CONF-970465--7

ACCURACY ESTIMATION FOR SUPERVISED LEARNING ALGORITHMS*

C. W. GLOVER
E. M. OBLOW
N. S. V. RAO

Intelligent Systems Section
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN.

Paper Submitted To:

SPIE AeroSense '97 Conference

April 21-26, 1997
Orlando, Florida

***Research sponsored by the Engineering Research Program of the Office of Basic Energy Sciences of the Department of Energy, under Contract No. DE-AC05-96OR22464 with Lockheed Martin Energy Research Corp.**

DISCLAIMER

CONF-970465--7

Accuracy estimation for supervised learning algorithms

C. W. Glover, E. M. Oblow, and N. S. V. Rao

Oak Ridge National Laboratory, P.O. Box 2008, MS-6355, Oak Ridge, TN 37831-6355

ABSTRACT

This paper illustrates and discusses the relative merits of three methods — k-fold Cross Validation, Error Bounds, and Incremental Halting Test — to estimate the accuracy of a supervised learning algorithm. For each of the three methods we point out the problem they address, some of the important assumptions that they are based on, and illustrate them through an example. Finally, we discuss the relative advantages and disadvantages of each method.

Keywords: supervised learning, holdout methods, cross validation, PAC learning, neural networks, statistical accuracy estimation

1. Introduction

This paper illustrates and discusses the relative merits of three methods — k-fold Cross Validation, Error Bounds, and Incremental Halting Test — estimating the accuracy of a supervised learning algorithm. What we seek is an estimate of an algorithm's *expected* accuracy on the population of all possible samples which is a measure of its generalization accuracy. For most applications this is impossible since we can never sample the entire population and we do not know the population's probability distribution. Therefore, we must back away from this goal and define and solve a different problem that is related. For each of the three methods we will point out the problem they address, some of the important assumptions that they are based on, and illustrate them through an example.

One often sees the performance of an algorithm reported as the accuracy (or its inverse, the error) it achieves on a test set of sample data. This performance measure by itself is meaningless. It represents a single instantiation of the random variable. The sample data set (training plus test sets) represents a single, finite-sized set of examples drawn from the population whose probability distribution is unknown, *a priori*. Another sample set will produce a different instantiation of the random variable. What is needed is a characterization of the probability distribution for a random variable representing the error, and this is the purpose of the methods in this paper.

2. SUPERVISED LEARNING — A PROBLEM STATEMENT

The goal of a supervised learning algorithm is to accurately approximate a target function based on a finite sample of empirical data. A target function, y, produces a value (regression) or label (classification) for each point, x, in the population's domain, D. A supervised learning algorithm induces a parameterized function, $f(x, \alpha)$ that also produces a value for each point. The random variable, Q, represents the cost of an incorrect prediction, e.g., number of incorrect values, distributed over the population according to the unknown probability distribution $F(x, y)$; its *expected* value is

$$R(\alpha) = \int Q(y, f(x,\alpha))\, dF(x,y) \equiv \int Q(z,\alpha)\, dF(z). \qquad (1)$$

The goal of a supervised learning algorithm is to select a function, $f(x, \alpha_0)$, from a family of functions $f(x, \alpha)$ characterized by the parameter set, $\alpha \in \Lambda$, that minimizes $R(\alpha)$. The labeled empirical data, $D_\ell = \{(x_1,$

$y_1), \bullet\bullet\bullet, (x_\ell, y_\ell)\} \equiv \{z_1, \bullet\bullet\bullet, z_\ell\}$, is drawn randomly from D according to $F(x, y)$. The resulting sample data set has its own empirical frequency distribution, $F_\ell(x, y)$ that we hope is similar to F.

Now the algorithm's problem of minimizing $R(\alpha)$ is replaced by the problem of selecting a function $f(x, \alpha_\ell)$ that minimizes the *mean* empirical functional,

$$R_\ell(\alpha) = \frac{1}{\ell}\sum_{i=1}^{\ell} Q(z_i, \alpha), \tag{2}$$

where the ℓ -subscript is carried to mind us that the estimate is dependent on the sample set. In the language of statistics, $R(\alpha)$ is a population parameter of the random variable $Q(z, \alpha)$ and $R_\ell(\alpha)$ is its associated statistic. Thus, the performance of the algorithm is addressed by investigating the difference between the functionals $R(\alpha)$ and $R_\ell(\alpha)$.

Each of three methods illustrated in this paper represent various attempts to estimate a form of the following equation,

$$Pr\left[\left|R(\alpha) - R_\ell(\alpha)\right| \le \varepsilon\right] \ge 1 - \delta, \tag{3}$$

for *a priori* specified accuracy, ε, and confidence, δ, parameters. This form of performance measure is also a random variable and is shown to be insensitive to the population's distribution F for a fairly general set of function classes[1,2], $\{f(x, \alpha): \alpha \in \Lambda\}$.

The following performance estimation methods will be illustrated:

1) The Holdout methods — Bootstrap and Cross Validation — are unbiased estimators to first-order. Each method essentially uses the sample data D_ℓ as though it were the parent population to randomly generate new sample data sets that will be used to train and test an algorithm and gather information about the statistics of the performance measure given by Equation (3).

2) The Empirical Risk Minimization method[3] produces upper and lower worst-case bounds, $|R(\alpha)| \le R_\ell(\alpha) \pm \Phi(\ell, \varepsilon, G_\Lambda)$ with probability 1 - δ, by bounding the system with a uniform rate of convergence for Equation (3) and a finite growth function, G_Λ (see Ref. 3 for details). The problem is proving that the growth function is bounded for supervised learning algorithms.

3) The Incremental Halting Test[4] exploits the combinatorial analysis of waiting times[5] to determine how many samples must be drawn to insure the conditions in Equation (3) are met for the particular distribution $F(z)$.

It should be noted that (1) and (3) are distribution specific methods and (2) is a distribution independent, worst-case estimate. We will illustrate each of the methods in the following sections on example problems to illuminate the technique and their strengths and weaknesses.

3. k-FOLD CROSS VALIDATION

The holdout method is an unbiased first-order technique to validate a statistic P_ℓ of a population parameter P from a finite sample set. The holdout method consists of ℓ samples drawn from a set D with a an unknown probability distribution to obtain a sample set D_ℓ. The sample set D_ℓ is divided into a training set D_{trn} and a test set D_{tst}. The bootstrap and k-fold cross validation techniques are the two most widely used variations of the holdout method.

The bootstrap method randomly draws members for the training and test sets by sampling from D_ℓ with replacement. The algorithm is trained from D_{trn} and tested on D_{tst}. This process is repeated with different

training and test sets to build an estimate P_ℓ and its variance s_ℓ^2. The confidence interval is determined from s_ℓ^2 accordingly.

The k-fold cross validation technique partitions the data in k mutually exclusive partitions. The first k - 1 partitions are used for D_{trn} and the kth for D_{tst} where the algorithm's accuracy is computed. Next, partitions $\{1, 2, \ldots, k-2, k\}$ are used for D_{trn} and the k - 1 partition is used for D_{tst}; then partitions $\{1, 2, \ldots, k-3, k-1, k\}$ are used for D_{trn} and the k - 2 partition is used for D_{tst}; and so on until k accuracy measures have been obtained along with their variance.

Both of these methods are unbiased in the limit that training set consists of ℓ - 1 samples and the test set is the single remaining example, and where the accuracy estimate and variance are computed from all ℓ - 1 permutations of these sets. This is the leave-one-out holdout method. The unbiased guarantee disappears if these conditions are not met. In practice, all ℓ - 1 permutation sets are seldom used because of the labor involved in evaluating algorithms for a large number of training and test sets. Usually, most researchers divide the data into one training and one test set with an equal number of samples in each set, or at best a small number of training and test sets. Thus, one is left with a potentially highly biased accuracy measure with a tight confidence bound that may not be remotely close to the population's accuracy. The researcher may therefore be led into a false sense of security about the performance.

Kohavi[6] compared the bias and variance tradeoff between the bootstrap and k-fold cross validation techniques as a function of the number of training/test sets. He found that bootstrap method has a smaller variance than k-fold cross validation, but the bias is much larger. For this reason, Kohavi concluded that k-fold cross validation may provide a better operational estimate of a classifiers' accuracy than bootstrap. In addition, he showed for the k-fold cross validation technique that ten or more partitions are sufficient for the sample accuracy with a 95% confidence interval to enclose the population's accuracy. For this reason, we employed the k-fold cross validation technique with a $k = 10$ partitions to estimate the following samples problem's accuracy and confidence intervals.

Example: k-Fold Cross Validation for an Artificial Neural Network (ANN)

We want to estimate the ANN performance on all future samples presented to it after training. This is clearly impossible unless the underlying probability distribution that the training samples were drawn from is exactly equal to the probability distribution from which the future examples are drawn. The final application's probability distribution must be "similar" to the training probability distribution for a measure of accuracy to have any meaning. This statement is true for any regression and/or pattern recognition method.

Even if this caveat is true and the probability distributions are equal, we can only provide an estimate P_ℓ of a population P and bound our estimate with a confidence interval because we used a finite training set; P_ℓ is then a random variable. One method to estimate the confidence interval is to use the de Moivré-Laplace Theorem and the assumptions it entails. It states that the proportion of *successes* drawn from a Bernoulli population is P. Then the confidence limits for P are given by $P_\ell \pm z\, s_\ell$, where the confidence coefficients are defined by,

$$Pr\left[-z < \frac{P_\ell - P}{s_\ell} < z\right] \approx \int_{-z}^{z} N(x)dx \;; \tag{4}$$

this assumes that the distribution approaches a normal distribution, $N(x)$. Only in the limit of the leave-one-out method does the confidence approach one. Any estimate of a statistic describing the accuracy of an ANN is a random number and by itself is meaningless unless it is accompanied by a corresponding confidence interval.

The proportion of successful predictions an ANN produces on the population, P, is the parameter we would like to estimate from a finite sample set as our measure of the ANN's accuracy. The proportion of successful predictions P_ℓ the ANN produces on a test set D_{tst} after its weight parameters, α, have been fixed to α^* by training is given by

$$P_t = \frac{1}{\ell} \sum_{\langle x_i, y_i \rangle \in D_{tst}}^{\ell} Q\left[y_i, f\left(x_i, \alpha^* \right) \right] ,$$

where Q is an indicator function that produces a value of one when a sample x_i is drawn from a test set D_{tst} and the ANN yields the correct output value y_i, and otherwise it is zero. Simply put, this is the average accuracy of the ANN on the test set D_{tst}.

We applied this method to a geophysical parameter estimation problem[7]. We used an ANN to estimate a parameter of an oilfield's reservoir given seven seismic parameters as input data. The data was randomly partitioned into ten subsets. These partitions were combined according to the k-fold-cross-validation procedure into ten training and testing pairs, such that

Training Set #1 = Partitions {1, 2, ... , 9} and
Test Set #1 = Partition {10};

Training Set #2 = Partitions {1, 2, ..., 8, 10} and
Test Set #2 = Partition {9};

•
•
•

Training Set #10 = Partitions {2, 3, ... , 9} and
Test Set #10 = Partition {1};

These sets were used to obtain all accuracy estimates and their associated confidence intervals.

The first question we investigated is whether one or two hidden layer ANN architectures produce *statistically significant* different accuracies. Initially, a single hidden layer was used with 10 nodes. All weights were initialized with values between ±0.1 and Training Set #1 shuffled for presentation to the ANN. The ANN's error on Test Set #1 was monitored along with the training set's error as a function of the number of epochs (1 epoch = 1 full presentation of the training set's examples). As the number of epochs increases both the training set error and test error decrease, up to a point where the training set error continues to decrease and the test set error starts to increase. Just prior to this point we extract the error on Test Set #1 and use this value in our accuracy estimate. This process is repeated for the remaining nine sets. The ten accuracy estimates are then averaged, variances extracted, and their 95% confidence intervals are computing by $P_t \pm 3\, s_t$.

Figure 1 displays the mean accuracy and 95% confidence intervals for ANNs with one and two hidden layers and with a differing number of nodes in the hidden layers. The lower curve represents the mean accuracy for an ANNs with 10, 20, 30, and 40 nodes in a single hidden layer. The upper curve represents the mean accuracy for an ANNs with 10 (5x5), 20 (15x5), 25(15x10), 30 (20x10), and 40 (25x15) nodes in a two hidden layers. The two layer ANN architecture provides a higher accuracy than a single hidden layer ANN.

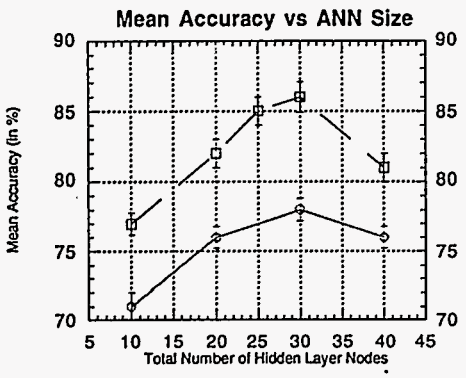

Figure 1: The mean accuracy and the 95% confidence interval for ANNs with one and two hidden layers.

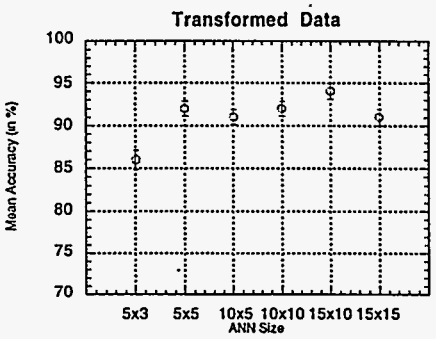

Figure 2: The mean accuracy and the 95% confidence interval for ANNs with two hidden layers, where a Whitening transform was applied to the input data.

The second question we addressed was whether a two hidden layer ANN produced more *statistically significant* accurate parameter estimates when a Whitening Transform[8] was applied to the input data. Figure 2 displays the mean accuracy and confidence intervals for ANNs two hidden layers and where the *input data were transformed into the eigenspace.* These results show that the transformed data consistently yields a higher accuracy than the untransformed data and that the ANN size is smaller. These conclusions could not be justified by using a single training and test set.

4. ERROR BOUNDS

We will illustrate the essential properties of the Error Bounds method with a 1-dimensional example.

The example is the game "guess a real number y between 0 and 1." An algorithm is required to guess a target number y to within an agreed upon accuracy, ε, from a set of examples $D_\ell = \{(x_1, I(|x_1 - y|, 2^{-n})), \bullet\bullet\bullet, (x_\ell, I(|x_\ell - y|, 2^{-n}))\}$, where $I(|x_1 - y|, 2^{-n})$ is 1 if $|x_1 - y| \leq 2^{-n}$ and is 0 otherwise.

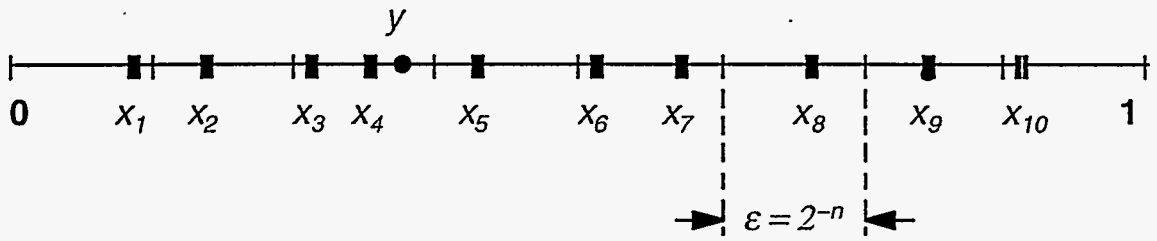

Figure 3: The line segment [0, 1] for the "guess the number" game is divided into 2^3 intervals. The target number y and 10 batch samples drawn to train a supervised learning algorithm are shown.

In Figure 3, the segment [0,1] is divided into bins of size $\varepsilon = 2^{-n} = 2^{-3}$, where n is an *a priori* accuracy parameter and $\ell = 10$ sample points have been drawn. For this example, $I(|x_3 - y|, 2^{-3}) = I(|x_4 - y|, 2^{-3}) = 1$, and all other $I(\bullet, 2^{-3}) = 0$. It indicates whether one or more of the samples points occupies a the bin with the number y. The configuration space indicates the number of samples points in each bin; for this example the configuration is [1,1,2,1,2,1,1,1]. The algorithm we will use to guess the number is simple,

$$f(x,\alpha)=\begin{cases}\sum_i^{\ell} x_i\, I\big(|x_i-y|,\alpha\big)\Big/\sum_i^{\ell} I\big(|x_i-y|,\alpha\big) & \text{if } \sum_i^{\ell} I\big(|x_i-y|,\alpha\big)\neq 0\\ \frac{1}{\ell}\sum_i^{\ell} x_i & \text{if otherwise}\end{cases};$$

it will produce as a guess the average value of all sample points in a bin whose indicator value is not zero, otherwise it will guess the average value of all sample points.

We now wish to estimate Equation (3) for a all possible ℓ sample points and α values. To do this let us define a success as the event satisfying the condition

$$\sup_{\alpha}\big|R(\alpha)-R_{\ell}(\alpha)\big|\le\varepsilon.$$

Clearly, we will only meet this condition for any y if we have at least one sample per bin in our sample set. Thus, the probability of success is the probability of drawing at least one sample per bin out ℓ samples.

As a baseline case, we can compute the probability of success for the case where a point in each has an equal probability of being chosen. Thus a uniform distribution over the bins gives a probability,

$$Pr[\text{ Success }]=1-\left(1-\frac{1}{2^n}\right)^{\ell}, \tag{6}$$

of having at least one point chosen in each bin. Figure 4 displays this equation as a function of the number of samples ℓ for n = 3. About 15 samples must be drawn before we can be 80% assured that we have drawn a sample from each of the 8 bins

We will now compute the same quantity by using an intuitive derivation of the Error Bounding method. The form of a Standardized Statistical Variable is $(S-\mu)/\sigma$, where the total difference between our algorithm's output and the target value is

$$S_{\ell}=\big|y-f(x_1,\alpha)\big|+\cdots+\big|y-f(x_{\ell},\alpha)\big|,$$

the expected difference is $\mu=0$, and the variance,

$$s_{\ell}=\sigma_1^2+\cdots+\sigma_{\ell}^2=\ell\sigma^2,$$

is the same over all possible ℓ-sample sets. By using the Central Limit Theorem we can write Equation (3) in the following form:

$$Pr\big[S_{\ell}<\sqrt{\ell}\,s_{\ell},\varepsilon\big]\ge 1-\delta\ <\ 1-\tfrac{1}{\sqrt{2\pi\ell}\varepsilon}e^{-\frac{1}{2}\varepsilon^2\ell},$$

where we have used a standard approximation[5] to bound the tail δ of the distribution. A tighter bound can be achieved by using the Hoeffding's inequality[2,3]; it produces

$$Pr\big[S_{\ell}<\sqrt{\ell}\,s_{\ell},\varepsilon\big]\ge 1-\delta\ <\ 1-2e^{-\frac{1}{4}\varepsilon^2\ell}.$$

This quantity is plotted in Figure 4. This bound indicates that on the order of 1200 samples need to be drawn before one can produce a confidence of 80% in the probability of success.

This result is the best estimate that can be made without invoking assumptions about the probability distribution for drawing samples. This estimate is valid for the worst-case distribution (i.e., the theoretically hardest distribution to learn from), and as can be seen it is clearly valid for the uniform

distribution. It overestimates by *three* orders of magnitude the number samples necessary to be assured of at least an 80% confidence value!

5. INCREMENTAL HALTING TEST

Despite the general nature of the Error Bound's result, in many practical problems we expect that much fewer examples are required to meet accuracy goals because the sample probability distributions for most problems are not close to the worst-case distribution. The usefulness of the Error Bounds methods for these cases is limited as was clearly seen for the uniform case. Therefore, for specific problems it is useful to have a general method for finding problem-specific bounds. Even more important is being able to estimate these bounds incrementally, while the algorithm is learning. This will be a reduction in labor as compared with Holdout Methods. The Incremental Halting Test forms the basis for a problem-specific incremental approach.

The essence of the Incremental Halting Test can be simply illustrated by considering a Bernoulli trails framework with our 1-dimensional example from the previous section. We define a *success* as drawing a sample point from a new, unsampled bin; our algorithm will learn something new and reduce its error in estimating the unknown number. A *failure* is drawing a sample from a previously sampled bin; it fails to reduce the algorithm's error. The basis for the Incremental Halting Test is the fact that after the $(j-1)^{st}$ success the probability $h(p_j)$ of having a string of m failures from any distribution is

$$h(p_j) \equiv (1-p_j)^m . \tag{7}$$

If the remaining success probability p_j were actually ε then this *m-failure* probability would be $h(\varepsilon)$. Thus a string of m failures indicates that our algorithm may be accurate to within ε, and one or more *successes* means that our algorithm is still capable of making an error $\geq \varepsilon$. In the case where no new successes were recorded in m additional trails, the selection process can be halted with the final remaining success probability $\leq \varepsilon$ to within some confidence δ that is determined later. Clearly, $h(\varepsilon)$ represents the probability of halting the algorithm's learning where the remaining success probability is ε or less. Equation (3) can be bounded in this formulation by,

$$Pr\left[\left|R(\alpha) - R_t(\alpha)\right| \leq \varepsilon\right] \;\geq\; 1-\delta = 1-(1-\varepsilon)^m . \tag{8}$$

This allows us to find the number of additional samples needed to ensure the (ε, δ) conditions in Equation (8). This number is

$$m = \frac{ln\left(\frac{1}{\delta}\right)}{ln\left(\frac{1}{1-\varepsilon}\right)} \approx \frac{1}{\varepsilon}\, ln\left(\frac{1}{\delta}\right) \tag{9}$$

for small ε. Using this test, if no additional successes are found in *m future* samples, then the algorithm can be halted with assurance that both ε and δ criteria are satisfied.

For a uniform distribution the Incremental Halting Test produces the same result as given in Equation (6) and shown in Figure 4.

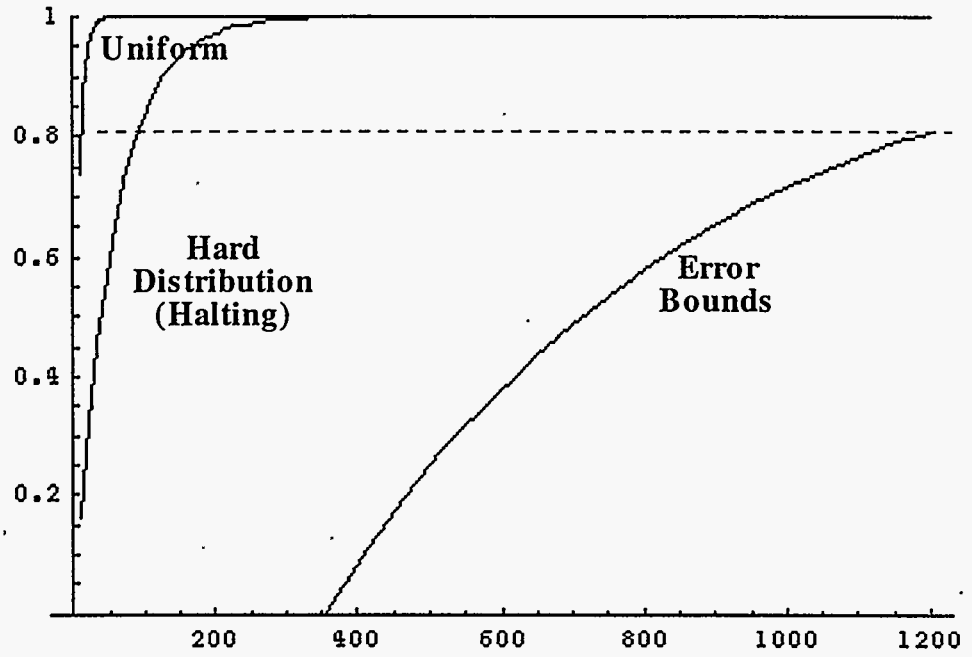

Figure 4: The plot of the number of samples ℓ vs. 1-δ. The curves represent the growth in the number of samples needed to achieve a confidence bound, 1-δ, for an $\varepsilon = 2^{-3}$ in the one dimensional example of "guess the number".

To demonstrate that the Incremental Halting Test is sensitive to the population's probability distribution, we define a "hard" probability distribution that is harder to learn than the uniform distribution for bin b_l as

$$p(b_i) = \begin{cases} \dfrac{1-\varepsilon}{2^n - 1} & \text{if} \quad i \neq b_y; \\ \varepsilon & \text{if} \quad i = b_y; \end{cases}.$$

For this distribution most of the probability is spread uniformly across all bins except for the bin contain the target value, y; it has a smaller probability, ε. Thus most of the samples will come from bins other than the one containing our target; this is why this distribution is hard to learn. Figure 4 shows that about 90 samples are needed to reach the 0.8 confidence value.

The Incremental Halting Test is designed to be used will an algorithm is learning. For example, one selects a target (ε, δ) pair and computes the string of samples m necessary to meet the conditions from Equation (9). Initially, the algorithm is a batch of m samples drawn with replacement from the set D_t, and it most likely produces one or more successes since the algorithm's parameters are randomly chosen. Then the algorithm uses these m samples to adjust its parameters α to reduce its training error. Another batch of m samples is given to the algorithm, if it produces no successes then it has learned the target to within (ε, δ). Otherwise, the algorithm will use these samples to again adjust its parameters. This process repeats itself until the algorithm learns the target values to within (ε, δ). If the algorithm has not halted by a user specified number of batches is reached then the target function may not be learnable to within (ε, δ) by algorithm.

6. SUMMARY

We have looked at three methods — k-fold Cross Validation, Error Bounds, and Incremental Halting Test — to estimate the accuracy of a supervised learning algorithm.

The k-fold Cross Validation provides the least bias holdout method to estimate the accuracy statistic, P_t, corresponding to the accuracy population parameter, P. This conclusion is based on Kohavi's work[6] investigating the bias-variance tradeoff between the bootstrap and k-fold Cross Validation on a finite number training and test sets. This is important since the unbiased guarantee is not valid when the number of sets is less than the leave-one-out holdout method.

We showed how the accuracy statistic of an ANN can be estimated using the k-fold Cross Validation technique. This estimate relied on an assumption about the statistic's distribution about the population parameter. In our example, we assumed a Bernoulli distribution and invoked the de Moivré-Laplace Theorem to bound the population's accuracy parameter with the corresponding sample accuracy statistic and variance. This process for the ANN is quite time consuming. and labor intensive.

Next, we considered a simple one dimensional example. In this example, our supervised learning algorithm was to learn a target number to within ε from a finite sample set. We showed that the Error Bounds method gave an upper bound on the algorithm's accuracy. This accuracy estimate makes no assumptions about the population's probability distribution. These bounds are good for the theoretically worst-case distribution. However, for most practical applications Error Bounding method produces bounds that are several orders of magnitude too large from what is found experimentally. It is not sensitive to the specific problem's probability distribution.

Finally, the Incremental Halting method was applied to the same one dimensional example as the Error Bounding method. This technique is a general method for finding problem-specific bounds. This was demonstrated in the one dimensional example by finding different bounds for two different probability distributions imposed on the population's sample space. Of great importance is its ability to estimate these bounds incrementally, while an algorithm is learning. This will be a reduction in labor as compared with k-fold Cross Validation method. The Incremental Halting Test forms the basis for a problem-specific incremental approach to estimating accuracy bounds that may be tighter the Error Bounds methods for specific problems and less labor intensive the k-fold Cross Validation method.

ACKNOWLEDGMENTS

Research sponsored by the Engineering Research Program of the Office of Basic Energy Sciences, of the U.S. Department of Energy, under Contract No.\ DE-AC05-96OR22464 with Lockheed Martin Energy Research Corp.

REFERENCES

1. W. Feller, *An Introduction to Probability Theory and Its Applications Vol. II*, John Wiley & Sons, New York, 1971.
2. A. W. van der Vaart and J. A. Wellner, *Weak Convergence and Empirical Processes with Applications to Statistics*", Springer, New York, 1996.
3. V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, 1995.
4. E. M. Oblow, "Implementing Valiant's Learnability Theory Using Random Sets", Machine Learning, Vol. 8, pp. 45-73, 1992.
5. W. Feller, *An Introduction to Probability Theory and Its Applications Vol. I*, John Wiley & Sons, New York, 1971.
6. R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection", *14th International Joint Conference on Artificial Intelligence*, p. 1137, Montreal, CA, 1995.
7. C. W. Glover, F. Aminzadeh, N. Toomarian, and J. Barhen, "Neural Network Application to Seismic Parameter Estimation", *(in preparation for the IEEE Transactions on Geoscience and Remote Sensing)*.
8. K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, San Diego, 1990.