Bias and Variance of Validation Methods for Function Approximation Neural Networks Under Conditions of Sparse Data

Janet M. Twomey
Department of Industrial and Manufacturing Engineering
Wichita State University
Wichita, Kansas

Alice E. Smith, Senior Member IEEE
Department of Industrial Engineering
University of Pittsburgh
Pittsburgh, Pennsylvania

Accepted to *IEEE Transactions on Systems, Man, and Cybernetics*

January 1998

Bias and Variance of Validation Methods for Function Approximation Neural Networks Under Conditions of Sparse Data

Abstract

Neural networks must be constructed and validated with strong empirical dependence, which is difficult under conditions of sparse data. This paper examines the most common methods of neural network validation along with several general validation methods from the statistical resampling literature as applied to function approximation networks with small sample sizes. It is shown that an increase in computation, necessary for the statistical resampling methods, produces networks that perform better than those constructed in the traditional manner. The statistical resampling methods also result in lower variance of validation, however some of the methods are biased in estimating network error.

1. INTRODUCTION

To be beneficial, system models must be validated to assure the users that the model emulates the actual system in the desired manner. This is especially true of empirical models, such as neural network and statistical models, which rely primarily on observed data rather than analytical equations derived from first principles. Validation of these models using problem-specific information, such as theoretic relationships or experiential knowledge, should be performed where possible. This paper focuses on the empirical aspect of model validation. Substantial amounts of data must be available to both construct and validate empirical models in the traditional manner, however, for some systems data is only available at great expense. The quandary becomes how to best use the relatively sparse data to both construct the model, and then validate it, prior to use. To this end, the research in this paper focuses on the adaptation and evaluation of statistical validation methods to the field of neural networks.

While the domain of neural networks originally arose from researchers trying to develop

computing mechanisms to emulate the biological brain, neural networks can also be viewed as a super set of statistics. Neural networks are a super set because they have been theoretically proven to serve as universal approximators when they include nonlinear activation functions [42]. The trained weights of a neural network are a vector-valued statistic, and training is the process of computing that statistic. The relationship between neural network models and statistical models has been the subject of several recent papers by well known statisticians [4, 5, 27] with the general conclusion that there are many important parallels between the development of neural network models and the computation of statistical models. Therefore, the approach of this paper is to construct and validate neural networks under conditions of sparse data[1] by adapting the statistical literature on evaluating estimators of model prediction error. Judging the effectiveness of an estimator can be broken into two fundamental aspects — bias and variance. Bias measures the expected value of the estimator relative to the true value and variance measures the variability of the estimator about the expected value. An ideal estimator (and an ideal model) will have no bias and low variance. To achieve low variance and no bias under conditions of small samples, the sample observations must be leveraged maximally. That is, they must all contribute to both the final model itself and to the model validation. The statistical literature contains several error estimation methods that achieve this by substituting computational effort for the unavailable larger data sample.

Described below are five error estimation methods used for evaluating prediction models:

---

[1] Under conditions of ample data, model building is less dependent on decisions concerning allocating the sample between model building and model validation.

resubstitution, cross-validation (CV), jackknife, bootstrap and train-and-test (also known as data splitting). The notation and terminology in this paper are adapted from Efron's work that examines three resampling methods (CV, jackknife, bootstrap) used to estimate the error of statistical prediction models [6-9]. The resampling methods allow the final model to be built on the entire sample of *n* observations, while the error estimate is derived from repeated sampling of *n* to construct multiple validation models. Throughout this paper, the model that is constructed and ultimately applied to the problem is referred to as the *application model*. Models constructed to estimate the prediction error of the application model are referred to as *validation models*. This paper centers on neural network models so that *network* is used interchangeably with *model*, however the discussion may be applicable to other data-driven models as well.

2. ERROR ESTIMATION METHODS

Assume that there is a population *F* from which a random sample, T_n, of size *n* is drawn, $T_n=\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$; where $t_i=(\mathbf{x}_i, y_i)$ is a realization drawn iid from *F*. A training pair $t_i=(\mathbf{x}_i, y_i)$, consists of a predictor (feature) vector, $\mathbf{x}_i$, and the corresponding output (response) variable, y_i. Both $\mathbf{x}_i$ and y_i may be continuous, or in cases where the prediction problem is classification, binary; furthermore y_i may be a vector but, without the loss of generality, is restricted to a scalar for purposes of simplifying this discussion. $\hat{F}$ is the empirical distribution (the sample of *n* observations) putting equal mass (1/*n*) on each observation t_i of T_n, for i = 1 to *n*. From T_n, a prediction model $\hat{f}(T_n, \mathbf{x})$ is constructed, so that for any $\mathbf{x}_o$, randomly but independently chosen from the same population *F*, y_o can be predicted.

Prediction error is measured according to some specified loss function *L*, typically the squared or absolute error. For the remainder of this paper *L* will be the absolute error, as it is

commonly used in both statistical and neural network models of continuous and classification relationships. The expected true error, Err, of the model $\hat{f}(\mathrm{T}_n, \mathbf{x})$ is defined as the expected error the model makes on any observation $\mathbf{x}_o$ taken from *F*:

$$\mathrm{Err} = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \left| \mathrm{y}_i - \hat{f}\left[\mathrm{T}_n, \mathbf{x}_i\right] \right| \qquad (1)$$

Err can only be calculated by a census of all N members of the population *F*; the primary validation objective is to estimate Err from the available sample, and this estimate is denoted as $\hat{\mathrm{E}}\mathrm{rr}$.

There are several methods available to perform the validation task. The most common statistical methods are resubstitution, CV and variants, and bootstrap and variants. The most common neural network method of prediction error estimation is train-and-test, more commonly known as data splitting in the statistical literature.

Resubstitution: True error is estimated using the same sample that is used to construct the model (all data is *resubstituted* back into the model [18, 29]):

$$\hat{\mathrm{E}}\mathrm{rr}_{\mathrm{Resub}} = \frac{1}{n} \sum_{i=1}^{n} \left| y_i - \hat{f}\left[\mathrm{T}_n, \mathbf{x}_i\right] \right| \qquad (2)$$

The resubstitution error is also termed *apparent error* in the statistical literature. Many papers of statistical models report $\hat{\mathrm{E}}\mathrm{rr}_{\mathrm{Resub}}$ to be a badly biased downwards estimator of Err [8, 12, 18, 19, 24, 30, 31]. Neural networks can be prone to over-parameterization, especially where data is sparse, therefore there is a greater likelihood for neural network $\hat{\mathrm{E}}\mathrm{rr}_{\mathrm{Resub}}$ to be more seriously biased downwards. The resubstitution method is computationally efficient and requires construction of only one model, which is used for both application and validation. Furthermore, the model utilizes the entire sample of *n* observations so that the application network is built from

the entire set of available data and the validation is over all *n*.

Cross-validation, Group Cross-validation, and Jackknife: Lachenbruch and Mickey [18] are responsible for the refinement of cross-validation (CV). $\hat{\mathrm{Err}}_{\mathrm{CV}}$ is determined by constructing *n* models on partitions of all data points of $\hat{F}$ leaving out one (size *n*-1), and then testing on the single omitted point. The cross-validation estimate is:

$$\hat{\mathrm{Err}}_{\mathrm{CV}} = \frac{1}{n}\sum_{j=1}^{n}\left| \mathrm{y}_j - \hat{f}\left[\mathrm{T}_{(j)}, \mathbf{x}_j\right] \right| \qquad (3)$$

where *j* is the excluded point and $\mathrm{T}_{(j)}=\{(\mathbf{x}_1,\mathrm{y}_1), (\mathbf{x}_2\ \mathrm{y}_2), ..., (\mathbf{x}_{j-1},\mathrm{y}_{j-1}), (\mathbf{x}_{j+1},\mathrm{y}_{j+1}),... , (\mathbf{x}_n,\mathrm{y}_n)\}$ is the data set except point *j* used to construct the model $\hat{f}\left[\mathrm{T}_{(j)}, \mathbf{x}_j\right]$. According to CV, *n* validation models plus one application model are built. The application model is built using all *n* observations. A computationally less burdensome variation of cross-validation is group cross-validation (GCV), where (usually) equal-sized groups of data are removed for each validation model, instead of a single observation. If $n = GH$, then *G* = total number of groups and *H* = total number of observations per group. The GCV estimate of Err is:

$$\hat{\mathrm{Err}}_{\mathrm{GCV}} = \frac{1}{n}\sum_{g=1}^{G}\sum_{h=1}^{H}\left| y_{(g-1)H+h} - \hat{f}\left[\mathrm{T}_{(g)}, \mathbf{x}_{(g-1)H+h}\right] \right| \qquad (4)$$

where *g* indexes the group left out, and $\mathrm{T}_{(g)}=\{(\mathbf{x}_1,\mathrm{y}_1), (\mathbf{x}_2,\mathrm{y}_2), ..., (\mathbf{x}_{(g-1)H},\mathrm{y}_{(g-1)H}), (\mathbf{x}_{(g)H+1},\mathrm{y}_{(g)H+1}),... , (\mathbf{x}_n,\mathrm{y}_n)\}$ is used to construct the prediction rule $\hat{f}\left[\mathrm{T}_g, \mathbf{x}_{(g-1)H+h}\right]$. *G* validation networks plus one application network are built. The original work on the jackknife methodology was by Quenouille [22, 26]. It was introduced to reduce the bias of a serial correlation estimator by splitting the data

into halves. Since the jackknife method of calculating $\hat{\text{Err}}$ is equivalent to the CV method, for all discussions in this paper the jackknife is considered equivalent to CV[2].

Bootstrap and Variants: Efron developed the bootstrap method of estimation and showed that it gives the nonparametric maximum likelihood estimate of the excess error of a prediction rule [6, 9]; i.e., the bootstrap corrects for the bias of the resubstitution error. The bootstrap, like CV, is a resampling technique, however it differs in the manner in which $\hat{F}$ is resampled. Bootstrap data sets are created by resampling $\hat{F}$ with replacement, whereas CV resamples $\hat{F}$ without replacement.

Bootstrap samples are generated as follows. Let $\hat{F}$ be the empirical distribution function for T_n with mass $1/n$ on $t_1, t_2, \ldots, t_n$; and let T_n^* be a random sample of size n taken iid with replacement from $\hat{F}$, where t_i^* is a single random observation, $t_i^*=(\mathbf{x}_i, y_i)$. Thus if an observation is selected twice, probability mass $2/n$ is assigned to that observation. True error is estimated through independent bootstrap training sets T^{*1}, T^{*2}, ..., T^{*B}; where B is the total number of bootstrap samples, each generated as described above. For each T^b, a prediction model is constructed, $\hat{f}\left[T^{*b}, \mathbf{x}_i\right]$, built using all n observations. The first term of the bootstrap estimate of Err is the resubstitution error of the application model. The second term is the difference, summed over all bootstrap validation models, between the average error over the original sample and the average error over the bootstrap sample. In other words, this second term subtracts off

---

[2] The jackknife differs from CV is its calculation of resubstitution error of a prediction model.

the mean resubstitution error from the mean error over the original sample, using the same bootstrap model. This is repeated *B* times.

$$\hat{\mathrm{Err}}_{\mathrm{BOOT}} = \frac{1}{n}\sum_{i=1}^{n}\left| \mathrm{y}_i - \hat{f}\left[\mathrm{T}_n, \mathbf{x}_i\right] \right| + \frac{1}{B}\sum_{b=1}^{B}\left(\frac{1}{n}\sum_{i=1}^{n}\left| y_i - \hat{f}\left[\mathrm{T}^{*b}, \mathbf{x}_i^*\right]\right| - \frac{1}{n}\sum_{i=1}^{n}\left| y_i^* - \hat{f}\left[\mathrm{T}^{*b}, \mathbf{x}_i^*\right]\right|\right) \quad (5)$$

There are a number of variations on the bootstrap. Three variations that are parsimonious in their required number of models were investigated in this research: E0, .632E and .632E′. The E0 [7] estimate of Err is the expected error of those observations not included in the bootstrap sample. Letting $\mathrm{A}_b = \{i \mid \mathrm{P}_i^{*b} = 0\}$ denote the number of training vectors that do not appear in the *b*th bootstrap sample, Err is estimated by averaging over *B* bootstrap training sets:

$$\hat{\mathrm{Err}}_{\mathrm{EO}} = \frac{\sum_{b=1}^{B}\sum_{\mathrm{A}_b}\left| \mathrm{y}_i - \hat{f}\left[\mathrm{T}^{*b}, \mathbf{x}_i\right] \right|}{\sum_{B} | \mathrm{A}_b |} \quad (6)$$

The .632E combines the resubstitution error with the EO error in proportions derived from the expected probability of an observation being included in a given bootstrap sample. Efron [7] reported the .632E estimator to be superior in his experiments. The expected true error is estimated by:

$$\hat{\mathrm{Err}}_{.632\mathrm{E}} = 0.368 * \frac{1}{n}\sum_{i=1}^{n}\left| \mathrm{y}_i - \hat{f}\left[\mathrm{T}, \mathbf{x}_i\right] \right| + 0.632 * \frac{\sum_{b=1}^{B}\sum_{\mathrm{A}_b}\left| \mathrm{y}_i - \hat{f}\left[\mathrm{T}^{*b}, \mathbf{x}_i\right] \right|}{\sum_{B}\left| \mathrm{A}_b \right|} \quad (7)$$

Efron showed that GCV with $G = 2$ and E0 are asymptotically equivalent [7] for large *n*. Due to the computational efficiency of using $\hat{\mathrm{Err}}_{\mathrm{GCV(2)}}$ as an estimate of E0, a third variation on the bootstrap, .632E′, was examined.

$$\hat{\mathrm{Err}}_{.632E'} = 0.368 * \frac{1}{n}\sum_{i=1}^{n}\left| \mathrm{y}_i - \hat{f}\left[\mathrm{T}, \mathbf{x}_i\right] \right| + 0.632 * \frac{1}{n}\sum_{g=1}^{2}\sum_{h=1}^{n/2}\left| y_{(g-1)n/2+h} - \hat{f}\left[\mathrm{T}_{(g)}, \mathbf{x}_{(g-1)n/2+h}\right] \right| \quad (8)$$

B validation models plus one application model (using all *n* observations) are constructed for all versions of the bootstrap (where *B* has traditionally ranged upwards of 20) excepting the .632E′ which constructs two validation models plus the application model.

Train-and-Test: The most commonly employed method of neural network evaluation is to obtain the estimate of Err from an independent set of data not used to train the model. This method is often referred to as "cross validating the network" and is reported as *test set error*. In the statistical literature it is called "data splitting." Given a sample of size *n*, subdivide it into two sub-samples such that $n_1 + n_2 = n$; where one sample, of size of n_1, is used for model construction $\hat{f}\left[\mathrm{T}_{n_1}, \mathbf{x}\right]$, and one sample, of size n_2, is used for validation.[3]. $\hat{\mathrm{Err}}_{\mathrm{T/T}}$ is the error over the validation set of size n_2:

$$\hat{\mathrm{Err}}_{\mathrm{T/T}} = \frac{1}{n_2}\sum_{\mathrm{j}=n_1+1}^{n}\left| \mathrm{y}_j - \hat{f}\left[\mathrm{T}_{n_1}, \mathbf{x}_j\right] \right| \quad (9)$$

Unlike the resubstitution and resampling methods, train-and-test does not utilize all available data in the construction of the application network. Instead it is built on a reduced set of data, n_1. Like the resubstitution method, only one model is constructed.

The methods just described have been studied in the statistical literature for a number of decades. CV was shown to provide unbiased estimates of Err for linear regression and statistical

---

[3] A variation of this uses a partition of three randomly chosen sets from *n*, one for training: one for testing the final trained network, and one for testing intermittently during training to prevent over-training.

classification problems by several authors [16, 18, 31]. Efron and Gong [6, 7, 9, 12] compared and contrasted the resampling methods on several regression problems involving dichotomous (0, 1) predicted values. According to Efron the methods can give considerably different estimates when applied to practical problems. CV gives a nearly unbiased estimate of Err but with unacceptably high variability, especially when *n* is small. The ordinary bootstrap gives a downwardly biased estimate of Err, especially in overfitted situations, but with low variability. Jain et al. [15] compared CV to the ordinary bootstrap and .632E bootstrap for pattern classification. They found that in most cases the bootstrap confidence intervals for the classifier error were narrower than those of CV. As an alternative to the resubstitution method, the train-and-test method has been shown in the statistical literature to provide unbiased estimates of Err for large sample sizes [13, 16, 18]. However when *n* is not large, the train-and-test method tends to give variable and overly pessimistic estimates of Err. When *n* is small, $\hat{\text{Err}}_{\text{T/T}}$ depends heavily on the partitioning of that data set [16, 31, 32, 40].

In the neural network literature, network models are primarily validated empirically [3, 19]. While many authors have compared various partitioning schemes for train-and-test, there are fewer works that attempt to use more computationally laborious forms of validation. CV and GCV have been suggested for use in evaluating neural network models [2, 4, 14, 20, 25, 38, 40, 42]. Moody [23] proposes a CV variation for nonlinear models which uses the final weights of the application networks as the initial weights of the validation networks. According to Moody, the nonlinear cross-validation (NCV) method provides a more suitable estimate since it is based on the application model. Although the bootstrap has been shown to provide very good estimates of error for statistical models, there are few instances of its use in the neural network literature. Recently Ankenbrand and Tomassini applied the bootstrap to a financial application [1], and

Lippman, et al. [21] and Huston et al. [14] utilized the bootstrap to evaluate networks built for medical applications. There has been some interest in the bootstrap as a method to construct network confidence intervals, and as a method to improve network generalization [17, 35, 36].

This paper provides a thorough and systematic analysis of four general methods for neural network validation. Specifically, resubstitution, various partitions for train-and-test, CV with various group sizes, the ordinary bootstrap with various *B*, and three bootstrap versions are compared for a simple function approximation problem under three levels of data sparseness. Several comparisons are pertinent: bias of the validation method, variance of the validation method, accuracy of the application network and computational effort. Additionally, the validation errors of the alternative approaches are partitioned into bias and variance components. While the results hereafter pertain to the function approximation problem, the results are consistent with those in the statistical literature and with other problems examined by the authors [14, 33, 34, 36, 37].

3. EXPERIMENTAL METHOD

A function approximation problem, also studied by Geman et al. [11], Efron [10] and Wahba and Wold [39], was used to examine and compare in detail the performance of the validation methods applied to neural network prediction models. All networks were trained to predict *y* given *x*,

$$y = 4.26(e^{-x} - 4e^{-2x} + 3e^{-3x}) \tag{10}$$

where *x* ranges from 0.0 to 3.10, as shown in Figure 1. This function was chosen because it has been previously studied in the statistical validation literature, its small size enabled the many computational experiments required, and its nonlinearity makes it a typical “difficult” function.

3.1 Experimental Design

A single experiment, or trial, proceeds as follows. One random sample, T_n, is generated, where T_n is the data set of n input/output pairs independently randomly sampled from the population F defined by the domain of the function to be approximated (equation 10). T_n represents the total sample available for building and evaluating the neural network. To eliminate effects of the exact sample chosen in any single trial, 100 trials were conducted and these 100 T_n were used to examine the performance of the validation methodologies.

The train-and-test method included four levels of how n is partitioned for training and testing (Table 1) and GVC varied on the number of groups G (Table 2). Three levels of total sample available were used, n = 5, 10 and 20. Therefore, the single research trial just described is repeated for each level of sample size. A network trained on 5 observations may be considered excessively small; however since equation 10 is a univariate function, n = 5 is the minimal acceptable sample size.

3.2 Methods of Assessment

To remove dependence on sampling for both training and testing, a single trial was replicated 100 times as stated above; T_n^k is a single realization of the random sample, and k = 1 to 100. In other words, for each of the three levels of n, 100 iid samples were pulled from F. The performance of the validation methodologies was evaluated and compared on several measures using these 100 T_n^k. The first performance measure is the prediction accuracy of the application network, where the true error, Err, of a single application network is approximated from N = 5000 independent randomly chosen observations, i.e., a separate set of 5000 was chosen to calculate the prediction error over the population F. The expected value of Err over the 100 samples drawn from F is given in equation 11.

$$E(\text{Err}) = \frac{1}{100}\sum_{k=1}^{100}\left(\frac{1}{5000}\sum_{i=1}^{5000}\left| y_i - \hat{f}[T_k, \mathbf{x}_i] \right| \right) \quad (11)$$

The variance of Err measures the sensitivity of the prediction performance of a network to the sample used to construct it, and is calculated over the 100 samples by:

$$V(\text{Err}) = \frac{1}{100}\sum_{k=1}^{100}\left(\text{Err}_k - E(\text{Err})\right)^2 \quad (12)$$

An informative measure of a validation method's performance is the mean squared error (MSE) of $\hat{\text{Err}}$, which pairs each $\hat{\text{Err}}_k$ with its corresponding Err_k. The MSE approximates the expected conditional disparity between $\hat{\text{Err}}_k$ and Err_k. The relevant equations are:

$$SE_k = (\hat{\text{Err}}_k - \text{Err}_k)^2 \quad (13)$$

$$MSE = \frac{1}{100}\sum_{k=1}^{100} SE_k \quad (14)$$

The variance of SE over the 100 networks is given by:

$$V(SE) = \frac{1}{100}\sum_{k=1}^{100}(SE_k - MSE)^2 \quad (15)$$

Of practical interest to the modeler is the probability that the error estimate of an individual network ($\hat{\text{Err}}_k$) will underestimate the true error of that network (Err_k). This is approximated by:

$$P(\hat{\text{Err}} < \text{Err}) = \frac{1}{100}\sum_{k=1}^{100} I_k \quad [\ \text{if } (\hat{\text{Err}}_k < \text{Err}_k)\ I_k = 1;\ \text{o.w. } I_k = 0\ ] \quad (16)$$

3.3 Network Architecture and Training

Prior to actual experimentation, network architecture, training parameters and stopping criteria were selected through experimentation and examination of networks trained on twenty independent samples T_n, at each level of n. The network architecture used was a multi-layered

fully connected perceptron with one input neuron (x), one hidden layer with three neurons and one output neuron (y). The transfer function was the unipolar sigmoid, and a traditional backpropagation learning algorithm [28, 41] was used with the learning rate = 0.1 and the momentum (smoothing) factor = 0.9. All networks were initialized to the same set of random weights between ±0.5. Network training ceased when mean weight change was below a given threshold (0.01) for 1000 generations. This termination approach was selected because it permits the evaluation of each validation method without the confounding effects of network termination by training set error or by testing set error.

4. RESULTS

This section begins with an analysis of the performance of the application networks. It continues with results of each method in estimating Err, both in terms of expected value (bias) and variability (variance).

4.1 Application Network Performance

Recall that the application network is the network constructed to be operational, that is, the outcome of the construction and validation effort. For the train-and-test method this network is constructed on the partition n_1 of the total sample, T_n, available. Therefore, the partitioning of T_n into the training sample, n_1, and the testing sample, n_2, may profoundly affect the performance of the application network. For resubstitution and the resampling methods, the application network is constructed on the entire sample, T_n, which is of size n. Therefore the application networks of all of these methods are identical.

Table 3 displays the expected Err and variance of Err over the 100 samples of each sample size, n = 5, 10, and 20. These are calculated using equations 11 and 12 for the set of 5000 that represent the population F. Figure 2 plots the expected Err vs. percentage of n used to construct

the application network. As anticipated, increasing a network's construction set size improves performance. Also shown in Figure 2, is the variability of Err vs. percentage of total available sample used for training purposes. The plots reveal substantial increases in the variance of Err as the training set size decreases. The large variability in Err results from the network's sensitivity to the composition of small training sets; i.e. when *n* is small, the Err of a network is greatly dependent upon which observations are chosen to make up the training set. Figure 3 shows the combination of Err and variance of Err for each application network. An ideal network would lie in the lower left corner (low error and low variability). It can be seen that larger sample sizes move the application networks toward the origin, while the application networks built using the entire sample (all but the train-and-test method) are the best networks for each value of *n*. To summarize these results, performance degrades and variability of performance increases as the size of the sample used to construct the application network decreases. These effects are seen most acutely in the sparser data sets.

4.2 Estimation of Err

This section considers the performance of the validation methods in estimating Err, and the variability of that estimate across the 100 samples at each sample size. Recall that the estimate of true error ($\hat{E}rr$) is given by equations 2 (resubstitution), 3-4 (cross-validation approaches), 5-8 (bootstrap and variations) and 9 (train-and-test). An ideal validation method would have $\hat{E}rr$ equal to Err on average (be unbiased) and have minimal $V(\hat{E}rr)$ (be insensitive to the exact sample chosen to estimate Err).

Table 4 provides expected $\hat{E}rr$ and variance of $\hat{E}rr$ for *n* = 5, 10 and 20 for the most

typical versions of the validation methods. These are resubstitution, CV, two fold GCV[4], train-and-test with 75% for training and 25% for testing, and the ordinary bootstrap at $B = 20$. For comparison, the expected Err over the population set of 5000 is also provided. The last column is the probability of underestimation of true error from equation 16. An unbiased validation method would have this value equal to 0.50.

Table 4 confirms the hypothesized results. As sample size, n, increases, both the error of validation and the variance of the validation estimate decrease. The resubstitution method is badly biased downwards, but has low variability. The train-and-test method is highly variable compared to all other methods and is biased upwards. CV is more precise and less variable than GCV, however this comes at a computational cost. The two fold creates only 2 validation models, while the non-group version creates 5, 10 and 20 validation models, respectively, for each sample size. Both versions of CV are relatively unbiased. The bootstrap requires 20 validation models, but results in less variance than CV. However, the bootstrap is biased downwards, though not nearly as severely as the resubstitution method.

4.3 Detailed Examination of Results for Train-and-Test and Cross-Validation

Figure 4 shows the MSE (as calculated using equation 14) and 95% confidence intervals (C.I.) for MSE given by the train-and-test method under different partitions. The C.I. are formed by:

---

[4] A "fold" for cross validation or jackknife refers to the number of groups, G, created for each sample. For example, a two fold for $n = 10$, would involve 2 validation models with a group size of 5 each. A five fold for $n = 100$, involves 5 validation models with a group size of 20 each.

$$\text{MSE} \pm z_{\frac{\alpha}{2}} \left(\frac{\sqrt{\text{V(SE)}}}{\sqrt{100}} \right) \tag{17}$$

where MSE is from equation 14 and V(SE) is from equation 15. For 95% C.I., $\alpha = 0.05$ and $z_{1-\alpha/2} = 1.96$. Note that all figures showing MSE confidence intervals have a truncated *y* axis since each confidence interval is symmetric. As *n* increases, the estimate is better and less variable. An interesting result is that the 90%/10% split yields highly variable estimates. This split is commonly found in the neural network literature, however it appears that a 10% test set for small sample sizes is inadequate for model validation.

Table 5 shows details of the cross-validation for different grouping strategies. Clearly the non-grouped cross-validation has superior results; however given the trade off in computational effort, the grouped versions are fairly effective. Figure 5 highlights this by examining the 95% MSE C.I. for sample sizes of 10 and 20. For $n = 20$, hardly anything is lost by going from non-grouped (20 validation networks) to a two fold approach (2 validation networks). This is an order of magnitude reduction in computational effort. However, for smaller sample sizes more groups are clearly better.

4.4 Comparison of Bootstrap Versions

The results of the ordinary bootstrap are presented in Table 6 for levels of bootstrap samples, $B = 1, 2, 5, 10, 20, 50, 60$ and 100. Almost counter-intuitively, the results do not indicate improved accuracy of $\hat{\text{Err}}$ with an increase in *B*. The improvement in $\hat{\text{Err}}_{\text{BOOT}}$ with an increase in *B* is shown only in the reduction in variability. $\hat{\text{Err}}_{\text{BOOT}}$ underestimates the true Err (shown under each sample size) at all levels of *B* (consistent with Efron's findings [7]), to a greater extent for smaller sample sizes.

The results on bootstrap variations E0, .632E, and .632E′ are provided in Table 7, along with the ordinary bootstrap results. The ordinary bootstrap, E0 and .632E are at $B = 20$[5]. For comparison, the expected true error (as calculated over the population of 5000) are 0.0874 ($n = 5$), 0.0551 ($n = 10$), and 0.0389 ($n = 20$). Table 7 shows that the .632E and .632E′ versions correct the downwards bias of the ordinary bootstrap. The EO actually over-corrects and results in an upwardly biased estimator. The best performing estimator, considering both variability and bias, is the .632E. The .632E′ also does well considering the order of magnitude reduction in computational effort.

4.5 Validation Methods Compared at the Same Level of Sample Size

This section focuses on comparing the validation methods in terms of MSE at the same level of n as shown in Figure 6. The vehicle for comparison is the 95% C.I. of MSE as given by equation 17. Note that the scales change between graphs A, B and C because of the effect of sample size. The train-and-test method produces the most variable estimates regardless of sample size, with extremely high variability using a small testing set (90%/10% split). As indicated earlier, this is worrisome because this is a typical validation strategy in the neural network literature. The resubstitution method has low variance, but is more biased for small samples. All resampling methods have low variance and good performance, except for the two fold GCV. These results must be considered in conjunction with the performance of the application network (discussed in Section 4.1), where inferior application networks resulted from the train-and-test

[5] Recall that the .632E′ method requires only two validation networks.

method.

Turning to comparison of computational effort, both the train-and-test and the resubstitution methods require the construction of only one network. CV requires the construction of n+1 networks, while GCV requires G+1. The bootstrap methods (except for .632E′) require B+1 networks. Choosing the number of validation networks for the bootstrap is a decision that can be made *a priori* or iteratively by the analyst. In this case, B = 20 gives low variability, and it is also the lower bound recommended by Efron. The .632E′ method uses the two fold GCV to estimate the error on data not included in the construction sample, and requires two validation networks plus an application network. It is slightly more variable and more biased than the .632E, but when considering the order of magnitude computational reduction, the .632E′ appears quite promising as a practical neural network validation method.

5. DECOMPOSING THE ERROR OF THE ESTIMATE INTO BIAS AND VARIANCE

Nonparametric estimators, such as neural networks, tend to be highly variable, whereas parametric estimators tend be biased. For small data sets, nonparametric estimators may be too sensitive to the chosen data sample, leading to increased variance. There is a bias-variance trade off; i.e., the decrease in one may cause an increase in the other. Research in estimation theory aims to achieve an estimator with the lowest possible MSE. The decomposition of MSE into bias and variance is useful to determine which term contributes the most. Once established, the effort to reduce MSE will focus on that particular term, or as an alternative, will seek to increase a positive covariance term (cov($\hat{\text{Err}}$, Err)). The general expression for MSE is expanded to explicitly consider the two components of the difference between $\hat{\text{Err}}$ and Err, *viz*. bias and variance, by:

$$\text{MSE} = \text{Bias}^2 + \text{Variance} \tag{18}$$

where the squared bias of the estimate is given by:

$$\text{Bias}^2 = \frac{\sum_{k=1}^{100} (\hat{\text{Err}}_k - \text{Err}_k)^2}{100} \tag{19}$$

and the variance is given by:

$$\text{V}(\hat{\text{Err}} - \text{Err}) = \frac{\sum_{k=1}^{100} \left(\hat{\text{Err}}_k - \text{E}(\hat{\text{Err}})\right)^2}{100} + \frac{\sum_{k=1}^{100} \left(\text{Err}_k - \text{E}(\text{Err})\right)^2}{100} - 2\left(\frac{\sum_{k=1}^{100} \left(\left(\hat{\text{Err}}_k - \text{E}(\hat{\text{Err}})\right)\left(\text{Err}_k - \text{E}(\text{Err})\right)\right)}{100}\right) \tag{20}$$

where the last term is twice the covariance between the estimate of Err and Err itself for each of the *k* networks.

The bias term is a measure of how well $\hat{\text{Err}}$ performs as an estimator of Err; i.e. bias is due to an inappropriate formulation of the estimate. The variance of the estimate, $\text{V}(\hat{\text{Err}})$, measures the stability of the estimate of true error over all realizations of the training set T_n^k. The variance of Err, V(Err), measures the stability of the actual true error over the same T_n^k. The covariance term (-2cov ($\hat{\text{Err}}$,Err)) is a measure of how the estimate and the true error vary together for each training sample.

Figures 7 through 9 show the decomposition by terms for the validation methodologies for the sample sizes 5, 10 and 20, respectively. The two components from equation 18 are shown in bold, dashed for the bias^2 term and solid for the variance term. The three terms which sum to the variance, given in equation 20, are shown, with the variance of the estimate dashed, the variance of the true error solid and the covariance term shaded. Please note that the scales on the three graphs are not identical — both bias and variance decrease with an increase in sample size — and

that zero is located at the midpoint of each axis. This is due to the negative value of the covariance term. The graphs are a succinct way to examine the relative magnitudes of the components that make up the MSE of the estimate. The train-and-test versions are the most variable, with the two fold GCV the second most variable. The covariance term is negative for the train-and-test, indicating a positive correlation between the error estimate and the true error across the 100 data samples. The variance of the true error is identical (and minimal) for all methods except the train-and-test. It can be viewed as a lower bound of total variance, as the variance of the true error depends only on the sample size used to construct the application network (where the method of model construction was held constant). For the train-and-test, true error is most variable for the 25/75 split because this version reduces the training set by 75%, resulting in a highly variable application network. As this construction sample grows larger, the dependency of the application network on the sample reduces, evidenced by reduced variability of the true error.

Figures 10 through 12 show the relative contributions of the terms of equation 18 to MSE for each sample size, along with the MSE located on top of each histogram. It is clear that the resubstitution method is the only one to contain substantial bias, relative to the total MSE. $V(\hat{Err})$ is the main component of MSE for CV, GCV and the train-and-test methods. This indicates that these methods are more dependent on the sample used for network construction and validation. The bootstrap's main contributor to MSE is the variability of the true error itself, which cannot be avoided, as the performance of predictive models for the population will always be dependent on the sample used to construct the model, especially for small *n*. Only an increase in *n* will reduce this variability. Also note that the value of MSE for each method is at top of the histograms, and the bootstrap is clearly the smallest.

The train-and-test methods offset some of the variability of the estimate by a positive covariance between the true error and the estimate of true error. Counter-intuitively, the correlation between Err and $\hat{E}rr$ for bootstrap and CV were generally negative. These results are consistent with Efron's results [7], who states there is no way to ensure a positive correlation for statistical prediction models, but the improvement exhibited by the .632E method is a consequence of driving the correlation to zero.

6. CONCLUSIONS

Based on the investigations of this paper and supported by further investigations by the authors [14, 33, 34, 36, 37], Table 8 has been assembled to offer general guidance in selecting a validation method for supervised neural network models. The table gives simplified, but succinct, information concerning these methods. The analyst can trade off the desired accuracy in the estimate of Err and the predictive accuracy of the application network with the computational effort allowed.

There are several standards that should be adopted by all neural network analysts when faced with constrained amounts of data. First, use the entire sample, *n*, to construct the final application network. Assuming a valid sample, a network constructed on more data will generally be superior to one constructed on less data. Second, the success of the traditional train-and-test validation using nearly all the sample for training (e.g. 90%) is highly dependent on the small number of observations left in the test set. This variability creates risk for the neural network builder and user. Third, the traditional resampling techniques — bootstrap, jackknife and CV — as enacted by the statistical community may be computationally impractical for the neural network community. The less computationally intensive versions of these — GCV and .632E′ — offer an attractive trade off. They lose little in the way of variability and bias, but reduce the number of

prediction models by order(s) of magnitude. The .632E′ appears particularly encouraging as a reasonable and effective procedure to validate neural network predictive models.

Finally, validation is a crucial part to the development of a sound empirical model. Neural networks will not realize their full potential in solving real problems until users are assured that the models will operate as intended. This can only be accomplished through validation. For large samples, validation will be adequate regardless of the method used, however for small samples, the validation approach can result in drastically different application networks and estimates of performance.

There are a number of extensions to the investigations reported herein. The experiments performed maintained the same architecture, initial weight set, training algorithm and parameters, and termination criterion to avoid confounding with the validation results. Research considering the effects of alterations in network construction and the validation methods would be beneficial. For example, validation might be more robust over different sets of initial random weights. Another important aspect for further research is to consider alternative data sampling methodologies, including imperfect sampling. This paper employed iid sampling (i.e., random uniform sampling over the data set available), however there are many instances when the neural network analyst will deliberately select a sample for training or validation using a design of experiments or problem-specific knowledge. There are also instances when sampling is imperfect (biased) either by choice or by happenstance. The influence of sampling on the validation methods is a second important area of further study. A third extension concerns pattern classification models, which have inherently different error metrics than function approximation models. Investigating the use of resampling methods for validation of classification neural networks may yield differing conclusions.

ACKNOWLEDGMENTS

The first author acknowledges the support of an NSF EPSCoR First Award. The second author gratefully acknowledges the support of NSF CAREER grant DMI 9502134.

References

[1] Ankenbrandm, T. and M. Tomassini (1996) "Predicting multivariate finical time series using neural networks: The Swiss bond case," *Proceedings of the IEE/IAFE 1996 Conference on Computational Intelligence for Financial Engineering*, New York, NY: IEEE Press, 27-33.

[2] Ben Brahim, S., A. E. Smith and B. Bidanda (1993) "Relating product specifications and performance data with a neural network model for design improvement," *Journal of Intelligent Manufacturing*, **4**, 367-374.

[3] Burke, L. (1993) "Assessing a neural net: Validation procedures," *PC AI*, March/April, 20-24.

[4] Cheng, B. and D. M. Titterington (1994) "Neural networks: A review from a statistical perspective," *Statistical Science*, **9**, 2-54.

[5] Cherkassky, V., J. H. Friedman, and H. Wechsler (Eds.) (1994) *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*. (NATO ASI Series) New York, NY: Springer-Verlag.

[6] Efron, B. (1982) *The Jackknife, the Bootstrap, and Other Resampling Plans.* SIAM NSF-CBMS Monograph, **38.**

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

[8] Efron, B. (1986) "How biased is the apparent error rate of the prediction rule?" *American Statistical Association*, **81**, 461-470.

[9] Efron, B. and R. Tibshirani (1986) "Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy," *Statistical Science*, **1**, 54-77.

[10] Efron, B. (1979) "Computers and the theory of statistics: Thinking the unthinkable," *SIAM Review: Society for Industrial and Applied Mathematics*, **21**, 460-480.

[11] Geman, S., E. Bienenstock and R. Doursat (1992) "Neural networks and the bias/variance dilemma," *Neural Computation*, **4**, 1-58.

[12] Gong, G. (1986) "Cross-validation, the jackknife, and the bootstrap: Excess error estimation in forward logistic regression," *Journal of the American Statistical Association*, **81**, 108-113.

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

[14] Huston, T. L., A. E. Smith and J. M. Twomey (1994) "Artificial neural networks as an aid to medical decision making: Comparing a statistical resampling technique with the train-and-test technique for validation of sparse data sets," *Artificial Intelligence in Medicine: Interpreting Clinical Data*, AAAI Press Technical Report SS-94-01, 70-73.

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

[16] Kanal, L. and B. Chandrasekaran (1971) "On the dimensionally and sample size in statistical

pattern classification," *Pattern Recognition*, **3**, 225-234.

[17] Kindermann, J., G. Paass and F. Weber (1995) "Query construction for neural networks using the bootstrap," *International Conference on Artificial Neural Networks '95*, **2**, 135-140.

[18] Lachenbruch, P. and M. Mickey (1968) "Estimation of error rates in discriminant analysis," *Technometrics*, **10**, 1-11.

[19] Larsen, L. E., D. O. Walter, J. J. McNew and W. R. Adey (1970) "On the problem of bias in error rate estimation for discriminant analysis," *Pattern Recognition*, **3**, 217-223.

[20] Lawrence, J. (1991) "Data preparation for a neural network," *AI Expert*, **11**, 34-41.

[21] Lippmann, R. P., L. Kukolich and D. Shahian (1994) "Predicting the risk of complications in coronary artery bypass operations using neural networks," in G. Tesauro, D. Touretzky and T. Leen (editors) *Advances in Neural Information Processing Systems 7*, Cambridge, MA: MIT Press.

[22] Miller, R.G. (1974) "The Jackknife - a review," *Biometrika*, **61**, 1-15.

[23] Moody, J. (1994) "Prediction risk and architecture selection for neural networks," in V. Cherkassky, J.H. Friedman, and H. Wechser (editors) *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*. (NATO ASI Series) New York, NY: Springer-Verlag.

[24] Mosier, C. I. (1951) "Symposium: The need and the means of cross-validation. I. Problems and design of cross-validation," *Education and Psychological Measurement*, **11**, 5-11.

[25] Nivelle, F., V. Rouy and P. Vergnaud (1993) "Optimal design of neural networks using resampling methods," *Proceedings of the Sixth International Conference on Neural Networks and Their Industrial and Cognitive Applications*, 95-106.

[26] Quenouille, M. (1949) "Approximate tests of correlation in time series," *Journal of the Royal Statistical Society Series B*, **11**, 18-84.

[27] Ripley, B. D. (1994) "Flexible non-linear approaches to classification," in V. Cherkassky, J.H. Friedman, and H. Wechler (Eds.) *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*. (NATO ASI Series) New York, NY: Springer-Verlag.

[28] Rumelhart, D. E., G. E. Hinton and R. J. Williams (1986). "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations.* D. E. Rumelhart and J. L. McClelland, and the PDP group, editors, MIT Press, Cambridge, MA, 318-362.

[29] Smith, C. A. B. (1947) "Some examples of discrimination," *Annals of Eugenics*, **18**, 272-278.

[30] Snee, R. D. (1977) "Validation of regression models: Methods and examples," *Technometrics*, **19**, 415-428.

[31] Stone, M. (1974) "Cross-validatory choice and the assessment of statistical predictions," *Journal of the Royal Statistical Society Series B*, **36**, 111-147.

[32] Toussaint, G.T, (1974) "Bibliography on estimation of misclassification," *IEEE Transactions on Information Theory*, **20**, 472-479.

[33] Twomey, J. M. (1995) *Nonparametric Error Estimation Methods For Validating Artificial Neural Networks*, unpublished Ph.D. Dissertation, Department of Industrial Engineering, University of Pittsburgh.

[34] Twomey, J. M. and A. E. Smith (1993) "Nonparametric error estimation methods for validating artificial neural networks," *Intelligent Engineering Systems Through Artificial Neural Networks, Volume 3*, (C. H. Dagli, L. I. Burke, B. R. Fernandez, J. Ghosh, editors), ASME Press, 233-238.

[35] Twomey, J. M. and A. E. Smith (1995) "Committee networks by resampling," *Intelligent Engineering Systems Through Artificial Neural Networks, Volume 5* (C. H. Dagli, M. Akay, C. L. P. Chen, B. R. Fernandez and J. Ghosh, editors), ASME Press, 153-158.

[36] Twomey, J. M. and Smith, A. E. (1996) "Artificial neural network approach to the control of a wave soldering process," *Intelligent Engineering Systems Through Artificial Neural Networks, Volume 6*, ASME Press, 889-894.

[37] Twomey, J. M. and Smith, A. E. (1997) "Validation and Verification," in *Artificial Neural Networks for Civil Engineers: Fundamentals and Applications* (N. Kartam, I. Flood and J. H. Garrett, editors), ASCE Press, New York, 44-64.

[38] Twomey, J. M., A. E. Smith and M. S. Redfern (1995) "A predictive model for slip resistance using artificial neural networks," *IIE Transactions*, **27**, 374-381.

[39] Wahba, G. and S. Wold (1975) "A completely automatic French curve: Fitting spline functions by regression by cross-validation," *Communications in Statistics Series A*, **4**, 1-17.

[40] Weiss, S. M. and C. A. Kulikowski (1991) *Computer Systems that Learn*. Morgan Kaufmann Publishers, Inc., San Mateo, CA.

[41] Werbos, P.J. (1974) *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, unpublished Ph.D. Dissertation, Harvard University.

[42] White, H. (1990) "Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings," *Neural Networks*, **3,** 535-549.

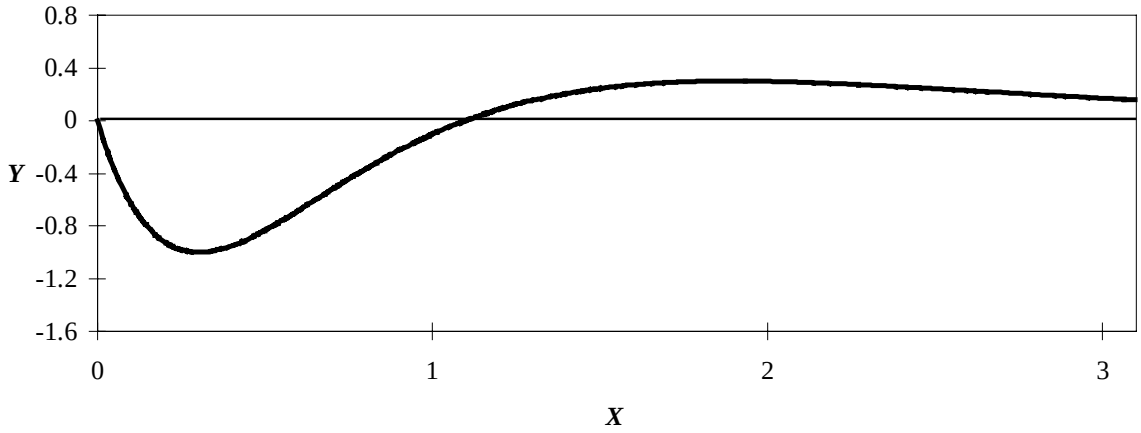


Figure 1. Function Approximation Problem.

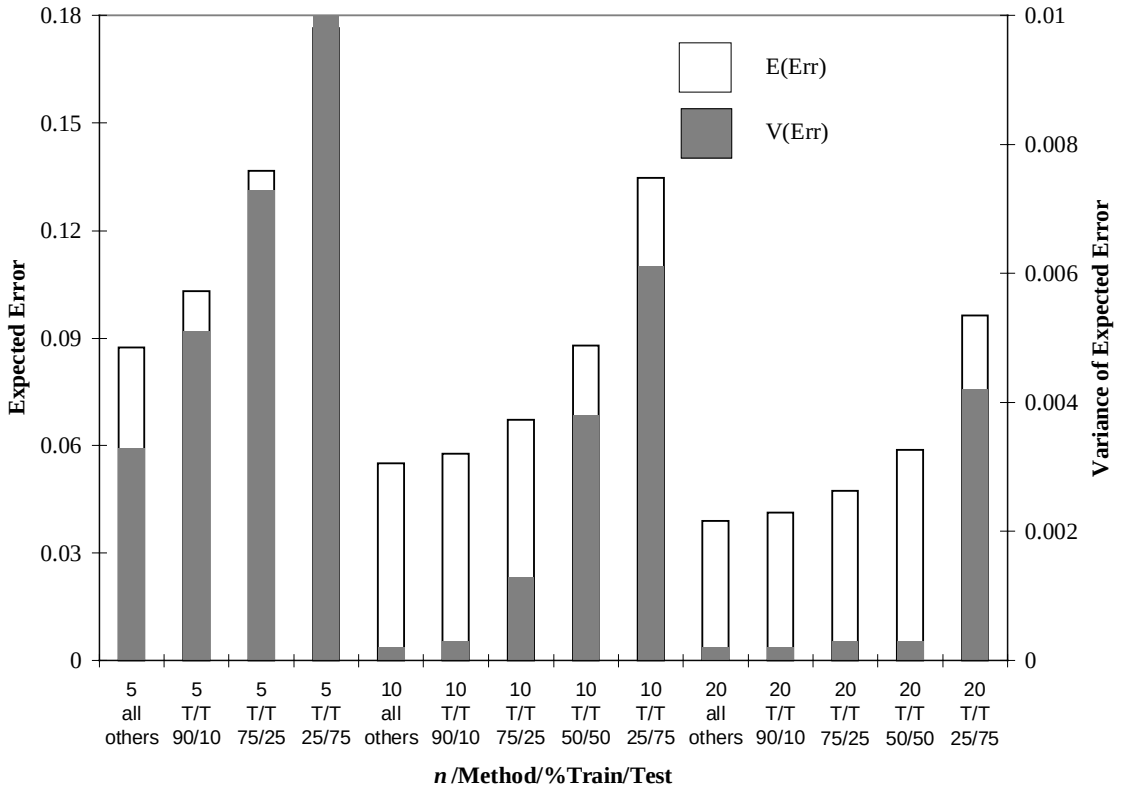


Figure 2. Expected Error and Its Variance of Application Networks.

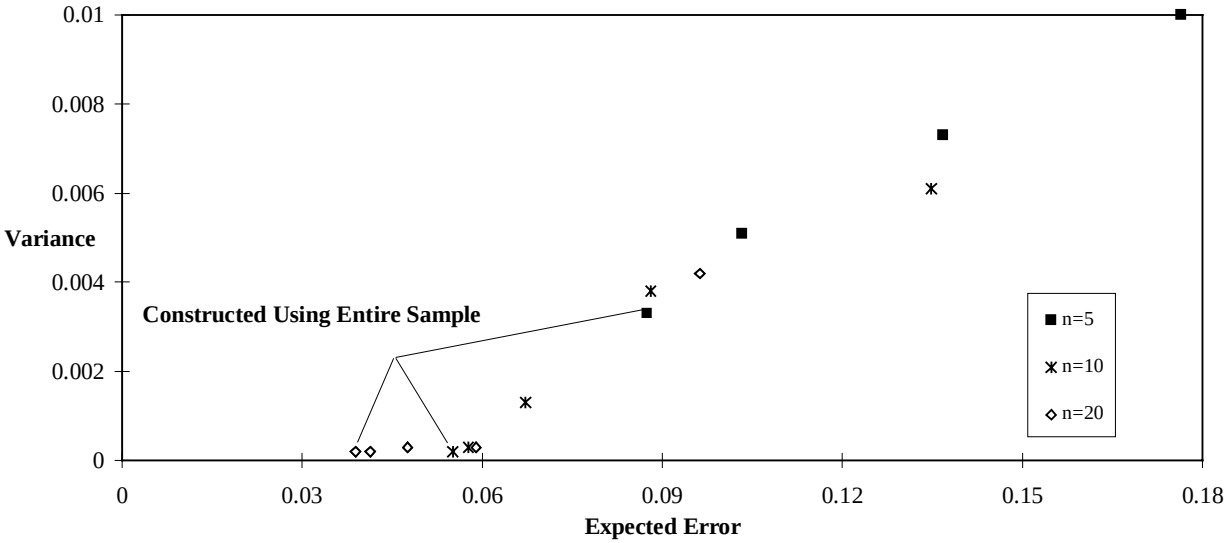


Figure 3. Variance vs. Expected Error of Application Networks.

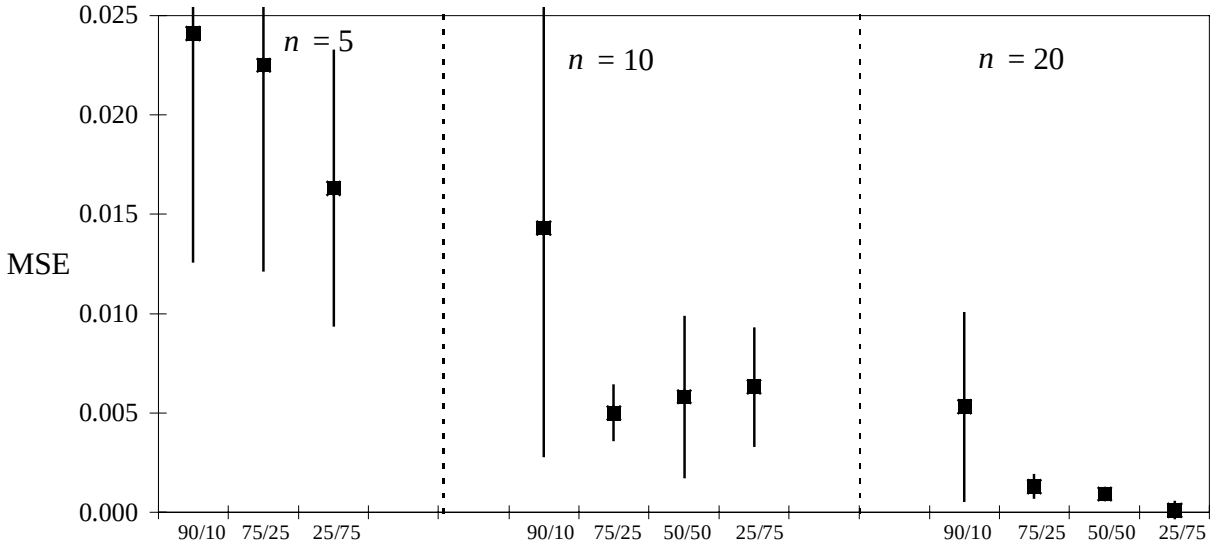


Figure 4. 95% C.I. for the MSE of $\hat{\text{Err}}_{\text{T/T.}}$

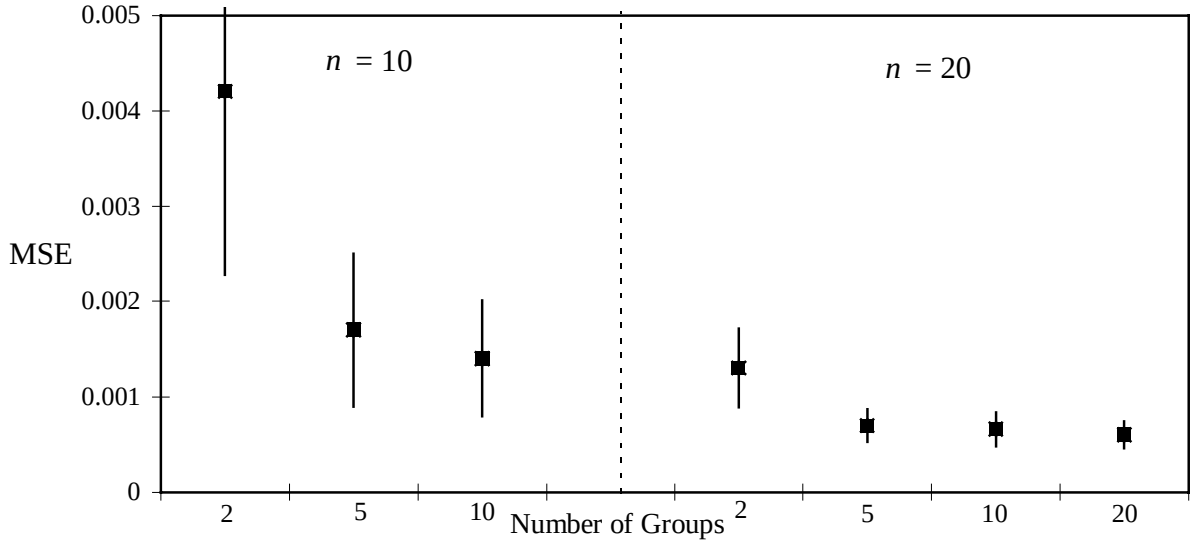
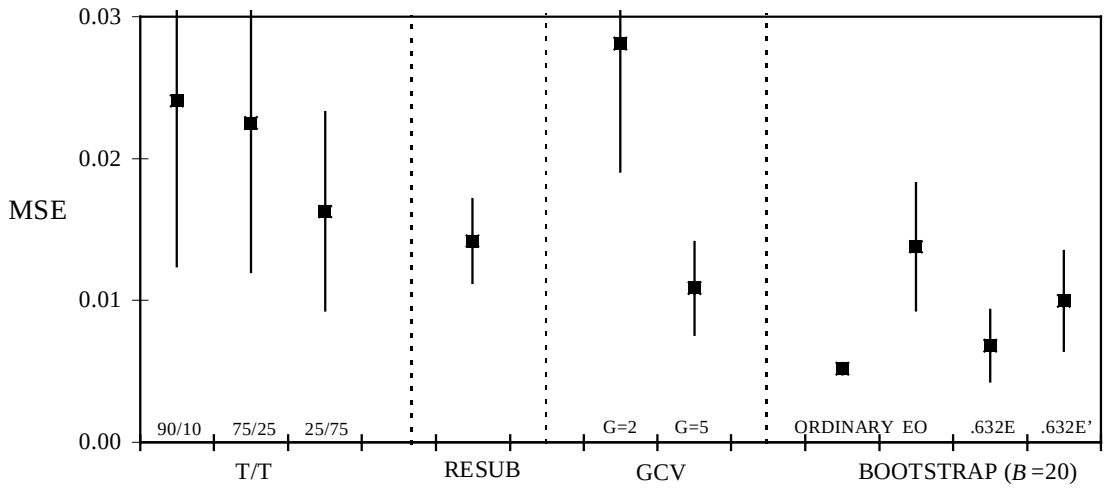
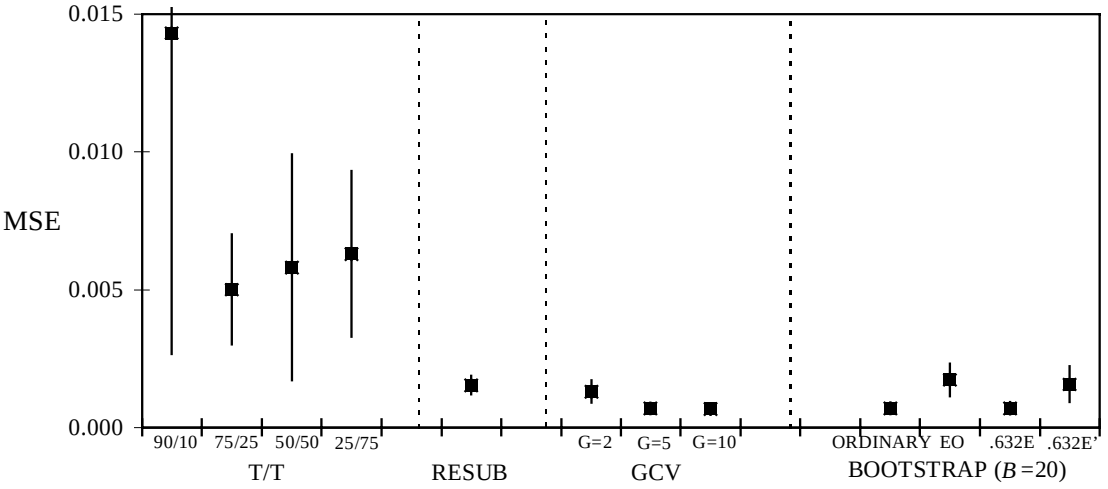


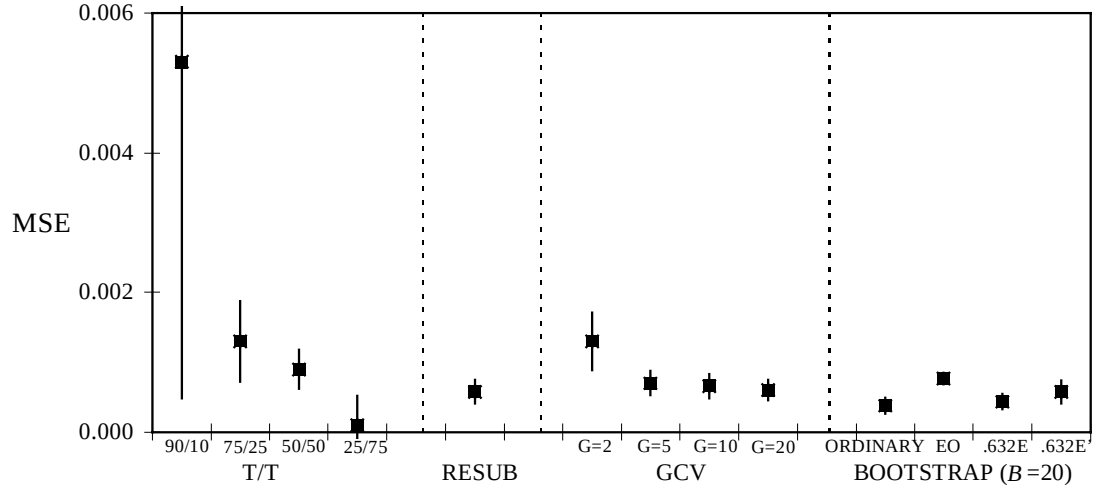
Figure 5. 95% Confidence Intervals for Different Number of Groups in GCV.



A. $n = 5$.



B. $n = 10$.



C. $n = 20$.

Figure 6. 95% Confidence Intervals for the Validation Methods.

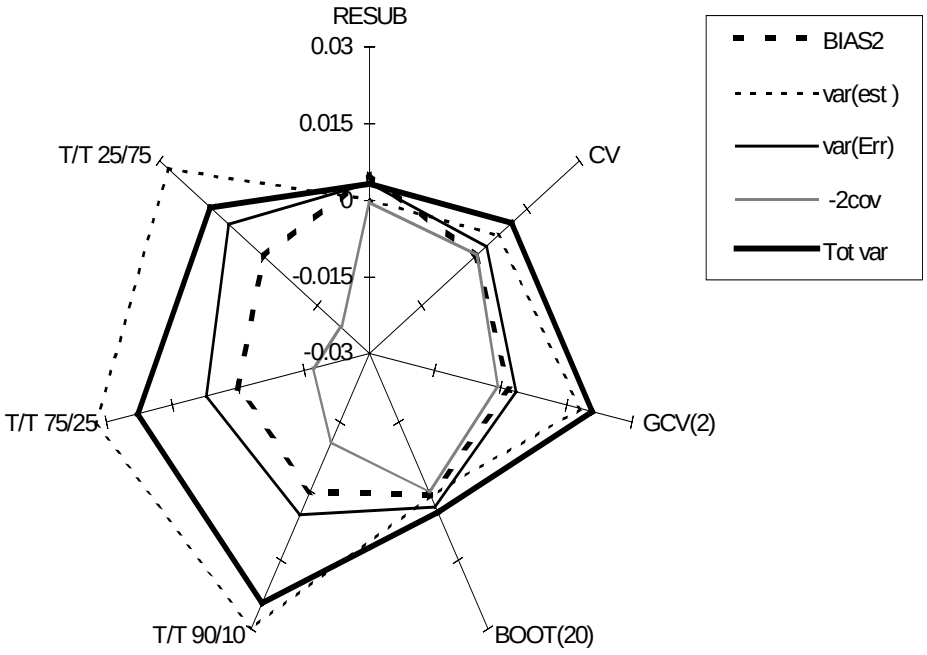


Figure 7. Components of MSE of $\hat{\text{Err}}$ for $n = 5$.

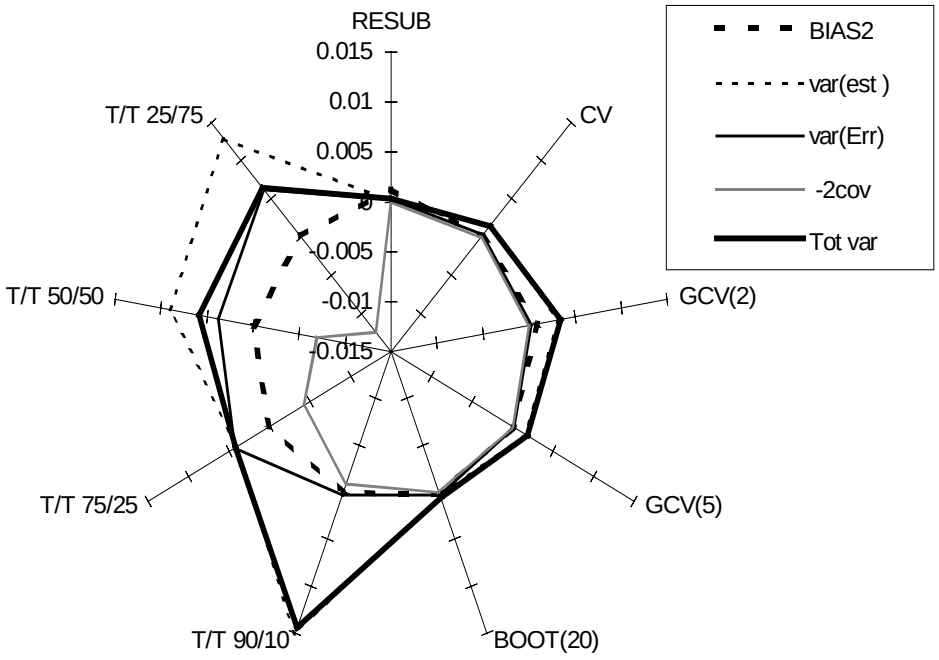


Figure 8. Components of MSE of $\hat{\text{Err}}$ for $n = 10$.

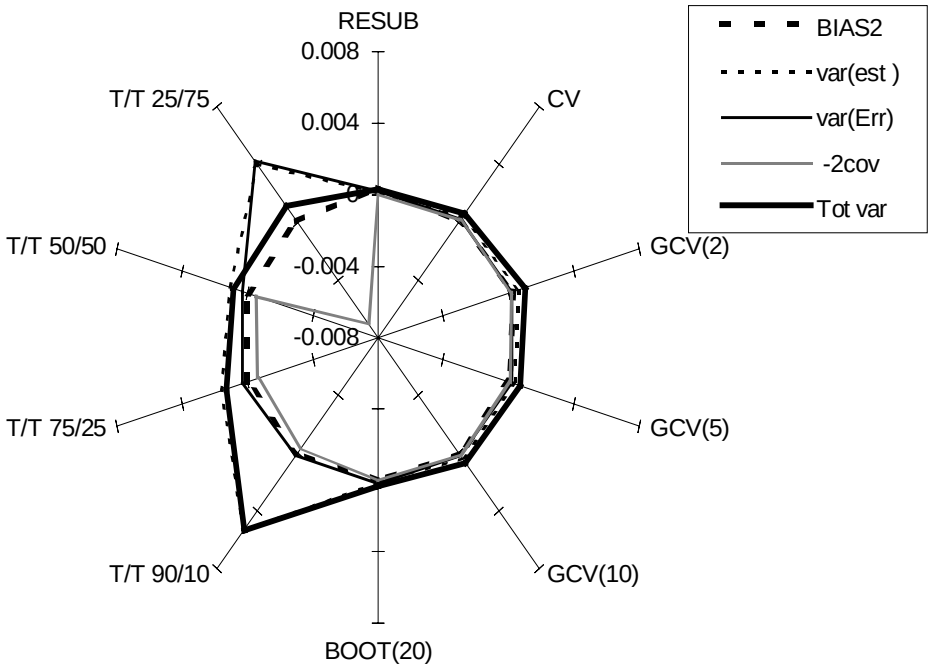


Figure 9. Components of MSE of $\hat{\text{Err}}$ for $n = 20$.

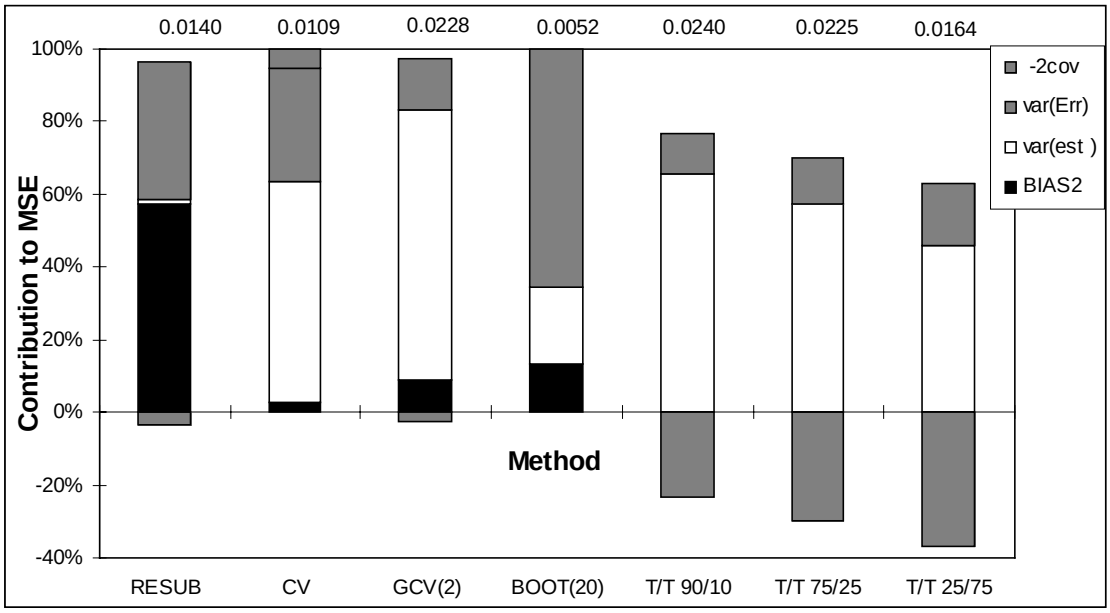


Figure 10. Relative Contribution of Bias and Variance Terms to MSE for $n = 5$.

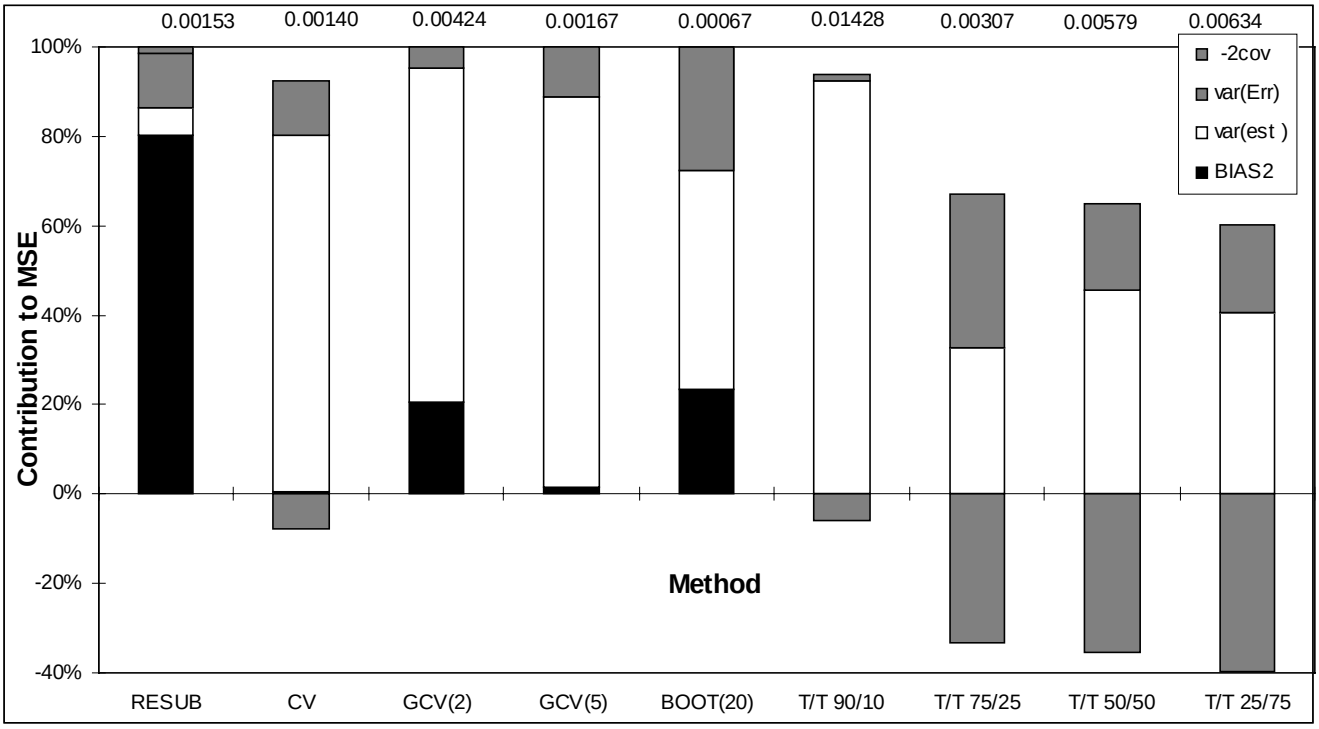


Figure 11. Relative Contribution of Bias and Variance Terms to MSE for $n = 10$.

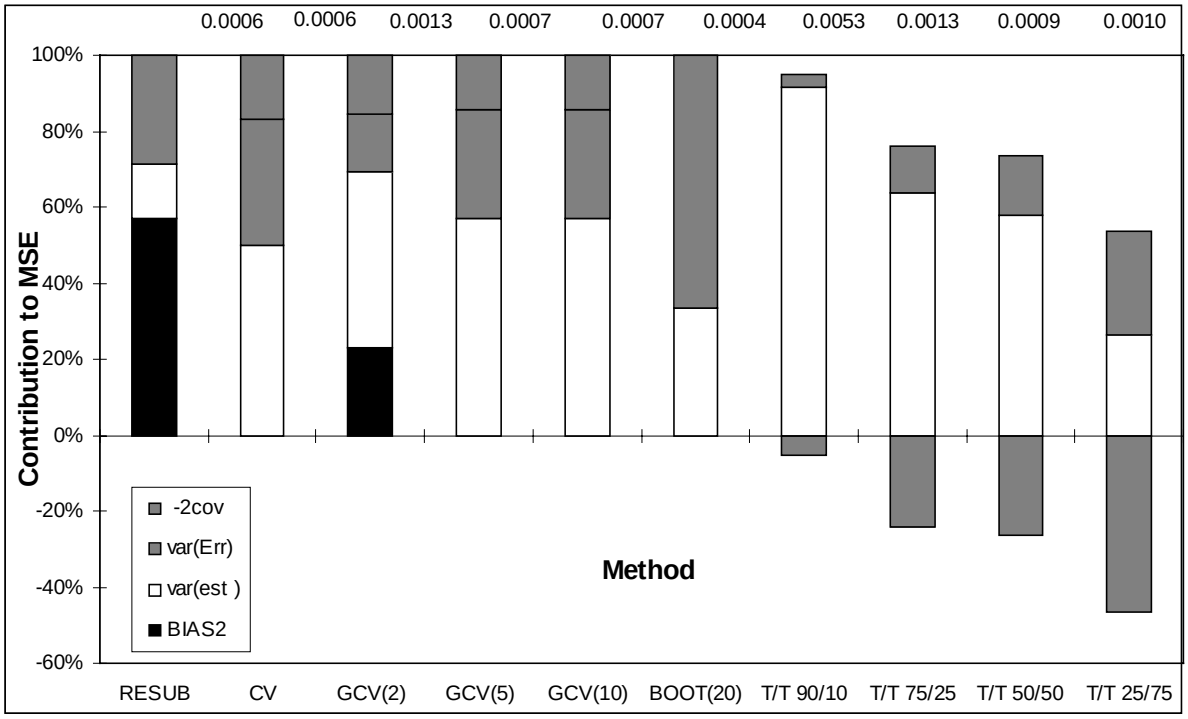


Figure 12. Relative Contribution of Bias and Variance Terms to MSE for $n = 20$.

Table 1. Variations of Train-And-Test.

Levels of Sample Size	**Levels of Train/Test (% train / % test)**
***n* = 5**	(90/10)* (75/25)* (25/75)*
***n* = 10**	(90/10) (75/25)[+] (50/50) (25/75)[+]
***n* = 20**	(90/10) (75/25) (50/50) (25/75)

* Actual % train/% test is (80/20) (60/40) and (40/60)

[+] actual % train/% test is (70/30) and (30/70)

Table 2. Variations of the Group Cross-Validation.

Levels of Sample Size	Levels of Groups
***n* = 5**	2 groups GCV (3 train & 2 test)
***n* = 10**	2 groups GCV (5 obs./grp) 5 groups GCV (2 obs./grp)
***n* = 20**	2 groups GCV (10 obs./grp) 5 groups GCV (4 obs./grp) 10 groups GCV (2 obs./grp)

Table 3. True Error of Application Networks Over 100 Samples on Population *F* of 5000.

Sample Size	Method	E(Err)	V(Err)
5	Resubstitution, CV, GCV, Bootstrap	0.0874	0.0033
	T/T (90/10)	0.1032	0.0051
	T/T (75/25)	0.1367	0.0073
	T/T (25/75)	0.1764	0.0100
10	Resubstitution, CV, GCV, Bootstrap	0.0551	0.0002
	T/T (90/10)	0.0577	0.0003
	T/T (75/25)	0.0672	0.0013
	T/T (50/50)	0.0880	0.0038
	T/T (25/75)	0.1348	0.0061
20	Resubstitution, CV, GCV, Bootstrap	0.0389	0.0002
	T/T (90/10)	0.0413	0.0002
	T/T (75/25)	0.0475	0.0003
	T/T (50/50)	0.0589	0.0003
	T/T (25/75)	0.0962	0.0042

Table 4. Error of Validation for Typical Methods.

Method	*n*	E(Err)	E($\hat{\text{Err}}$)	V($\hat{\text{Err}}$)	P($\hat{\text{Err}}$ <Err)
Resubstitution	5	0.0874	0.0157	0.0001	1.00
	10	0.0551	0.0200	0.0001	0.99
	20	0.0389	0.0197	0.0002	0.98
CV	5	0.0874	0.1047	0.0067	0.47
	10	0.0551	0.0577	0.0014	0.59
	20	0.0389	0.0415	0.0003	0.47
GCV - 2 Fold	5	0.0874	0.1336	0.0181	0.45
	10	0.0551	0.0605	0.0016	0.56
	20	0.0389	0.0426	0.0004	0.48
Train-and-Test (75%/25%)	5	0.1367	0.1423	0.0325	0.34
	10	0.0672	0.0640	0.0046	0.33
	20	0.0475	0.0499	0.0016	0.38
Bootstrap - *B* = 20	5	0.0874	0.0605	0.0011	0.64
	10	0.0551	0.0420	0.0004	0.79
	20	0.0389	0.0326	0.0001	0.59

Table 5. Comparison of Grouping Strategies for CV.

Sample Size (*n*)	**Groups**	**E(Err)**	**E($\hat{Err}_{cv}$)**	**V($\hat{Err}_{cv}$)**	**P($\hat{Err}_{cv}$ <Err)**
5	**2**	0.0874	0.1336	0.0181	0.45
	5	0.0874	0.1047	0.0067	0.47
10	**2**	0.0551	0.0848	0.0032	0.35
	5	0.0551	0.0605	0.0016	0.56
	10	0.0551	0.0577	0.0014	0.59
20	**2**	0.0389	0.0573	0.0006	0.32
	5	0.0389	0.0450	0.0003	0.46
	10	0.0389	0.0426	0.0004	0.48
	20	0.0389	0.0415	0.0003	0.47

Table 6. Ordinary Bootstrap Results.

	***n* = 5**		***n* = 10**		***n* = 20**	
	E(Err) = 0.0874		**E(Err) = 0.0551**		**E(Err) = 0.0389**	
B	**$E(\hat{Err}_{boot})$**	**$V(\hat{Err}_{boot})$**	**$E(\hat{Err}_{boot})$**	**$V(\hat{Err}_{boot})$**	**$E(\hat{Err}_{boot})$**	**$V(\hat{Err}_{boot})$**
1	0.0556	0.0049	0.0422	0.0009	0.0322	0.0003
2	0.0613	0.0027	0.0416	0.0006	0.0336	0.0002
5	0.0637	0.0018	0.0427	0.0005	0.0332	0.0002
10	0.0610	0.0013	0.0419	0.0004	0.0330	0.0001
20	0.0605	0.0011	0.0420	0.0004	0.0326	0.0001
50	0.0582	0.0009	0.0417	0.0004	0.0327	0.0001
60	0.0573	0.0008	0.0419	0.0004	0.0328	0.0001
100	0.0572	0.0008	0.0419	0.0003	0.0328	0.0001

Table 7. Comparisons of Bootstrap Alterations at $B = 20$.

Measure	*n*	Boot	E0	.632E	.632E′
E($\hat{\text{Err}}$**)**	5	0.0605	0.1289	0.0873	0.0903
	10	0.0420	0.0714	0.0525	0.0610
	20	0.0326	0.0492	0.0383	0.0434
V($\hat{\text{Err}}$**)**	5	0.0011	0.0083	0.0033	0.0072
	10	0.0004	0.0015	0.0007	0.0014
	20	0.0001	0.0004	0.0002	0.0003
P($\hat{\text{Err}}$ **<Err)**	5	0.64	0.30	0.44	0.58
	10	0.79	0.41	0.60	0.54
	20	0.59	0.42	0.51	0.46

Table 8. Summary of Validation Methods.

Method	Computational Effort	Application Net Performance	Bias of Error Estimate	Variability of Error Estimate
Resubstitution	Minimal (1)	Maximum	Downwards, sometimes severely depending on amount of training	Low
Cross-validation	Large (n+1)	Maximum	Unbiased	Low
Group Cross-validation	Small to Medium (3 to n)	Maximum	Unbiased	Medium for G = 2, Low $G > 2$[*]
Bootstrap	Large (10+)	Maximum	Mildly Downwards	Lowest[+]
.632E′ Bootstrap	Small (3)	Maximum	Unbiased	Low
Train-and-Test	Minimal (1)	Medium to Minimal[@]	Mildly Upwards	Highly variable (especially for 90/10)

[*]Modest improvement with increase in $G > 2$

[+]Reduced variability until $B = 20$

[@]Best partition is 75/25

Biographical Sketches

Janet M. Twomey is Assistant Professor of Industrial and Manufacturing Engineering at Wichita State University. She has degrees from Duquesne University and the University of Pittsburgh, from which she received the Ph.D. degree in Industrial Engineering in 1995. Her research in engineering design, simulation and manufacturing processes has been funded by the Boeing Corporation and the National Science Foundation. Her articles have appeared in *IIE Transactions, Engineering Design and Automation* and *Journal of Mathematical and Computer Modelling*. She is a member of IIE and INFORMS.

Alice E. Smith is Associate Professor of Industrial Engineering and Board of Visitors Faculty Fellow at the University of Pittsburgh. She joined the University of Pittsburgh in 1991 after ten years of industrial experience with Southwestern Bell Corporation, and has degrees in engineering and business from Rice University, Saint Louis University and University of Missouri - Rolla. Her research in analysis, modeling and optimization of manufacturing processes and engineering design has been funded by the National Institute of Standards, Lockheed Martin, ABB Daimler-Benz Transportation, U.S. Steel, the Ben Franklin Technology Center of Western Pennsylvania and the National Science Foundation, from which she was awarded a CAREER grant in 1995. Dr. Smith is an associate editor of *INFORMS Journal on Computing, IEEE Transactions on Evolutionary Computation, International Journal of Smart Engineering System Design* and *Engineering Design and Automation* and she is on the Design and Manufacturing Editorial Board of *IIE Transactions*. Her articles have appeared in many journals including *IIE Transactions, IEEE Transactions on Reliability, INFORMS Journal on Computing, International Journal of Production Research, The Engineering Economist, IEEE Transactions on Evolutionary*

Computation and *Computers and Operations Research*. She is a senior member of IIE, IEEE and SWE, a member of INFORMS and ASEE, and a Registered Professional Engineer in Industrial Engineering in the Commonwealth of Pennsylvania.