Evaluating Machine Learning Models for Engineering Problems

Yoram Reich and S V Barai
Department of Solid Mechanics, Materials, and Structures,
Faculty of Engineering, Tel Aviv University, Ramat Aviv 69978, Israel
email: {yoram, skbarai}@eng.tau.ac.il

To appear in *Artificial Intelligence in Engineering*

Keywords: Artificial neural networks, research methodology, performance evaluation, statistical tests, modeling

Abstract: The use of machine learning (ML), and in particular, artificial neural networks (ANN), in engineering applications has increased dramatically over the last years. However, by and large, the development of such applications or their report lack proper evaluation. Deficient evaluation practice was observed in the general neural networks community and again in engineering applications through a survey we conducted of articles published in *AI in Engineering* and elsewhere. This deficient status hinders understanding and prevents progress. This paper goal is to remedy this situation. First, several evaluation methods are discussed with their relative qualities. Second, these qualities are illustrated by using the methods to evaluate ANN performance in two engineering problems. Third, a systematic evaluation procedure for ML is discussed. This procedure will lead to better evaluation of studies, and consequently to improved research and practice in the area of ML in engineering applications.

1 Introduction

The use of machine learning techniques for building models from data is growing steadily. Building such models requires intimate understanding of the data and knowledge of available ML tools and their properties. A systematic approach that ensures that all important aspects of the modeling are addressed can lead to building good quality models (Reich, 1997). ML models can be built for different roles: models for improving understanding of the data and models for prediction. Independent of their role, one of the most critical steps when building models is their evaluation. Evaluation determines whether the stated goal of the modeling activity has been achieved; it allows to compare different modeling approaches and to direct future research. In spite of its importance, evaluation has received little and sometimes no attention compared to the other modeling steps.

The long term goal of this paper is to remedy this situation. In order to set the basis for discussion, Section 2 reviews several methods for evaluating prediction models and their properties. Section 3 illustrates these properties using two case studies: marine propeller behavior and material corrosion analysis. The purpose of these studies is to demonstrate two critical assertions:

Assertion 1: The choice of evaluation method is critical. The results obtained by different methods may be very different.

Assertion 2: The relations between different evaluation results vary with the properties of the data, the learning goal, and the learning system. Therefore, it is not possible to determine these relations *a priori*..

Consequently, the only way to assess the quality of ML models is through their careful evaluation with appropriate methods. The key is to identify which methods are appropriate for given modeling contexts.

Section 4 reports on a review of studies published in *AI in Engineering* that used artificial neural networks (ANN) for building prediction models. The review focuses on the methods used for evaluating the models developed in these studies. It demonstrates the deficient status of evaluation in engineering applications. In order to improve present evaluation practice, Section 5 outlines a systematic procedure to evaluating ML models. It is hoped that following this paper, the evaluation procedure or similar approaches will be used in future studies in this area and that careful evaluation will become a prerequisite for publishing ML modeling studies.

2 Evaluating the accuracy of ML models

In this paper we focus on the evaluation of models created for prediction. We first define the problem of evaluating prediction models, discuss some general properties, and review basic evaluation methods. A general discussion on evaluation that includes ML models for understanding can be found elsewhere (Reich, 1995). We use input from (Feelders and Verkooijen, 1995; Geman et al, 1992; Kohavi, 1995) and others in this section.

2.1 Theoretical background

Given an input vector $\mathbf{x}$ and a response vector $\mathbf{y}$, where $(\mathbf{x}, \mathbf{y})$ obey some unknown joint probability distribution F, learn to predict $\mathbf{y}$ from $\mathbf{x}$ as follows. A training set, D of size n, $(\mathbf{x}_i, \mathbf{y}_i), i = 1, \ldots, n$ is drawn randomly and independently from F. In a classification task, $\mathbf{y}$ is a scalar that takes as a value the possible class labels. For function mapping (regression), $\mathbf{y}$ is a scalar whose values are real numbers. In the more general case, $\mathbf{y}$ is a vector whose elements could be discrete, ordered, or numeric.

A ML program P uses D to build a model $\hat{f}(\mathbf{x})$ that estimates $\mathbf{y} = f(\mathbf{x})$, i.e., $\hat{\mathbf{y}} = \hat{f}(\mathbf{x})$. Since $\hat{f}$ depends on P and D we note it by $\hat{f}(\mathbf{x}; D, P)$. We are interested in evaluating the performance of P or the quality of $\hat{f}$ with respect to f. Given D and a particular $\mathbf{x}$, independent of D, the performance, or error θ, of $\hat{f}$ is often measured by a square loss function. (We assume from now that $\mathbf{y}$ is a scalar y. This simplifies understanding the concepts. The formulation can be extended to deal with the vector case. See more in section 5.2)

$$\theta = E[y - \hat{y}] = E[(y - \hat{f}(\mathbf{x}; D, P))^2 | \mathbf{x}, D], \tag{1}$$

where $E[\cdot]$ is the expected value with respect to F. For classification tasks, the error may be formulated using a zero-one loss function:

$$\theta = E[1 - \delta(y, \hat{f}(\mathbf{x}; D, P)) | \mathbf{x}, D], \tag{2}$$

where δ is the Kronecker delta.

In general, E is unknown and therefore, θ could only be estimated. One way to estimate it is to test $\hat{f}(\mathbf{x}; D, P))$ on an independent set T of size l, $(\mathbf{x}_i, y_i), i = 1, \ldots, l$ drawn randomly from the same population F. The estimated value of θ will be (for a square loss function):

$$\hat{\theta}(T, D, P) = \frac{1}{l} \sum_{i=1}^{l} (y_i - \hat{f}(\mathbf{x}_i; D, P))^2. \tag{3}$$

When $l \to \infty, \hat{\theta} \to \theta$. For classification tasks, $l > 1000$ guarantees accurate results with confidence above 0.95 (Highleyman, 1962). Other confidence intervals for different test sizes can be estimated as well given that each test can be viewed as a Bernoulli trial (Kohavi, 1995). Such general calculations of confidence intervals do not exist for function mapping.

There are two situations in which testing on large number of instances is possible: (1) the database is very large (and for classification, 1000 or more examples can be sampled for testing while others are used for training); or (2) there is a generator (simulator) that can provide additional examples on demand. In most real situations, however, we have a fixed size database that must be used for training and testing. Consequently, other error estimation methods have to be used. These methods differ in their process and the way they use the database. Before discussing any of them, two general properties of evaluation methods, *bias* and *precision* (or *variability*), should be discussed.

The *estimation bias* reflects the difference between the expected value of the estimation, $E(\hat{\theta})$, and the parameter being estimated, θ, whose value is never known for real data (but can be calculated precisely for simulated data), i.e.,

$$bias = E[\hat{\theta}] - \theta. \tag{4}$$

The *method precision* or *variability* σ reflects the variability of the error estimation derived by the evaluation method. It is often measured by the standard error σ of the method distribution.

$$\sigma^2 = E[(\hat{\theta} - E[\hat{\theta}])^2] \tag{5}$$

The method variability might be caused by many factors that directly depend upon the modeling process. In our formulation, these are D, P, and T. In the context of ANN the following sources are relevant for solving a particular learning problem:

- D: Given the problem, sample the data D from the general data population F. The problem and data may be artificial or real.
- P: Select an ANN architecture (e.g., number of layers, number of units, training rule and parameters); select process parameters (e.g., limit on number of epochs or system square error); and initialize ANN training with some weights.
- T: Manage the database for training and testing (e.g., subdivide the data into training and testing sets).

The ideal evaluation method would be the least sensitive to these choices for all learning problems. Such ideal method does not exist (Schaffer, 1994; Wolpert, 1996). However, better methods for different practical problems can be identified.

For classification tasks, where each test instance can be viewed as a Bernoulli trial, and given large enough test set, the variance of a method can be estimated to be $\theta(1-\theta)/n$ or even $\hat{\theta}(1-\hat{\theta})/n$. In general, however, the value of σ cannot be estimated easily. (Although see the following discussion on bootstrap, Section 2.2.4.)

If an evaluation method is based upon averaging several independent estimations then given a sample of I *independent* estimations calculated by Equation 3, $\hat{\theta}_i$, $i = 1 \ldots I$, the mean error is estimated by $\bar{\hat{\theta}} = (\sum_{i=1}^{I} \hat{\theta})/I$ and its variability can be estimated by

$$\hat{\sigma}_M = \sqrt{\frac{\sum_{i=1}^{I}(\hat{\theta}_i - \bar{\hat{\theta}})^2}{(I-1)I}}. \tag{6}$$

When using any of the evaluation methods we present later (with a fixed size database), the I estimations are not independent thus Equation 6 is only an approximation and estimating the variability precisely is non-trivial. A question arises regarding the influence of this dependency on inferences we might wish to make about absolute program performance error or the relative performance of ML methods. Independent of this complication, the method variability determines our confidence in the estimated error.

For I independent estimations and if I is small (so that normal distribution of the sample cannot be assumed), the confidence interval can be determined by using the t distribution. The t value is calculated by

$$t = \frac{\bar{\hat{\theta}} - \theta}{\hat{\sigma}_M}. \tag{7}$$

The confidence interval of $1 - \alpha$ for the value of θ would be:

$$\theta \in [\bar{\hat{\theta}} \pm t(\alpha/2; I-1) \cdot \hat{\sigma}_M], \tag{8}$$

where $t(\alpha/2; I-1)$ is the t value for level of significance α and $I-1$ degrees of freedom. That is, with a confidence of $1-\alpha$ the value of the true θ will fall in the range calculated by Equation 8.

Instead of using Equation 6 as an approximation to $\hat{\sigma}_M$ and Equation 8 to calculate confidence intervals, we can use the bootstrap (B) procedure discussed later (Section 2.2.4) which allows to empirically calculate different properties (e.g., variance, confidence intervals) of an arbitrary statistic. B also does not assume anything about the distribution of the statistic.

Going back to Equation 1, it is easy to show that the prediction error θ can be decomposed into three terms: bias, variance, plus some noise factor. For a square loss function (Geman et al, 1992) the decomposition is:

$$\begin{aligned} E[(y - \hat{f}(x; D, P))^2 | x, D] &= \text{“}noise\text{''} \\ &+ \ (E_D[\hat{f}(x; D, P)] - E[y|x])^2 \qquad \text{“}bias\text{''}^2 \\ &+ \ E_D[(\hat{f}(x; D, P) - E_D[\hat{f}(x; D, P)])^2] \quad \text{“}variance\text{''}, \end{aligned} \tag{9}$$

where, $E_D[\cdot]$ represents expectation with respect to the possible D, for fixed sample size n. The "bias" and "variance" terms correspond to those in Equations 4 and 5, respectively. For a zero-one loss function there is a different decomposition (Kohavi and Wolpert, 1996).

Typically, there is a tradeoff between the bias and variance terms. The variance can be reduced by smoothing or by modeling data with a restricted number of degrees of freedom. Such constraints on the model would lead to high bias. In contrast, fitting a complicated model may lead to low bias but increase the variability. We cannot obtain an *a priori* optimal choice between bias and variance that minimizes the overall error (by manipulating P or T). Different error estimation methods have different tradeoffs between bias and variance that influence their suitability as discussed next.

2.2 Error estimation methods

There are several methods that have been used to estimate the performance of ML models in general as well as ANN models including: (1) resubstitution (R), (2) hold-out (H), (3) cross-validation (CV) such as leave-one-out (L) or k-fold cross validation (K), and (4) bootstrap (B). There is a growing interest in the ML community in understanding the properties of these tests (Bailey and Elkan, 1993; Breiman et al, 1984; Dietterich, 1996; Efron, 1983; Efron and Tibshirani, 1993; Flexer, 1995; Jain et al, 1987; Kohavi, 1995). Such properties are derived empirically from many tests on artificial and real databases. By and large, most of this work was conducted on classification tasks. In contrast, in this paper we are mainly interested in function mapping (regression). We assume that these properties approximately generalize to regression; however, more theoretical and empirical work is required to study error estimations of function mappings.

2.2.1 Resubstitution (R)

In resubstitution the complete data set D is used to train the network. The model created by the network is subsequently tested on the same data set ($T = D$). The estimation of generalization error of resubstitution θ_R is highly optimistic, i.e., its error estimation is biased downward. The performance of R is highly dependent on the sampling of D from F, i.e., it has high variability. If we restrict the ability of P to overfit the data D (e.g., by early stopping in ANN training), θ_R might be less biased, but in view of the bias-variance tradeoff, the variability might increase even further. Consequently, R is a poor evaluation method.

2.2.2 Hold-out (H)

H is the most common method for estimating generalization error. The data set D is randomly divided into disjoint training and testing sets (part of the influence of T). It is common to select 2/3 of the set for training and the remaining 1/3 for testing. After training, the network is evaluated on the test set and the errors give an estimate of the generalization error θ_H. The results of H have high variability that depends upon this random subdivision, in addition to the variability due to the sampling of D from F. Also, the results of H may be pessimistic because not all available data is used for training, i.e., its error estimation is biased upward.

As mentioned before for classification tasks, in order to produce results with confidence above 0.95, the testing set should include more than 1000 instances (Highleyman, 1962). No such general bounds exist for regression.

In smaller databases, a variation of H called random subsampling and denoted here by H^I, is performed. Random subsampling is simply repeating H I times with random subdivisions into training and test sets. Each such repetition is called an *iteration*. The resulting estimation is the mean of the I iterations. Note that these I iterations are not independent, having used the same database. Therefore, special care should be exercised when interpreting results from multiple runs of H.

2.2.3 Cross-validation (CV): k-fold (K) or leave-one-out (L)

In *k-fold cross-validation*, one divides the data D into k subsets of roughly equal size. The ML program P is trained k times, each time leaving out one of the subsets from training, and using it for testing (see Figure 1). The error estimation θ_K is the average accuracy of the k runs (i.e., the *k internal iterations*). This estimate is dependent upon the subdivision to k subsets. To account for this variability source a *complete CV* could be performed in which all the $\binom{n}{n/k}$ possibilities of selecting the testing subset of n/k instances are exercised. Still the results depend upon the sampling of D from F. If P is stable for the problem (Breiman, 1996), θ_K will be unbiased. If k is too small, θ_K is pessimistically biased because fewer data points are used for training, however, it might have lower variability.

Instead of running the costly complete CV test, K can be run I times with different subdivisions into k sets. This test will be denoted by K^I and its estimation would be the average of the I K estimations. As in H^I, we need to approximate the standard deviation or confidence intervals of this test or evaluate them empirically by B.

If $k = n$, K is called a *leave-one-out (L)* test. L is always a complete CV test. It has been common in general ML studies to use a 10-fold CV method when the number of instances, n, exceeds 100, or a leave-one-out method for small databases (Henry, 1994; Michie et al, 1994; Weiss and Kapouleas, 1989).

2.2.4 Bootstrap

In the bootstrap, a sample of n instances is drawn *with replacement* from the original n instances, where each instance has equal probability to be sampled. On an average, $1-1/e = 0.632$ of the original instances are drawn into this sample. The new sample is used for training and the old sample for testing. The result of this testing provides a measure of the optimism of resubstitution. I such samples are drawn and their optimism measure is calculated. Experiments show that I should be in the range of $50 \div 200$. The final estimation is the average of these measures added to the resubstitution estimation. Bootstrap reduces the variability observed in previous methods and is only slightly optimistic. Other versions of bootstrap such as the .632 bootstrap (.632B) improve upon this bias in some experiments (Efron, 1983). The .632B is calculated as follows:

$$\theta_{.632B} = \frac{1}{I}\sum_{i=1}^{I}(0.632 \cdot \theta_{H_i} + 0.368 \cdot \theta_R), \tag{10}$$

where θ_{H_i} is the holdout estimate for the $I - th$ iteration (i.e., training on n instances sampled from D with replacement and testing on the instances that *were not sampled* into the training set) and θ_R is the resubstitution estimate (i.e., training and testing on the complete database D).

More generally, bootstrap can be used to estimate the value of any statistic and not just the error statistic. Its advantage is that no assumption regarding the distribution F is made. For example, the variability of an evaluation method X could be estimated as follows:

(1) Sample with replacement I bootstrap samples D^i, $i = 1, \ldots, I$, from D.
(2) Calculate the error estimation $\theta_{X_i}(D^i)$ for each sample $i = 1, \ldots, I$.
(3) Estimate the variability of X by calculating the standard deviation of the sample θ_{X_i}, $i = 1, \ldots, I$.

This procedure will fail if we try to estimate some "narrow feature of the original sampling process" (Stine, 1990, p. 286) such as the maximum of a sample.

Confidence intervals can also be calculated not using any distribution assumption but using the *percentile method* (Efron and Tibshirani, 1993). An $1 - \alpha$ confidence interval can be estimated by taking the $\alpha/2$ and $1 - \alpha/2$ percentiles of the sample of I bootstrap estimations. Such procedure works when the number of bootstrap samples is large, e.g., $I > 1000$. It also assumes that the estimation is unbiased.

2.2.5 Summary of evaluation methods

The above methods are the very basic evaluation tests. They do not include any distinction between different types of errors (e.g., false positive or negative in classification, or positive vs. negative errors in function prediction), although different types of errors might be very different in terms of cost or severeness for the particular engineering application. Also, they do not include improvements such as stratified methods for classification tasks where, for example, training and test sets are sampled randomly but with the constraint that they have the same distribution of classification values (Feelders and Verkooijen, 1995). These and other issues that arise in the context of specific applications require further elaboration. Additional information on evaluation methods can be found in the statistics and general ML literature (e.g., Breiman et al, 1984; Dietterich, 1996; Efron, 1983; Gascuel and Caraux, 1992; Henry, 1994; Kohavi, 1995; Weiss and Kapouleas, 1989).

Table 1 summarizes the four evaluation methods. In the Table, the *number of internal iterations* denotes the number of times a basic procedure is executed in order to obtain one estimation, $\hat{\theta}$, of the true accuracy θ. The *number of iterations* denotes the number of times the complete process is performed. The two iteration figures influence the computational cost of the method. Figure 2 roughly illustrates the expected bias and variability (or confidence interval) of some of these evaluation methods relative to the true error.

There are results in the literature that differ from these expected values. In some cases these discrepancies have good explanations. For example,

- Jain et al, 1987 suggested that .632B is inappropriate for nearest neighbor classifiers because they do not make use of duplicate equal instances generated in bootstrap samples.
- Weiss, 1991 found B to perform poorly when applying nearest neighbor classifiers to small databases in cases with high true error rates. B was worse in the case of 1-NN than 3-NN. This discrepancy can be explained by the ability of 3-NN to partially benefit from multiple similar instances compared to the lack of 1-NN to benefit from it.
- Bailey and Elkan, 1993 found B to perform poorly on FOIL. This might result from FOIL's inability to benefit from the duplicate equal instances generated in bootstrap samples.
- Shao, 1993 found L to perform poorly for selecting between linear models because L is asymptotically inconsistent (Stone, 1977). (A consistent estimator of a parameter satisfies that the probability of getting the parameter within some range around the estimation converges to 1 as the data size grows.)

From these examples and theoretical work, no method is always better than another (Schaffer, 1994; Wolpert, 1996); however, some evaluation methods are better than others on some classes of problems. Figure 2 expresses the general practical relation between these methods as suggested by past experiments. A case where deviations are found between results of these methods on some problems and the figure is a trigger that this learning context might have special features worth studying.

3 Case studies

3.1 General description

Two case studies were performed to prove the assertions from the introduction. The systematic procedure outlined by Reich (1997) is followed in discussing them. For each case, we conducted five tests: R, H (2/3 for training, 1/3 for testing, with 10 repetitions), L, K (10 folds with 10 repetitions), and .632 bootstrap with (100 resamplings). The results for each case are tabulated and displayed graphically.

The tables provide the raw data. In the case of R and L they give the results and in the case of .632B, K, and H they give the minimum and maximum values in all iterations, and the mean and standard deviation of the iterations sample.

The figures display the values of R, L, and B, and the mean of H and K with their 0.95 confidence intervals. The confidence intervals for H and K where calculated from Equation 8 assuming that the I iterations are independent and that the data reflects the overall method variability. Since these assumptions are incorrect (as discussed above), and furthermore, the latter assumption is not conservative, we performed several B tests to assess the confidence intervals of H, K, and also R and found the discrepancies to be minimal. For the purpose of this paper, the approximation used is sufficient. The confidence interval of B was not calculated. In order to calculate it, we would have to perform many B evaluations with different original databases. Alternatively, we could assess confidence intervals of B by using B itself. This, however, is a very time consuming and impractical exercise. Also, note that the confidence intervals neglect the dependency between the different output parameters. Consequently, they cannot be the basis for hypothesis testing in general. (See more in Section 5.2.)

In the discussion of the results we refer to K and H. However, since K^I and H^I are means of several K and H tests, stronger statements (i.e., with tighter confidence intervals) apply to them as well.

3.2 Example 1 : Marine propeller behavior

- **Problem Definition**

The goal of this study was to establish a mapping between propellers design parameters and their performance.

- **Data Collection**

In establishing the actual performance of marine propeller, one has to go through the laborious, intensive design process followed by extensive tank or open sea experimentations. We took a typical example of the USN series data of marine propeller design for the present study (Denny et al, 1989). The data were created in actual sea trials using suitably modified USN 36 ft boat. The experimental data of 301 instances cover the parameters thrust coefficient (K_T), torque coefficient (K_Q), efficiency (η) versus advance coefficient (J) for various values of pitch diameter ratio ($\frac{P}{D}$), expanded area ratio (EAR), number of blades (z), and cavitation number (σ). The accuracy in data collection during the experiment was within the ranges of 0.5% for propeller thrust and 0.2% for propeller torque. We used data extracted from the original graphical data by Neocleous and Schizas, 1995.

- **Preliminary Data Analysis**

Five parameters - J, $\frac{P}{D}$, EAR, z, and σ - were selected as inputs and three - K_T, K_Q, and η - as outputs. Simple normalization was performed on the parameters.

- **Selection of Neural Networks Model**

The multilayer perceptron was selected due to its recognized ability to perform regression.

- **Selection of Neural Networks Model Parameters**

The neural network architecture had two hidden layers with 30 hidden units in each layer. The program was implemented using improved backpropagation in MATLAB Neural Networks Toolbox (Demuth and Beale, 1994). After several exercises, the sum square error was selected as 0.5 and learning rate as 0.02, keeping a compromise between the accuracy and the computational time. Note that no optimization of the architecture or training parameters was performed. In contrast, we selected common architecture and parameters and set the parameters so that the time consuming exercises will take reasonable time.

- **Evaluating and Interpreting Results**

The computed average relative error for the parameters K_T, K_Q, and η are are tabulated in Table 2. The graphs depicting the results of the tests are shown in Figure 3. We used the relative error and not the square loss function which measures absolute error because the former is more informative from the ANN user perspective.

Also, we did not sum the errors of the three outputs into one measure because this would have given us only one coarse measure. Note that this complicates statistical analysis when comparing between two programs because the dependence between the results of the output parameters requires that instead of using t test one has to perform T^2 test for testing between *vectors of means* (Morrison, 1990) (see more in Section 5.2).

- As expected, H gives pessimistic results relative to the other tests and with higher variance than K. From Figure 3, the results of H are statistically significant far beyond 0.95 confidence from the rest of the tests. In this case this conclusion is conservative; a T^2 test on the three dependent parameters, K_T, K_Q, and η, outputs a confidence level of 0.9999 that K is different from H.
- As expected, resubstitution gave optimistic results compared to the other tests. Based on the confidence interval of K and H, in most cases these results are different from R with confidence beyond 0.95.
  The results would have been further away if we did not use early stopping which caused R not to overfits the data and therefore, produce higher (less optimistic) errors that were close to the other tests.
- L gave poorer results compared to K due to data overfitting that occurred during this test.
- Out of the 100 .632B samples, several caused the ANN to "diverge". No such behavior was observed in any of the other many runs we performed. We hypothesize that this behavior results from the replication of some of the instances due to the use of bootstrap samples. We removed those samples from the analysis. In Table 2 the I in the .632B test denotes the number of samples maintained in the analysis.

In spite of all the above significant or insignificant differences, we must always focus on the engineering relevance of these evaluations. The performance errors in the evaluations range from 7.8% between the maximum value of H and R for K_T to 2% between the same for η. If we eliminate H as an evaluation method, the errors between the other methods range from 2.1% to 0.6%. From an engineering perspective the first set of ranges may be unacceptable while the latter acceptable. Therefore, for this application, it might not matter which test we use out of R, B, L or K.

Note, however, that the data we have does not take into account the variability due to sampling of D from F, nor the variability due to the choice of learning parameters and ANN architecture. These will definitely increase these ranges. Consequently, the conservative user will still have to look for the most appropriate tests.

3.3 Example 2: Corrosion data analysis

• Problem Definition

The goal of this exercise was to analyze empirical data on the stress corrosion cracking (SCC) of a sensitized, wrought type 304 (UNS S304000) stainless steel and establish relationships between the environmental conditions and their effects on the stainless steel.

• Data Collection

We used experimental corrosion data of sensitized wrought type 304 stainless steel tested in argon gas and lithiated water doped with 20 ppm SO_4^{2-}, 20 ppm + SO_4^{2-} 20 ppm Cl^-, 100 ppm SO_4^{2-} and 100 ppm Cl^- (Congleton et al, 1995). In the data set of 93 samples, the environmental conditions are temperature (T), potential, solution types and effects are crack-length (CL), ultimate tensile strength (UTS), time to failure (TF) and reduction of area (RA).

• Preliminary Data Analysis

Preliminary data analysis suggested that the data is sparse and required careful modeling. However, for this study only simple normalization was performed on the parameters.

- **Selection of Neural Networks Model**

Again, due to their general capability to perform regression, multilayer perceptron was selected.

- **Selection of Neural Networks Model Parameters**

A multilayer perceptron with two hidden layers and 18 hidden nodes per layer was used in this study. Learning rate was set to 0.02 and based on few runs, the cut-off threshold was set to 25000 epochs for training. Again, no optimization of architecture or parameters was performed; rather, the parameter selection was a compromise between reasonable accuracy and training speed.

- **Evaluating and Interpreting Results**

The ANN performance was evaluated and the results for two parameters (CL and RA) were found unsatisfactory (errors ranging beyond 100%). Therefore, we decided to clean the data manually. We focused on improving the prediction of the RA parameter. We identified data points that gave high errors of this parameter in both R and L tests and removed those that looked as potentially erroneous. Also, several repetitive instances were removed. As a result we obtained two data sets:

- *Raw data case:* The original data given in Congleton et al. (1995) with 93 instances.
- *Cleaned data for RA:* Remaining 79 instances after cleaning.

Some of the instances in the data have output parameters CL and RA with zero values. For these instances, the computation of the percent error of the ANN prediction was impossible. Therefore, for the corrosion data we represented the errors as the common mean square error. The results of the two cases are tabulated in Tables 3 and 4 and their graphic presentations are shown in Figures 4 and 5, respectively.

- By and large, the results of the two cases follow the observations of the results obtained for the propeller behavior exercise. However, there are some discrepancies. For example, the results of the K and H tests while different are not statistically different with confidence of 0.95. A T^2 test found that K and H are different with confidence of 0.90 in the first case and 0.93 in the second.
- Some of the discrepancies between the results and the anticipated relations from Figure 2 might be a consequent of the ANN parameters used, the training schedule, or the initial weight setting.
- In general, when moving toward the "cleaner" data in the second case, the performance graphs resemble more the anticipated behavior shown in Figure 2.
- We did not find clear cases of divergence when using B as in the marine propeller case. This might result from the general high error rates of RA and CL that in many cases exceeded 100% (not shown). Also, it might be that sparse data is less susceptible to such phenomena.
- Although not shown, the differences between the tests for parameters CL and RA are serious from an engineering perspective. Therefore, in this application selecting an appropriate evaluation test is critical.

3.4 Summary of exercises

From the case studies we can summarize the following main observations:

- *Assertion 1:* Independent of data quality (e.g. good as the propeller behavior or bad as the material corrosion analysis), R gives optimistic results and H pessimistic results relative to each other. In all cases but one these results are statistically significant with confidence above 0.95 while in other cases, less significant.

- *Assertion 2:* The relations between the evaluation methods, whether statistically significantly different or not, varies with data quality. Therefore, one cannot replace one test with another and only evaluation methods appropriate for the context may be used.

- Similarly, the engineering relevance of choosing an appropriate method varies with applications and cannot be determined *a priori*.

- The selection of ANN model, architecture, and training parameters, influence the evaluation and therefore, influence the relations between the results of the different evaluation methods. For example, early stopping in training may result in R estimates that are close to L or K estimates. However, these pseudo optimal parameters cannot be determined *a priori*. We did not perform sensitivity analysis or multiple evaluations with many parameters in our exercises because our goal was to prove the assertions. Nevertheless, careful evaluation in applications should consider performing such multiple tests.

- Among the two CV tests, we found K to be comparable to L in susceptibility to data overfitting or other data quality problems. Therefore, for databases larger than about 100 it is acceptable to use it. Given its observed small variability due to different subdivisions into 10 sets, a small number of such subdivisions can lead to good accuracy estimation. This recommendation is in line with other researchers (e.g., Bailey and Elkan, 1993; Kohavi, 1995; Weiss and Indurkhya, 1994).

4 Review of evaluating ANN models for prediction

ANN are by far the ML tools most heavily used for building prediction models from data. They are used due to their demonstrated capabilities to create mappings from input to output data even if the data is noisy and when no model of the data exists. They can handle a variety of data types as well as time-variable data. Careful empirical evaluation of models created by ANN is critical to selecting ANN architectures, setting their parameters, as well as to selecting between different ANN approaches. Unfortunately, there is little awareness to evaluation issues within the neural network community (Flexer, 1995; Prechelt, 1996). Flexer, 1995 also mentioned several minimal requirements for evaluating neural networks models and the percentage of their use in studies in two leading ANN journals (see Table 5).

The status of testing in engineering applications of ANN is similar and sometimes worse. In some studies, no report on testing is mentioned. Other studies provide one anecdotal example showing what the technique might be doing in a particular situation. Even studies that perform some testing may be deficient if their tests are methodologically wrong, or if they do not complete the testing procedure, leaving readers to interpret the results (Reich, 1997).

We performed a detailed study of twenty six articles published in *Artificial Intelligence in Engineering* on neural networks modeling from 1992 to 1997. The review analyzed each application (total of 32 in the 26 articles) according to the modeling steps identified in Reich (1997) for developing practical ML applications. The review results related to evaluation are summarized in Table 6.

Clearly ANN models in engineering applications have been tested improperly: 31.3%, 18.7%, 3.1%, and 12.5% have used respectively R, H, L, and TTV tests. (These TTV tests divided the data into three parts: one used for training, second for testing and tuning parameters, and third for testing the optimal ANN. Both tuning and testing employed the H method. See more on TTV in Section 5). Most of these did not mention the basis for selecting their testing methods. The remaining studies (34.4%) did not clearly mention which approach was used for evaluation.

Among the CV methods L was used only once and K never. B was never used. While H or TTV may also give reasonable results for classification when the testing set size exceeds 1000, such testing set sizes were used only twice, and not on classification tasks. Therefore, these testing sizes do not guarantee reliable results, especially if the domain is complex and data is sparse. The remaining tests, in particular, resubstitution, cannot be considered as reliable or as a basis for future comparisons.

5 A systematic approach to evaluating ML models

In the general ML community, ML problems can be classified based on several aspects (see e.g., Dietterich, 1996):

- G *Goal of study.* This will involve either an absolute evaluation or a relative comparison between several programs.
- D *Domain or data studied.* In some cases a single domain is studied and in others programs are compared relative to multiple domains. The data might be large or small (limited), real or artificial.
- P *Nature of the modeling activity or the ML program studied.* Depending on the problem, the ML modeling might be supervised (e.g., regression, classification) or unsupervised.

Proper evaluation depends upon the goal of the study. The focus or goals of engineering applications of ML are different than those of the general ML community. In engineering applications, whether in research or practice, solving an engineering problem is the focus rather than the solution method or its generalization across problems or domains.

When solving an unsolved problem, we would be content with one reliable solution. Subsequently we may seek improvements. Even if we adopt a method that already solved one class of problems, we are not necessarily interested in the old problem but rather in solving the new problem. Thus, generality across classes of problems is not critical. Consequently,

- G initially we would be interested in reliable absolute evaluations and subsequently in reliable comparisons.
- D We are less interested in multiple domain comparisons and are interested in artificial data only as a way to understand how to work with real data.
- P While we might be interested in both supervised and unsupervised learning, we will focus only on evaluating supervised learning.

The following discussion focuses on absolute evaluation and comparisons between supervised ML programs.

5.1 Absolute evaluation of ML programs

Absolute evaluation requires the least biased method (and one with small variability). Most empirical studies found K to be reasonably unbiased and with reasonable variability. In general, K was found to be more stable than L, and given its reasonable computational requirements, *K is the recommended test for absolute evaluation.* For small databases one has to use L or B although there are cases where both fail (see Section 2.2.5). *When better estimations are required and if computational resources are available K^I should be used.* It is sufficient to use $I = 10$ for this test.

In order to get most out of the evaluation process, *we recommend executing all the evaluation methods* and plotting graphs similar to Figure 2. Any deviation from the anticipated results requires explanation and might lead to better understanding of the data, ML program, the evaluation method, or their interaction.

An important aspect when solving practical problems is obtaining the best possible performance out of the data. Therefore, it is natural to wish to optimize ML program parameters. However, this requires special attention related to evaluation. Figure 6 illustrates an estimation method called training-testing-validation (TTV) that can be used for tuning operational parameters and options of programs, and finally, estimating the performance accuracy of the ML model. In the first step, the data is subdivided into data for model learning and model testing. In the second step, the data for model learning is used to select the best model (i.e., learning approach) and operational parameters. The evaluation in this step is done by K in order to generate better ground for selecting between the different parameters. In the third step, a model is created from the complete model learning set by the best approach and best operational parameters. This model is validated on the testing set. Obviously, the final experiment is a hold-out estimation method with all its limitations.

When optimization of parameters is not performed and parameters are selected based on "common practice" it is useful to *analyze the sensitivity of the evaluation to the parameter choices.* This can be done by conducting several parametric studies.

Any absolute evaluation should be accompanied with some measure of precision (or variability). When using *real data*, there is no direct way to estimate the variability of the evaluation caused by the sampling of D from F. The variability due to operational parameters could be addressed by sensitivity studies and the variability due to the testing process can be assessed by subsampling. As discussed earlier, different subsampling iterations are statistically dependent and care should be given to the calculation of confidence intervals. It could be approximated by assuming independence, but such confidence intervals better be checked using B. For real limited data, all the variability sources (including the one due to the sampling of D from F) could be estimated by B as described in Section 2.2.4. In our experiments we found small differences between confidence intervals calculated by B and the approximation by Equation 8.

When using *simulated data*, all these variabilities could be estimated by simply sampling different databases D and conducting experiments on them. Assessment of precision is time consuming especially for ANN. It is hardly ever estimated and consequently, evaluations lose much of their anticipated reliability.

5.2 Comparing between ML programs

So far we have addressed the performance evaluation of a single ML program. However, in order to improve solutions, we develop new methods and compare them to existing solutions. It has been common to compare between different ML programs by calculating their performance accuracy using K without performing any statistical analysis. This procedure may lead to wrong conclusions due to the variability of K as observed in our exercises. As in absolute evaluation, any comparison must be accompanied with a calculation of confidence interval or statistical significance.

Many researchers recommend the use of K or even K^I. Therefore, when comparing between programs, we could carry out K^I on each and test whether the difference between the two estimates (which involves comparing between two means) is statistically significant. First, t' is given by:

$$t' = \frac{(\bar{\hat{\theta}}_1 - \bar{\hat{\theta}}_2) - (\theta_1 - \theta_2)}{\hat{\sigma}_{diff}}, \tag{11}$$

where $\bar{\hat{\theta}}_1$ and $\bar{\hat{\theta}}_2$ are the estimates from applying K^I on both programs,

$$\hat{\sigma}_{diff} = \sqrt{\hat{\sigma}^2_{\bar{\hat{\theta}}_1} + \hat{\sigma}^2_{\bar{\hat{\theta}}_2}}, \tag{12}$$

and $\hat{\sigma}_{\bar{\hat{\theta}}_1}$ and $\hat{\sigma}_{\bar{\hat{\theta}}_2}$ are the standard errors of the I evaluations of K within each K^I as calculated by Equation 6. (Again, this calculation assumes independence and is only an approximation.)

The following consideration apply sequentially for obtaining a value for t' for some desired confidence:

(1) If I is large enough (say, above 30), we can use z values of the normal distribution instead of t' (Winer, 1962).
(2) If we assume equal standard error of the evaluation methods, and the observed errors from the samples are different we can use a pooled standard error. For I equal in both methods, $\hat{\sigma}_{diff}$ stays as above. We can then use the t distribution to obtain the value of $t' = t(\alpha/2; \nu)$ for $\nu = 2 \cdot (I - 1)$ degrees of freedom, and the desired significance α.
(3) If $\sigma_{\bar{\hat{\theta}}_1}$ is different from $\sigma_{\bar{\hat{\theta}}_2}$ then t' *does not follow* a normal or a t distribution and the next three approximations could be used.
(4) When comparing between two programs, if we maintain equal number of iterations I in both evaluations, we could use a t distribution as in item 2 because the comparison becomes less sensitive to deviations from the test assumptions (Hays, 1988).
(5) Otherwise, we can use the t distribution with corrected degrees of freedom (Welch, 1947). For I equal in both methods, $\nu = (I-1)/(c^2 + (1-c)^2)$, where $c = \hat{\sigma}^2_{\bar{\hat{\theta}}_1} / (\hat{\sigma}^2_{\bar{\hat{\theta}}_1} + \hat{\sigma}^2_{\bar{\hat{\theta}}_2})$.
(6) In any case, if the two evaluation tests were run with different number of iterations I_1, I_2, using the t distribution with degrees of freedom $\nu = \min(I_1 - 1, I_2 - 1)$ would give conservative results, i.e., differences found significant are guaranteed to be significant when using a correct test.

After choosing the correct t' for the confidence $1 - \alpha$, a confidence interval can be calculated by:

$$\theta_1 - \theta_2 \in [(\bar{\hat{\theta}}_1 - \bar{\hat{\theta}}_2) \pm t' \cdot \hat{\sigma}_{diff}], \tag{13}$$

That is, with confidence $1 - \alpha$ the difference between the estimation will be in the range. If the lower bound exceeds 0, it means that with confidence $1 - \alpha$ the true estimations are different.

There are three complications that arrise when comparing between programs. First, the evaluations of two programs are dependent since they use the same database for training and testing. A simple improvement for dealing with this dependency is the use of a *paired sample t* test. This requires that all programs use the same training and testing data in parallel. This poses a problem because one cannot use published results that do not specify how to replicate the data subdivisions for the purpose of employing this statistical test.

When using pairing, the standard error $\hat{\sigma}_{diff}$,

$$\hat{\sigma}_{diff} = \sqrt{\hat{\sigma}^2_{\bar{\hat{\theta}}_1} + \hat{\sigma}^2_{\bar{\hat{\theta}}_2} - 2cov(\bar{\hat{\theta}}_1, \bar{\hat{\theta}}_2)}, \tag{14}$$

replaces the one from Equation 12 (Hays, 1988). In most cases pairing leads to positively correlated $\bar{\hat{\theta}}_1$ and $\bar{\hat{\theta}}_2$. Hence, $\hat{\sigma}_{diff}$ becomes smaller and the confidence intervals will be smaller. In this case therefore, a paired sample test without considering this dependence becomes conservative.

We can even avoid considering this dependence if instead of comparing between the results of the methods on the complete set, we do a comparison on each testing instance (Hays, 1988). This will lead to a simples t test but with $I - 1$ instead of $2(I - 1)$ degrees of freedom which makes the test less powerful.

The second complication arises when using ANN in regression to predict several output (dependent) variable. In our formulation, we return to the vector notation of $\mathbf{y}$ instead of the scalar y. If we wish to compare between the performance of two ML programs as reflected by the results of this vector of variables, we must use multivariate tests to account for the dependency between them. In the simple case of comparing between two programs we could use the T^2 test, which could be viewed as the extension of the t test to multivariate populations (Morrison, 1990). In more complicated comparisons we would have to use other tests based on multivariate analysis of variance (MANOVA) (Morrison, 1990).

Figure 7 illustrates the difference between uni- and multivariate analyses. Suppose the test is to check whether the mean of the population $\mathbf{y} = [y1, y2]$ is different from $[0, 0]$ with confidence beyond 0.95. In (I), the distribution is such that the independent use of the univariate t tests would not reveal any statistical difference (i.e., the confidence intervals both overlap $[0, 0]$) while the multivariate test would reveal significance. In (II), we see that both tests reveal statistical significance beyond the desired confidence. In such cases, if we find significance with the univariate test, we would be able to skip the more complicated one. Nevertheless, since the use of simple and complicated tests is available in commercial statistical packages, one can easily use the appropriate tests. The complexity only lies in interpreting the results.

The third complication involves, in the general case, a comparative evaluation of several ML programs tested on several databases. In this case, previous tests need to be modified to account for the *multiplicity effect*. Consider that the evaluation of each program on each database gives us a sample by executing a test several times. The overall comparison would involve multiple comparisons between means. If we compared I means by using $J = \binom{I}{2} = I(I-1)/2$ independent comparisons and found in each comparison statistically significant difference with confidence $1-\alpha$ we would have a chance of $1-(1-\alpha) = \alpha$ to make an error in one judgment and $1-(1-\alpha)^J$ to make at least one error in all J comparisons. To illustrate, if we compared between 3 programs on 2 databases we would have 6 means and 15 comparisons. Given an original confidence of 0.95, we would have a chance of $1 - 0.95^{15} = 0.54$ to make at least one error in these comparisons. In order to reduce this chance one can adjust the significance level α to α/J, also called Bonferroni adjustment (Hays, 1988). For a small number of tests this adjustment is appropriate but it is overly conservative for a large number of tests. Other corrections or tests have been developed and are available in commercial statistics packages. Again, the problem is not using these tests but selecting the right test and interpreting its results.

5.3 General considerations

There are several general considerations related to evaluation.

(1) There is no best ML method or error estimation method. One has to become familiar with the interaction between evaluation methods, ML methods, and problems to fit the former to the others. Following a preliminary attempt to generate a mapping between ML program applicability and problem characteristics (Gama and Brazdil, 1995), one could try and extend such or different analysis to the applicability of evaluation tests given ML programs and problems. This would be more complex than the above attempt.

(2) When comparing between programs care should be exercised to have a sound basis for comparison, for example, give the programs similar amount of training. When dealing with ANN, there are two ways to interpret "equal training": use equal sum system error or equal number

of epochs. In the marine propeller exercise we used the former and in the material corrosion the latter. Both may be acceptable if we avoid excessive overfitting. Equal number of epochs is harder to control in this respect. Different settings of initial weights can lead to very different training speeds. Insisting on a fixed number of epochs can easily drive an easily trained network to the range of overfitting.

(3) K and H have large variabilities. K^I and H^I, being means of several estimates, have much smaller variabilities.

(4) The use of non-parametric methods such as B for estimating the precision of a statistic allows to remove assumptions about the distribution of the statistic but increases the size of the resulting confidence interval. Therefore, whenever parametric methods are applicable they should be used.

(5) When evaluation results contradict those anticipated from Figure 2, they need to be studied and justified.

(6) Engineering relevance must play a critical role in selecting between evaluation methods. There is no need to spend effort on selecting between methods or using time consuming one (e.g., K^I) if the use of a simpler method will lead to acceptable results from the engineering perspective.

6 Conclusions

The status of evaluating ANN applications is poor as revealed by a survey of articles that appeared from 1992 to 1997 in *Artificial Intelligence in Engineering*. This is also the prevalent status of evaluation elsewhere. Given the critical nature of evaluation it is essential to improve this situation. The first step involves improving our understanding of evaluation methods. Towards this end, we reviewed several error estimation methods and discussed their properties. The discussion identified methods that lead to good error estimates and some that yield poor estimates. We demonstrated the two assertions from the introduction in two case studies: (1) the evaluation methods indeed are very different from one another and (2) there is no way to determine *a priori* the relations between the results obtained by running them. Therefore, in order to estimate the accuracy of ML models correctly, one has to select carefully the appropriate methods.

While no evaluation method is superior to others in all contexts, some are better than others in certain practical settings and others are clearly deficient in certain contexts. It is important to build classification knowledge that given a learning problem and a program will assist us in identifying appropriate evaluation methods and warn us against others.

There are many issues that compound the problem of performance evaluation such as data quality, whether certain error calculations can be performed (e.g., percentage error with correct zero values cannot be calculated), statistical assumptions underlying evaluation procedures, etc. Therefore, modifications to existing tests might be inescapable. In such cases, it is critical to explain and justify such deviations.

In order to evaluate ML programs or models one has to become familiar with statistics beyond the basic level. We identified several complications that require special treatment and there can be others depending on the particular evaluation context. It is necessary to understand statistical tests including their assumptions in order to select between them and interpret their results.

Without proper evaluation there is no meaning to results. Therefore, appropriate report of the evaluation performed in a study must be a prerequisite to its publication. Such report should include explanation of the nature of the data and program(s), the reason for selecting an evaluation method, the data necessary for checking the evaluation, the data needed to verify statistical tests that are reported, as well as explanations of deviations from anticipated results.

References

Bailey, T. L. and Elkan, C. (1993). "Estimating the accuracy of learned concepts." In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 895–900, Morgan Kaufmann.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*, Belmont, Waldsworth, CA.

Breiman, L. (1996). "Heuristics of instability and stabilization." *The Annals of Statistics*, 24(6):2350–2383.

Congleton, J., Berrisford, R., and Yang, W. (1995). "Stress corrosion cracking of sensitized type 304 stainless steel in doped high-temperature water." *Corrosion Science*, 51(12):901–910.

Demuth, H. and Beale, M. (1994). *Neual Networks Toolbox - For Use with MATLAB*, The Mathworks Inc., Natick, MA.

Denny, S. B., Puckette, L. T., Hubble, E. N., Smith, S. K., and Najarian, R. F. (1989). "A new usable propeller series." *Marine Technology*, 26(3):173–191.

Dietterich, T. G. (1996). "Statistical tests for comparing supervised classification learning algorithms." Technical report, Department of Computer Science, Oregon State University.

Efron, B. and Tibshirani, R. (1993). *An Introduction to the Bootstrap*, Chapman & Hall, New York, NY.

Efron, B. (1983). "Estimating the error rate of a prediction rule: Improvement on cross-validation." *Journal of the American Statistical Association*, 78(382):316–331.

Feelders, A. and Verkooijen, W. (1995). "Which method learns most from the data?." In *Preliminary papers of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 219–225.

Flexer, A. (1995). "Statistical evaluation of neural network experiments: minimum requirements and current practice." In Trappl, R., editor, *Proceedings of the Thirteenth European Meeting on Cybernetics and Systems Research*, pages 1005–1008, Vienna, Austria, Austrian Society for Cybernetic Studies.

Gama, J. and Brazdil, J. (1995). "Characterization of classification algorithms." In Pinto-Ferreira, C. and Mamede, N., editors, *Progress in Artificial Intelligence, 7th Portuguese Conference on Artificial Intelligence, EPIA-95*, pages 189–200, Berlin, Springer Verlag.

Gascuel, P. and Caraux, G. (1992). "Statistical significance in inductive learning." In Neumann, B., editor, *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 435–439, New York, NY, Wiley.

Geman, S., Bienstock, E., and Doursat, R. (1992). "Neural networks and the bias/variance dilemma." *Neural Computation*, 4(1):1–58.

Hays, W. L. (1988). *Statistics. Fourth edition*, Holt, Rinehart & Winston, New York.

Henry, R. J. (1994). "Methods of comparison." In Michie, D., Spiegelhalter, D., and Taylor, C. C., editors, *Machine Learning, Neural and Statistical Classification*, pages 107–124, Chichester, England, Ellis Horwood Publishers.

Highleyman, W. (1962). "The design and analysis of pattern recognition experiments." *Bell System Technical Journal*, 41(1962):723–744.

Jain, A. K., Dubes, R. C., and Chen, C. (1987). "Bootstrap techniques for error estimation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI 9(5):628–633.

Kohavi, R. and Wolpert, D. H. (1996). "Bias plus variance decomposition for zero-one loss functions." In *Procedings of the Thirteenth International Conference on Machine Learning*, Morgan Kaufmann.

Kohavi, R. (1995). "A study of cross-validation and bootstrap for error estimation and model selection." In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1137–1143, Morgan Kaufmann.

Michie, D., Spiegelhalter, D., and Taylor, C. C. (1994). *Machine Learning, Neural and Statistical Classification*, Ellis Horwood Publishers, Chichester, England.

Morrison, D. F. (1990). *Multivariate Statistical Methods*, McGraw-Hill, New York, NY, 3rd edition.

Neocleous, C. C. and Schizas, C. N. (1995). "Artificial neural networks in marine propeller design." In *Proceedings of ICNN'95 - International Conference on Neural Networks, Vol.2*, pages 1098–1102, New York, NY, IEEE Computer Society Press.

Prechelt, L. (1996). "A quantitative study in experimental evaluation of neural network learning algorithms: Current research practice." *Neural Networks*, 9(3):457–462.

Reich, Y. (1995). "Measuring the value of knowledge." *International Journal of Human-Computer Studies*, 42(1):3–30.

Reich, Y. (1997). "Machine learning techniques for civil engineering problems." *Microcomputers in Civil Engineering*, 12(4):307–322.

Schaffer, C. (1994). "A conservation law for generalization performance." In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 259–265, Morgan Kaufmann.

Shao, J. (1993). "Linear model selection via cross validation." *Journal of the American Statistical Association*, 88(422):486–494.

Stine, R. A. (1990). "An introduction to bootstrap methods." *Sociological Methods and Research*, 18:243–291.

Stone, M. (1977). "Asymptotics for and against cross-validation." *Biometrika*, 64(1):29–35.

Weiss, S. M. and Indurkhya, N. (1994). "Decision tree pruning: Biased or optimal?." In *Proceedings of the 12th National Conference on AI*, pages 626–632, AAAI Press.

Weiss, S. M. and Kapouleas, I. (1989). "An empirical comparison of pattern recognition, neural nets, and machine learning classification methods." In *Proceedings of The Eleventh International Joint Conference on Artificial Intelligence, Detroit, MI*, pages 781–787, San Mateo, CA, Morgan Kaufmann.

Weiss, S. M. (1991). "Small sample error rate estimation for k-nearest neighbor classifiers." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):285–289.

Welch, B. L. (1947). "The generalization of Student's problem when several different population variances are involved." *Biometrika*, 34:28–35.

Winer, B. J. (1962). *Statistical Principles in Experimental Design*, McGraw-Hill, New York, NY.

Wolpert, D. H. (1996). "The lack of a priori distinctions between learning algorithms." *Neural Computation*, 8:1341–1390.

Tables

Table 1: Properties of error evaluation methods

Estimation method	Size of training set	Size of testing set	Number of internal iterations	Number of iterations	Method variability	Method bias
Resubstitution (R)	n	n	1	1	very high	very optimistic
Hold-out (H)	$(0.6 \div 0.8) \cdot n$	$(0.2 \div 0.4) \cdot n$	1	1	high	pessimistic
Random subsampling (H^I)	$(0.6 \div 0.8) \cdot n$	$(0.2 \div 0.4) \cdot n$	1	$I << n$, O(10)	moderate-high	pessimistic
Cross-validation						
K-fold CV (K)	$n(k-1)/k$	n/k	$k(\sim 10)$	1	moderate-high	nearly unbiased
I K-fold CVs (K^I)	$n(k-1)/k$	n/k	$k(\sim 10)$	$I << n, O(10)$	moderate	nearly unbiased
Leave-one-out (L)	$n-1$	1	n	1	moderate-high	nearly unbiased
.632 Bootstrap (B)	n (see text)	n (see text)	1	$I(50 \div 200)$	low	slightly optimistic

Table 2: Error rates for marine propeller behavior

Exercise	KT	KQ	Efficiency
(1) Resubstitution	5.92	3.46	3.41
(2) .632 Bootstrap ($I = 92$)			
Minimum	5.19	3.12	3.15
Maximum	8.35	4.94	4.31
Mean	6.81	3.90	3.59
Std. Dev.	0.69	0.32	0.23
(3) Cross-Validation			
(i) Leave-one-out	8.04	4.21	3.62
(ii) K-fold analysis			
Minimum	7.49	4.03	3.93
Maximum	8.52	4.93	4.34
Mean	7.91	4.57	4.18
Std. Dev.	0.36	0.28	0.12
(4) Hold-out			
Minimum	6.93	4.57	4.23
Maximum	13.69	7.31	5.38
Mean	9.79	5.79	4.60
Std. Dev.	1.46	0.70	0.26

Table 3: Mean square error for material corrosion analysis: raw data

Exercise	Crack length ($\times 10^{-3}$)	UTS ($\times 10^{-3}$)	Time of failure ($\times 10^{-3}$)	Reduction of area ($\times 10^{-3}$)
(1) Resubstitution	0.53	0.63	0.26	0.95
(2) .632 Bootstrap ($I = 100$)				
Minimum	0.50	0.40	0.20	3.80
Maximum	7.00	2.70	11.40	51.20
Mean	2.10	1.00	1.30	17.50
Std. Dev.	1.70	0.40	2.70	9.10
(3) Cross-validation				
(i) Leave-one-out	4.60	1.60	2.50	44.0
(ii) K-Fold Analysis				
Minimum	3.19	1.48	0.74	33.2
Maximum	5.62	2.11	2.73	47.6
Mean	4.42	1.77	1.02	41.1
Std. Dev.	0.87	0.20	0.59	4.3
(4) Hold-out				
Minimum	1.52	0.89	0.73	31.86
Maximum	28.41	3.82	1.56	83.94
Mean	6.09	2.29	1.10	54.33
Std. Dev.	8.06	0.97	0.33	19.05

Table 4: Mean square error for material corrosion analysis: cleaned data for RA

Exercise	Crack length ($\times 10^{-3}$)	UTS ($\times 10^{-3}$)	Time of failure ($\times 10^{-3}$)	Reduction of area ($\times 10^{-3}$)
(1) Resubstitution	0.40	0.60	0.20	1.40
(2) .632 Bootstrap ($I = 100$)				
Minimum	0.20	0.30	0.10	1.20
Maximum	8.40	7.20	1.06	3.22
Mean	1.60	1.10	0.60	6.70
Std. Dev.	1.50	0.80	1.50	4.70
(3) Cross-validation				
(i) Leave-one-out	3.20	1.50	0.50	15.40
(ii) K-Fold Analysis				
Minimum	2.77	1.51	0.54	13.20
Maximum	6.81	2.44	0.90	32.70
Mean	3.92	2.03	0.71	19.40
Std. Dev.	1.28	0.30	0.12	5.80
(4) Hold-out				
Minimum	1.00	1.37	0.44	13.39
Maximum	8.37	3.96	5.14	72.03
Mean	4.29	2.69	1.32	35.64
Std. Dev.	2.59	0.82	1.39	17.35

Table 5: Survey of empirical studies of algorithms in leading ANN journals (Flexer,1996)

	Requirement	Requirement was met		
		Yes	No	Unclear
1.	The use of different training and testing set	72.2 %	1.6 %	26.2 %
2.	Computation of multiple runs using an appropriate resampling techniques	57.3 %	36.1 %	6.6 %
3.	The use of 3rd independent data set in case of parameter tuning	4.9 %	0.0 %	95.1 %
4.	Reporting of mean, variance, confidence intervals	27.7 %	72.3 %	0.0 %
5.	Statistical test for comparison of performances	4.9 %	93.5 %	1.6 %

Table 6: Use of evaluation methods in published in *AI in Engineering*

	Evaluation method	Nature of Data		Total
		Simulated	Real	
1	R	7	3	10
2	H	1	5	6
3	K	0	0	0
4	L	0	1	1
5	B	0	0	0
6	TTV	2	2	4
7	NM	10	1	11

Resubstitution (R), Holdout (H), Leave-one-out (L), K-fold (K), Training-testing-validation (TTV), Not (clearly) Mentioned (NM)

Figure Captions

1. Performance accuracy estimation using a k-fold cross-validation method
2. Relative performance of different error estimation methods
3. Error rates for marine propeller behavior
4. Mean square error for material corrosion analysis: raw data
5. Mean square error for material corrosion analysis: cleaned data for RA
6. Optimizing parameters using a 3-stage evaluation process
7. Univariate versus multivariate confidence intervals

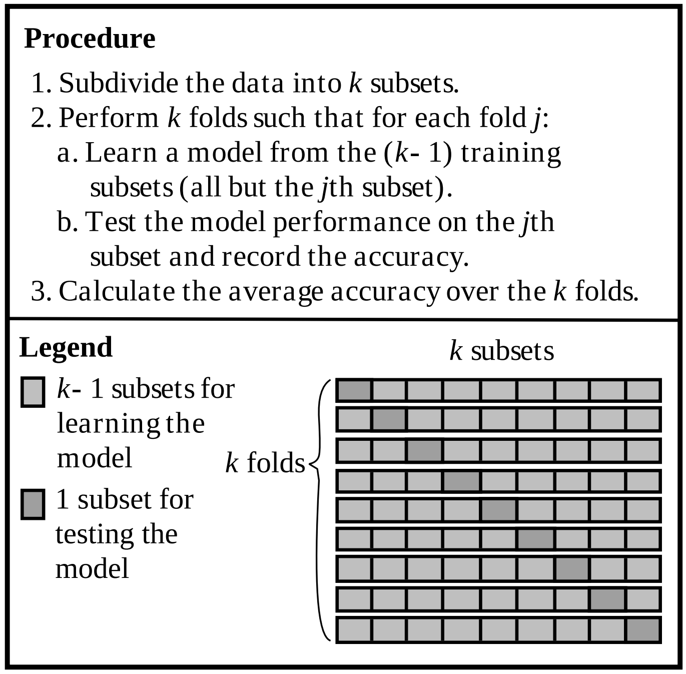

Figure 1: Performance accuracy estimation using a k-fold cross-validation method

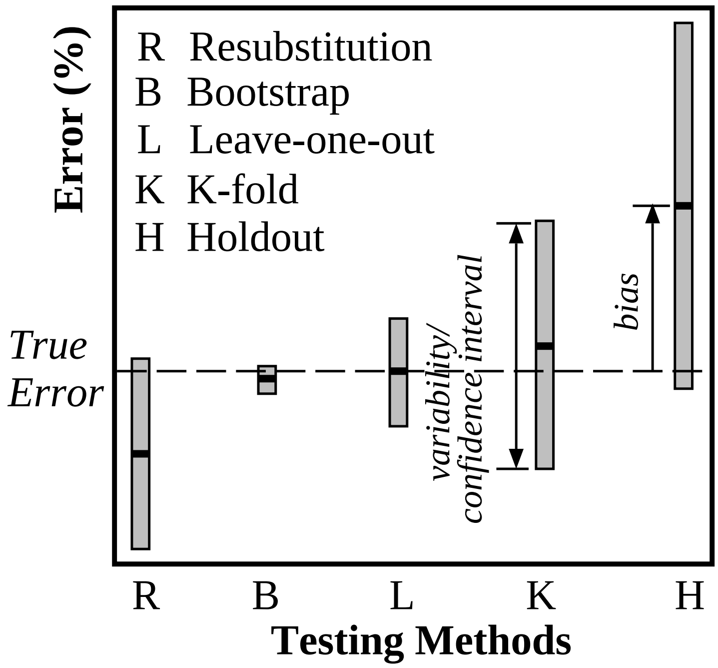

Figure 2: Relative performance of different error estimation methods

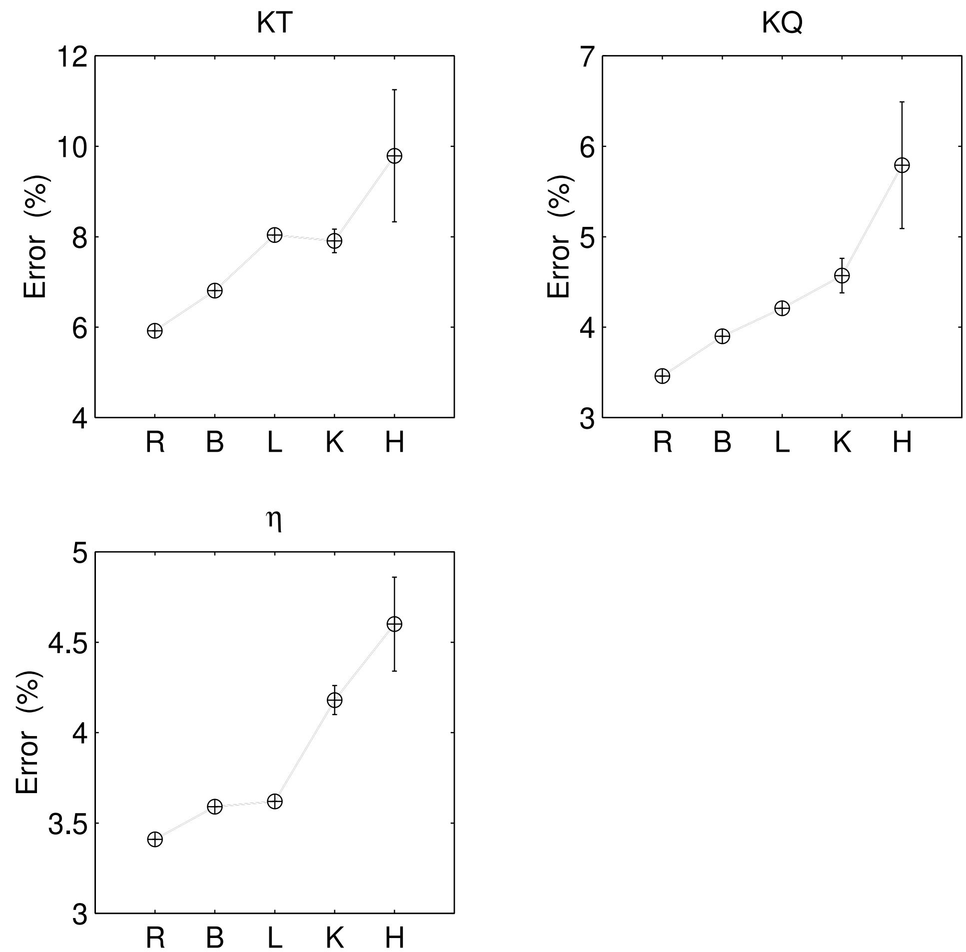

Figure 3: Error rates for marine propeller behavior

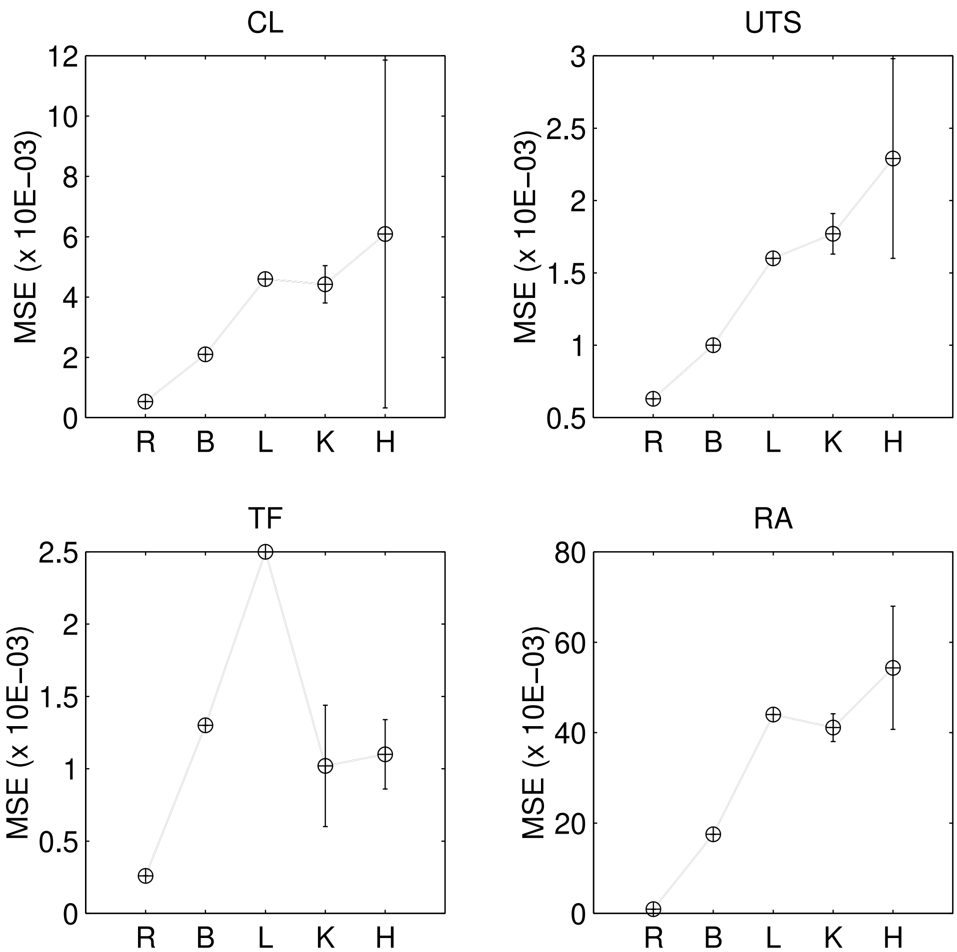

Figure 4: Mean square error for material corrosion analysis: raw data

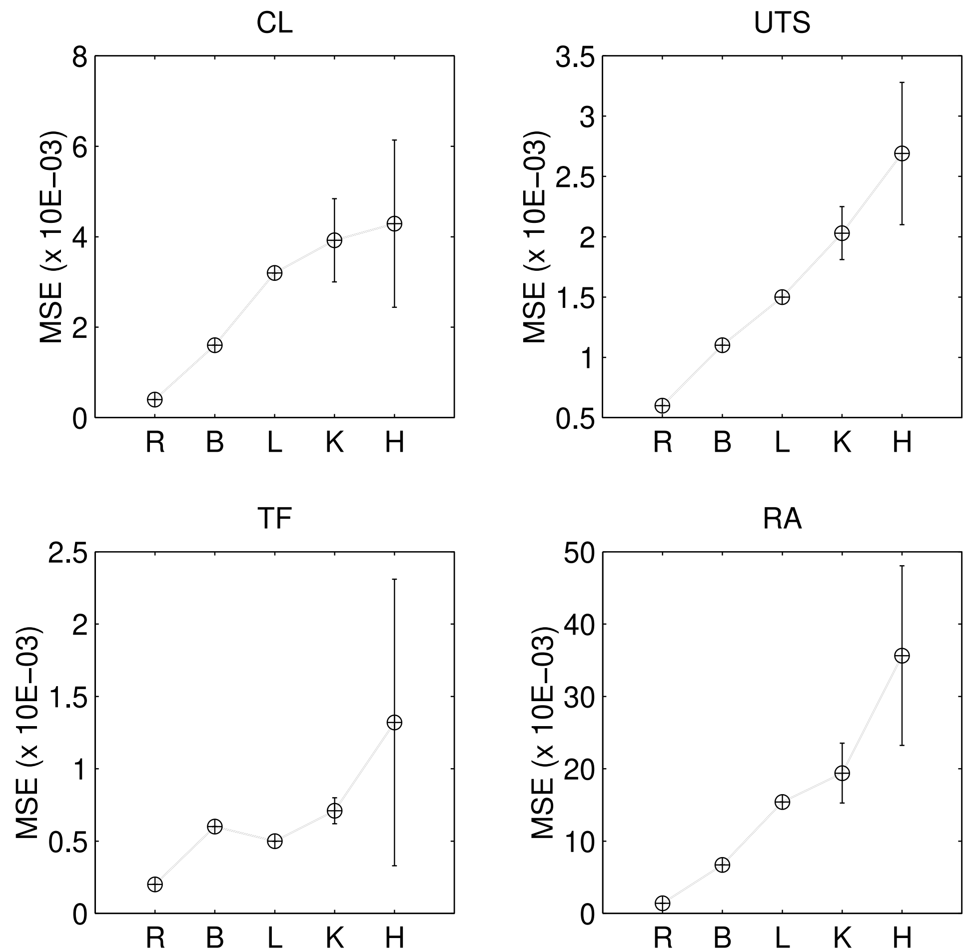

Figure 5: Mean square error for material corrosion analysis: cleaned data for RA

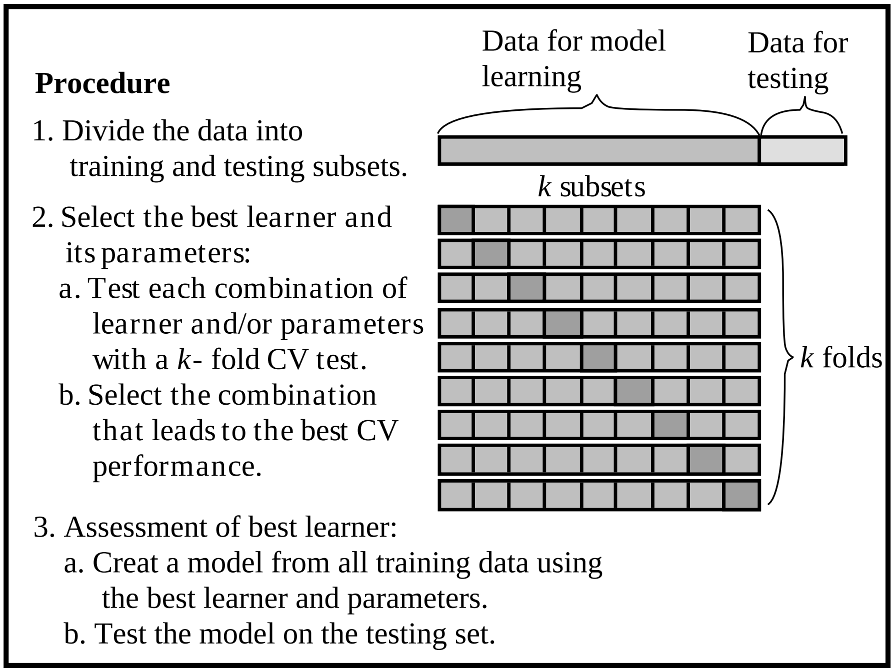

Figure 6: Optimizing parameters using a 3-stage evaluation process

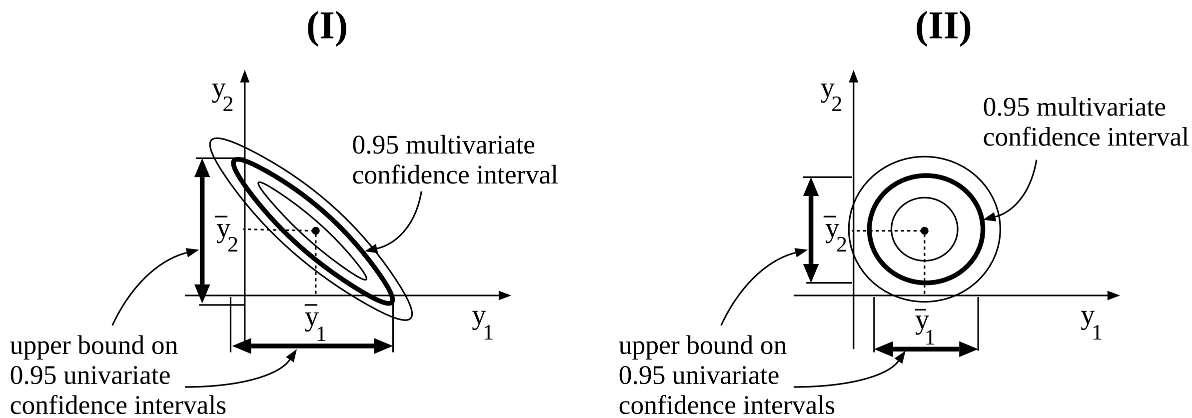

Figure 7: Univariate versus multivariate confidence intervals