Ordering and Finding the Best of $K > 2$ Supervised Learning Algorithms

Olcay Taner Yildiz and Ethem Alpaydin, *Senior Member*, *IEEE*

Abstract—Given a data set and a number of supervised learning algorithms, we would like to find the algorithm with the smallest expected error. Existing pairwise tests allow a comparison of two algorithms only; range tests and ANOVA check whether multiple algorithms have the same expected error and cannot be used for finding the smallest. We propose a methodology, the MultiTest algorithm, whereby we order supervised learning algorithms taking into account 1) the result of pairwise statistical tests on expected error (what the data tells us), and 2) our prior preferences, e.g., due to complexity. We define the problem in graph-theoretic terms and propose an algorithm to find the "best" learning algorithm in terms of these two criteria, or in the more general case, order learning algorithms in terms of their "goodness." Simulation results using five classification algorithms on 30 data sets indicate the utility of the method. Our proposed method can be generalized to regression and other loss functions by using a suitable pairwise test.

Index Terms—Machine learning, classifier design and evaluation, experimental design.

1 Introduction

In machine learning literature, there exist several supervised learning algorithms, and for any application, we need to find the algorithm that generalizes the best, i.e., the one with the smallest probability of misclassifying an instance unseen during training. To check for a statistically significant difference, the algorithms are trained and tested on several training, validation folds and a sample of validation errors are obtained. Statistical tests are used to compare the means of populations which these validation error values are sampled from, i.e., their expected error.

Tests in the literature are pairwise and compare the means of two populations [1], [2]. Generally, these tests are two-sided and, in our case of expected error comparison, check whether two supervised learning algorithms yield the same expected error. If the test accepts, we conclude that they have the same expected error, but if the test rejects and there is a statistically significantly difference, we do not know which one is smaller; that requires a one-sided test.

To the best of our knowledge, there exists no method in the statistical literature that, given $K > 2$ populations, finds the population with the statistically significantly smallest mean, or in our case, finds the supervised learning algorithm with the smallest expected error. Finding the smallest mean is a particular case of the more general problem of ordering $K > 2$ populations in terms of increasing mean, which can be solved by iteratively applying the same method and removing the smallest at each iteration. Such a method is the topic of this paper.

This paper is organized as follows: Section 2 contains a literature survey on statistical methods used for comparing the error rates of learning algorithms. The MultiTest algorithm is given in Section 3. In Section 4, we give experimental results using five classification algorithms on 30 data sets, and we conclude in Section 5.

• *The authors are with the Department of Computer Engineering, Boğaziçi University, TR-34342, Istanbul, Turkey.*
E-mail: yildizol@cmpe.boun.edu.tr, alpaydin@boun.edu.tr.

Manuscript received 17 Jan. 2005; revised 12 July 2005; accepted 26 July 2005; published online 13 Jan. 2006.
Recommended for acceptance by L. Kuncheva.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0034-0105.

2 Statistical Methods for Comparing the Error Rates of Algorithms

2.1 Resampling Methods

In order to compare the error rates of learning algorithms, a sample of their error values are obtained by training the algorithm on a *training set* and then testing it on a *validation set*. We do this training and validation multiple times to be able to estimate the distribution of validation error values; typically, we would like to estimate the *expected error rate* of a learning algorithm, as well as how much the error rate can vary around this expected value in any run.

When the data set is small, we need resampling methods to generate multiple training/validation sets from a single data set. In *k-fold cross-validation,* we divide the data into k equal parts and in each fold, we use one part for validation and the remaining $k-1$ parts for training. Note that two training sets share $k-2$ parts. Typically, k is taken as 10 or 30.

In 5×2 *cross-validation* [1], we do two-fold cross-validation five times, where, at each time, we randomly divide the data into two, use one part for training, use the other for validation to get the first half of the fold, and swap the roles of the two parts to get the second half of the fold, giving us a total of 10 training and validation set pairs to use. The choice of five replications is not arbitrary: fewer replications gives us few data points (to estimate distribution parameters, e.g., mean) and increases the variance of estimates made; with more replications, the folds overlap too much and the independence assumption of folds may no longer be tenable.

In *bootstrap* [3], we randomly draw instances with replacement from a data set where each drawn sample

contains the same number of instances as the data set. This is done k times, producing k bootstrap data sets for training, and the full set is used for estimating the generalization error. Since the bootstrap data sets overlap, the error estimates look unrealistically good.

2.2 Hypothesis Testing

In hypothesis testing, we use a sample to test a hypothesis about the population from which the sample is drawn. We define a statistic that falls in a certain interval if the *null hypothesis* H_0 holds. We calculate the value of the statistic using the sample, accept the hypothesis if the statistic is indeed in the interval, and reject it otherwise. If we reject when the hypothesis is correct, this is called *type I error*; if we accept a hypothesis when it is not correct, this is called *type II error*. The *power* of a test is the probability of rejecting the null hypothesis when it is not correct. We want hypothesis tests to have small type I and type II errors (or large power).

For example, let us say we have a sample $\mathcal{X} = \{x^t\}_{t=1}^N$ drawn from unknown $p(x)$ with unknown mean μ, and we want to check if μ is equal to a given value of μ_0

$$H_0 : \mu = \mu_0 \text{ versus } H_1 : \mu \neq \mu_0. \tag{1}$$

The sample estimate of μ is $m = \sum_t x^t/N$ and we accept the hypothesis if $m \in (\mu_0 - c, \mu_0 + c)$, or equivalently, if $|m - \mu_0| < c$, where c depends on $p(x^t)$ (assumed symmetric here, e.g., Gaussian, t, etc.) and the significance level α. Equation (1) can be rewritten as $H_0 : \mu_1 - \mu_2 = 0$ versus $H_1 : \mu_1 - \mu_2 \neq 0$, and we test if the mean of the sample of *paired differences* is 0 or not.

If we would like to test whether one mean is smaller, i.e., $H_0 : \mu_1 \leq \mu_2$ versus $H_1 : \mu_1 > \mu_2$, this is a *one-sided test*, and can be rewritten as $H_0 : \mu_1 - \mu_2 \leq 0$ versus $H_1 : \mu_1 - \mu_2 > 0$. We reject the null hypothesis if $(m_1 - m_2) > c$, and accept it otherwise., i.e., if $(m_1 - m_2) \in (-\infty, c)$.

2.3 Pairwise Tests for Two Populations

2.3.1 Notation

r_{ij}^t denotes the desired output, $f(x_{ij}^t)$ denotes the predicted output by the fitted model (in classification, r_{ij}^t, $f(x_{ij}^t) \in \{0, 1\}$), and X_{ij}^t denotes the Bernoulli random variable representing the outcome of classifier $i = 1, \ldots, K$ on instance $t = 1, \ldots, N$ of validation fold $j = 1, \ldots, L$. $X_{ij}^t = 1$ if $r_{ij}^t \neq f(x_{ij}^t)$, 0 otherwise. The average error of learner i on fold j is

$$Y_{ij} = \frac{\sum_{t=1}^N X_{ij}^t}{N}. \tag{2}$$

Y_{ij} are the sum of independent and identically distributed random variables (X_{ij}^t) and by the central limit theorem are approximately normal distributed with mean μ_i. For each classification algorithm i, the *expected error* is the sample average m_i (estimator to μ_i), calculated as

$$m_i = \frac{\sum_{j=1}^L Y_{ij}}{L}. \tag{3}$$

2.3.2 The k-Fold Cross-Validation t Test

This is a two-sided test which uses k-fold cross-validation to generate k training/validation folds and then uses a paired t test for $H_0 : \mu_1 - \mu_2 = 0$ versus $H_1 : \mu_1 - \mu_2 \neq 0$. p_i is the paired difference between the errors on fold i; so, $p_1 = Y_{11} - Y_{21}$. The estimators for the mean and variance of the differences are ($L = k$, the number of folds)

$$m = \frac{\sum_{i=1}^L p_i}{L} \quad S^2 = \frac{\sum_{i=1}^L (p_i - m)^2}{L - 1}. \tag{4}$$

Under the null hypothesis that the two error rates are the same, the paired differences are approximately normally distributed with 0 mean and unknown variance σ^2, and

$$t' = \frac{\sqrt{L}m}{S} \tag{5}$$

is t with $L - 1$ degrees of freedom. H_0 is accepted with α confidence if $t' \in (-t_{\alpha/2,L-1}, t_{\alpha/2,L-1})$. Similarly, the one-sided test accepts $H_0 : \mu_1 \leq \mu_2$ if $t' \in (-\infty, t_{\alpha,L-1})$.

2.3.3 The 5×2 cv t Test

This is a paired t test using 5×2 cross-validation with $H_0 : \mu_1 = \mu_2$ [1]. $p_i^{(j)}$ is the difference between the errors on replication i of fold j; so $p_i^{(1)} = Y_{1(2i-1)} - Y_{2(2i-1)}$ and $p_i^{(2)} = Y_{1(2i)} - Y_{2(2i)}$. s_i^2 is the estimated variance of the replication i: $s_i^2 = (p_i^{(1)} - \overline{p_i})^2 + (p_i^{(2)} - \overline{p_i})^2$, where $\overline{p_i}$ is the average of the differences on replication i: $\overline{p_i} = (p_i^{(1)} + p_i^{(2)})/2$. Under H_0, $p_i^{(j)}/\sigma$ is unit normal ($\mathcal{Z}$), and assuming that $p_i^{(1)}$ and $p_i^{(2)}$ are independent normals, s_i^2/σ^2 is chi-square distributed with one degree of freedom ($\mathcal{X}_1^2$). Assuming also that the s_i^2 are independent,

$$M = \frac{\sum_{i=1}^5 s_i^2}{\sigma^2} \tag{6}$$

is chi-square distributed with five degrees of freedom. We know that if $Z \sim \mathcal{Z}$ and $X \sim \mathcal{X}_n^2$ and if Z and X are independent, then $Z/\sqrt{X/n}$ is t distributed with n degrees of freedom. Then,

$$t' = \frac{p_1^{(1)}/\sigma}{\sqrt{M/5}} = \frac{p_1^{(1)}}{\sqrt{\sum_{i=1}^5 s_i^2/5}} \tag{7}$$

is t-distributed with five degrees of freedom. Dietterich [1] proposed the two-sided test where $H_0 : \mu_1 = \mu_2$ is accepted with α confidence if $t' \in (-t_{\alpha/2,5}, t_{\alpha/2,5})$, and has shown this to have lower type I error than the k-fold t test. We can derive a one-sided test and accept $H_0 : \mu_1 \leq \mu_2$ if $t' \in (-\infty, t_{\alpha,5})$. This is the one-sided pairwise test we use in the rest of the paper. The combined 5×2 cv F test [4] is an improved version of the 5×2 cv t test, but is two-sided and because it uses the squares of the differences, it cannot be used to define a one-sided test.

2.4 Multiple Populations

2.4.1 Anova Test

Analysis of variance (Anova) [5] tests whether K samples are drawn from populations with the same mean, and can

be used to test whether $K > 2$ learning algorithms induce learners with the same expected error:

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_K. \tag{8}$$

To apply Anova, we calculate the mean standard error between the sample means

$$MST = L\frac{\sum_{i=1}^{K}(m_i - m)^2}{K-1}, \tag{9}$$

where $m = \sum_i m_i/K$ is the grand mean. The total mean standard error around sample means is

$$MSE = \frac{\sum_{i=1}^{K}\sum_{j=1}^{L}(Y_{ij} - m_i)^2}{KL-K} \tag{10}$$

and the test statistic for Anova

$$f = \frac{MST}{MSE} \tag{11}$$

under the null hypothesis that all population means are equal, is F distributed with $K-1, K(L-1)$ degrees of freedom, and H_0 is accepted with α confidence if $f \in (0, F_{\alpha,K-1,K(L-1)})$. If Anova accepts, then all algorithms induce learners with the same expected error and we can choose any of them. If Anova rejects, there is an inequality somewhere but we cannot pinpoint the algorithm with the smallest expected error.

2.4.2 Newman-Keuls Test

A multiple range test, similar to Anova, checks for the equality of the means of subsets of populations. One such test is the Newman-Keuls test. There are also multiple range tests due to Duncan and Tukey, but the Newman-Keuls test is favored over them ([6], p. 87). In our case of comparing expected error, the multiple range test is used to find subsets of algorithms with the same expected error. For example, given algorithms 1, 2, 3, 4, 5, a range test can conclude as

$$\underline{5\ 2\ \underline{4}\ 3}\ 1.$$

The algorithms are sorted in ascending average error. An underline implies that there is no statistically significant difference between the means of the underlined populations. In the example above, the test concludes that there is no statistically significant difference between the expected errors of 5, 2, and 4, and also that there is no difference between 4 and 3. We see that, for example, there is difference between 2 and 3, and also between 3 and 1. Note that a range test checks for equality and a rejection, that is, the absence of an underline, does not imply an ordering. For example, we know that 3 and 1 have significantly different expected errors and that the average of errors of 3 over the validation folds is less than the average of errors of 1, but this does not imply that 3 has *significantly* less error than 1; it might, or it might not; a range test does not check for this.

In the Newman-Keuls test, we first test for equality of K means $(m_1, \ldots, m_K)$. If they are equal, we stop. Otherwise, we check for equality of $K-1$ means $((m_1, \ldots, m_{K-1}), (m_2, \ldots, m_K))$, ..., up to two means $((m_1, m_2), (m_2, m_3), \ldots, (m_{K-1}, m_K))$. The test statistic to compare P means between m_i and m_j in $(m_i, \ldots, m_j)$ is

$$t = (m_j - m_i)\sqrt{\frac{L}{MSE}} \tag{12}$$

and the null hypothesis $H_0 : m_i = \ldots = m_j$ is accepted with α level of confidence if $t \in (0, q_{\alpha,K(L-1),P})$, where q is the studentized range distribution. If the test accepts, we do not check for any of the subsets between m_i and m_j, and assume all to have equal means.

2.5 Related Work

Comparing statistics from multiple populations is called *multiple comparison procedures* [7], [8], and failures to adjust the statistical properties of multiple comparisons may lead to attribute selection errors, overfitting, and oversearching [9]. Several solutions were proposed to overcome these errors, including randomization, cross-validation and Bonferroni adjustment. The disadvantages of these solutions are: Randomization tests are computationally expensive; they require k randomized samples where k must be > 100 if one wants to make distinctions between probabilities that differ by less than 1 percent. Cross-validation is also computationally expensive, although not as much as randomization (k = 10) and the results can be highly variable. Although Bonferroni correction is well suited for our purposes of getting α level of confidence for hypotheses $H_1, H_2, \ldots, H_n$ simultaneously, we must set very high significance levels to each hypothesis, and in that case, we may reject hypotheses which have high significance individually.

In addition to the parametric tests we discussed, there are also nonparametric tests [10]; for example, Kruskal-Wallis' test is the nonparametric version of Anova. In contrast to parametric tests, nonparametric tests do not assume a particular population probability distribution. Contingency table analysis may also help in finding out if two or more algorithms have the same expected error. These tests use a single training/validation set and can be used in cases where learning and/or validation is so costly that they can only be done once, assuming that the internal variability of the supervised learning algorithms is small.

McNemar's test [11] is such a pairwise test having lower type I error and reasonable power [1]. Let n_{01} denote the number of instances misclassified by the first classifier but not by the second and n_{10} denote the number of instances misclassified by the second but not by the first. We accept the null hypothesis that both classifiers have the same mean with α level of confidence if

$$t = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}} > \chi^2_{1,\alpha}. \tag{13}$$

Similarly, Looney's test [12] uses contingency table analysis and checks for the equality of K means, as in Anova. Since it checks whether multiple populations have the same mean, it cannot be used for finding the population with the smallest mean. On the other hand, when training/validation can be done only once, it is used instead of Anova. Let $p_{\cdot j}$ denote the proportion of N test instances that are correctly classified by classifier j, $p_{i\cdot}$ denote the

proportion of the K classifiers that correctly classify test instance i, and $p_{..}$ denote the total proportion of correct classifications. Then, the sum of squares for classifiers and the sum of squares for test instances are

$$SSA = N\sum_{j=1}^{K} p_{.j}^2 - NKp_{..}^2 \quad SSB = K\sum_{i=1}^{N} p_{i.}^2 - NKp_{..}^2 \quad (14)$$

and the total sum of squares and the sum of squares for classifier and instance interaction are

$$SST = NKp_{..}(1-p_{..}) \quad SSAB = SST - SSA - SSB. \quad (15)$$

Then, the ratio

$$F = \frac{MSA}{MSAB} = \frac{SSA/(K-1)}{SSAB/(K-1)(N-1)} \quad (16)$$

has an F distribution with $(K-1)$ and $(K-1)(N-1)$ degrees of freedom. Looney prefers the adjusted F^+ test, where the degrees of freedoms are multiplied with an appropriate $\hat{\epsilon}$ adjustment.

Dietterich has shown that the 5×2 cv t test has low power and the k-fold cv t test has high type I error [1]. Bouckaert [13] claims that the reuse of the same data causes the effective degrees of freedom to be lower than theoretically expected and calibrates the effective degrees of freedom empirically. On synthetic problems with binary features, the calibrated tests perform better when compared to 5×2 and k-fold tests proper.

Receiver Operating Characteristic (ROC) analysis is another tool to compare the performances of classification algorithms. ROC graphs depict trade-offs between true positive and false positive rates. In applications where the class distributions become skewed or when misclassification losses are not equal, accuracy-based comparisons may break down. Provost et al. [14] suggest using ROC analysis in classifier comparisons and, to this aim, propose the incremental ROC convex hull method (ROCCH) [15], which allows clear visual comparisons and sensitivity analyses. ROCCH selects the classifiers that are potentially optimal; therefore, only these classifiers must be kept for further comparisons. ROCCH is an incremental algorithm and incorporating new classifiers is an easy task. Provost and Fawcett also argue that their algorithm allows studying important phenomena without having precise prior class and cost distributions. The most important limitation of ROCCH is that it is only applicable to binary-class problems.

In the multinomial selection problem (MSP) procedure [16], for each test data point of class j, we compare class j posterior probabilities of each classifier and select the one with the maximum posterior. The best classifier is the one that has the maximum number of wins across the test set. With this procedure, it makes sense to compare algorithms which produce sensitive posterior probabilities. For example, with k-nearest neighbor, posterior probabilities are multiples of $1/k$, whereas with a neural network, one can have virtually any value between 0 and 1.

3 The Multitest Algorithm

3.1 Ordering Two Learning Algorithms

Though expected error is the most important criterion in favoring one algorithm over another, it is by no means the only one. Various measures of cost, e.g., space/time complexity of training and test, costs of features (if learners use different input representations), interpretability, and ease of programmability affect our choosing an algorithm over another one; various types of cost are discussed in detail in [17]. Then, if we have two or more algorithms with the same expected error, it is logical that we choose the simplest one based on the cost measure we consider important in that particular application.

In this work, we are going to use the fact that, given a number of supervised learning algorithms for a particular application, we can always order them in terms of such a preference, e.g., based on their complexities. That is, given any two algorithms, if they have the same expected error, we favor one over another based on this preference. In our method, this information of our *prior* preferences (before looking at the data) is combined with what the data tells us through the statistical tests, to give us a final ordering of the algorithms. But, the expected error remains the most important criterion and it overrides our prior preference.

For comparing error rates, we use a one-sided test and use the prior ordering based on preference in choosing how to apply the test. We only test for

$$H_0 : \mu_1 \leq \mu_2, \quad (17)$$

where we have a prior preference of 1 over 2, e.g., because it is simpler. Since we assume a prior ordering and we would like to test whether it holds, the hypothesis follows the prior and is one-sided. If the test accepts, we choose 1 over 2, either because 1) $\mu_1 < \mu_2$, i.e., 1 has less expected error than 2; in such a case, 1 is chosen over 2 both because it has less expected error and it is simpler, or 2) $\mu_1 = \mu_2$, i.e., they have the same expected error; we can choose either one and we choose the preferred one. Only when the test rejects, do we know that $\mu_1 > \mu_2$, and in this case, we choose 2 over 1, the test overriding our prior preference.

Therefore, we choose an algorithm A over another one B, and say that A is "better than" B by taking into account both our prior preferences of A and B and the statistical test comparing their expected error. A is better than B either because 1) A has less expected error than B, or 2) they have the same expected error and A is preferred to B, e.g., because it is simpler.

In the next section, we discuss how to combine the results of such pairwise orders to find the best of $K > 2$ algorithms, or order them in terms of their "goodness" in the general case.

3.2 Combining Pairwise Orders

Before applying the tests, we assume that we are given a full, linear ordering of supervised learning algorithms in terms of our prior preferences. Depending on the particular requirements and constraints of the particular application at hand, such a linear ordering can always be defined. We denote supervised learning algorithms by their indices in this ordering as $1, 2, \ldots, K$, such that 1 is the most preferred,

```
1  Best MultiTest(1, ..., K; T; α)
2      E ← ∅ /* Edges of the graph */
3      for i = 1 to K
4          for j = i + 1 to K
5              Test H0: μi ≤ μj using T
6              if H0 rejected E ← E ∪ e[i, j] (α/(K(K − 1)/2)) /* Bonferroni */
7      Sk = {x : ∀e[j, k] ∈ E, j ≠ x}; /* Find the nodes with no outgoing edges */
8      l = ∀j ∈ Sk, l ≤ j; /* Select node l with the lowest index */
9      return l;
```

Fig. 1. Pseudocode of MultiTest: $1, \ldots, K$: Algorithms in decreasing order of prior preference, T: The one-sided test used for pairwise comparison, α: Required confidence level. Lines 3-6 form the directed graph, lines 7-9 find the "best." If we iterate lines 7-9, removing l and edges incident to it after each iteration, we get an ordering in terms of "goodness."

2 is preferred over $3, 4, \ldots, K$, and K is the least preferred. We then apply the one-sided test of (17) as

$$H_0 : \mu_i \leq \mu_j, \tag{18}$$

where $i < j$ with $i, j = 1, \ldots, K$. There are $K(K-1)/2$ tests to be applied. Note that the real cost is the training and validating the K algorithms L times, where L is the number of folds, e.g., 10 if we are using 5×2-fold cross-validation. Once the K algorithms are trained and validated L times and the $K \cdot L$ validation errors are recorded, applying the tests is simple.

We use the one-sided version of the 5×2 cv t test that we discuss in Section 2.3.3 because it has lower type I error than the 10-fold cv test [1]. The confidence of the one-sided test of (18) is set to $\alpha/(K(K-1)/2)$. This is because we are making $K(K-1)/2$ multiple tests to get the resulting final ordering and to have a confidence level of α, we need this Bonferroni correction [7]. When K is large, Bonferroni correction may be too conservative and we can use Holm correction [18] instead to have higher confidence levels for the one-sided statistical tests.

We use graph theory to represent the result of the tests. The algorithm is given in Fig. 1. The graph has K vertices corresponding to the K algorithms. For all $i < j$ where $i, j = 1, \ldots, K$, we test $H_0 : \mu_i \leq \mu_j$, and if the test *rejects*, we place a directed edge from i to j to indicate that we are overriding the prior order. This corresponds to a binary relation R defined on the set of supervised learning algorithms where jRi implies that $j > i$ and j has significantly less expected error than i ($\mu_i \leq \mu_j$ is rejected). If we have $jR\!\!\!/i$, this means that the hypothesis test is accepted and our choice of i over j stands. The resulting directed graph has thus directed edges where the test is rejected for its incident vertices. The number of incoming edges to a node j is the number of algorithms that are preferred over j but have higher expected error. The number of outgoing edges from a node i is the number of algorithms that are less preferred than i but have less expected error. The resulting graph need not be connected. For example, when all algorithms have the same expected error, there are no edges.

Once the directed graph is formed, we choose the "best" node. For this:

1. We find the nodes with no outgoing edges to produce the set S_k. If there is no outgoing edge from a node, there is no other algorithm that has less expected error.
2. From S_k, we select the node with the lowest index and report it. This selected algorithm is the one that is the most preferred among all with the least expected error.

This calculates the "best"; if we want to find an ordering, we iterate steps 1 and 2 above, removing the best node and its incident edges at each iteration to get a *topological sort* [19].

Fig. 2 shows a sample execution of the MultiTest algorithm on four ($K = 4$) algorithms numbered 1 to 4. They are sorted in decreasing order of prior preference 1, 2,

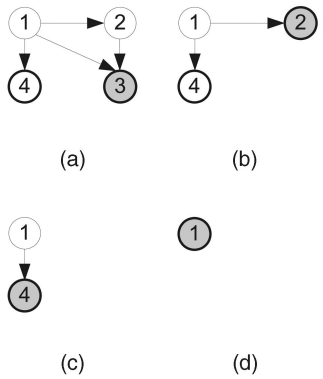

Fig. 2. Sample execution of the MultiTest algorithm on four algorithms, 1, 2, 3, 4, in decreasing order of preference (increasing order of complexity). Nodes with thick lines indicate candidates at each step and among them, the one with the lowest index (the most preferred) is taken (shown shaded). The best one is 3 and if we continue iterating the ordering found is $\mathbf{3} < 2 < 4 < 1$.

TABLE 1
Average and Standard Deviations of the Misclassification Error Rates of Five Classification Algorithms on 5×2-Fold in 1,000 Runs

Dataset	MAX (1)	NMC (2)	LGC (3)	C4.5 (4)	NN (5)
balance	55.46±1.29	54.01±2.88	4.10±1.59	40.26±7.53	40.10±2.81
breast	34.48±1.81	3.80 ±0.86	3.11±0.70	6.88 ±1.69	4.72 ±0.88
bupa	42.09±2.84	42.83±4.23	31.93±2.93	40.15±4.66	39.63±3.34
car	29.98±1.12	47.77±2.25	6.89±0.77	15.55±1.95	21.82±1.36
cmc	57.30±1.27	57.65±2.02	49.21±1.39	53.22±3.46	55.72±1.51
credit	44.55±2.08	19.16±1.85	14.29±1.41	14.85±1.87	22.68±1.88
cylinder	42.22±2.13	38.75±2.93	27.31±2.46	33.32±4.84	26.68±2.40
dermatology	69.43±2.50	4.23 ±1.30	2.73±0.92	8.42 ±3.55	6.00 ±1.42
ecoli	57.44±2.67	21.28±4.07	13.95±1.96	23.65±4.61	20.01±2.53
flare	11.15±1.76	34.30±7.64	12.09±1.64	11.26±1.80	17.06±2.70
glass	67.51±3.14	54.35±5.83	37.35±3.53	42.41±8.77	32.45±3.99
haberman	26.47±2.54	28.28±3.80	25.62±1.27	26.90±2.80	33.77±2.96
hepatitis	20.65±3.22	19.03±3.51	17.25±3.44	21.24±3.91	18.78±3.51
horse	36.96±2.54	24.98±2.92	13.74±2.23	23.11±6.67	21.96±2.55
iris	70.68±2.25	13.59±3.53	4.14±1.89	6.29 ±4.47	6.51 ±2.38

3, and 4. After applying $4(4-1)/2$ tests, let us say that we place edges from 1 to 2, 1 to 3, 1 to 4, and 2 to 3 (shown in Fig. 2a). Then, we start generating the full ordering: There are two nodes, 3 and 4, having no outgoing edges (shown with a thicker line); this means that there is no algorithm having less expected error than these two. In this case, we choose the simpler of them, 3, as the best algorithm (shown shaded). Assuming that we want not only to find the best one but to order all algorithms, we continue. We remove node 3 and all edges incident to 3 and have the graph shown in Fig. 2b. Here, nodes 2 and 4 have no outgoing edges, so we choose the simpler 2 as the second best. After removing node 2 and its incident edges, we have the graph shown in Fig. 2c. Now, 4 has no outgoing edge and we select it as the third and 1 as the last (Fig. 2d).

4 Simulation Results

4.1 Learning Algorithms Used

The five classification algorithms we use in decreasing prior preference because of their increasing time/space complexity are (c is the number of classes, d is the number of inputs, and N is the number of training instances):

1. MAX decides based on the prior class probability without looking at the input. All test instances are assigned to the class with the MAXimum prior. It has c parameters. It is not a learning algorithm in the usual sense, but *any* plausible learning algorithm must have a smaller error rate than MAX; it is indeed surprising that MAX is sometimes quite accurate.
2. NMC is the Nearest Mean Classifier, which keeps the mean vector for each class and assigns instance to the class whose mean has the smallest Euclidean distance to the instance [2]. It has $c \cdot d$ parameters. This corresponds to assuming that classes are Gaussian distributed with a shared covariance matrix whose diagonals are equal and whose off-diagonals are 0.
3. LGC [20] is the LoGistic Classification algorithm with a linear model. We use gradient-descent for learning. Discrete features are converted to numeric features by 1-of-n encoding. It has $c(d+1)$ parameters.
4. C4.5 [21] is the archetypal decision tree method. We use postpruning with 20 percent of the data reserved for pruning. The tree changes depending on the problem but generally the complexity of the tree is between those of linear and nearest-neighbor estimators.
5. NN [20] is the 1-Nearest Neighbor classification algorithm and uses the Euclidean distance. It has $N \cdot d$ parameters. This is the most complex (least preferred) classifier since it stores all training data and it is also the most time consuming algorithm.

4.2 Methods Alternative to MultiTest

Comparing with the results of MultiTest, we remember that Anova and Newman-Keuls only check for equality of

TABLE 1 (Continued)
Average and Standard Deviations of the Misclassification Error Rates of Five Classification Algorithms on 5×2-Fold in 1,000 Runs

Dataset	MAX (1)	NMC (2)	LGC (3)	C4.5 (4)	NN (5)
ironosphere	35.90±2.56	20.16±5.26	12.13±1.84	13.59±4.53	14.77±2.57
monks	51.93±1.47	33.82±2.27	33.89±2.31	14.99±8.34	26.41±4.96
mushroom	48.21±0.57	12.64±0.44	0.09±0.06	0.11 ±0.13	0.00 ±0.01
nursery	67.02±0.44	62.90±2.71	7.51±0.25	6.54 ±0.48	23.86±0.55
optdigits	90.71±0.34	9.47 ±0.57	4.34±0.41	15.92±1.27	2.94 ±0.35
pendigits	90.05±0.25	15.70±0.49	6.24±0.40	7.45 ±0.73	0.74 ±0.13
pima	34.90±1.72	26.82±1.88	23.35±1.60	29.68±4.24	30.37±1.91
segment	86.77±0.39	15.66±0.97	8.81±0.75	8.00 ±1.22	5.07 ±0.66
spambase	39.40±0.71	10.56±0.61	8.15±0.52	9.89 ±1.04	10.35±0.61
tictactoe	34.66±1.54	32.87±3.31	1.70±0.40	23.51±3.27	31.57±1.63
vote	38.62±2.32	12.69±1.86	4.86±1.32	4.53 ±1.12	7.30 ±1.59
wave	66.91±0.82	18.94±0.61	13.70±0.53	25.62±1.39	23.84±0.70
wine	62.94±5.02	3.42 ±1.62	2.41±1.67	13.93±6.08	5.40 ±2.19
yeast	69.53±1.59	48.63±1.75	41.10±1.32	48.79±4.63	49.19±1.50
zoo	59.47±4.99	7.98 ±4.14	5.88±2.76	19.17±9.23	7.70 ±4.42

means and do not provide ordering, whereas the pairwise test used by MultiTest is for ordering. The result of Anova can be used to choose the best one only when it accepts and in this case, we choose the simplest algorithm. The result of Newman-Keuls can be used to find the best learner if one of the following conditions hold:

- The first one, namely the algorithm with the smallest average, is not underlined. For example, if Newman-Keuls result is $3\ \underline{5\ 4}\ \underline{2\ 1}$, the best can be taken as 3.
- There is a line under the first one and this line does not overlap with any other line(s), e.g., if Newman-Keuls result is $\underline{5\ 4\ 3}\ 2\ 1$, the best is 3 because it is simpler than 4 and 5.
- There is a line under the first one and this line overlaps with one or more lines but the overlap does not include the first one; e.g., with $\underline{2\ 4\ 5}\ \underline{3\ 1}$, the best is 2.
- If we have the case above and the overlap does not contain a simpler algorithm, the most simple is selected as the best; e.g., if Newman-Keuls result is $\underline{5\ 2\ 4\ 3\ 1}$, the best is 2.

If neither of these four cases occur, Newman-Keuls test cannot yield the best. For example, if Newman-Keuls result is $\underline{5\ 4\ 2\ 1}\ 3$ the first underline chooses 2, the second underline chooses 1, which is simpler than 2. But we cannot choose 1 as it has higher error than 5. These indicate that Anova and Newman-Keuls test results should be further processed to generate orderings and in certain cases, they may be unable to generate an ordering. This is expected because they, unlike MultiTest, are designed not to generate orderings but to find subsets of equality.

For comparison, we also propose a simple method that one would normally use to find the best. This *TestFirst* algorithm sorts the algorithms in increasing order of average error. Then, TestFirst *tests* the *first* algorithm, the algorithm with the smallest average error, to be the best by comparing it with the other $K-1$ algorithms using the one-sided pairwise test and accepts it if 1) it has less expected error, or 2) it has equal expected error and is simpler. Otherwise, TestFirst cannot find the best. For example, if the order in terms of average error is $3 < 4 < 1 < 2 < 5$, TestFirst tries to select 3 as the best. Since 3 is simpler than 4 and 5, the first condition holds for those two. Since 3 is more complex than 1 and 2, we check if 3 has less error than 1 and 2. If so, 3 is the best; otherwise, we say that TestFirst cannot find the best.

4.3 Results

These classification algorithms are tested on 30 data sets from the UCI Machine Learning Repository [22]. The overall α is taken as 0.05 and Bonferroni correction is used. The full results on all 30 data sets are given in Tables 1 and 2. In Table 1, we report the average and standard deviations of the error rates on validation folds of 30 data sets in 1,000 runs. In Table 2, we give the most and second most selected algorithms and their percentages by TestFirst,

TABLE 2
Most and Second Most Selected Algorithms (and Percentages)

Dataset	TestFirst		Newman-Keuls		MultiTest	
	(1)	(2)	(1)	(2)	(1)	(2)
balance	LGC (100)	-	LGC (100)	-	LGC (100)	-
breast	* (100)	-	NMC (100)	-	NMC (100)	-
bupa	* (96)	LGC (4)	LGC (100)	-	MAX (91)	NMC (8)
car	LGC (100)	-	LGC (100)	-	LGC (100)	-
cmc	* (69)	LGC (31)	LGC (100)	-	MAX (60)	NMC (29)
credit	* (89)	LGC (11)	LGC (99)	NMC (1)	NMC (94)	LGC (6)
cylinder	* (86)	LGC (14)	LGC (97)	NN (3)	NMC (44)	LGC (35)
dermatology	* (99)	LGC (1)	NMC (78)	LGC (22)	NMC (100)	-
ecoli	* (95)	LGC (5)	LGC (100)	-	NMC (99)	LGC (1)
flare	MAX (69)	* (31)	MAX (100)	-	MAX (100)	-
glass	* (100)	-	LGC (50)	NN (50)	NMC (56)	LGC (43)
haberman	* (99)	MAX (1)	MAX (100)	-	MAX (100)	-
hepatitis	* (100)	-	MAX (78)	* (16)	MAX (100)	-
horse	* (68)	LGC (32)	LGC (100)	-	NMC (74)	LGC (14)
iris	* (84)	LGC (16)	LGC (100)	-	NMC (88)	LGC (12)

Newman-Keuls and MultiTest on 5×2-fold in 1,000 runs. If a best algorithm cannot be found, we show it by a *.

We discuss our results in more detail on two data sets. On *hepatitis*, all tests says that all five algorithms have the same expected error (Fig. 3). Anova accepts; Newman-Keuls underlines all five. With MultiTest, all pairwise tests are accepted and in the absence of any other information, the classification algorithms are sorted in terms of prior preference. TestFirst cannot find the best algorithm because 3 (LGC) has the smallest average error but it is not the simplest.

On *pendigits* (Fig. 4), there is an exact ordering of algorithms. Anova rejects. Newman-Keuls does not underline any: 5 3 4 2 1. With MultiTest, all tests except $H_0 : \mu_3 \leq \mu_4$ are rejected and the ordering is $5 < 3 < 4 < 2 < 1$ and 5 is chosen. TestFirst also selects 5.

4.4 Discussion

We give the total frequencies of algorithms chosen as best by Anova, Newman-Keuls, TestFirst, and MultiTest in Table 3. Anova can find the best algorithm only if it accepts the null hypothesis, which occurs only 2 percent of the time. In that case, Anova will select the simplest algorithm (in our case, MAX) as the best algorithm. The TestFirst algorithm can find the best algorithm only if the algorithm with the smallest error rate is significantly better than the simpler algorithms, otherwise it cannot find the best algorithm, which occurs in 70 percent of the cases. When simple algorithms do not have the smallest average, or when complex algorithms have the smallest average but do not have significantly less error, TestFirst cannot find the best algorithm. As we have said before, Newman-Keuls is not able to find the best algorithm if none of the four cases applies (Section 4.2). In our simulations, on two data sets, namely, *hepatitis* and *glass*, there are cases where Newman-Keuls cannot find the best algorithm.

On *balance*, *breast*, *car*, *dermatology*, *flare*, *haberman*, *hepatitis*, *mushroom*, *nursery*, *pendigits*, *tictactoe*, *wave*, *wine*, and *zoo*, in most of the cases, Newman-Keuls and MultiTest select the same algorithm as the best algorithm. On the other data sets, the best algorithms selected by MultiTest and Newman-Keuls are different. This occurs because of three reasons:

- Newman-Keuls uses the studentized test, whereas MultiTest (in this case) uses the 5×2 cv t test. The 5×2 cv t test tends to reject H_0: $\mu_i \leq \mu_j$ less than the studentized test. Because the 5×2 cv t test accepts more, it favors simpler algorithms more than Newman-Keuls does.
- Since MultiTest uses a Bonferroni correction, the confidence of each comparison is higher than the original confidence, whereas Newman-Keuls always uses the same confidence. If the confidence level is

TABLE 2 (Continued)
Most and Second Most Selected Algorithms (and Percentages)

Dataset	TestFirst		Newman-Keuls		MultiTest	
	(1)	(2)	(1)	(2)	(1)	(2)
ironosphere	* (99)	LGC (1)	LGC (100)	-	NMC (99)	MAX (1)
monks	* (77)	C4.5 (23)	C4.5 (100)	-	NMC (85)	C4.5 (10)
mushroom	* (100)	-	LGC (100)	-	LGC (97)	C4.5 (3)
nursery	* (89)	C4.5 (11)	LGC (75)	C4.5 (25)	LGC (96)	C4.5 (4)
optdigits	* (74)	NN (26)	NN (100)	-	LGC (89)	NN (11)
pendigits	NN (100)	-	NN (100)	-	NN (100)	-
pima	* (98)	LGC (2)	LGC (99)	NMC (1)	NMC (55)	MAX (45)
segment	* (85)	NN (15)	NN (100)	-	LGC (67)	C4.5 (28)
spambase	* (70)	LGC (30)	LGC (100)	-	NMC (87)	LGC (13)
tictactoe	LGC (100)	-	LGC (100)	-	LGC (99)	NMC (1)
vote	* (90)	LGC (10)	LGC (100)	-	NMC (53)	LGC (47)
wave	LGC (92)	* (8)	LGC (100)	-	LGC (77)	NMC (23)
wine	* (93)	NMC (6)	NMC (99)	LGC (1)	NMC (100)	-
yeast	* (56)	LGC (44)	LGC (100)	-	NMC (78)	LGC (22)
zoo	* (95)	NMC (5)	NMC (99)	LGC (1)	NMC (99)	MAX (1)

higher, the probability of rejecting the null hypothesis will be lower, and as above, MultiTest favors simpler algorithms over more complex ones.

- If Newman-Keuls finds an equality between algorithms i and j, it does not check for algorithms whose average error rates are between those two and assumes that all between i and j have the same expected error. On the other hand, MultiTest does all the pairwise tests, checks for all differences and has more information at its disposal.

These results show that Anova results can be converted to an ordering only if it accepts; we order in terms of our prior preference. But this occurs rarely, in only two data sets out of 30. Converting Newman-Keuls results to an ordering is possible in some cases but not always. We only compare five algorithms in this paper; if there are more, we may

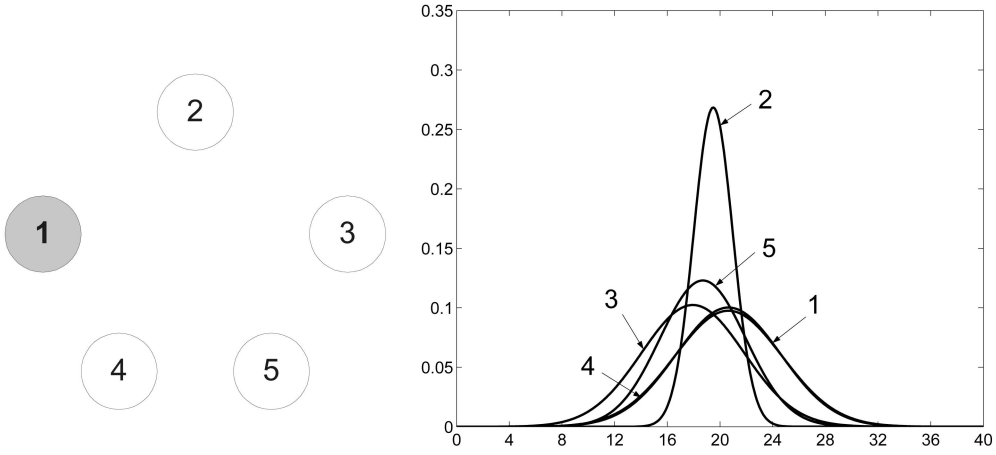

Fig. 3. Results on *hepatitis*. The most frequently occurring graph and the corresponding error distributions of the classifiers are shown.

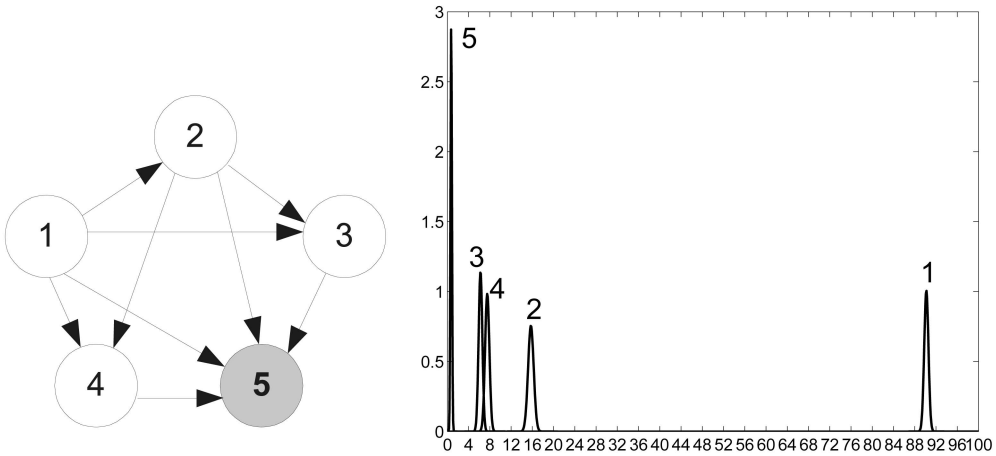

Fig. 4. Results on *pendigits*.

expect to have more groups of algorithms having similar expected error and more occurrences of this problematic case where multiple underlines overlap. There are similar problems with a straightforward approach such as the heuristic TestFirst algorithm, which is not always able to choose a best algorithm. These results show that a methodological approach as proposed by the MultiTest algorithm is necessary and that MultiTest is always able to find a best algorithm.

5 Conclusions

We introduce the MultiTest method, which allows choosing the "best" of an arbitrary number of learning algorithms where the "goodness" takes into account the results of pairwise statistical tests on expected error and our prior preferences. We apply it to classification by making use of the 5×2 pairwise t test. It is applicable to regression and other loss functions by using a suitable test [23]. MultiTest is always able to find an algorithm as the "best" one (or order the algorithms in the general case) combining the expected error and prior preferences. Though Anova and Newman-Keuls or the pairwise test results using TestFirst can be extended to find a "best" algorithm, these do not always work, justifying the methodology we propose with MultiTest.

The MultiTest method does not increase the computational and/or space complexity because the real cost is the training and validation of the algorithms. Once the validation errors of algorithms are recorded, applying the calculation necessary for MultiTest is simple in comparison.

Acknowledgments

This work has been supported by the Turkish Academy of Sciences, in the framework of the Young Scientist Award Program (EA-TÜBA-GEBİP/2001-1-1) and Boğaziçi University Scientific Research Projects 02A104D and 03K120250. The authors would like to thank the associate editor and referees for the constructive comments which improved the content and the presentation of the manuscript.

TABLE 3
The Frequency and Percentages of Best Classification Algorithms Found by TestFirst, Newman-Keuls, and MultiTest for All Data Sets

Test	Not Found	MAX (1)	NMC (2)	LGC (3)	C4.5 (4)	NN (5)
Anova	29,388	612	0	0	0	0
	97.96%	2.04%	0.00%	0.00%	0.00%	0.00%
TestFirst	21,505	694	107	5,929	348	1,417
	71.68%	2.31%	0.36%	19.76%	1.16%	4.72%
Newman-Keuls	165	2,783	3,781	18,490	1,251	3,530
	0.55%	9.28%	12.60%	61.63%	4.17%	11.77%
MultiTest	0	5,314	13,720	9,343	464	1,159
	0.00%	17.71%	45.73%	31.14%	1.55%	3.86%

REFERENCES

[1] T.G. Dietterich, "Approximate Statistical Tests for Comparing Supervised Classification Learning Classifiers," *Neural Computation*, vol. 10, pp. 1895-1923, 1998.
[2] E. Alpaydin, *Introduction to Machine Learning*. The MIT Press, 2004.
[3] B. Efron, "Computers and the Theory of Statistics," *SIAM Rev.*, vol. 21, pp. 460-480, 1979.
[4] E. Alpaydin, "Combined 5×2 cv f Test for Comparing Supervised Classification Learning Classifiers," *Neural Computation*, vol. 11, pp. 1975-1982, 1999.
[5] S.M. Ross, *Introduction to Probability and Statistics for Engineers and Scientists*. New York: Wiley, 1987.
[6] R.G. Miller, *Simultaneous Statistical Inference*. New York: Springer Verlag, 1981.
[7] A. Dean and D. Voss, *Design and Analysis of Experiments*. New York: Springer Verlag, 1999.
[8] Y. Hochberg and A.C. Tamhane, *Multiple Comparison Procedures*. John Wiley and Sons, 1987.
[9] D.D. Jensen and P.R. Cohen, "Multiple Comparisons in Induction Algorithms," *Machine Learning*, vol. 38, pp. 309-338, 2000.
[10] W.J. Conover, *Practical Nonparametric Statistics*. New York: John Wiley and Sons, 1999.
[11] B.S. Everitt, *The Analysis of Contingency Tables*. London: Chapman and Hall, 1977.
[12] S.W. Looney, "A Statistical Technique for Comparing the Accuracies of Several Classifiers," *Pattern Recognition Letters*, vol. 8, pp. 5-9, 1988.
[13] R.R. Bouckaert, "Choosing between Two Learning Algorithms Based on Calibrated Tests," *Proc. 20th Int'l Conf. Machine Learning*, 2003.
[14] F. Provost, T. Fawcett, and R. Kohavi, "The Case against Accuracy Estimation for Comparing Induction Algorithms," *Proc. 15th Int'l Conf. Machine Learning*, 1998.
[15] F. Provost and T. Fawcett, "Robust Classification for Imprecise Environments," *Machine Learning*, vol. 42, pp. 203-231, 2001.
[16] S.G. Alsing, K.W. Bauer, and J.O. Miller, "A Multinomial Selection Procedure for Evaluating Pattern Recognition Algorithms," *Pattern Recognition*, vol. 35, pp. 2397-2412, 2002.
[17] P.D. Turney, "Types of Cost in Inductive Concept Learning," *Proc. Workshop Cost-Sensitive Learning, 17th Int'l Conf. Machine Learning*, pp. 15-21, 2000.
[18] S. Holm, "A Simple Sequentially Rejective Multiple Test Procedure," *Scandinavian J. Statistics*, vol. 6, pp. 65-70, 1979.
[19] K.H. Rosen, *Discrete Mathematics and Its Applications*. New York: McGraw-Hill, 1995.
[20] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York: Springer Verlag, 2001.
[21] J.R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, Calif.: Morgan Kaufmann, 1993.
[22] C.L. Blake and C.J. Merz, "UCI Repository of Machine Learning Databases," http://www.ics.uci.edu/~mlearn/MLRepository.html, 2000.
[23] O.T. Yildiz, "Tuning Model Complexity Using Cross-Validation for Supervised Learning," PhD thesis, Dept. of Computer Eng., Boğaziçi Univ., 2005.

Olcay Taner Yildiz received the BS, MS, and PhD degrees in computer science from Boğaziçi University, Istanbul, in 1997, 2000, and 2005, respectively. He is currently doing postdoctoral study at the University of Minnesota. His research interests include model selection, decision trees, neural networks, and robotics.

Ethem Alpaydin received the PhD degree in computer science from Ecole Polytechnique Fédérale de Lausanne, Switzerland, in 1990 and did postdoctoral work at the International Computer Science Institute (ICSI), Berkeley, California, in 1991. Since then, he has taught in the Department of Computer Engineering at Boğaziçi University, Istanbul, where he is now a professor. He had visiting appointments at MIT in 1994, ICSI (as a Fulbright scholar) in 1997, and IDIAP, Switzerland, 1998. He received the young scientist award from the Turkish Academy of Sciences in 2001 and the scientific encouragement award from the Turkish Scientific and Technical Research Council in 2002. His book *Introduction to Machine Learning* has recently been published by The MIT Press. He is a senior member of the IEEE and the IEEE Computer Society.