] test various classification algorithms on various feature sets to predict protein functions; the performance of the methods are compared over Yeast protein sequences.

As our second case study, we compare Rip, Mdt, Mlp, Svm, and Knn on *donors* data set in terms of error, AUC-ROC, and AUC-PR. The histograms are given in Figure 2. We see that though the five algorithm seem very different in terms of error and AUC-ROC, they seem more similar in terms of AUC-PR, again indicating that the difference in behavior is due to the negative instances.

For all three measures, ANOVA rejects the null hypothesis that all algorithms have the same performance. We apply 5×2 cv F test as a pairwise post-hoc test as per our discussion in Section IV-B and find the following orderings and cliques:

- Error: Mdt Svm Mlp Rip Knn
- AUC-ROC: Svm Mlp Knn Mdt Rip
- AUC-PR: Svm Mlp Knn Rip Mdt

TABLE XIII
ERROR DIFFERENCES OF Mlp−RnF ON THE 11 TUMOR DATA SETS.

9tum	11tum	14tum	brai1	brai2	dlbcl
−3.64	−0.49	12.16	−12.81	1.11	−10.37

leuk1	leuk2	lung	prost	srbct
−12.4	0.4	−3.91	−3.71	−1.38

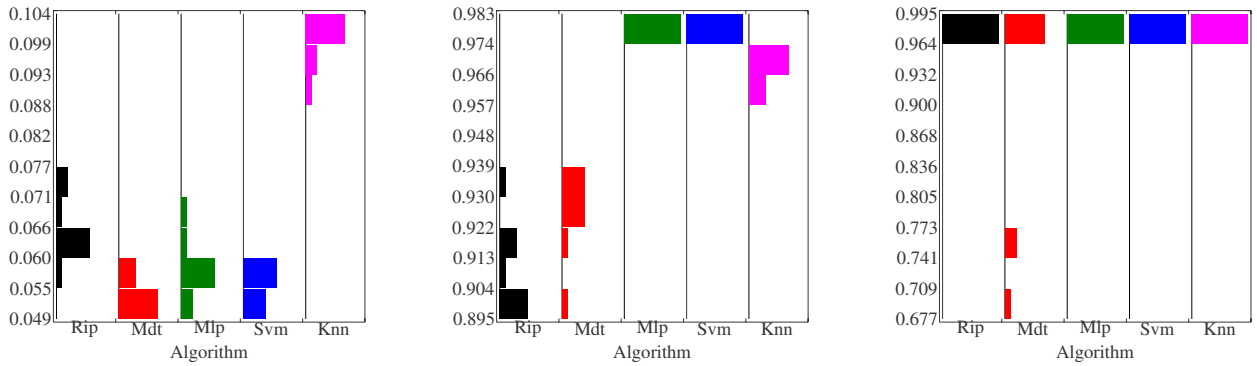
In terms of error, since Rip is significantly different from Mdt but not from Svm nor Mlp, (Mdt, Svm, Mlp) and (Svm, Mlp, Rip) form a clique. On the other hand, as seen in the figure, Knn is significantly worse than all other algorithms. In terms of AUC-ROC, Mdt and Rip have similar performance, they form a single group and perform worse than the other algorithms. There is no significant difference between Mlp and Svm or Knn, but since the last two are significantly different from each other, two cliques are formed: (Svm, Mlp) and (Mlp, Knn).

In terms of AUC-PR, Mdt is not significantly different from Svm using 5×2 cv F test, so although ANOVA rejects the null hypothesis that all algorithms have the same AUC-PR, we say that all five algorithms form a single clique. This may happen in real life, tests for the same purpose may decide differently due to different properties (ANOVA is not a paired test) or assumptions.

We can use MultiTest [32] to get rid of the underlines and get a full ordering. For example, when we apply MultiTest with error as the performance measure and average space complexity as the cost measure, the ordering we get is (from best to worst: '$<$' means "preferred to"): Mdt $<$ Rip $<$ Mlp $<$ Svm $<$ Knn; if we use average training time to prefer faster algorithms, we get Mlp $<$ Mdt $<$ Rip $<$ Svm $<$ Knn.

C. Comparing Two Algorithms on $M > 1$ Data Sets

Some examples of this scenario can be found: MacDonald and Beiko [48] propose a rule mining method named CPAR to extract microbial genotype-phenotype association rules and compare it against the existing NETCAR algorithm over multiple data sets. In converting multi-class problems to a set of two-class problems, Taipa et al. [49] compare one-against-all and error-correcting output codes over various data sets.



(a) Error (b) AUC-ROC (c) AUC-PR

Fig. 2. Comparison of Rip, Mdt, Mlp, Svm, and Knn on *donors* data set.

As a case study, we compare Mlp and RnF on the 11 tumor data sets. Because nine of the 11 data sets are multi-class, we cannot use AUC-ROC and AUC-PR directly, so we use error only. We calculate the average error of each algorithm on the ten folds of each data set and use Wilcoxon's signed-rank test as per our discussion in Section IV-C.

Error differences (Mlp−RnF) are shown in Table XIII. We see that the negative differences occur a lot more than the positive differences and are also bigger in magnitude and that is why Wilcoxon's signed-rank test rejects the null hypothesis that the average ranks of the two algorithms are same. Overall, Mlp performs better than RnF on these 11 tumor data sets.

D. Comparing $L > 2$ Algorithms on $M > 1$ data sets

Examples of this scenario are found in the literature: Zhu et al. [50] propose a novel feature selection method before SVM and compare their method against various other dimensionality reduction techniques over multiple data sets. Liu et. al. [51] propose a sparse SVM method for biomarker identification and compare their method with three other methods over three data sets, including a synthetic data set.

As a case study, we compare Rip, Mlp, RnF, Svm, and Knn on the 11 tumor data sets in terms of error. Table XIV shows the error rates of Rip, Mlp, RnF, Svm, and Knn. First we apply Friedman's test which rejects that the algorithms have equal expected error. The result of the post-hoc Nemenyi's test can be seen in Figure 3, which can be rewritten as:

Svm Mlp RnF Knn Rip

We see that there are three cliques: (Svm, Mlp), (Mlp, RnF, Knn), and (RnF, Knn, Rip). We can not directly conclude that Svm is the best because there is no significant difference between Svm and Mlp; we can not choose Mlp either because RnF and Knn are as good (but worse than Svm).

We can use Multi^{2}Test [34] here to get a full ordering. If we use space complexity as the cost measure and Nemenyi's test as the pairwise test on error, we get Rip $<$ Mlp $<$ RnF $<$ Svm $<$ Knn, whereas with training time as the cost measure, we get Knn $<$ Rip $<$ RnF $<$ Svm $<$ Mlp.

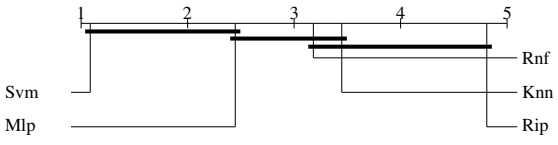


Fig. 3. The result of post-hoc Nemenyi's test.

VIII. Conclusions

As in all machine learning applications, in bioinformatics too, the correct use of experiment design and analysis is of paramount importance for results to be considered significant. Our contributions here are as follows:

- We review the basics of the design and analysis of experiments discussing the correct use of resampling methods and hypothesis testing in the comparison of machine learning methods.
- We give the results of a survey of over 1500 papers published in the last two years in three major bioinformatics journals to check for the current practice, good and bad. To summarize, our principal findings are:
  - Most applications are two-class problems.
  - Not only accuracy/misclassification error, but measures such as precision/recall, ROC/AUC-ROC are relevant and indeed are widely used.
  - Most bioinformatics data is not large. Nearly half has fewer than 1000 instances.
  - Most bioinformatics data is high dimensional. Nearly half has more than 1000 dimensions.
  - Dimensionality reduction hence is an important research topic and such methods are heavily used.
  - There does not seem to be any learning method heavily favored. The use of decision trees and support vector machines seem to be slightly more frequent.
  - The need for resampling seems to be accepted by the community. Around 70 per cent of the papers use some sort of cross-validation.
  - Though resampling is popular, statistical tests to check for significant difference is rare, only in 21

TABLE XIV
ERROR RATES OF Rip, Mlp, RnF, Svm, AND Knn ON 11 TUMOR DATA SETS.

Dataset	Rip	Mlp	RnF	Svm	Knn
9tumors	72.73±8.57	86.36±0.00	65.91±9.40	51.36±12.87	62.27±10.29
11tumors	30.00±1.74	60.16±10.08	33.77±5.42	16.72±2.42	33.28±5.89
14tumors	64.41±3.01	82.97±2.63	58.11±3.19	51.71±2.66	70.27±3.96
braintumor1	25.31±6.50	37.50±0.00	31.56±5.60	14.38±3.02	18.75±4.42
braintumor2	42.22±14.63	72.22±0.00	33.33±8.69	31.67±9.09	34.44±9.37
dlbcl	20.74±5.00	24.81±3.51	23.33±2.50	11.11±3.90	12.96±5.59
leukemia1	25.20±8.44	46.00±6.32	21.20±5.98	12.40±6.10	8.80±4.54
leukemia2	20.40±2.95	48.00±12.36	14.00±4.71	7.20±4.54	14.40±10.70
lungtumor	10.00±3.64	24.06±5.56	17.83±3.75	5.51±1.65	13.91±3.82
prostatetumor	25.43±4.94	19.14±4.87	21.43±9.45	10.86±5.68	17.71±4.00
srbct	23.45±6.66	40.35±7.10	12.07±7.13	7.93±6.09	10.69±8.36

per cent. Some papers show only mean and standard deviations without any test, and some use only a single value. This is probably our most significant finding and indicates the relevance of this paper.

- We define four scenarios which we observe frequently in the machine learning applications in bioinformatics and for those, we discuss the proper statistical methodology.
- For each of these scenario, we include a case study where we show an example use of the proposed methodology on a real-world bioinformatics application with state-of-the-art learning algorithms.
- A section on related work shows the evolution of statistical methodology and contains pointers to related papers.

Our discussion in this paper is for classification; though regression algorithms are used less frequently in bioinformatics, a similar study can also be carried out for regression.

ACKNOWLEDGMENTS

We would like to thank the Editor and the Reviewers for their constructive comments, suggestions, pointers to related literature, and pertinent questions which allowed us to better situate our work as well as organize the manuscript and improve the presentation. This work has been supported by the Turkish Scientific Technical Research Council (TÜBİTAK) EEEAG 109E186 and Boğaziçi University Research Funds BAP 5701.

REFERENCES

[1] E. Alpaydın, *Introduction to Machine Learning*, 2nd ed. The MIT Press, 2010.
[2] D. C. Montgomery, *Design and Analysis of Experiments*, 7th ed. Wiley, 2009.
[3] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, pp. 861–874, 2006.
[4] V. V. Raghavan, G. S. Jung, and P. Bollmann, "A critical investigation of recall and precision as measures of retrieval system performance," *ACM Transactions on Information Systems*, vol. 7, pp. 205–229, 1989.
[5] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in *International Conference on Machine Learning*, 2006, pp. 233–240.
[6] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning classifiers," *Neural Computation*, vol. 10, pp. 1895–1923, 1998.
[7] E. Alpaydın, "Combined 5×2 cv F test for comparing supervised classification learning classifiers," *Neural Computation*, vol. 11, pp. 1975–1982, 1999.
[8] B. Efron and R. Tibshirani, "Improvements on cross-validation: The .632+ bootstrap method," *Journal of American Statistical Association*, vol. 92, pp. 548–560, 1997.
[9] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
[10] P. R. Cohen, *Empirical Methods for Artificial Intelligence*. MIT Press, 1995.
[11] T. Mitchell, *Machine Learning*. McGraw-Hill, 1997.
[12] S. L. Salzberg, "On comparing classifiers: Pitfalls to avoid and a recommended approach," *Data Mining and Knowledge Discovery*, vol. 1, pp. 317–328, 1997.
[13] C. Nadeau and Y. Bengio, "Inference for the generalization error," *Machine Learning*, vol. 52, pp. 239–281, 2003.
[14] R. R. Bouckaert, "Estimating replicability of classifier learning experiments," in *International Conference on Machine Learning*, 2004, pp. 15–22.
[15] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 3rd ed. New York: Springer Verlag, 2011.
[16] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, pp. 29–36, 1982.
[17] C. X. Ling, J. Huang, and H. Zhang, "AUC: a better measure than accuracy in comparing learning algorithms," in *International Joint Conference on Artificial Intelligence*. Springer, 2003, pp. 329–341.
[18] J. Huang, J. Lu, and C. Ling, "Comparing naive Bayes, decision trees, and SVM with AUC and accuracy," in *IEEE International Conference on Data Mining*, 2003, pp. 553–556.
[19] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, pp. 1145–1159, 1997.
[20] J. A. Hanley and B. J. McNeil, "A method of comparing the areas under receiver operating characteristic curves derived from the same cases," *Radiology*, vol. 148, pp. 839–843, 1983.
[21] C. Cortes and M. Mohri, "Confidence intervals for the area under the ROC curve," in *Neural Information Processing Systems*. The MIT Press, 2004, pp. 305–312.
[22] H. C. Bravo, G. Wahba, K. E. Lee, B. E. K. Klein, R. Klein, and S. K. Iyengar, "Examining the relative influence of familial, genetic, and environmental covariate information in flexible risk models," in *Proceedings of the National Academy of Sciences*, vol. 106, 2009, pp. 8128–8133.
[23] G. M. Weiss and F. Provost, "Learning when training data are costly: The effect of class distribution on tree induction," *Journal Of Artificial Intelligence Research*, vol. 19, pp. 315–354, 2003.
[24] B. Hanczar, J. Hua, C. Sima, J. Weinstein, M. Bittner, and E. R. Dougherty, "Small-sample precision of roc-related estimates," *Bioinformatics*, vol. 26, no. 6, pp. 822–830, 2010.
[25] S. J. Swamidass, C.-A. Azencott, K. Daily, and P. Baldi, "A croc stronger than roc: Measuring, visualizing, and optimizing early retrieval," *Bioinformatics*, vol. 26, no. 10, pp. 1348–1356, 2010.
[26] A. Folleco, T. M. Khoshgoftaar, and A. Napolitano, "Comparison of four performance metrics for evaluating sampling techniques for low quality class-imbalanced data," in *International Conference on Machine Learning and Applications*, 2008, pp. 153–158.
[27] E. Bloedorn, I. Mani, and T. R. Macmillan, "Machine learning of user profiles: Representational issues," in *National Conference on Artificial Intelligence*, 1996, pp. 433–438.
[28] T. C. W. Landgrebe, P. Paclik, and R. P. W. Duin, "Precision-recall operating characteristic (P-ROC) curves in imprecise environments," in *International Conference on Pattern Recognition*, 2006, pp. 123–127.

[29] S. Clemencon and N. Vayatis, "Nonparametric estimation of the precision-recall curve," in *Proceedings of the 26th Annual International Conference on Machine Learning*, vol. 382, 2009, pp. 185–192.

[30] S. B. Bengio, J. Marithoz, and M. Keller, "The expected performance curve," in *International Conference on Machine Learning*, 2005, pp. 9–16.

[31] C. Drummond and R. C. Holte, "Cost curves: An improved method for visualizing classifier performance," *Machine Learning*, vol. 65, no. 1, pp. 95–130, Oct. 2006.

[32] O. T. Yıldız and E. Alpaydın, "Ordering and finding the best of $K > 2$ supervised learning algorithms," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 28, no. 3, pp. 392–402, 2006.

[33] P. D. Turney, "Types of cost in inductive concept learning," in *Workshop on Cost-Sensitive Learning in 17th International Conference on Machine Learning*, Stanford University, CA, 2000, pp. 15–21.

[34] A. Ulaş, O. T. Yıldız, and E. Alpaydın, "Cost-conscious comparison of supervised learning algorithms over multiple data sets," *Pattern Recognition*, vol. 45, pp. 1772–1781, 2012.

[35] A. Dean and D. Voss, *Design and Analysis of Experiments*. New York: Springer Verlag, 1999.

[36] S. García and F. Herrera, "An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons," *Journal of Machine Learning Research*, vol. 9, pp. 2677–2694, 2008.

[37] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, vol. 6, pp. 65–70, 1979.

[38] J. P. Shaffer, "Modified sequentially rejective multiple test procedures," *Journal of the American Statistical Association*, vol. 81, no. 395, pp. 826–831, 1986.

[39] G. Bergmann and G. Hommel, "Improvements of general multiple test procedures for redundant systems of hypotheses," in *Multiple Hypotheses Testing*, P. Bauer, G. Hommel, and E. Sonnemann, Eds., 1988, pp. 100–115.

[40] L. J. Jensen and A. Bateman, "The rise and fall of supervised machine learning techniques," *Bioinformatics*, vol. 27, no. 24, pp. 3331–3332, 2011.

[41] C. C. Chang and C. J. Lin, *LIBSVM: a library for support vector machines*, 2001. [Online]. Available: http://www.csie.ntu.edu.tw/∼cjlin/libsvm

[42] W. W. Cohen, "Fast effective rule induction," in *International Conference on Machine Learning*, 1995, pp. 115–123.

[43] O. T. Yıldız and E. Alpaydın, "Linear discriminant trees," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 19, no. 3, 2005.

[44] D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman, "A generalized hidden markov model for the recognition of human genes in dna," in *International Conference on Intelligent Systems for Molecular Biology*, 1996.

[45] A. Statnikov, C. Aliferis, I. Tsamardinos, D. Hardin, and S. Levy, "A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis," *Bioinformatics*, vol. 21, pp. 631–643, 2005.

[46] J. Song, H. Tan, H. Shen, K. Mahmood, S. E. Boyd, G. I. Webb, T. Akutsu, and J. C. Whisstock, "Cascleave: towards more accurate prediction of caspase substrate cleavage sites," *Bioinformatics*, vol. 26, pp. 752–760, 2010.

[47] J. C. Jeong, X. Lin, and X.-W. Chen, "On position-specific scoring matrix for protein function prediction," *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 8, pp. 308–315, 2011.

[48] N. J. MacDonald and R. G. Beiko, "Efficient learning of microbial genotype-phenotype association rules," *Bioinformatics*, vol. 26, pp. 1834–1840, 2010.

[49] E. Tapia, L. Ornella, P. Bulacio, and L. Angelone, "Multiclass classification of microarray data samples with a reduced number of genes," *BMC Bioinformatics*, vol. 12, pp. 1471–2105, 2011.

[50] S. Zhu, D. Wang, K. Yu, T. Li, and Y. Gong, "Feature selection for gene expression using model-based entropy," *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 7, pp. 25–36, 2010.

[51] Z. Liu, S. Lin, and M. T. Tan, "Sparse support vector machines with l_p penalty for biomarker identification," *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 7, pp. 100–107, 2010.

Ozan IRSOY received his BA in Mathematics and BSc in Computer Engineering (double major) from Boğaziçi University in 2012 and is starting soon the PhD program in computer science at Cornell University.

Olcay Taner YILDIZ received his BSc, MSc, and PhD degrees in computer science from Boğaziçi University in 1997, 2000, and 2005. He did postdoctoral work at the University of Minnesota in 2005. He is associate professor of Computer Engineering at Işık University. He worked on machine learning, specifically model selection and decision trees. His current research is on software engineering, natural language processing, and bioinformatics.

Ethem ALPAYDIN received his BSc from Boğaziçi University in 1987 and PhD from EPF Lausanne in 1990. He did his postdoctoral work at ICSI, Berkeley in 1991 and he visited MIT in 1994, ICSI, Berkeley in 1997 (as a Fulbright scholar) and IDIAP, Switzerland in 1998. He is a professor of Computer Engineering at Boğaziçi University since 1991. His book Introduction to Machine Learning was published by The MIT Press in 2004, which since has been translated to German, Chinese and Turkish.