A Method to Determine the Required Number of Neural-Network Training Repetitions

Mahesh S. Iyer and R. Russell Rhinehart

***Abstract*—Conventional neural-network training algorithms often get stuck in local minima. To find the global optimum, training is conventionally repeated with ten, or so, random starting values for the weights. Here we develop an analytical procedure to determine how many times a neural network needs to be trained, with random starting weights, to ensure that the best of those is within a desirable lower percentile of all possible trainings, with a certain level of confidence. The theoretical developments are validated by experimental results. While applied to neural-network training, the method is generally applicable to nonlinear optimization.**

***Index Terms*—Neural-network training, steady-state identification, weakest-link-in-a-chain.**

I. Introduction

Neural networks are used for a variety of engineering applications like process modeling and control, time series modeling, process fault diagnosis, and system identification [2], [6], [11], [15]–[17], [21]. Control of bioreactors is finding increasing application of neural networks [7], [18] and pattern recognition [5], [8], [22] has been one of the largest applications of neural networks. For all the above, neural networks need to be trained, i.e., an optimization procedure is to be undertaken. While the most popular algorithm for training of feedforward neural networks has been error backpropagation [9], global optimizers like the Levenberg–Marquardt optimizer [12] are now gaining popularity [14], [19].

An inherent problem with optimizers is their tendency, for nonlinear problems, to get stuck in local minima. Neural-network training is nonlinear. To alleviate this problem, often, neural networks are trained more than once, starting with a random set of weights. The best neural network is often selected as the one with the lowest error. Park *et al.* [15] have reported the results on prediction of sunspots using a feedforward neural network. Their best network was selected from among ten networks of the same architecture, each of which was initiated with a random set of weights. In another work, Sha *et al.* [19] have reported the use of 25 random starts in the use of neural networks for ship design.

This study establishes a theoretical basis for the choice of the number of random starts in neural-network training. The concept, used for this development, is the best-of-N or the well accepted weakest-link-in-the-chain analysis [1]. The theory is verified by experimental results, in the context of neural-network training. However, the study is generally applicable, and can be extended, to any nonlinear optimization procedure.

Manuscript received July 6, 1998; revised December 15, 1998. This work was sponsored by the State of Texas Advanced Research Program Grant (Project 003 644-165).

M. S. Iyer is with the Department of Chemical Engineering, Texas Tech University, Lubbock, TX 79409-3121 USA.

R. R. Rhinehart is with the Department of Chemical Engineering, Texas Tech University, Lubbock, TX 79409-3121 USA, on leave with the School of Chemical Engineering, Oklahoma State University, Stillwater, OK 74078-5021 USA.

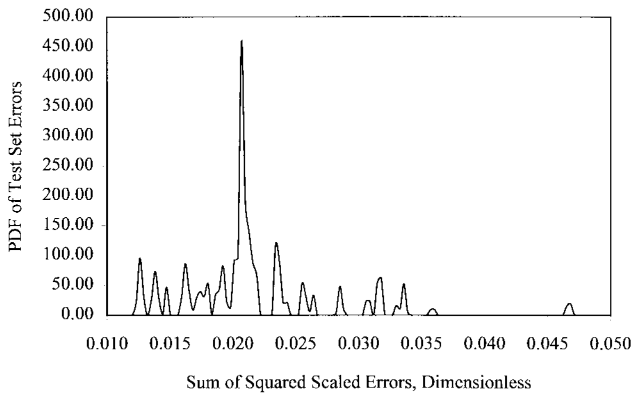
Publisher Item Identifier S 1045-9227(99)02670-3.

II. The Best-of-N or Weakest-Link-in-the-Chain Analysis [1]

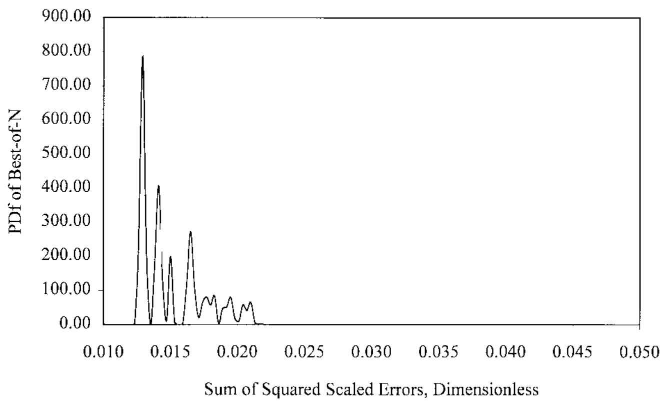
A chain is only as strong as its weakest link. In other words, the strength of a chain of N links, each of whose strengths is a distributed variable, is the strength of the weakest link. By analogy, the above concept is applicable to optimization problems, when an optimizer is used repeatedly, starting each time with randomly selected values of the decision variables. Each complete optimization, from one random start, is analogous to an individual link. The performance of the optimizer is typically measured by the sum-of-squared errors on a data set [9]. The lower the error, the better the optimizer's performance. The value of the error function, being minimized, is analogous to the strength of a link. The lowest error, of several random starts, is analogous to the strength of the weakest link. The "weakest link," from a certain number of random starts, would mean the best optimization, and not the poorest one, as the name may seem to suggest.

To develop the analogy further, consider the following thought experiment. Train a neural network many times (perhaps thousands), each from independent starting values for the weights. For each training, record X, the sum-of-squared errors on the independent test set, after training is complete. Create a histogram of the values of X. The curve which describes the histogram $f_X(x)$ is the probability density function (PDF) of X for a particular training problem. It may resemble the curve labeled as "PDF of test set errors" shown in Fig. 1(a). The x-axis location of peaks on the curve represent the sum-of-squared errors where the optimizer stopped. The height of the peaks on the curve represent the probability of the optimizer getting stuck in the local optima. It should be noted that the ordinate in Fig. 1(a) is the smoothed PDF obtained using the kernel method [20]. The global optimum has a value of 0.013 for the sum-of-squared errors, while the local optima have many values ranging from 0.015–0.53. The graph only shows the local optima with values less than 0.05, and it should be noted all errors are dimensionless, since scaled values of network prediction and desired outputs were used in the computation.

Continue the thought experiment. Train a particular network from each of N random starts (N is a small number, such as 10), and record the lowest of the N sum-of-squared



(a)



(b)

Fig. 1. (a) Sum of squared scaled errors, dimensionless. (b) Sum of squared scaled errors, dimensionless.

errors. The use of N independent random starts, in training a neural network, is equivalent to drawing an independent random sample of size N (with replacement) from the $f_X(x)$ distribution and selecting the best-of-N results. Create a histogram of these best-of-N values. It would resemble the PDF curve in Fig. 1(b), labeled "Best-of-N."

Reconsider the N independent training runs. The probability that any single optimization has an error value, x, less than or equal to "a" is $F_X(a)$, where $F_X(a) = \int_0^a f_x(x)\,dx$ is the value of the cumulative distribution function (CDF) at a. Then, the probability of $x > a$ is $1 - F_X(a)$, and the probability that all elements of the sample, of size N, have a value greater than a specific value, a, is $[1 - F_X(a)]^N$. Hence, the probability that at least one of the elements has a value lower than, or equal to, a is $1 - [1 - F_X(a)]^N$. Using $F_W(a)$ to represent the CDF for the weakest link, from N links

$$F_W(a) = 1 - (1 - F_X(a))^N. \tag{1}$$

Equation (1) explicitly defines the value of one of three variables, in terms of values of the other two. N is the number of random, independent optimization starts from which the best will be chosen. X is the sum-of-squared errors on any individual optimization. $F_X(a)$ is the fraction of random starts which would result in a value of X less than or equal to "a" and $0 \le F_X(a) \le 1$. If $F_X(a)$ has a value of 0.2, this means that the X-value for the sum-of-squared errors is one of the best (lowest) 20% possible values. W is the best (lowest) value for X, the sum-of-squared errors, out of N starts. $F_W(a)$ is the fraction of best-of-N X-values that result in a value of W less than or equal to "a" $0 \le F_W(a) \le 1$. If $F_W(a)$ has a value of 0.99, this means there is only a 1% chance of the best-of-N X-values being worse.

Equation (1) can be rearranged to solve explicitly for $F_X(a)$, given N and $F_W(a)$

$$F_X(a) = 1 - (1 - F_W(a))^{1/N}. \tag{2}$$

Alternatively, if the user decides on the desired values of $F_W(a)$, the level of confidence, and $F_X(a)$, the percentage vicinity of the lower tail of the distribution, which the best-of-N is expected to provide, then (1) can be arranged to give the required number of random starts

$$N = \frac{\ln(1 - F_W(a))}{\ln(1 - F_X(a))}. \tag{3}$$

For example, if you want to be 99% confident ($F_W(a) = 0.99$) that the best-of-N random starts will result in one of the best 20% values for the sum-of-squared errors ($F_X(a) = 0.20$) then $N = (\ln(1 - 0.99)/\ln(1 - 0.20)) \cong 20$.

III. Experimental Results

Equations (1)–(3) are widely accepted in analyzing reliability and failure of mechanical and electronic devices. In order to verify the theory presented above for optimization, three feedforward neural networks were trained. The details of the training exercise are as follows.

Case 1: A 1-20-2 network, with unipolar sigmoidal activation function in the hidden and output layers, was trained to learn the simulated variation in the reformate yield and octane number (network outputs) as functions of time (network input). Error backpropagation was used with a learning rate of 0.2 and a momentum constant of 0.7. Only the hidden layer was connected to a unit bias, while the output layer was not connected to any bias. There were 80 weights, but only 61 data points were used to train the network. A novel steady-state identification (SSID) algorithm [3], [4] was used to automatically stop training when the improvement in validation error was statistically insignificant [10], [14]. Each of the SSID algorithm parameters had a value of 0.05. The test set comprised 80 data points not seen by the network during training.

Case 2: An 8-4-1 network, with unipolar sigmoidal activation function in the hidden and output layers, was trained to learn the experimentally observed variation in the outlet pH (network output) in an in-line pH control system [13] as functions of eight different process variables (network inputs) using the Levenberg-Marquardt optimizer. Only the hidden layer was connected to a unit bias, while the output layer was not connected to any bias. There were 40 weights, and 75 data points were used to train the network. Training was stopped by the SSID algorithm, with values of 0.20 for each of the parameters in the algorithm. The test set comprised 120 data points not seen by the network during training.

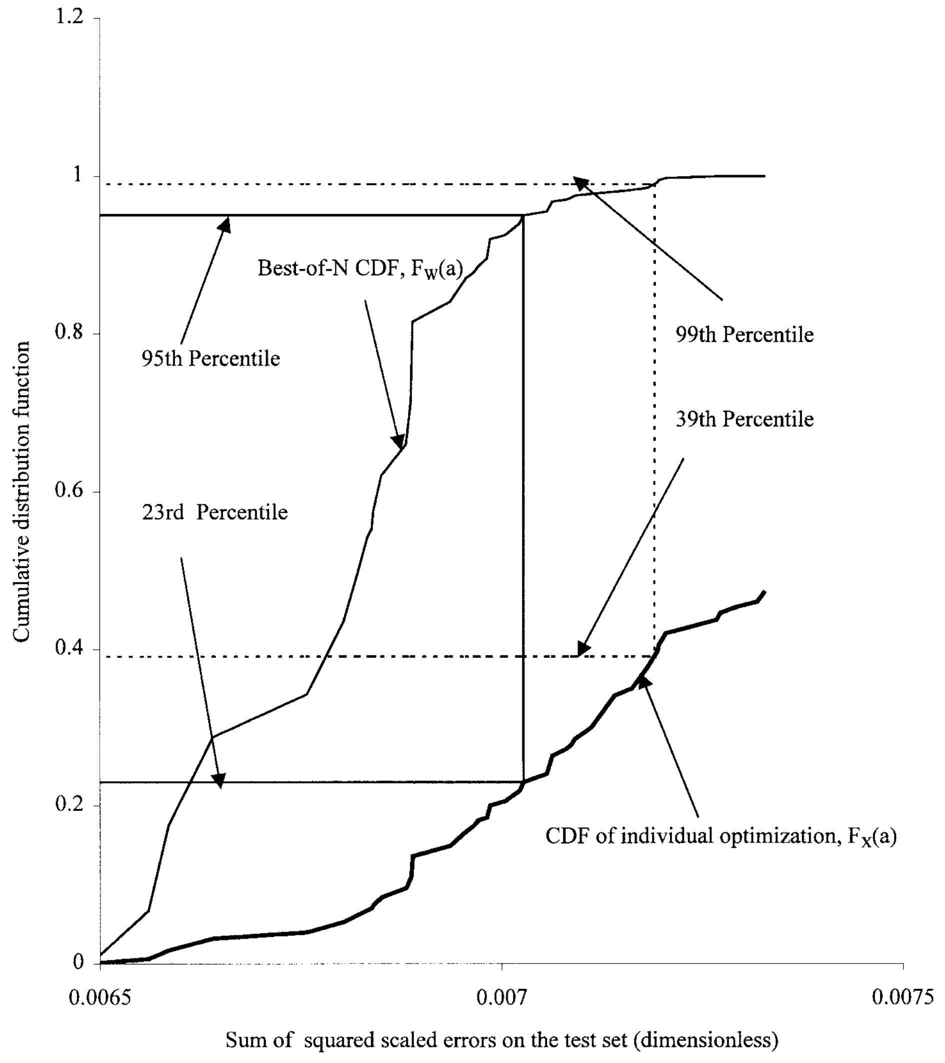


Fig. 2. CDF for test error and Best-of-N CDF for $N = 10$, Case 1.

Case 3: A 4-10-2 network, with unipolar sigmoidal activation function in the hidden and output layers, was trained to predict the simulated manipulated variable action (the reflux rate and the reboiler heat duty), corresponding to a particular feed rate and feed composition, and for control to target overhead and bottoms compositions (network inputs) for a methanol-water distillation column [17]. The Levenberg–Marquardt optimizer was used to determine the weights, and training was stopped by cross-validation. Only the hidden layer was connected to a unit bias, while the output layer was not connected to any bias. There were 70 weights, and 114 data points were used to train the network. The validation and test sets comprised 37 data points each.

In each of the cases described above, the inputs and outputs were scaled between 0.2 and 0.8 using, the entire data set available (including training, test and validation sets). The sum-of-squared errors was calculated on the scaled data.

In order to confirm the theoretical analysis, $f_X(x)$ and $F_X(x)$ must be known for each neural network. Hence, each neural network was trained 1000 times, starting each training with random initial values of the weights in the interval [−0.01, 0.01]. The 1000 observations on the test set error were accepted to represent the underlying distribution of the test set errors. We chose to test the analysis for values of N equal to five, ten, 20, and 50. The general procedure was to draw, with replacement, a random sample of size N from the 1000 observations. Fig. 1(a) is an example, from Case 2. The minimum test error of these N values was the experimental best-of-N values. For each N, this procedure was performed 400 times. The resulting 400 best-of-N data points was used to construct the best-of-N histogram. Fig. 1(b) is an example, from Case 2, for $N = 10$. We used the kernel technique [20] to smooth the PDF for both the test set and the best-of-N sum-of-squared errors.

Fig. 1(a) shows the presence of various local optima, values of X, where the optimizer gets stuck. The best-of-N PDF, Fig. 1(b) ($N = 10$), is located near the lower tail of the PDF of the sum-of-squared errors, as expected.

Figs. 2–4 show the plot of the CDF of sum-of-squared errors and the best-of-N CDF for a value of N equal to ten, for cases 1–3, respectively. The horizontal and vertical lines on these figures show that when ten random starts are taken, there

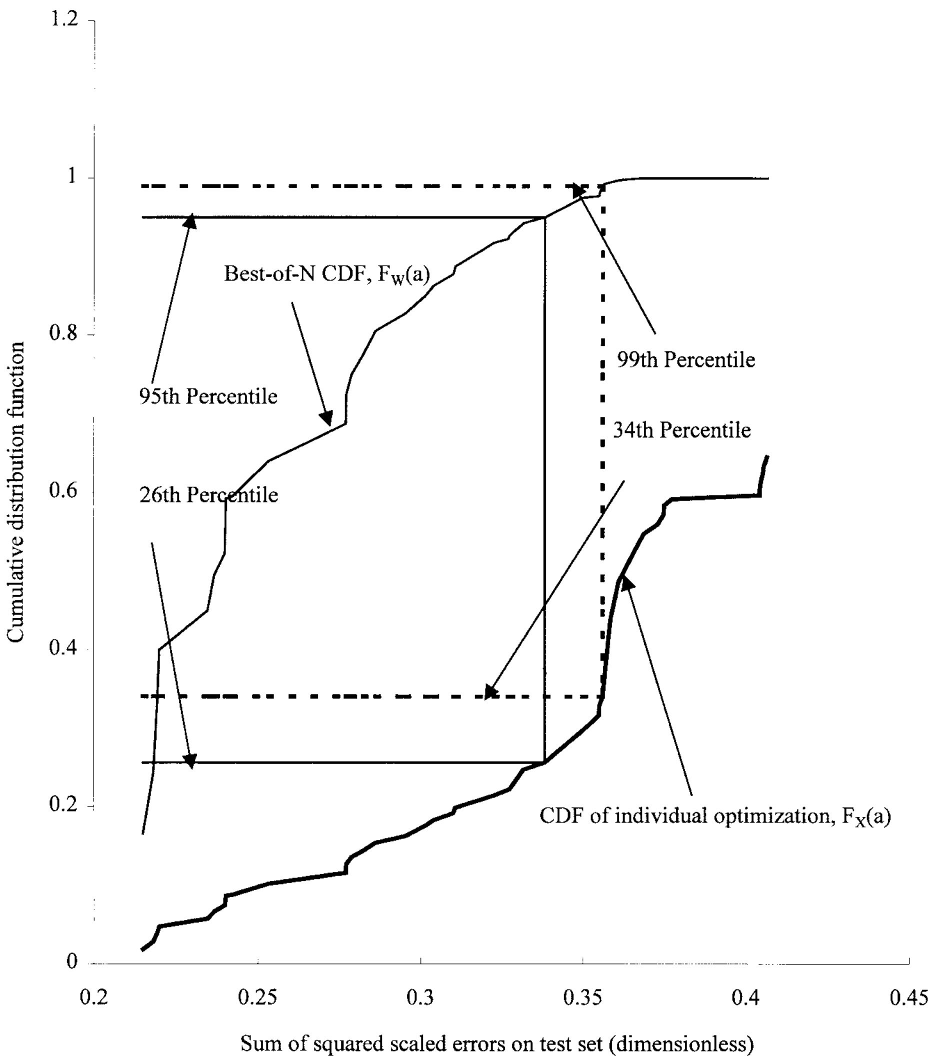


Fig. 3. CDF for test set error and Best-of-N CDF for $N = 10$, Case 2.

is a 95% probability that the best network from these ten starts would give a sum-of-squared errors that falls roughly in the lower 26% of all values comprising the underlying distribution of the sum-of-squared errors. Similarly, there is a 99% probability that the best network has a sum-of-squared errors corresponding to the lower 37% of all possible sum-of-squared errors that form the underlying distribution. Similar observations were made for several values of N (although not graphically depicted here).

The behavior of these observations were expected to obey (2). To verify it, use (2) to predict $F_X(a)$ from N and $F_W(a)$, and compare the theoretical values of $F_X(a)$ with those found experimentally from figures such as Figs. 2–4. The results obtained are presented in Tables I–III. Consider Table I. Theoretically, when the best network is selected from just five random starts, it is expected to result in a sum-of-squared errors that corresponds to the lower 45.08% of the values comprising the underlying distribution of the sum-of-squared errors, with a confidence of 95%. Experimentally, it was observed that the best-of-five random starts, for the same confidence, resulted in a network with sum-of-squared errors in the lower 44.6% of the underlying distribution. Statistically, 45.08 and 44.6% are indistinguishably different. Similarly, with 99% confidence, the best network from 50 random starts should give an error in the lower 8.8% of all possible errors. The experimentally observed figure was 9.13%. Results presented in Tables I–III, for various values of N, different neural-network architectures, two types of optimizers, three unique data sets, and two stopping techniques, all follow the theoretical pattern laid by the weakest-link-in-the-chain or best-of-N analysis.

IV. Important Features

The significant features of the study are as follows.

1) Equation (3) determines the required number of random starts of an optimizer, regardless of the application, optimization technique, and stopping criterion, activation function of neurons, etc. It was tested on a wide variety of neural-network training cases.
2) Tables I–III, and (2), indicate that when a network is trained with ten random starts (as has often been the case [15]), the lowest error found by the optimizer would

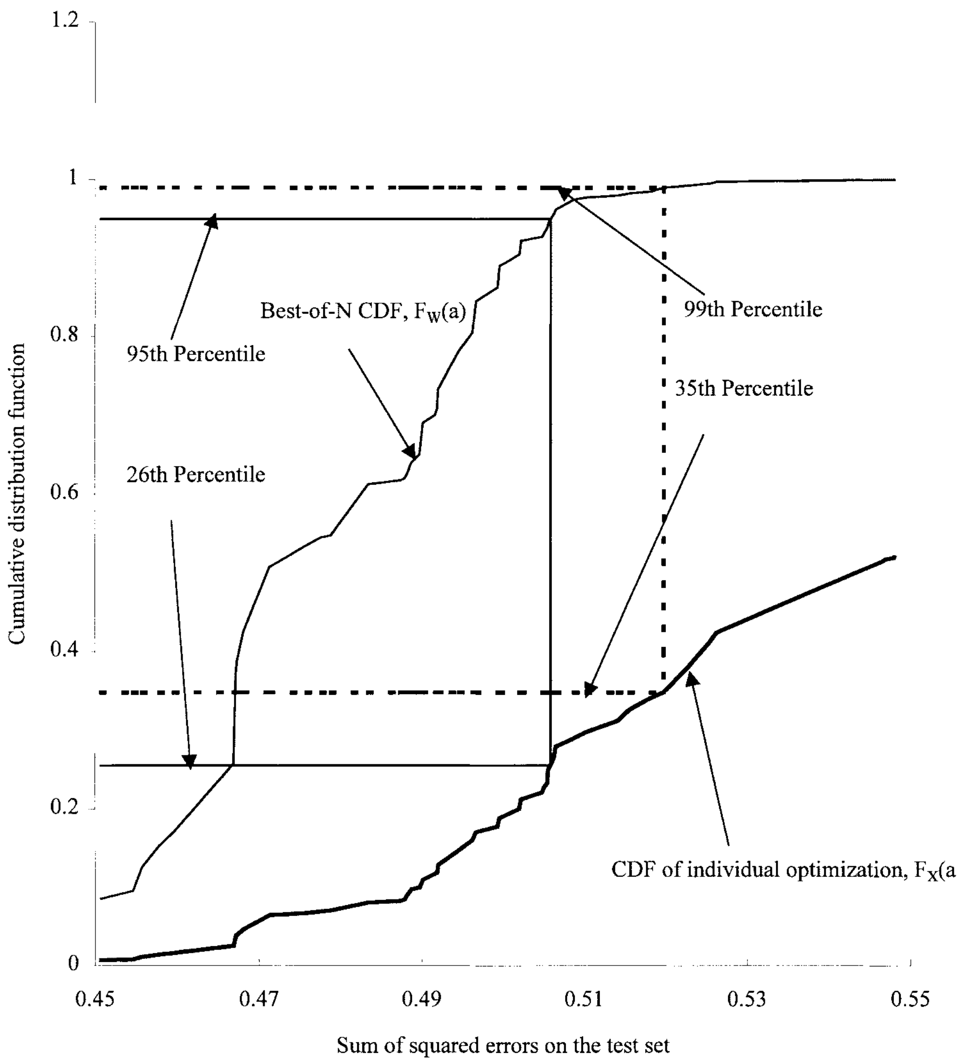


Fig. 4. CDF for test set error and Best-of-N CDF for $N = 10$, Case 3.

TABLE I
COMPARISON OF EXPERIMENTAL AND THEORETICAL RESULTS OF CASE STUDY (1)

Number of Random Starts	Lower Percentile for the Best of N			
	95% Confidence		99% Confidence	
	Theoretical	Experimental	Theoretical	Experimental
5	45.08	44.6	60.19	53.5
10	25.89	23.0	36.9	39.1
20	13.91	13.6	20.57	18.1
50	5.82	5.3	8.8	9.13

TABLE II
COMPARISON OF EXPERIMENTAL AND THEORETICAL RESULTS OF CASE STUDY (2)

Number of Random Starts	Lower Percentile for the Best of N			
	95% Confidence		99% Confidence	
	Theoretical	Experimental	Theoretical	Experimental
5	45.08	42.6	60.19	55.2
10	25.89	25.6	36.9	34.1
20	13.91	12.96	20.57	21.20
50	5.82	5.5	8.8	7

probably (99% confidence) be within the best 36.9% of the distribution of all possible errors. This may be acceptable, but we would prefer to probably have one of the best 20%, and therefore recommend that all networks be trained with at least 20 random starts.

3) The lower 20% of all possible errors (20 random starts) may not be as close to the global optimum as a particular application desires. Then (3) can be used to determine the number of random starts needed to obtain a trained network within a particular percentile of the generaliza-

TABLE III
COMPARISON OF EXPERIMENTAL AND THEORETICAL RESULTS OF CASE STUDY (3)

Number of Random Starts	Lower Percentile for the Best of N			
	95% Confidence		99% Confidence	
	Theoretical	Experimental	Theoretical	Experimental
5	45.08	45.5	60.19	59.2
10	25.89	25.5	36.9	34.8
20	13.91	13.2	20.57	18.2
50	5.82	5.88	8.8	8.31

tion errors, with a certain desirable level of confidence. To cite an example, if the best 2% of all errors is sought ($F_X(a) = 0.02$) with 99% confidence ($F_W(a) = 0.99$) then (3) yields the desired value of N as

$$N = \frac{\ln(1-0.99)}{\ln(1-0.02)} \cong 228.$$

4) The weakest-link-in-a-chain analysis provides a foundation to analytically evaluate the heuristic best-of-ten practice often used in neural-network training [15].

V. CONCLUSION

The close agreement between the theoretical expectations and the experimental observations, over a wide range of conditions, supports the best-of-N analysis to determine the number of random starts needed in neural-network training. The technique is equally applicable, regardless of the application, optimization algorithm, type of data set, and stopping technique.

REFERENCES

[1] R. M. Bethea and R. R. Rhinehart, *Applied Engineering Statistics.* New York, NY: Marcel Dekker, 1991.

[2] N. V. Bhat, P. A. Minderman, Jr., T. McAvoy, and N. S. Wang, “Modeling chemical process systems via neural computation,” *IEEE Contr. Syst. Mag.*, vol. 10, pp. 24–30.

[3] S. Cao and R. R. Rhinehart, “An efficient method for on-line identification of steady-state,” *J. Process Contr.*, vol. 5, no. 6, pp. 363–374, 1995.

[4] ——, “Critical values for a steady-state identifier,” *J. Process Contr.*, vol. 7, pp. 149–152, 1997.

[5] R. O. Duda, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.

[6] P. Dutta and R. R. Rhinehart, “Application of neural-network control to distillation and an experimental comparison with other advanced controllers,” *J Process Contr*, Feb. 1997.

[7] C. Emmandouilides and L. Petrou, “Identification and control of anaerobic digesters using adaptive, on-line trained neural networks,” *Comput. Chemical Eng.*, vol. 21, no. 1, pp. 113–143, 1997.

[8] K. Fukunaga, *Introduction to Statistical Pattern Recognition.* New York: Academic, 1972.

[9] S. Haykin, *Neural Networks, A Comprehensive Foundation.* New York: MacMillan, 1994.

[10] M. S. Iyer, Ph.D. dissertation, Texas Tech Univ., Lubbock, TX, 1997.

[11] N. V. Joshi, P. Murugan, and R. R. Rhinehart, “Experimental comparison of control strategies,” *Contr. Eng. Practice*, Mar. 1997, to be published.

[12] D. W. Marquardt, “An algorithm for least-squares estimation of nonlinear parameters,” *J. Soc. Indust. Appl. Math.*, vol. 11, no. 2, pp. 431–441, 1963.

[13] S. Natarajan and R. R. Rhinehart, “Implementation techniques for an inline pH controller,” in *Proc. 1994 Amer. Contr. Conf.*, Baltimore, MD, June 1994, pp. 3523–3527.

[14] ——, “Automated stopping criteria for neural network training,” in *Proc. 1997 Amer. Contr. Conf.*, Albuquerque, NM, June 1997, Paper #TP09-4.

[15] Y. R. Park, T. J. Murray, and C. Chen, “Predicting sun spots using a layered perceptron neural network,” *IEEE Trans. Neural Networks*, vol. 7, pp. 501–505, Mar. 1996.

[16] S. Ramaswamy, P. B. Deshpande, G. E. Paxton, and R. P. Hajare, “Consider neural networks for process identification,” *Hydrocarbon Processing*, vol. 74, no. 6, pp. 59–62, June 1995.

[17] S. Ramchandran and R. R. Rhinehart, “A very simple structure for neural-network control of distillation,” *J. Process Contr.*, vol. 5, no. 2, pp. 115–128, 1995.

[18] J. Schubert, R. Simutis, M. Dors, I. Havlik, and A. Lubbert, “Bioprocess optimization and control: Application of hybrid modeling,” *J. Biotechnol.*, vol. 35, pp. 51–68, 1994.

[19] O. P. Sha, T. Ray, and R. P. Gokarn, “An artificial neural network for preliminary ship design,” in *Proc. 8th Int. Conf. Computer Applicat. Ship Building*, p. 2, vol. 1014, pp. 10.15–10.30, vol. 2.

[20] B. W. Silverman, *Density Estimation for Statistics and Data Analysis.* London: Chapman and Hall, 1986.

[21] J. Thibault and B. P. A. Grandjean, “Neural networks in process control—A survey,” *IFAC Advanced Contr. Chemical Processes*, 1991, pp. 251–260.

[22] S. Watanabe, *Pattern Recognition: Human and Mechanical.* New York: Wiley, 1985.