Platt, John. 1990. "A Resource Allocating Neural Network for Function Interpolation." *To Appear.*

Press, W.H., and B.P. Flannery, S.A. Teukolsky, W.T. Vetterling. 1988. **Numerical Recipes in C.** Cambridge University Press, Cambridge.

Seber, G.A.F, and C.J. Wild. 1989. **Nonlinear Regression.** Wiley, New York.

Valiant, L.G. 1984. "A theory of the learnable." *Comm. of the ACM.* vol.27, no 11.

White, H. 1989. "Learning in Artificial Neural Networks: A Statistical Perspective." *Neural Computation*, vol.1, no.4, pp.425-464. MIT Press, Cambridge, MA.

White, H. 1990. "Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings." *Neural Networks*, vol.3, no.5, pp.535-549. Pergamon Press, New York.

Society. Presently available as: *TR# 90-011, International Computer Science Institute, University of California, Berkeley.*

Ash, A., and A. Hedayat. 1978. "An introduction to design optimality with an overview of the literature." *Communications in Statistics, Part A - Theory and Methods*, 7, 1295-1325.

Bartle, Robert G. 1966. **The Elements of Integration.** Wiley, New York.

Baum, E.B., and Haussler, D. 1989. "What size network gives valid generalization." *Neural Computation*, vol. 1, no.1, 151-160, MIT press, Cambridge MA.

Billingsley. 1986. **Probability and Measure.** Wiley, New York.

Blum, Avrim, and Ronald L. Rivest, 1988. "Training a 3-Node Neural Network is NP-Complete." *Computational Learning Theory (COLT).* MIT, August, 1988. Editors: Haussler (UCSC) and Pitt (Illinois). Sponsored by ACM SIGACT/SIGART, UC Santa Cruz, 1988.

Box,G., and N.Draper. 1987. **Empirical Model-Building and Response Surfaces.** Wiley, New York.

Cooray-Wijesinha, M., and A.I. Khuri. 1987. "The sequential generation of multiresponse D-optimal designs when the variance-covarance matrix is not known." *Comm. Statistics - Simulation*, 16(1), 239-259.

Cover, Thomas M. 1965. "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition" *IEEE Transactions on Electronic Computers*, EC-14, 326-334.

Faraway, Julian J. 1990. "Sequential design for the nonparametric regression of curves and surfaces." Technical Report #177, Department of Statistics, The University of Michigan.

Fedorov, V.V. 1972. **Theory of Optimal Experiments**. Academic Press, New York.

Gasarch, W.I. and M.B. Pleszkoch. 1989. "Learning via queries to an oracle." *Proc. COLT'89 (Second Annual Workshop on Computational Learning Theory.)*

Hellerstein, Lisa, and Marek Karpinski. 1989. "Learning read-once formulas using membership queries." *Proc. COLT'89 (Second Annual Workshop on Computational Learning Theory.)*

Hornik, Kurt, and M. Stinchcombe, H. White. 1989. "Multilayer feedforward networks are universal approximators." *Neural Networks.* vol.2, no.5, pp.359-366. Pergamon Press, New York.

Hwang, Jenq-Neng, J.J. Choi, Seho Oh, and Robert J. Marks III. 1990. "Query learning based on boundary search and gradient computation of trained multilayer perceptrons. " *Proc. IJCNN 1990, San Diego. The International Joint Conference on Neural Networks.* IEEE press.

Judd, Stephen, 1988. "On the Complexity of Loading Shallow Neural Networks." *Journal of Complexity*, vol.4, pp.177-192.

Judd, Stephen, 1988. "Learning in Neural Networks (extended abstract)." *Computational Learning Theory (COLT)*, MIT, August, 1988. Editors: Haussler (UCSC) and Pitt (Illinois). Sponsored by ACM SIGACT/SIGART, UC Santa Cruz, 1989.

Khuri, A.I., and J.A.Cornell. 1987. **Response Surfaces (Designs and Analyses)**. Marcel Dekker, Inc., New York.

Kurtz, S.A. and C.H. Smith. 1989. "On the role of search for learning." *Proc. COLT'89 (Second Annual Workshop on Computational Learning Theory.)*

Müller, Hans-Georg. 1984. "Optimal designs for nonparametric kernel regression." *Statistics and Probability Letters 2.* pp.285-290.

Myers, Raymond H., and A.I. Khuri, W.H. Carter, Jr. 1989. "Response Surface Methodology: 1966-1988." *Technometrics.* vol.31, no.2.

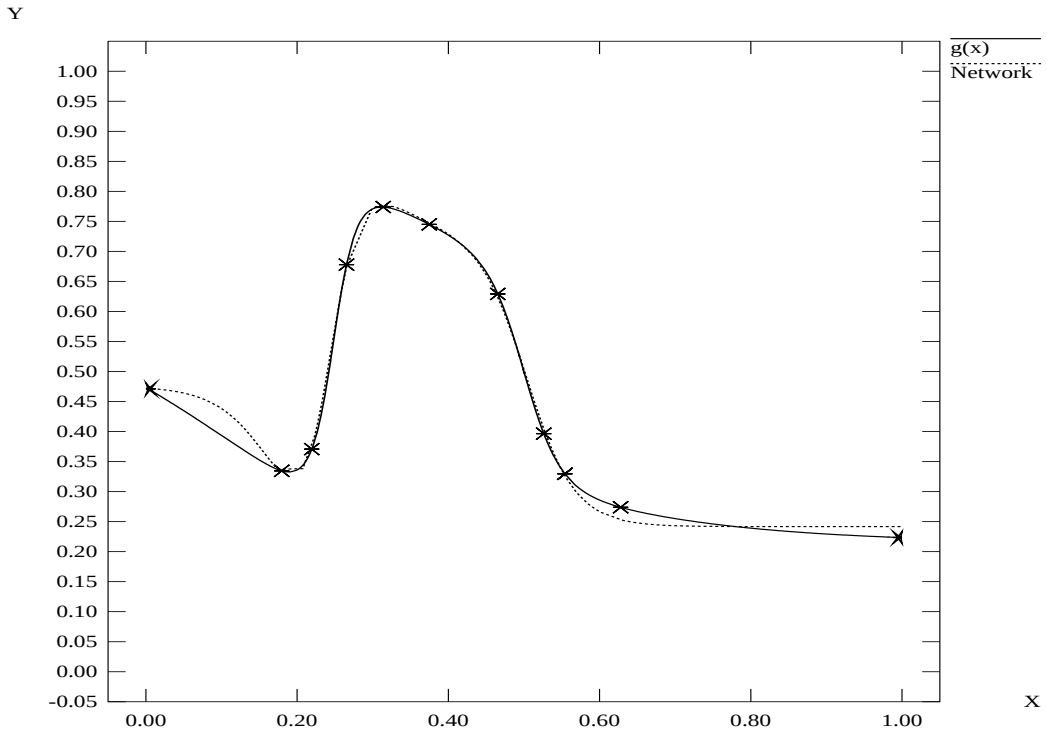

Figure 28: Network obtained by active selection of x^{11} (see previous section.)

8 Conclusion

The goal of this work is to provide a base from which to pursue a sampling criterion for noisy data. The selection of the IMSE criterion used to motivate the result obtained here is a step towards this goal. The result we obtain here for the special case of noiseless data is interesting in itself, as well. The sampling method obtained from this criterion is reasonable to compute and exhibits satisfying behavior on simple mappings. The training sets so obtained do not exhibit the clustering effects and gaps often present in training sets obtained by random sampling. These results indicate that this method would be of use when the training set is generated sequentially from a set of available candidates, such as in certain data-driven learning techniques. It may also provide an advantage in domains where data is abundant, by pruning the training set to a manageable size.

Empirical experience with the selection criterion indicates that aspects of its use require more study, in order to automate the technique more completely. In particular, to determine when it is appropriate to select another example, we need an objective means for determining whether network training is sufficiently stringent to ensure that the information content of the current set of examples has been fully exploited. We also need an objective means for regulating network complexity as the size of the training set increases. We expect that by minimizing the data requirement, data-adaptive means of achieving this latter goal may be made more tractable.

References:

Abu-Mostafa, Yaser S. 1989. "The Vapnik-Chervonenkis Dimension." *Neural Computation*, vol. 1, no.3, 312-317, MIT press, Cambridge MA.

Ahmad, Subutai, and Stephen Omohundro. 1990. "A network for extracting the locations of point clusters using selective attention." To appear: *12th Annual Conference of the Cognitive Science*

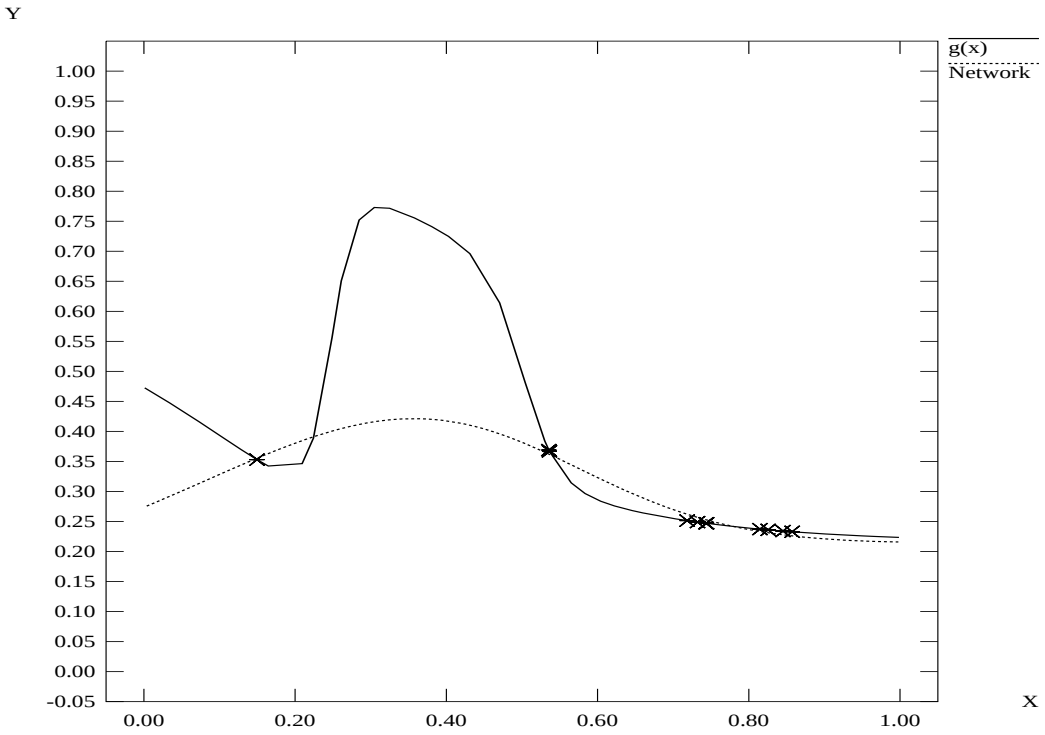

Figure 26: Worst network obtained by random selection of x^{11}.

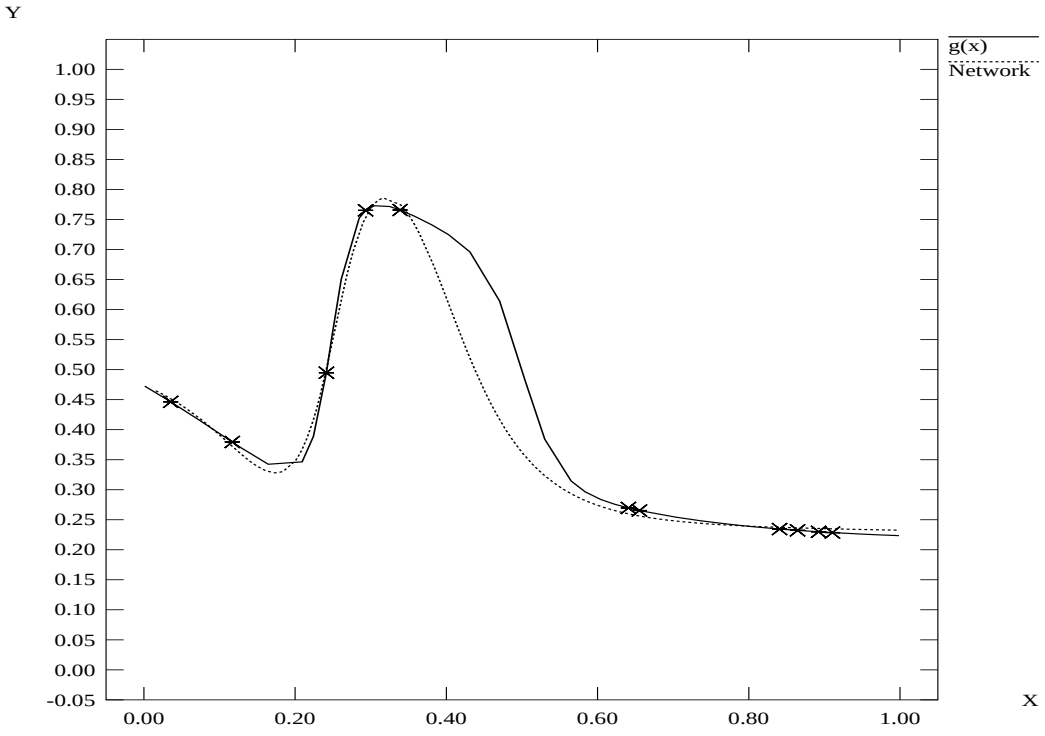

Figure 27: Typical example of a network obtained by random selection of x^{11}.

Statistics for Figures 25, 26, 27, and 28.				
	Training Error		Test Set Error	
n	mse	wse	mse	wse
Best	.00004	.00027	.00175	.02253
Worst	.00004	.00011	.02615	.12740
Typical	.00006	.00013	.00347	.04115
ΔISB	.00008	.00043	.00027	.00201

Table 3: Statistics of best, worst, and typical networks obtained from 25 random selections of x^{11}, compared with the network obtained by ΔISB in the previous section.

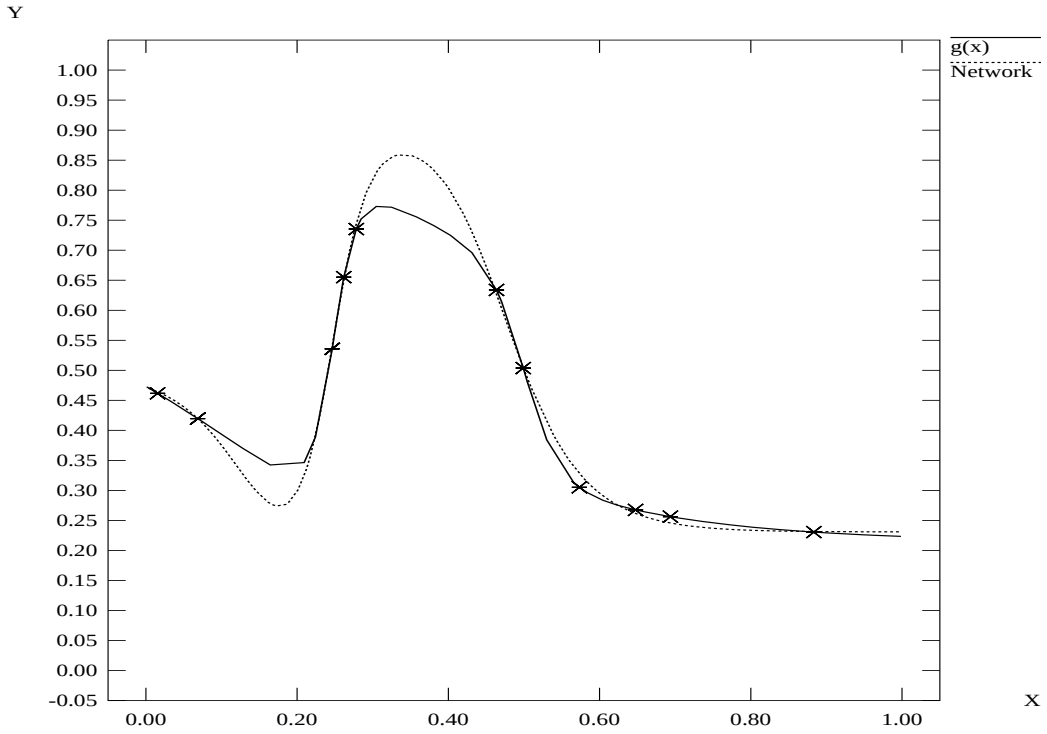

Figure 25: Best network obtained by random selection of x^{11}.

Statistics for Experiment 2. (See Figures 13 through 24.)							
	Training Error		Test Set Error		New Set Error		
n	mse	wse	mse	wse	mse	wse	New example
1	.00000	.00000	.03666	.14245	.07123	.14245	(.313, .775)
2	.00000	.00000	.07611	.32016	.00289	.00865	(.998, .224)
3	.00000	.00000	.06604	.29210	.05908	.23631	(.219, .371)
4	.00000	.00000	.02267	.20693	.00428	.02141	(.180, .335)
5	.00101	.00262	.01147	.13168	.02279	.13168	(.003, .471)
6	.00010	.00033	.00200	.02817	.00411	.02817	(.265, .678)
7	.00061	.00112	.00129	.01465	.00236	.01465	(.465, .629)
8	.00106	.00336	.00110	.00935	.00184	.00808	(.375, .745)
9	.00184	.00808	.00110	.00935	.00184	.00808	(.554, .330)
10	.00005	.00011	.00032	.00127	.00016	.00126	(.628, .274)
11	.00008	.00043	.00027	.00201	.00025	.00201	(.099, .394)
12	.00016	.00053	.00013	.00053	.00019	.00053	(.097, .396)

Table 2: Training statistics for second experiment. The network possesses a single hidden unit for $1 \leq n \leq 3$. For $4 \leq n \leq 9$, the network possesses two hidden units. For $n = 10, 11$ and 12, the network possesses 3 hidden units.

7.1 Comparison with random sampling

In this section, we compare the sampling distribution determined by ΔISB with sets of equal size chosen randomly. To do so, we compare the results of stringent training upon randomly selected sets with a stringently trained network obtained from active selection of examples. Recall, as discussed in the previous section, that we chose to quit training upon x^{12} before completely exploiting the information in the training set, in that training set mse still exceeds test set mse at the time training was halted. For this reason, we perform the following comparison using training sets composed of 11 examples, and compare the results obtained from stringent training upon these training sets with the result obtained from the stringently trained network for $n = 11$ above.

The subsequent figures show selected networks obtained by 25 different selections of x^{11} uniformly at random from the available set of examples. Table 4 provides the statistics for the networks obtained in this way. Rather than showing the results of all 25 runs, we give the best and worst runs, as well as a typical run, judged according to network error over the test set. In Figure 28 we show the network obtained by active selection of x^{11}, as discussed above. To allow comparison with the final result obtained for $n = 11$, the statistics for $n = 11$ tabulated in Table 2 are repeated in Table 4 in the row titled "ΔISB."

In these experiments we immediately observed that random sampling often resulted in clustering of the training examples. Even the "best" training sample obtained by random selection exhibits a sizeable gap near the mode of the desired function, (i.e., in the neighborhood of $x = 0.4$), resulting in a sizeable portion of the test set mse of the resulting network being due to errors within this region. In comparison, selecting examples by ΔISB apparently provided a sampling distribution which placed examples where the network needed them most, without the clustering effects and obvious gaps which were likely in a random sample of the same size.

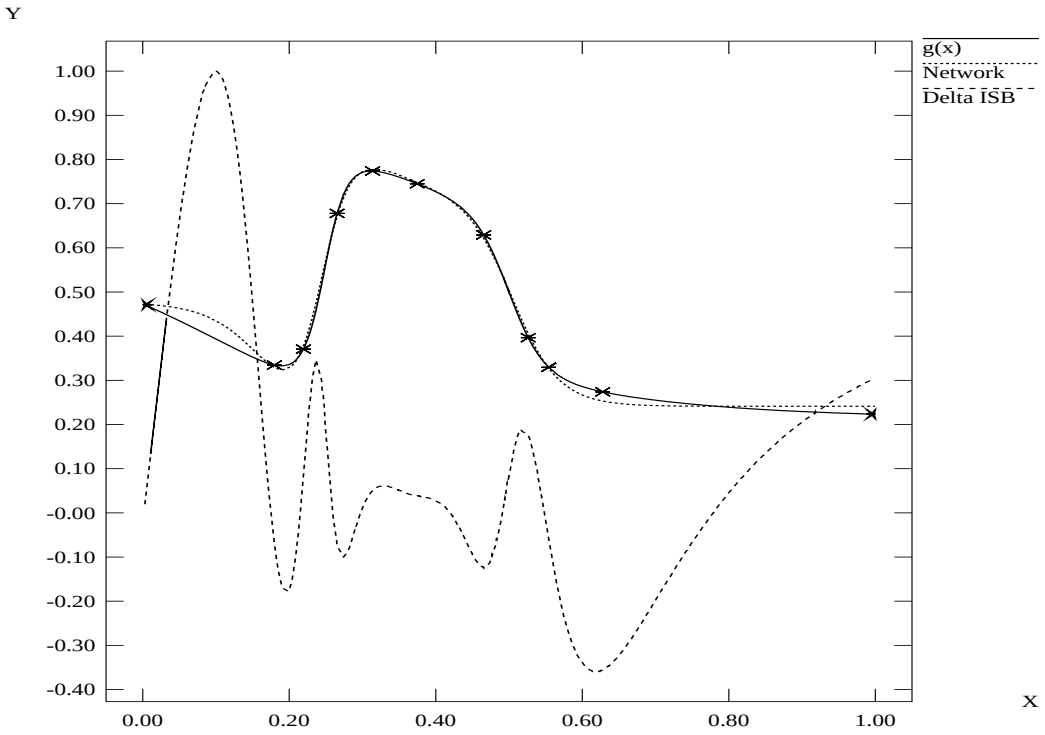

Figure 23: 3 hidden units. Trained on x^{11}.

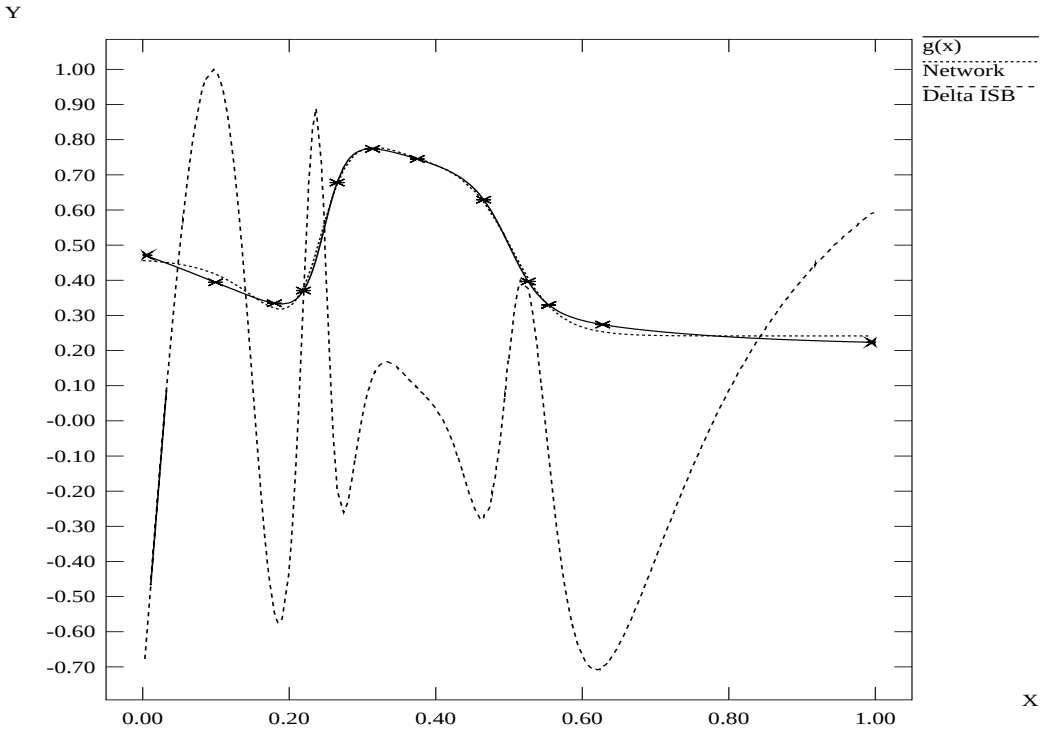

Figure 24: 3 hidden units. Final training on x^{12}.

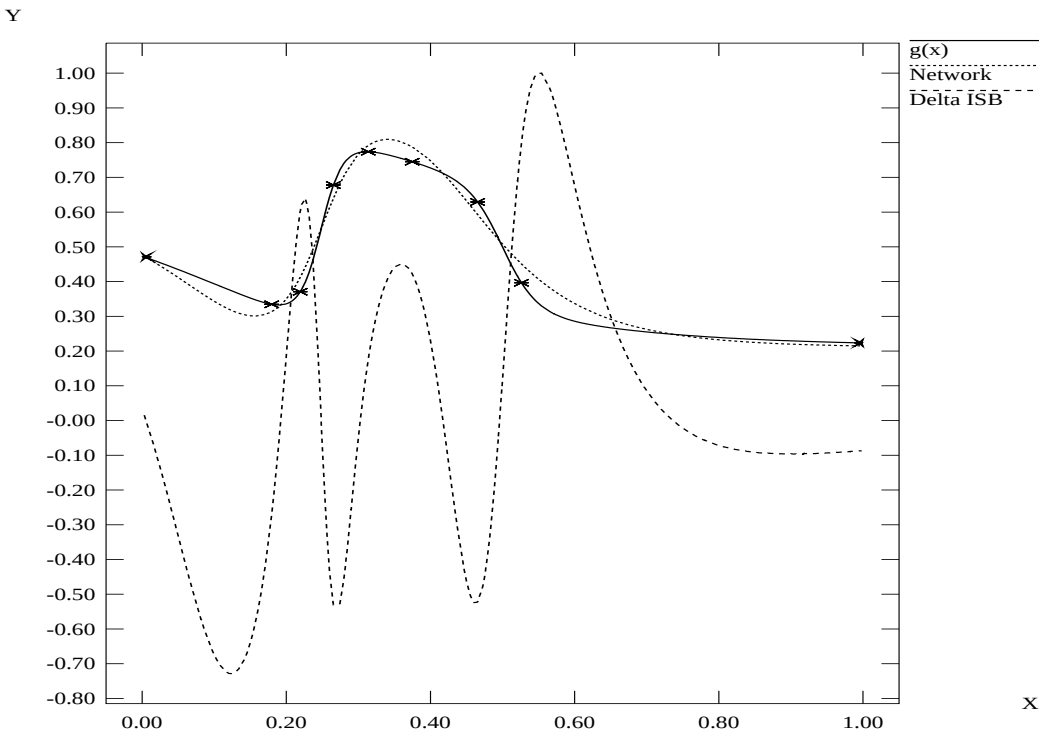

Figure 21: 2 hidden units. Training on x^9.

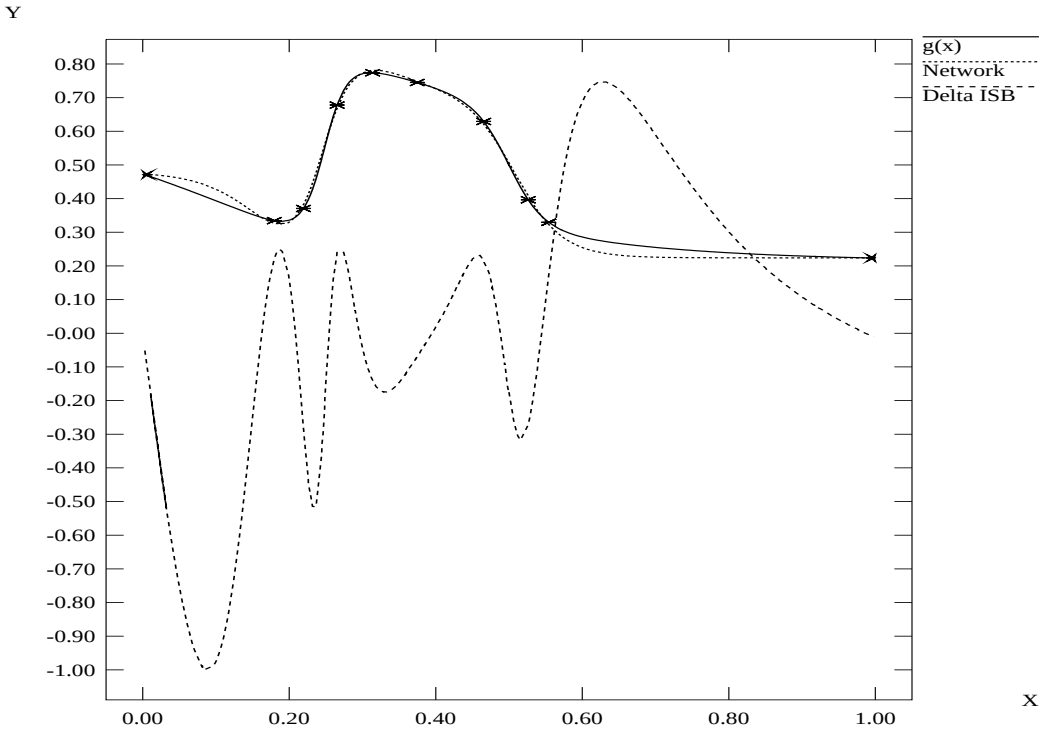

Figure 22: 3 hidden units, trained on x^{10}.

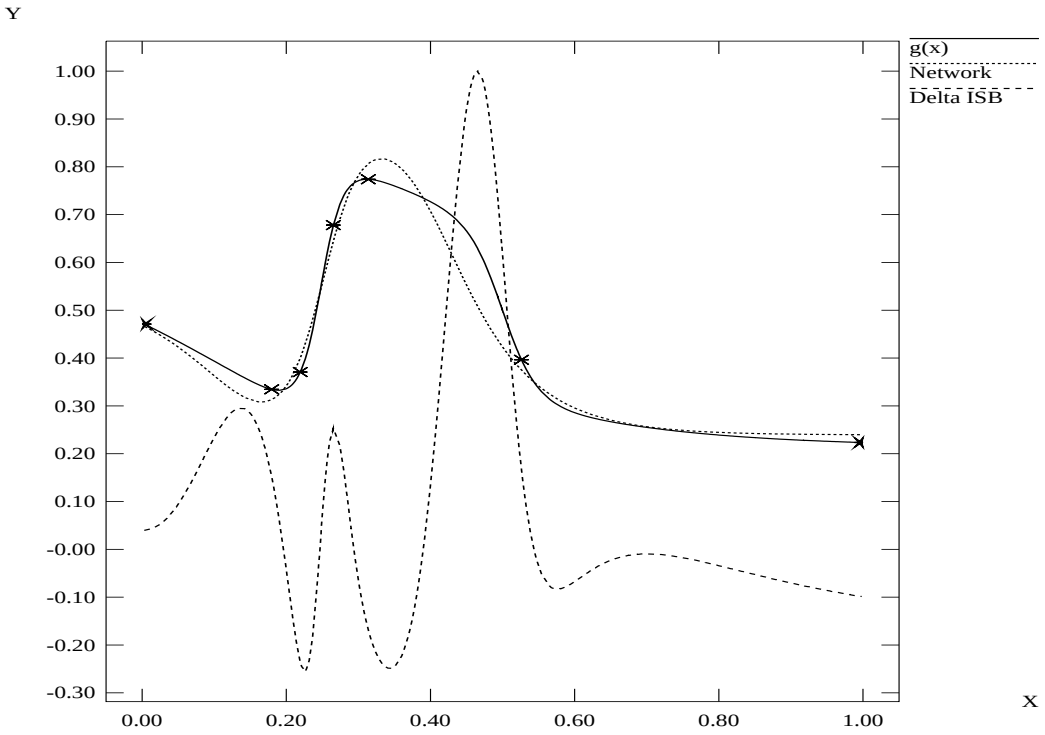

Figure 19: 2 hidden units, trained on x^7.

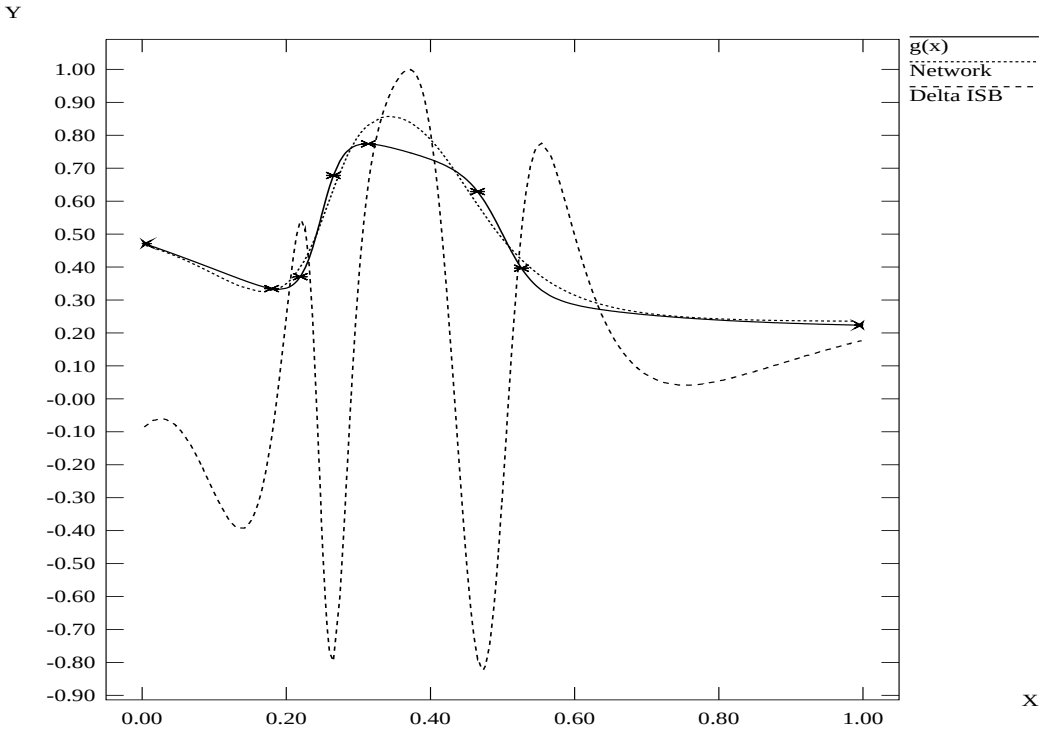

Figure 20: 2 hidden units, trained on x^8.

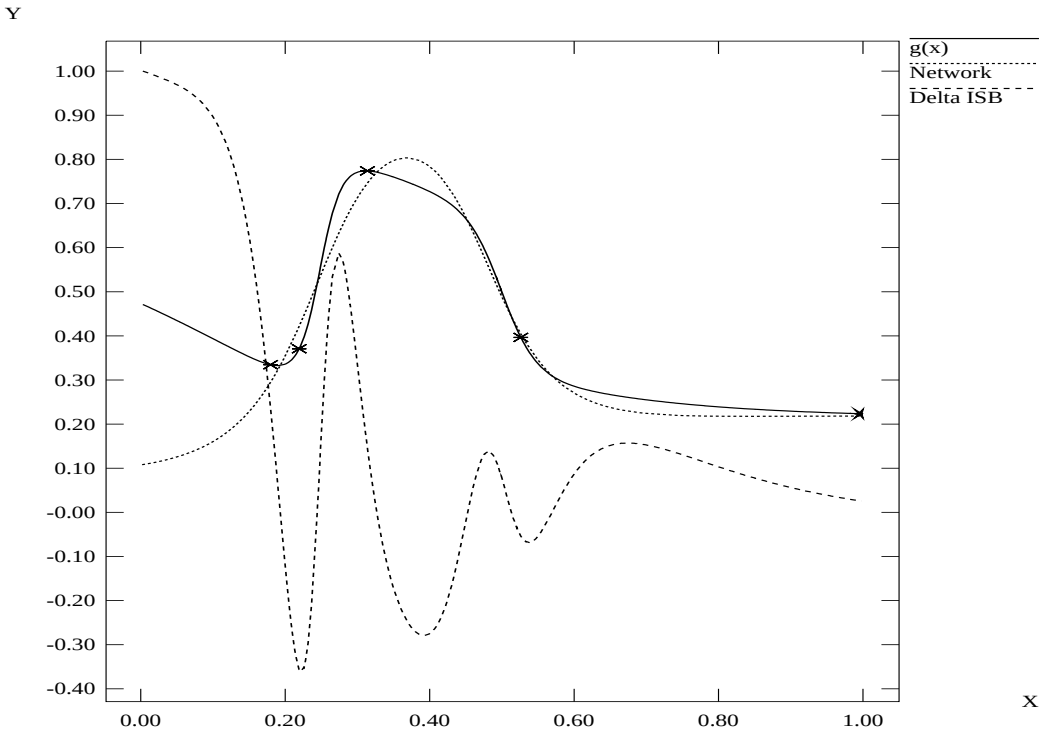

Figure 17: 2 hidden units, trained on x^5.

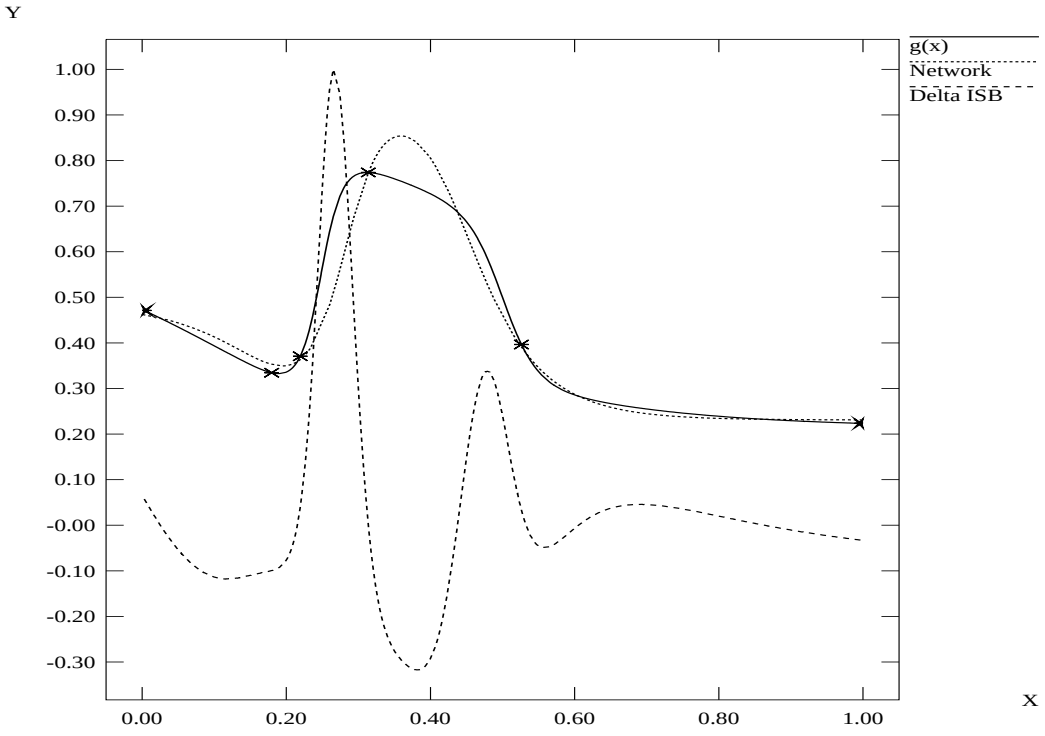

Figure 18: 2 hidden units, trained on x^6.

training upon x^{12}. Note also that the Training Set mse is greater than the Test Set mse. This is another indication that training should continue upon x^{12}, if we wish to obtain a better fit. Since the test set mse is reasonably low, this is a good stopping point for the learning process. We stop training at $n = 12$, with the network exhibiting a reasonably close fit to the desired function.

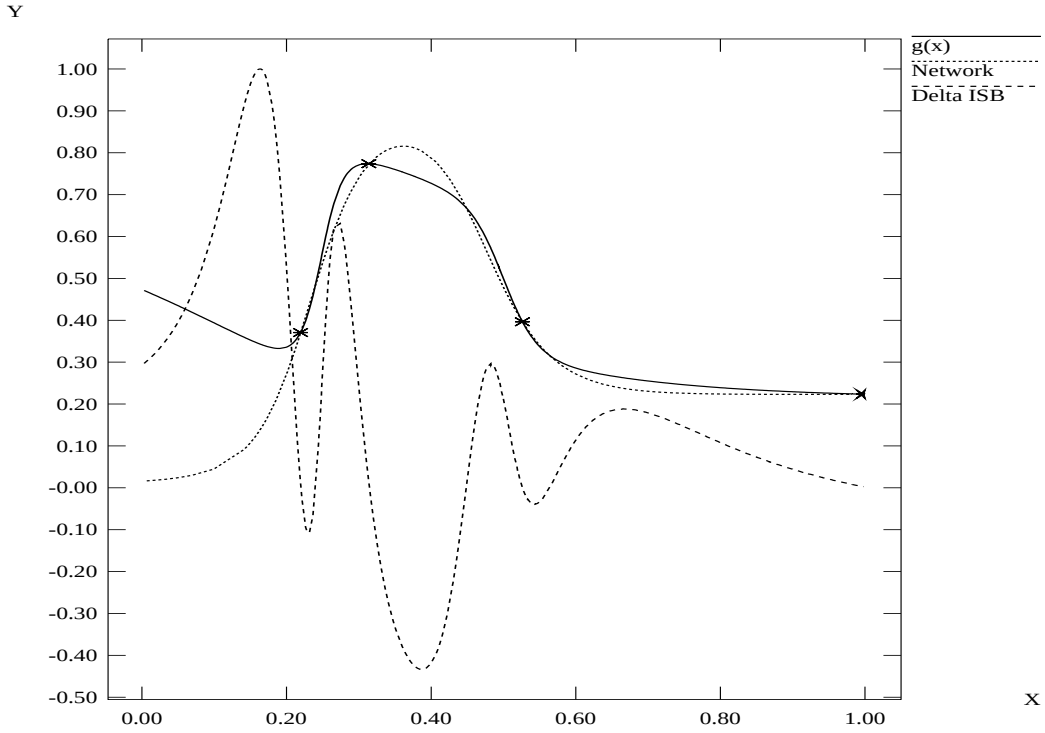

Figure 16: 2 hidden units, trained on x^4.

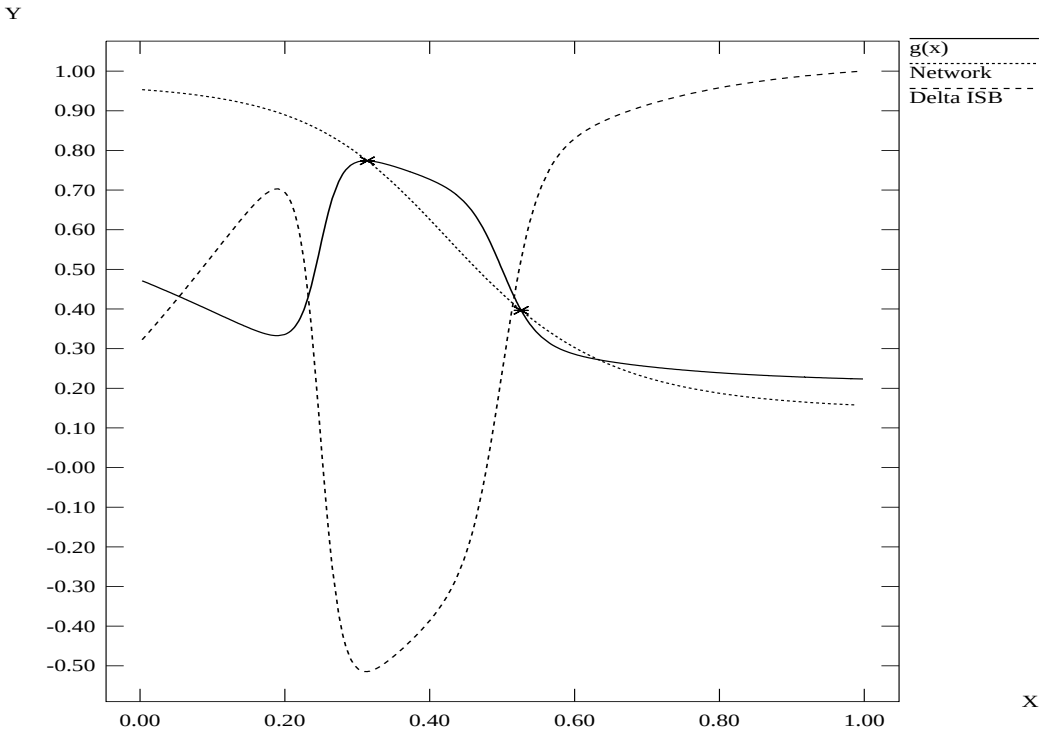

Figure 14: 1 hidden unit, trained on x^2.

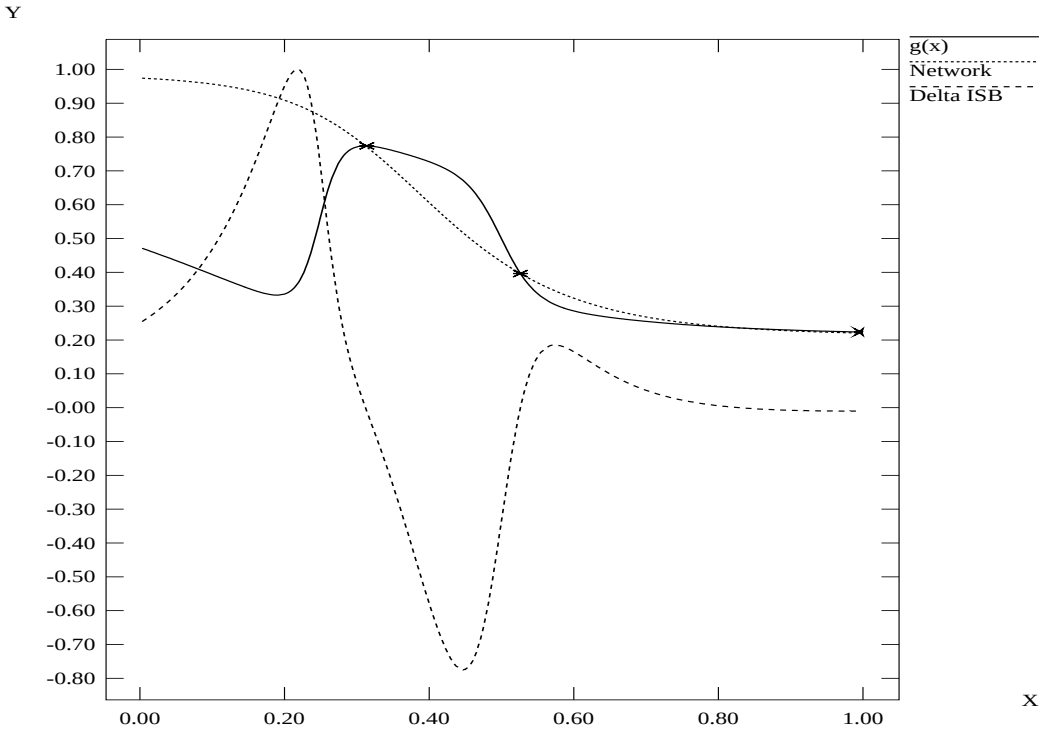

Figure 15: 1 hidden unit, trained on x^3.

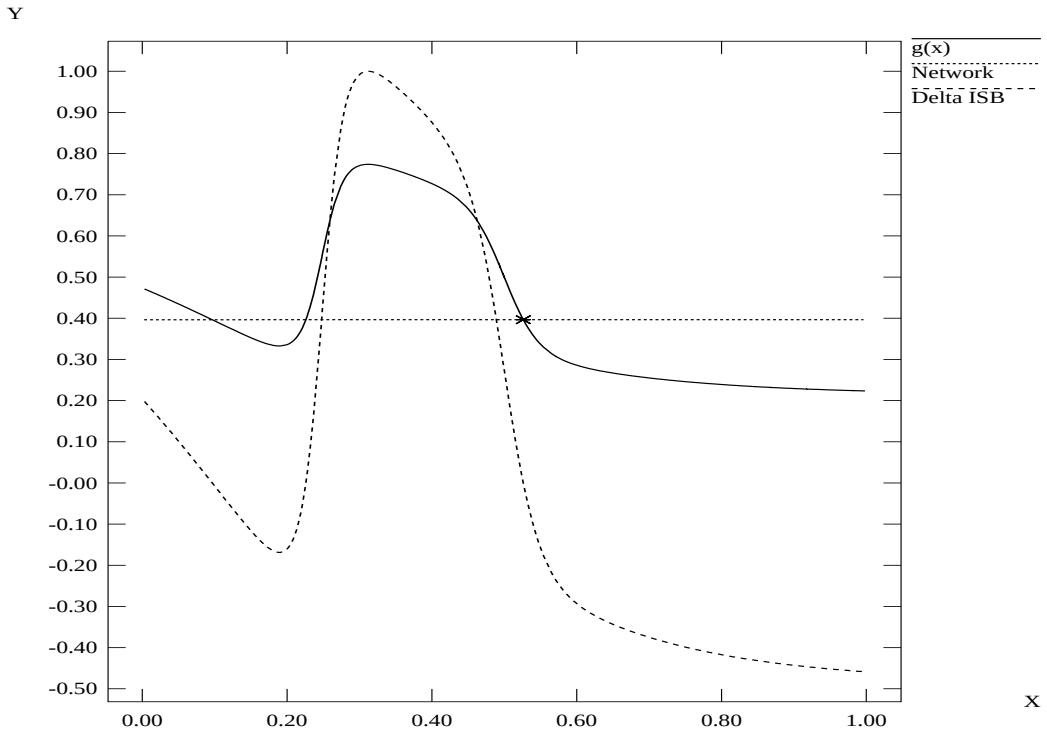

Figure 13: 1 hidden unit, trained on x^1.

Our choice of $g(x)$ was motivated by a desire for a function difficult to learn with a network of less than three hidden units, assuming no connections from inputs to outputs. In the first experiment with using the ΔISB criterion on this mapping, the network initially has one hidden unit. As new examples are added, when the network can no longer fit the training examples within close tolerance, we modify network complexity by adding another hidden unit.

The training regime is the same as for the experiment reported in the previous section, except for extension to growing the network. To choose x_1 we first calculated the average of $g(x)$ over x^N, resulting in a value of .415. Due to randomness in generating x^N from the network illustrated in Figure 12, none of the examples gave $g(x) = .415$. We chose the example x giving $g(x)$ closest to this value. This gave $x_1 = 0.563$, for which $g(x_1) = 0.397$.

For each n, the figures illustrate $f(x, w_n)$ and $\Delta ISB(x|x^n)$ along with $g(x)$. $\Delta ISB(x|x^n)$ is labelled "Delta ISB." Note for each n where the new point was obtained, corresponding to the point maximizing ΔISB in the plot for $n - 1$. At $n = 4$, optimization stalled, as readily evidenced by repeated attempts to obtain a better fit over the sample x^4 with a wide variety of parameter settings over several restarts. At this time the worst squared error over the training set was the same order of magnitude as the mean squared error over the training set, which we interpreted as another indication that the network was reaching its capable limit for fitting x^4. Hypothesizing that the network needed more complexity to capture subsequent points, we decided to add one hidden unit, and retrained upon x^4 before selecting x_5.

At $n = 9$, we added a third hidden unit. In Table 2 note that for $n = 8$ and $n = 9$, Test Set Error was unchanged, while Training Set Error increased to more than Test Set Error. We interpreted this as an indication to grow the network to provide an additional amount of flexibility in fitting the training examples. So, for $n = 10, 11, 12$, the network has three hidden units. Note that the next example that would be chosen at this time is relatively close to one chosen previously (for $n = 11$.) Hence, if we wish to continue training, it would behoove us to reject this example and continue

the desired function.

Statistics for Experiment 1. (See Figures 6 through 11.)							
	Training Error		Test Set Error		New Set Error		
n	mse	wse	mse	wse	mse	wse	New example
1	.00000	.00000	.08191	.20663	.10330	.20660	(.001, .818)
2	.00000	.00000	.01661	.12495	.04165	.12495	(.243, .795)
3	.00008	.00013	.02216	.03451	.00411	.01619	(.314, .242)
4	.00001	.00003	.00023	.00034	.00007	.00027	(.273, .651)
5	.00002	.00005	.00040	.00063	.00012	.00063	(.423, .183)
6	.00006	.00011	.00007	.00012	.00006	.00011	(.273, .651)

Table 1: Catalog of mean squared error (mse) and worst squared error (wse) for each n. Test Set refers to the set of all available examples. The New Set Error refers to the training set x^{n+1} obtained by adding the new example (tabulated in the rightmost column) to x^n. New Set Error is calculated immediately after adding the new example to the training set x^n, before commencing with training on the new training set.

7 Active Sampling Applied to a Misspecified Network

We now illustrate how the sampling criterion behaves upon a network which does not necessarily have the correct number of hidden units to fit the desired function exactly. We investigate the behavior of the sampling criterion using the skewed unimodal mapping $g(x)$ illustrated in Figure 13. This mapping can be realized exactly by a 3-hidden unit feedforward network with the connectivity and weight values shown in Figure 12, using the same type of sigmoided units used in Network 1.

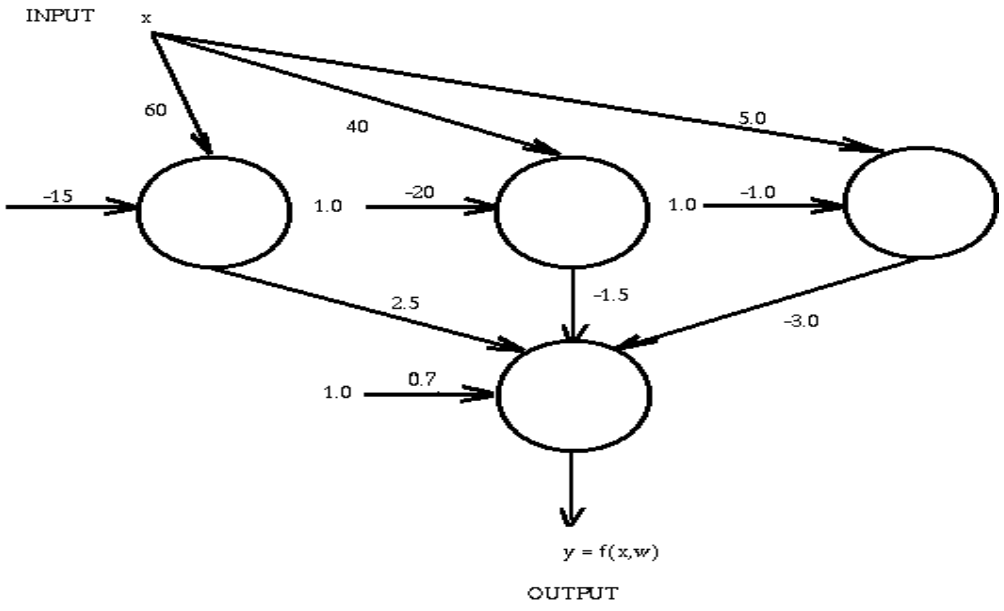

Figure 12: Network 2.

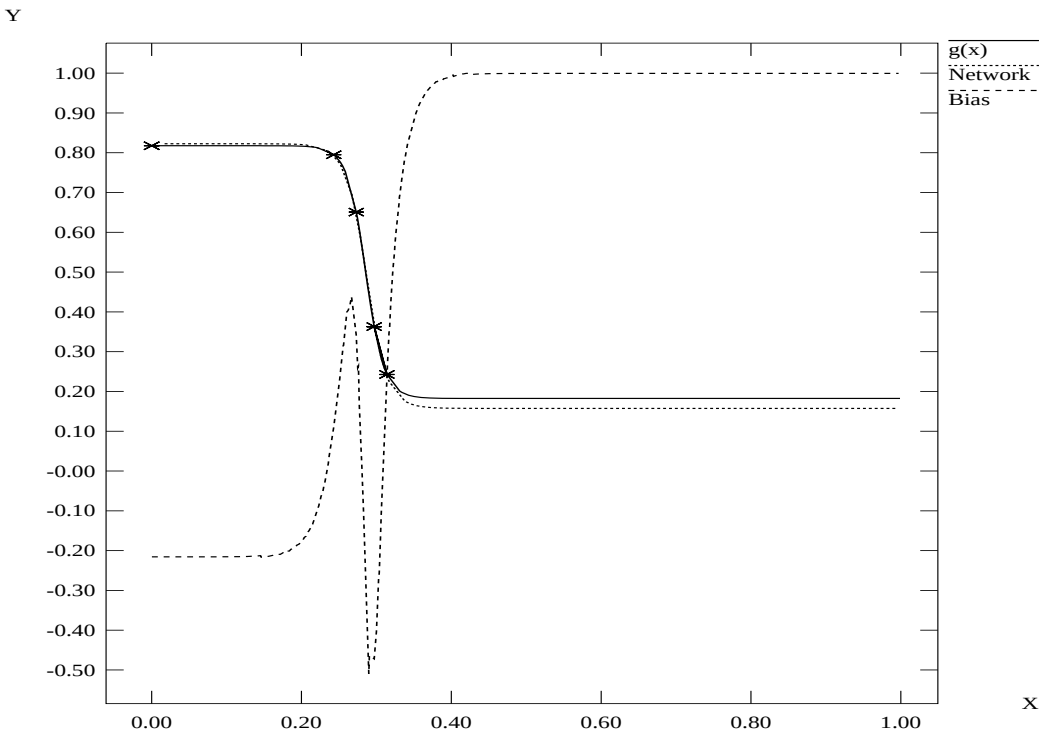

Figure 10: After training on x^5. $x_6 = .423$.

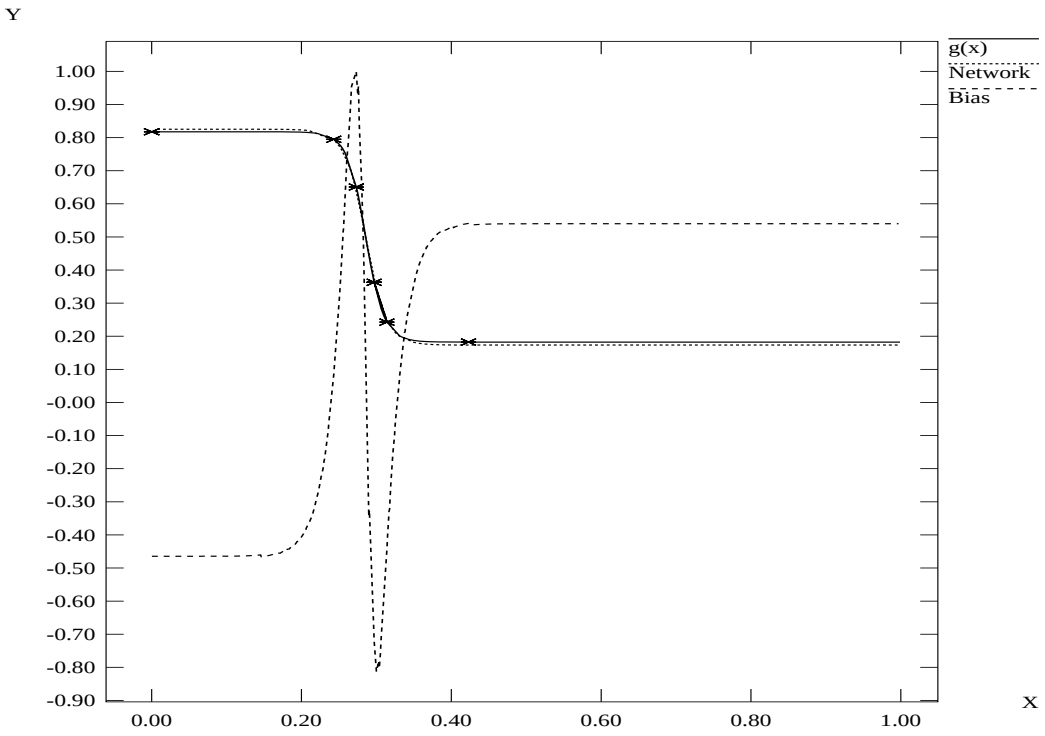

Figure 11: Final training on x^6.

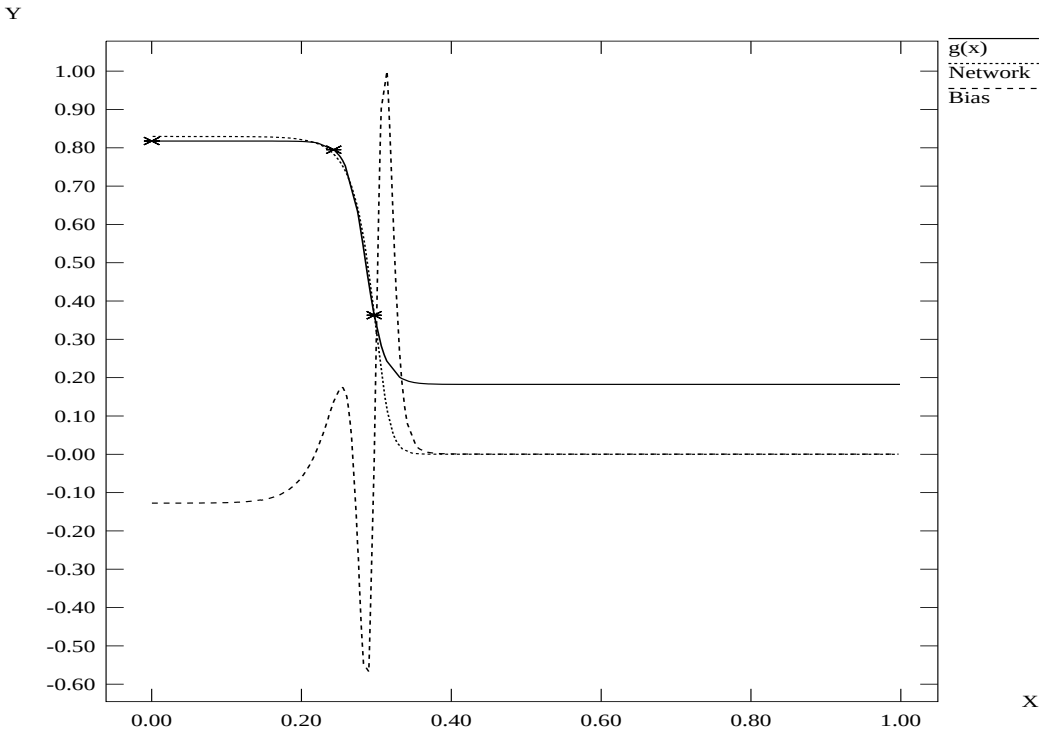

Figure 8: After training on x^3. $x_4 = .314$.

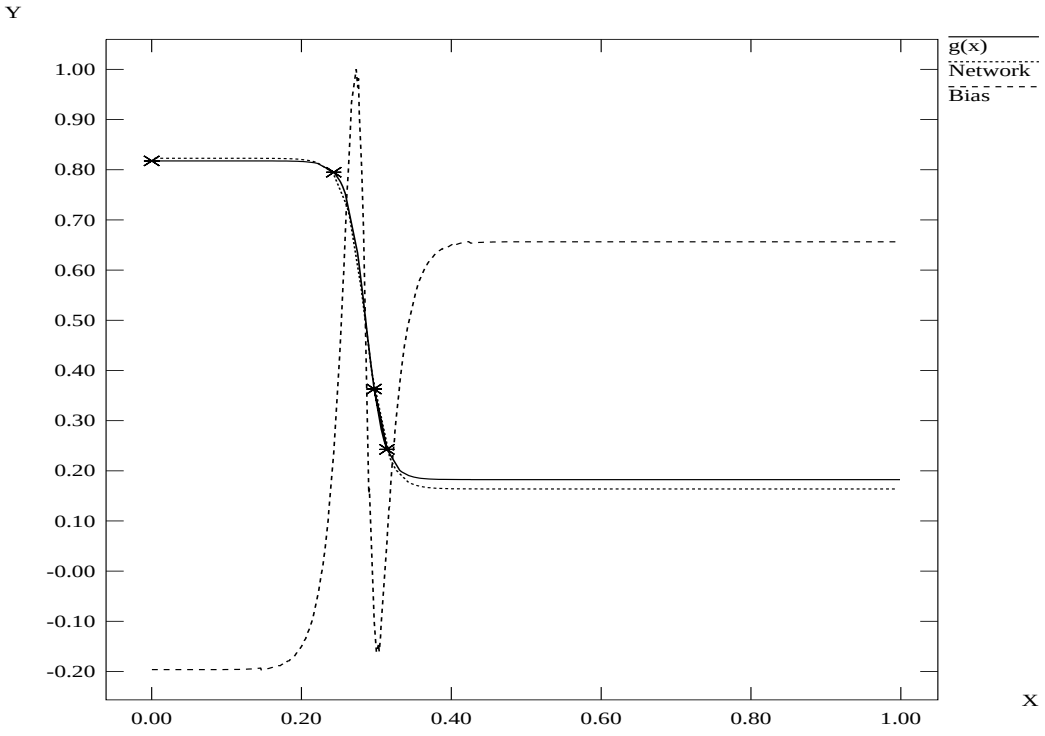

Figure 9: After training on x^4. $x_5 = .273$.

upon $g(x)$. Note how training on x^2 resulted in improvement in the lower plateau region of the mapping, even though x_2 was chosen from the upper plateau. After obtaining w_2, ΔISB was maximized at $x_3 = .243$, where $g(x_3) = .795$. Training set mean squared error was .00000. Worst case squared error (wse) over the training set was .00000. The wse gives the squared error for the example for which the network exhibits maximal error. Mean squared error (mse) over the 500 test set examples was .01661, worst case, .12495. After appending x_3 to the training set to obtain x^3, but before commencing training, the mse over x^3 was .04165, and the wse, .12495. These statistics are compiled in Table 1, along with the statistics for each other value of n. Recall that the training set is a subset of the set of all available examples, which for simplicity we refer to as the test set. The statistics for the case in which we desire the test set and training set to be disjoint may be easily obtained from the information given in the table.

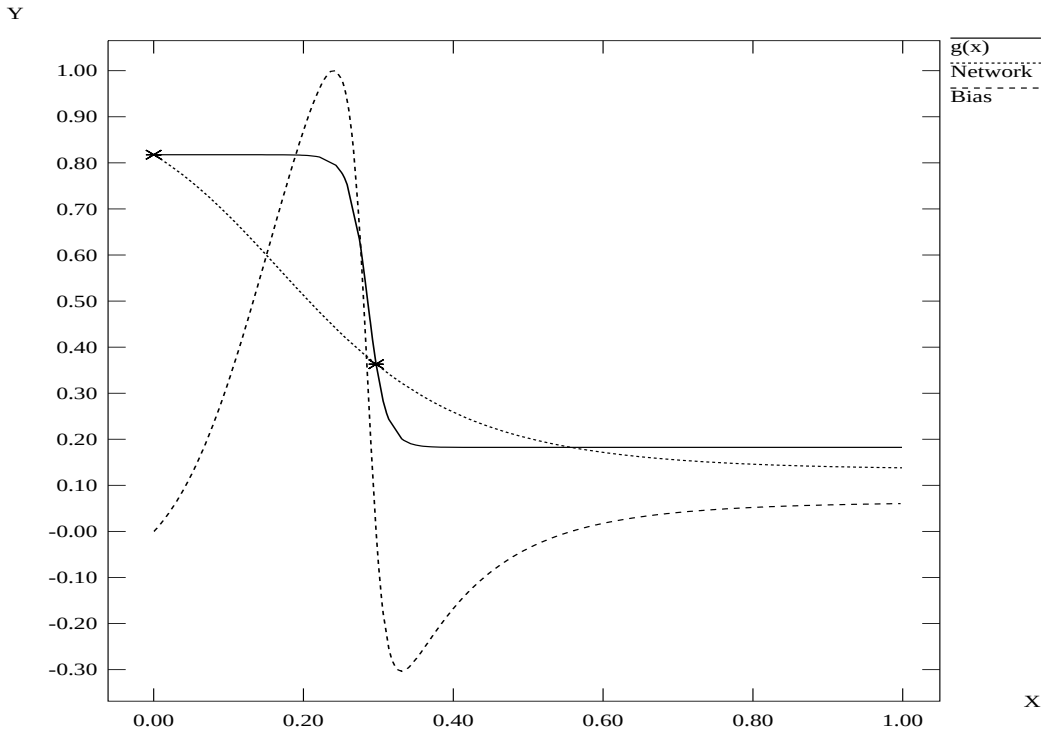

Figure 7: $x^2 = \{.295, .434\}$. Max ΔISB at $x_3 = .241$.

In Table 1, observe the wse under the column titled "New Set Error." This indicates the worst case squared error over the training set immediately after selecting a new example. For $n = 2$, note that the wse for the new training set is equal to the wse over the test set. From this we can tell that the new example was one for which network error was maximal. This occurrence is coincidental. While this is almost true as well for $n = 1$, it is certainly not true for $n = 3$ and $n = 4$. Hence, ΔISB need not coincide with the heuristic of training on examples according to maximal error. We expect this to be the case for the squared error learning rule we have assumed.

The experiment concludes in the close fit of $g(x)$ from 6 examples, as illustrated in Figure 11. Of note is the prevalence of examples near the steep portion of the curve, as compared with the sparsity of examples on the plateaus. The selection of x_4 and x_5 is interesting, if not counterintuitive. Training on x^4 results in the network function improving dramatically upon the lower plateau, even though no examples have been chosen from this region at the time. Even more counterintuitive is the choice of x_5 (as evidenced by the low New Set Error tabulated in Table 1 for $n = 4$.) However, the next example selected, x_6, results in the network mapping being fine tuned to within close tolerance of

the weight update appropriately for each particular weight. The benefit is that the gradient descent direction is improved by using the scaling information afforded by the diagonal terms of the Hessian matrix, without the bother of using the curvature information, which can be tricky to manage in practice. The inverse of the resulting matrix is easily computed, at low compuational cost. (See [Press, et.al., 1988] for a discussion of the related Levenberg-Marquardt technique.)

We began with a training set consisting of a single example. In order to select x_1, we calculated the average value of the outputs over the candidate examples, as discussed above. This value was .3788. We selected the example x for which $g(x)$ was closest to .3788. This turned out to be $x_1 = .297$, for which $g(x_1) = .363$. We initialized the network by randomizing the weights to values in $[-0.3, 0.3]$, and trained the network over x^1. The resulting network function was a constant function, approximately equal to $g(x_1)$.

In this case, $x_1 = .297$, where $g(.297) = .363$. See Figure 6. The initial training set is indicated in the figure at the point $(.297, .363)$, by a marker superimposed upon plot of the desired function, $g(x)$. After training the network on the initial training set x^1, obtaining w_1, we computed ΔISB for every example in x^N. ΔISB is normalized by its maximum absolute value, so that it may be plotted along with $f(x, w)$ and $g(x)$, as in Figure 6.

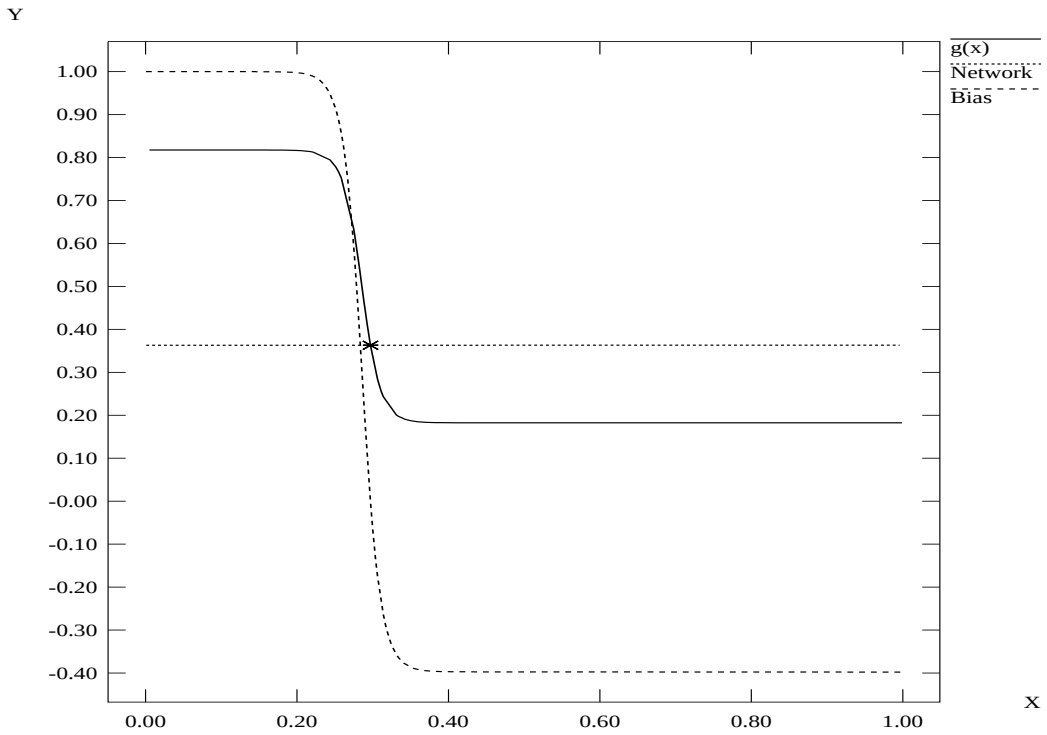

Figure 6: $x^1 = \{.295\}$. Max ΔISB at $x_2 = .001$.

Note the shape of the curve given by $\Delta ISB(x|x^1)$ indicates this criterion favors examples in the range $x \in (0.0, 0.3)$. Presumably this indicates that appending a single example in this range to the training set would result in better performance than would be obtained by appending an example in the range $x \in (0.4, 1.0)$, for which $\Delta ISB(x|x^1)$ is negative.

Recall that the next example selected by equation (14) is located by the example giving the maximum value of the Delta ISB curve. In this case, $x_2 = .001$, and the example generated at this location is appended to the initial training set x^1 to give the second training set x^2. Figure 7 shows the result of training on x^2, the two examples in x^2 being indicated, as before, by markers superimposed

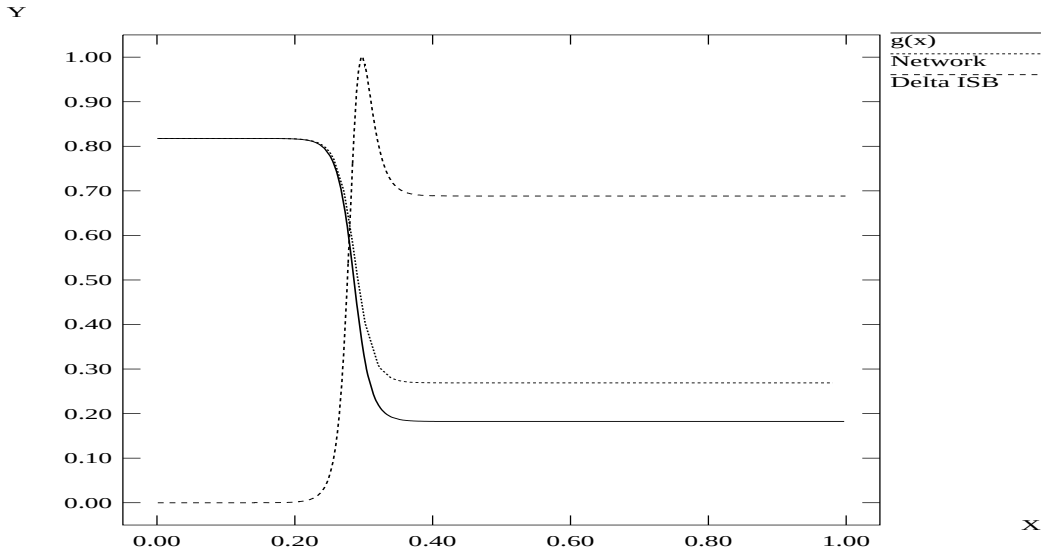

Figure 5: $w = (-20.0, 70.0, 1.5, -2.5)$.

Intuitively, these figures indicate that the criterion gives high value to candidate examples which lie in regions where more information is necessary, at least on this simple mapping. These examples are useful for illustrative purposes, but it is not obvious that the criterion will always act in accordance to our immediate intuition. In more general settings, the criterion may place high value on examples in regions where network error is substantially lower than the network error over other regions. This can occur by effect of μ which essentially weights each example according to the environmental probability law. While less obvious, it is also possible that this effect could be due solely to the form of the desired function, in conjunction with the representational flexibility of the network. This "flexibility" is determined by the class of functions the network is capable of representing. A particular example might plausibly lie at a sort of "pressure point", which, if learned along with the previously selected examples, brings the network function in line with the desired function in areas of the input space far away from the so-called pressure point. This is because (possibly local) changes in the network resulting from the attempt to fit a particular example can have a global impact upon the network function. We hope that the selection criterion will exploit regularities in the network flexibility in choosing new examples. These issues are explored experimentally in the following two sections.

6 Active Sampling During Learning

We investigate what sequence of examples will be chosen by the criterion while learning the mapping of Figure 1. [1] Examples will be chosen from a set of candidates, x^N. Here, $N = 500$. Between selection of successive examples, the network function is optimized over the training set, x^n, without reference to x^N. Subsequent to optimization over x^n, obtaining w_n, $\Delta ISB(x|x^n)$ is calculated for each $x \in x^N$, and x_{n+1} chosen from $\{x \in \max_x \Delta ISB(x|x^n)\}$. The training set is updated, obtaining $x^{n+1} = \{x^n, x_{n+1}\}$. We commence with optimizing $f(x, w)$ over x^{n+1} to obtain w_{n+1}, and so forth. As our selection criterion we use the implementation of equation (14) given by equation (13), with $\ddot{w}_{n+1}$ as defined in equation (12). We modified equation (12), using only the diagonal terms of $H(x^n, \hat{w}_n)$, setting the off-diagonal terms to zero. This in effect throws away local curvature information regarding the local neighborhood of $\hat{w}_n$, while retaining information necessary to scale

[1] Thanks to Timur Ash for his Dynamic Node Creation network simulator, which was easily modified to run these experiments.

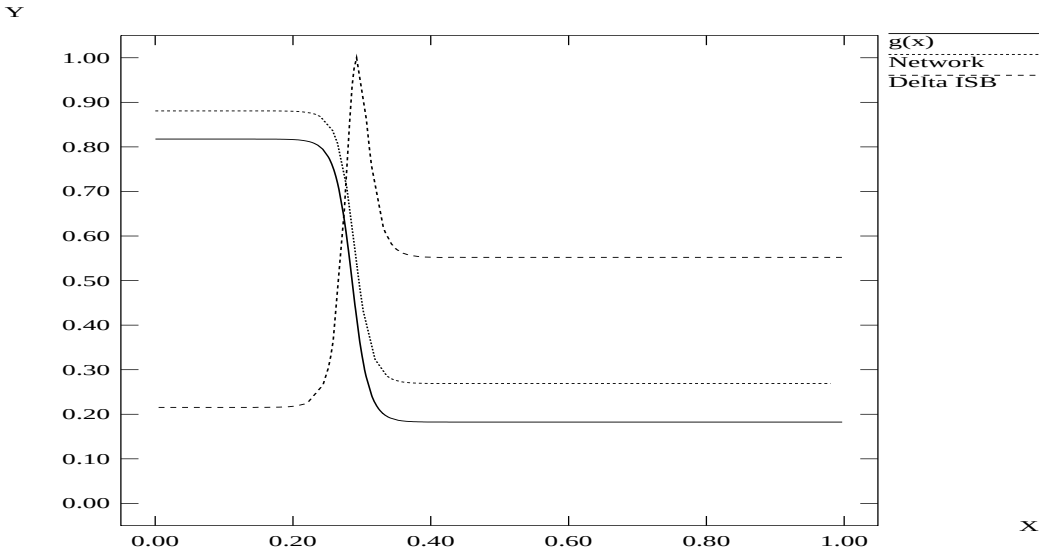

Figure 3: $g(x)$, along with ΔISB for a particular network function, $f(x, w)$.

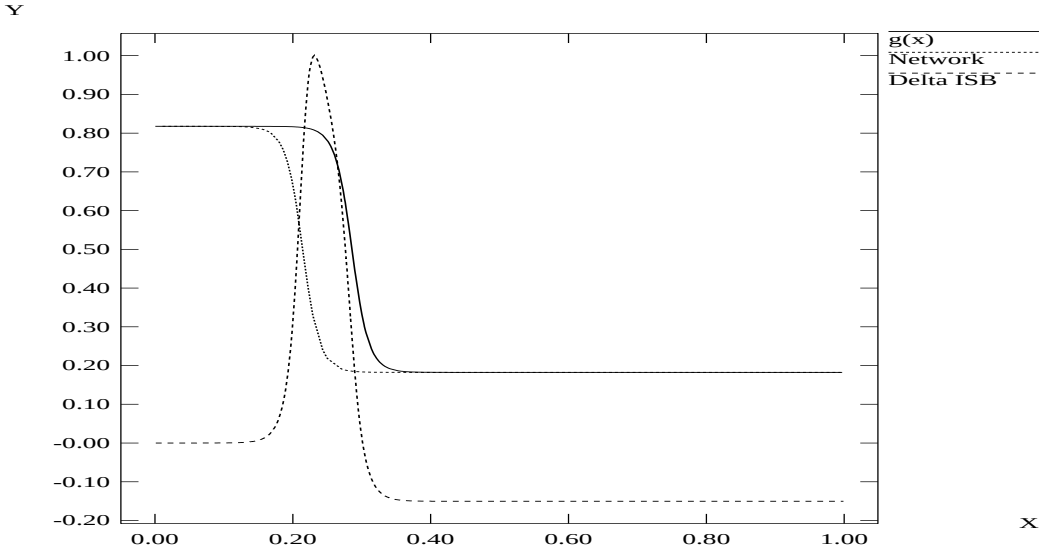

Figure 4: Another example, $w = (-15.0, 70.0, 1.5, -3.0)$.

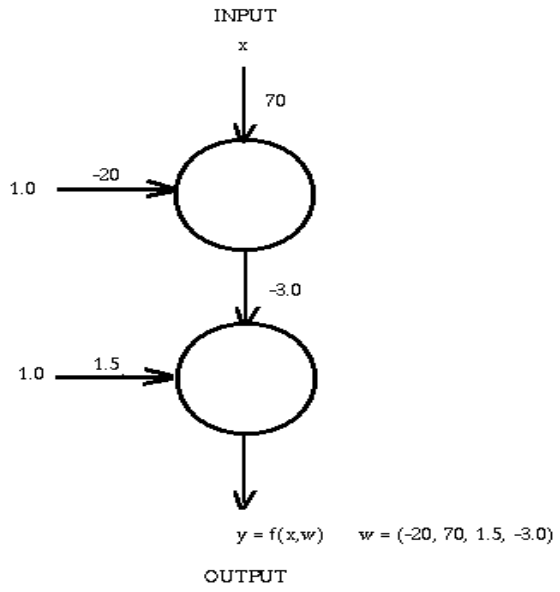

Figure 2: Network 1.

has the incidental benefit of illustrating the role each weight plays in the network mapping. We also have the benefit of knowing that the network can correct its mistakes by a single update to a single weight, and so in a loose sense w is in the vicinity of w^*. See Figure 3.

Here, weight w_3 was perturbed, by putting $w_3 = 1.9$. This results in an upward shifting of the entire mapping. The dashed line labelled "Delta ISB" gives the plot of ΔISB for the given network function, calculated over the set of candidates x^N, where here, $N = 500$. The candidate set was obtained by using the network illustrated in Figure 2 to generate 500 outputs in response to 500 inputs chosen uniformly at random over the input space. After being calculated for each of the 500 candidate examples, ΔISB is normalized to allow it to be plotted along with $f(x, w)$ and g, as illustrated in figure 3. Plotting ΔISB along with the $f(x, w)$ and $g(x)$ allows a comparison of the two functions, and provides a visualization as to how the ΔISB criterion grades the examples in x^N.

Figure 3 corroborates the intuition that the sampling criterion indicates the potential of a candidate example for reducing squared error over the input space. Although both plateaus are shifted upwards an equal distance, a training example chosen from the region of input space corresponding to the lower plateau is weighted more heavily than one chosen from the upper plateau. Presumably, that is because an example from the lower plateau might result in bringing the lower plateau of f back in line with g, and this would be better in comparison with choosing an example from the upper plateau, which covers a smaller region of the input space. To see where the new example would be selected by equation (14), observe the location on the x coordinate axis the location of the maximum value of the Delta ISB curve.

The next two figures provide more illustration of the behavior of the criterion. Compare the values of w in the figures with figure 2 to see which weight has been perturbed. Note that ΔISB can be negative, as in Figure 4.

premature selection of an example may result in a new example which adds little new information to the training set. As a deterministic stopping criterion for the selection criterion, consider the following:

- **Within feasible resource bounds, if $\int \nabla_w f(x, \hat{w}_n)(g(x) - f(x, \hat{w}_n))\mu(dx) \neq 0$ continue generating examples until the test set mse is sufficiently small.**

This condition quantifies the requirement that the possible gains to be made by selecting a new example be nonzero. The qualification acknowledges the case in which it is difficult to feasibly obtain this condition. Indeed, it may not be necessary in practice to meet this goal to obtain a set of training examples which allow the desired function to be learned sufficiently well.

5 Illustration of the technique

In this section we illustrate behavior of the ΔISB criterion upon a network function approximating a one dimensional instantiation of $g(x)$. See Figure 1. Here $g(x)$ is a function with a steep dropoff, with $g(x)$ at the upper and lower plateaus at about 0.2 and 0.8, respectively. The dropoff is shifted left of center, so that the lower plateau is more extensive than the upper plateau. For clarity of illustration, assume all inputs are equally likely.

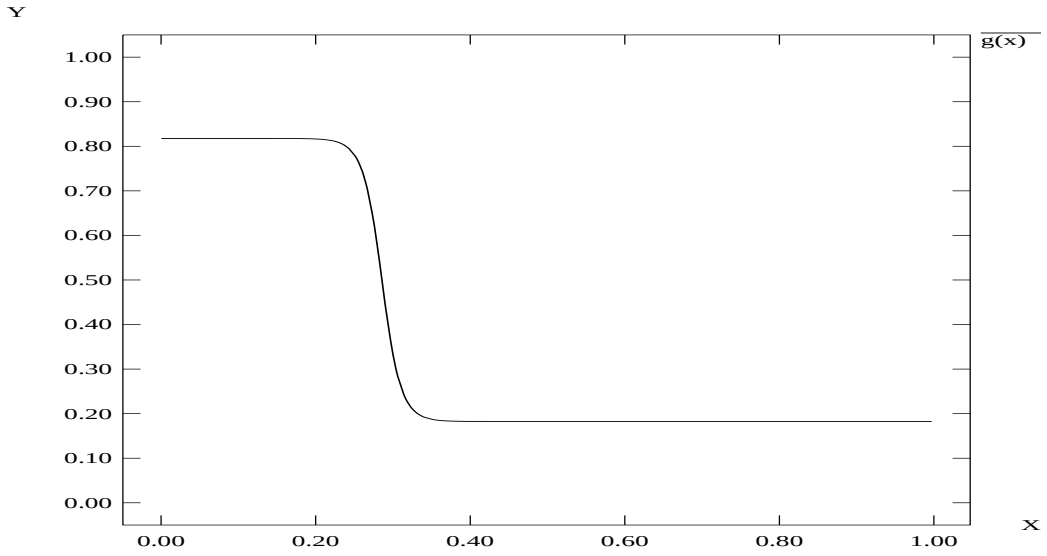

Figure 1: The desired function, $g(x)$.

We can realize g(x) exactly with a network function $f(x, w)$. See Figure 2. This diagram illustrates a feedforward network with 1 input, 1 output, and 1 hidden unit. Each unit employs for its transfer function the usual sigmoided output activation $o(net) = (1+exp(-net))^{-1}$, where net is the weighted sum of the inputs to the unit. This network implements the function of Figure 1 exactly. The arcs are labelled by real values, denoting the weights. The values of the weights in this figure give a set of weights for which $f(x, w) = g(x)$. This set of weights is $w^* = (-20.0, 70.0, 1.5, -3.0)$. Note the size of weights w_1 and w_2 relative to w_3 and w_4. This is a manifestation of the region in g of high first derivative, which the network function fits by employing large magnitude weights on the hidden unit.

5.1 Illustrating ISB on a simple mapping

The first three illustrations of the method are obtained by applying the criterion to a network function implementing a perturbation of the true mapping. We perturb one weight at at time. This

the number of hidden units in the neural network, then, given any set of $n \leq h$ examples, there exist weights w_n such that $\sum_{i=1}^{n}(g(x_i) - f(x_i, w_n))^2 = 0$. (Theorem 2.5, [Hornik, et.al., 1989].) Thus, in this case we know in advance that a set of weights exists that can fit the training examples exactly.

We also found it informative to compare the performance of the network over the training set with performance of the network over the set of available examples. For examples, see Tables 1 and 2, which catalog the performance statistics of experiments to be discussed in more detail below. Here, we consider the training set to be a subset of the set of all available examples, which we refer to for simplicity as the "test set." In the case that we desire the test set and training set to be disjoint sets, we may easily recalculate the relevant statistics from the information given in the tables. The mse (mean squared error) gives the average squared error over the set of examples indicated in the column heading. The wse (worst squared error) gives the maximal squared error for any particular example in the indicated training set. By comparing the mse over the training set (tabulated under "Training Error mse") with the mse over the set of all available examples (under "Test Set Error mse") we can surmise whether the training examples are becoming representative of the set of all available examples. In practice, network error over the training set tends to increase as new examples are added; network error over the test set will typically decrease accordingly.

In general, stringent optimization should be employed to extract as much information as possible from the training examples before selecting a new example. As stated above, we observed that if the training examples are not fit reasonably well, the selection criterion will tend to choose examples close to previously selected examples. Because the training data is noiseless, and since the training examples are, by hypothesis, selected to maximize the decrement in squared error over the set of all available examples, we recommend that training be continued upon the current set of training examples so long as a better fit is possible. This recommendation is supported by our empirical experience with the criterion. In practice, determining whether or not a better fit is possible can be quite difficult, and so determining when to select a new example can become a matter of judgement. We found a naive indicator useful in practice. When the average network error over the the training set was greater than the average network error over the test set, we found this to be an indication to continue with training upon the current training set. Typically, as new examples are selected, the training set becomes more and more difficult to fit, as evidenced by increasing resources required to train the network to a given level of accuracy. When training set mse continues to exceed test set mse after lengthy optimization, we interpreted this as meaning that the capability of the given learning rule in searching through the function space available to the current network was being pushed to feasible limits, assuming that a set of weights exists which provide a fit reducing the training set mse to less than the test set mse. The case in which the network complexity needs to regulated as the number of examples increases is indeed relevant here, but is outside the scope of this paper.

Summarizing the discussion so far, we recommend the following guidelines for regulating the selection criterion:

1 **While $\nabla_w f(x^n, \hat{w}_n)(g(x^n) - f(x^n, \hat{w}_n)) \neq 0$, continue training on the current training set. Otherwise, follow recommendation 2.**

2 **If n is less than or equal to the number of hidden units, continue training until network error over x^n is zero. Otherwise, follow recommendation 3.**

3 **If the training set mse is greater than the test set mse, continue training. Otherwise, follow recommendation 4.**

4 **Select a new training example.**

Note that recommendation 3 may effectively serve as a stopping criterion for learning in case it proves practically infeasible to continue the search for a better fit. Intuitively, it may be premature to select a new training example if information in the current training set has yet to be fully exploited. Recall also the recommendation we made previously: if the new example is close to a previously selected example, discard the new example and continue training upon the previous training set. Therefore,

Although we have arrived at this criterion by making use of approximations valid for large n, this criterion has an appealing interpretation apart from large sample considerations. Specifically, ΔISB can be interpreted directly as the square of the weighted Euclidean distance between the gradient of squared error with respect to network weights integrated over all candidate examples, and the gradient of squared error with respect to network weights for a single candidate example. (The matrix H^{-1} performs the weighting.) Thus, picking x_{n+1} to maximize ΔISB can be viewed as picking the single example having individual error gradient most highly correlated with error gradient of the entire set of examples. Learning with this example is therefore likely to be especially informative, relative to other available examples. The ΔISB criterion thus possesses heuristic appeal in training sets of any size. As n becomes large, the derivation above establishes that ΔISB approaches $ISB(x^n) - ISB(\check{x}_{n+1})$, providing further support for its use.

4 Implementation

We now consider the practical use of the ΔISB criterion (13) for evaluating examples for training a one-hidden layer feedforward network. We have available a set of N examples from which to select training examples. Call this set x^N, the set of candidate examples. Suppose we have trained the network using a subset of n examples x^n. Given x^n, we evaluate $\Delta ISB(\check{x}_{n+1}|x^n)$ for all $\check{x}_{n+1} \in x^N$. We realize the integral of (13) by a discrete summation over x^N. Observing equation (13), we note that the summation need be performed only once. The result of the summation is a vector of length p, where p is the number of weights in the network. Thereafter, each candidate example $\check{x}_{n+1}$ is evaluated by taking the inner product of the vector obtained from the discrete summation with the length p vector obtained for $\Delta\check{w}_{n+1}$. For simplicity, we begin by putting $n = 1$ and find x_1 such that $x^1 = \{x_1\} = \arg\min_{x^1} ISB(x^1)$. This is computationally appealing, since fitting a single example by a neural network can be accomplished efficiently. Given a priori knowledge about the class of functions that we can expect the network learning rule to utilize to fit a single example, it may be possible to calculate the optimal location of x_1 by employing summary statistics over the set of available examples. For instance, from extensive experience with network learning of a single example upon one dimensional mappings, we observed that (starting from weights initialized to small randomized values) the backpropagation learning rule will fit a single example with a constant function (see Figures 6 and 13.) Given such a reliable tendency of the learning rule, it is straightforward to choose the single example minimizing $ISB(x^1)$. Calculate $m = \frac{1}{N}\sum_{i=1}^{N} g(x_i)$, $x_i \in x^N$, and select $x_1 = \arg\min_{x \in x^N}(g(x) - m)^2$.

Given x_1, we proceed to grow the training set by successive optimization of ΔISB. Precisely, x_2 is chosen such that $x_2 = \arg\max_{x \in x^N} \Delta ISB(x|x^1)$. For $n = 1, 2, \ldots$, we choose

$$x_{n+1} = \max_{x} \Delta ISB(x|x^n). \tag{15}$$

As an aside, we emphasize that this is only one possible method of utilizing the ΔISB information. We experimented briefly with an iterative training regime, in which little optimization was performed between successive examples. There was a tendency for the criterion to sample repetitively near previously sampled points when the optimization between successive examples was not stringent. In some such cases a pathological condition emerged in which the technique entered a cycle, endlessly alternating between two examples. For this reason, we advocate stringent optimization at each step.

Returning to the original discussion, we now address the issue of the determining when to select a new example. Clearly, the first heuristic we can rely upon is to continue training so long as network error is dropping over the training set. And, guided by the theory, we expect that the selection criterion will work best if the network function gives a good fit over the training set. Although we may not know in general when an estimate is sufficiently close to a globally optimal solution, in particular circumstances we may employ capacity results for neural networks to determine whether this is the case. (cf., Vapnik-Chervonenkis dimension [Abu-Mostafa, 1989], [Baum and Haussler, 1989].) For example, assume that each unit employs the usual sigmoid squashing function. If h is

To calculate (8), we need to estimate $(\hat{w}_n - \check{w}_{n+1})$, given $\check{x}_{n+1}$. When n is large, it is generally true that $\check{w}_{n+1}$ is in a neighborhood of $\hat{w}_n$. The particular weight update we assume satisfies this requirement. For example, we may use the Gauss-Newton weight update [Seber and Wild, 1989, p.21]:

$$\hat{w}_n - \check{w}_{n+1} \approx -\left[\frac{1}{n+1}H(\check{x}^{n+1}, \hat{w}_n)\right]^{-1} \frac{1}{n+1}\nabla_w f(\check{x}^{n+1}, \hat{w}_n)(g(\check{x}^{n+1}) - f(\check{x}^{n+1}, \hat{w}_n)). \tag{9}$$

Here,

- $f(\check{x}^{n+1}, \hat{w}_n) = (f(x_1, \hat{w}_n), f(x_2, \hat{w}_n), \ldots, f(x_n, \hat{w}_n), f(\check{x}_{n+1}, \hat{w}_n))'$.
- $\nabla_w f(\check{x}^{n+1}, \hat{w}_n) = (\nabla_w f(x_1, \hat{w}_n), \ldots, \nabla_w f(x_n, \hat{w}_n), \nabla_w f(\check{x}_{n+1}, \hat{w}_n))$, a matrix with p rows and $n+1$ columns.
- $H(\check{x}^{n+1}, \hat{w}_n)^{-1} = (\nabla_w f(\check{x}^{n+1}, \hat{w}_n)\nabla_w f(\check{x}^{n+1}, \hat{w}_n)')^{-1}$ acts as an approximation to the so-called Hessian matrix of second derivative information. [Fedorov, 1972, p.35], [Press et.al., 1988, p.542], [Seber and Wild, 1989, pp.21-23]

The estimate of $\hat{w}_n - \check{w}_{n+1}$, equation (9), can be simplified by noting that for $i = 1, 2, \ldots, n$, $\nabla_w f(x_i, \hat{w}_n)(g(x_i) - f(x_i, \hat{w}_n)) = 0$. This is by definition of $\hat{w}_n$, which is optimal for x^n. (Note that this remains a valid assumption when $\hat{w}_n$ is approximated by a locally optimal estimate $\tilde{w}_n$ obtained by gradient descent.) As a result,

$$\nabla_w f(\check{x}^{n+1}, \hat{w}_n)(g(\check{x}^{n+1}) - f(\check{x}^{n+1}, \hat{w}_n)) = \nabla_w f(\check{x}_{n+1}, \hat{w}_n)(g(\check{x}_{n+1}) - f(\check{x}_{n+1}, \hat{w}_n)).$$

We also employ the approximation

$$\left[\frac{1}{n}H(x^n, \hat{w}_n)\right]^{-1} \approx \left[\frac{1}{n+1}H(\check{x}^{n+1}, \check{w}_{n+1})\right]^{-1}.$$

Collecting the above approximations together, define our estimate of $\hat{w}_n - \check{w}_{n+1}$ to be

$$\Delta\check{w}_{n+1} \equiv -\left[\frac{1}{n}H(x^n, \hat{w}_n)\right]^{-1} \frac{1}{n+1}\nabla_w f(\check{x}_{n+1}, \hat{w}_n)\,(g(\check{x}_{n+1}) - f(\check{x}_{n+1}, \hat{w}_n)), \tag{10}$$

giving as one implementation of (8),

$$\beta^2(\hat{w}_n) - \beta^2(\check{w}_{n+1}) \approx 2\Delta\check{w}_{n+1}{}' \int \nabla_w f(x, \hat{w}_n)(g(x) - f(x, \hat{w}_n))\mu(dx). \tag{11}$$

For use in comparing candidate examples with each other, note that the constant factors of equation (10) may be eliminated, giving

$$\Delta\check{w}_{n+1} = [H(x^n, \hat{w}_n)]^{-1}\nabla_w f(\check{x}_{n+1}, \hat{w}_n)\,(g(\check{x}_{n+1}) - f(\check{x}_{n+1}, \hat{w}_n)). \tag{12}$$

Ignoring the constant factor in (11) and combining with (12) leads to the criterion that we propose for selecting the next training example,

$$\Delta ISB(\check{x}_{n+1}|x^n) = (g(\check{x}_{n+1}) - f(\check{x}_{n+1}, \hat{w}_n))'\nabla_w f(\check{x}_{n+1}, \hat{w}_n)'[H(x^n, \hat{w}_n)]^{-1} \int \nabla_w f(x, \hat{w}_n)(g(x) - f(x, \hat{w}_n))\mu(dx). \tag{13}$$

The training example x_{n+1} is selected so that

$$x_{n+1} \in \arg\max_{\check{x}_{n+1}} \Delta ISB(\check{x}_{n+1}|x^n). \tag{14}$$

$$
\begin{aligned}
IMSE(x^n) &= \int B(x|x^n)^2 \mu(dx) \\
&= \int (g(x) - f(x, \hat{w}_n))^2 \mu(dx). \qquad (5)
\end{aligned}
$$

Note that $\varepsilon = 0$ is the limit of the case in which the noise is non-zero but measurement at the sample points is repeated ad infinitum, and the measurement averaged.

3.4 Minimizing ISB for nonlinear least squares regression

For a sample of size n, the optimal training set satisfies $x_*^n = \arg\min_{x^n} ISB(x^n)$. Finding such a training set is computationally impractical. Here we pursue a more modest goal. Given a training set x^n, we shall seek x_{n+1} that maximizes the decrement in ISB, $ISB(x^n) - ISB(x^{n+1}), x^{n+1} = (x^n, x_{n+1})$. Note that this property for x_{n+1} is a necessary condition for solving the computationally impractical global problem, but will not necessarily deliver the globally optimal solution. Nevertheless, this approach permits a computationally feasible and attractive method for sequential selection of training examples.

Let $\check{x}_{n+1}$ be a candidate input, and $\check{x}^{n+1} = \{x^n, \check{x}_{n+1}\}$. Let $\lambda_n(x^n) = l_n(x^n, g(x^n))$, (where $g(x^n)$ is as defined in the previous section), so that $\hat{w}_n = \lambda_n(x^n)$, and put $\check{w}_{n+1} = \lambda_{n+1}(\check{x}^{n+1})$. Thus we seek to maximize

$$
\begin{aligned}
ISB(x^n) - ISB(\check{x}^{n+1}) &= \beta^2(\lambda_n(x^n)) - \beta^2(\lambda_{n+1}(\check{x}^{n+1})) \\
&= \beta^2(\hat{w}_n) - \beta^2(\check{w}_{n+1}), \qquad (6)
\end{aligned}
$$

where $\beta^2(w) = \int [g(x) - f(x, w)]^2 \mu(dx)$. (The variable λ is introduced to make explicit the role of x^n in the noiseless environment; the variable β^2 is introduced to allow a change of argument emphasizing the role of the network weights.) We could pick x_{n+1} to maximize expression (6) directly, albeit at great expense. Fortunately, a much simpler procedure can be obtained by exploiting certain properties of the learning algorithm.

To proceed, we take a first order Taylor expansion of $\beta^2(\check{w}_{n+1})$ around $\beta^2(\hat{w}_n)$, yielding

$$
\beta^2(\hat{w}_n) - \beta^2(\check{w}_{n+1}) \approx (\hat{w}_n - \check{w}_{n+1})' \nabla_w \int (g(x) - f(x, \hat{w}_n))^2 \mu(dx).
$$

By Bartle [1966, Corollary 5.9] the gradient and integral can be interchanged to yield

$$
\nabla_w \int (g(x) - f(x, \hat{w}_n))^2 \mu(dx) = -2 \int \nabla_w f(x, \hat{w}_n) \, (g(x) - f(x, \hat{w}_n)) \mu(dx), \qquad (7)
$$

provided in addition to the compactness of W that $\partial f / \partial w$ exists on $X \times W$, and that $f(x, \cdot)$ and the elements of $\nabla_w f(x, \cdot)$ are dominated by square integrable functions of x only. We have already assumed implicitly that g is square integrable. Consequently,

$$
\beta^2(\hat{w}_n) - \beta^2(\check{w}_{n+1}) \approx -2(\hat{w}_n - \check{w}_{n+1})' \int \nabla_w f(x, \hat{w}_n)(g(x) - f(x, \hat{w}_n)) \mu(dx). \qquad (8)
$$

Equation (8) provides a computationally simpler estimate of equation (6). For each successive selection of an example from a set of candidates, the integrand of equation (8) need be computed once, and this can be approximated by summing over the set of candidate examples. Then, to evaluate each individual example in the set of candidates, we compute $(\hat{w}_n - \check{w}_{n+1})$, and take its inner product with the vector quantity obtained from the summation over the set of candidates.

3.2 Partitioning IMSE

Given x^n, take $E[f(x,\hat{w}_n)|\,x^n]$ as the expected output of the network at input x:

$$E\left[f(x,\hat{w}_n)|\,x^n\right] = \int f(x,\, l_n\,(x^n,y^n))\;\gamma^n(dy^n|x^n)$$

We can partition MSE as follows:

$$\begin{aligned}
M(x|x^n) &= E\left[(g(x)-f(x,\hat{w}_n))^2|\,x^n\right] \\
&= E\left[(g(x)-E[f(x,\hat{w}_n)|\,x^n]+E[f(x,\hat{w}_n)|\,x^n]-f(x,\hat{w}_n))^2|\,x^n\right] \\
&= (g(x)-E[f(x,\hat{w}_n)|\,x^n])^2 \\
&\quad + E\left[(E[f(x,\hat{w}_n)|\,x^n]-f(x,\hat{w}_n))^2|\,x^n\right] \\
&\quad +2\,E\left[(g(x)-E[f(x,\hat{w}_n)|\,x^n])\,(E[f(x,\hat{w}_n)|\,x^n]-f(x,\hat{w}_n))\,|\,x^n\right] \\
&= (g(x)-E[f(x,\hat{w}_n)|\,x^n])^2 + E\left[(E[f(x,\hat{w}_n)|\,x^n]-f(x,\hat{w}_n))^2|\,x^n\right] \\
&= B(x|x^n)^2 + V(x|x^n). \qquad (3)
\end{aligned}$$

The term $B(x|x^n) = g(x) - E[f(x,\hat{w}_n)|\,x^n]$ quantifies the *bias* of the estimator $\hat{w}_n$. $E[f(x,\hat{w}_n)|\,x^n] - f(x,\hat{w}_n)$ is the *sampling variation*. We have $B(x|x^n)^2 = (g(x)-E[f(x,\hat{w}_n)|\,x^n])^2$, the *Squared Bias*, and $V(x|x^n) = E[(E[f(x,\hat{w}_n)|\,x^n] - f(x,\hat{w}_n))^2|\,x^n]$, the *Network Variance*. Note that the covariance term vanishes because $B(x|x^n)$ is nonstochastic. Hence

$$\begin{aligned}
IMSE(x^n) &= \int \left[B(x|x^n)^2 + V(x|x^n)\right]\;\mu(dx). \\
&= ISB(x^n) + IV(x^n), \qquad (4)
\end{aligned}$$

where we call ISB the Integrated Squared Bias, and IV the Integrated Variance.

3.3 Integrated Squared Bias

Three cases arise in minimizing IMSE:

1 *Neither $ISB(x^n)$ nor $IV(x^n)$ is constant.*

2 *$ISB(x^n)$ is constant.*

3 *$IV(x^n)$ is constant.*

Case 2 has been studied mathematically in much depth, and methods derived from this approach can be applied to network learning; however, this approach requires quite stringent conditions. [Fedorov, 1972], [Ash and Hedayat, 1978], [Box and Draper, 1987], [Khuri and Cornell, 1987]. Our ulimate interest is in case 1. For simplicity, however, in this paper we investigate case 3, in which IMSE is minimized by minimizing the bias term. To do so requires conditions on $V(x|x^n)$ ensuring that decreases in $B(x|x^n)^2$ do not affect $V(x|x^n)$.

A particularly amenable special case occurs when $E[f(x,\hat{w}_n)|\,x^n] = f(x,\hat{w}_n)$, so that $V(x|x^n) = 0$. Consider $\varepsilon \equiv 0$. In this case, for any x^n the training sample becomes $z^n = (x^n, g(x^n))$, where we abuse notation by writing $g(x^n) = (g(x_1), g(x_2), \ldots, g(x_n))'$. In this case, y^n is completely determined by x^n, as is $\hat{w}_n$. Therefore, in a noiseless environment, $E\left[f(x,\hat{w}_n)|\,x^n\right] = f(x,\hat{w}_n)$, and the variance term disappears. Consequently, equation (4) becomes

$\nu(A \times \Re^r \cap \Re \times B) = \gamma(A|B)\mu(B)$. Knowledge of γ completely describes the relationship between X and Y, in particular, the conditional expectation of Y given X, denoted $E(Y|X)$.

We consider the task of learning $E(Y|X)$. We know that whenever $E(Y|X)$ exists, there exists a function g such that $g(X) = E(Y|X)$ (Billingsley, 1986, Theorem 20.1). Let the random expectational error $\varepsilon \equiv Y - E(Y|X)$. Then, $Y = g(X) + \varepsilon$. As a consequence, ε has mean zero.

Let f denote the output function of a neural network parameterized by weights w, where w is an element of a weight space $W \subset \Re^p$ (we assume W compact). Then, for some particular input vector x, and weight vector w, the network output is $f(x, w)$.

We will measure network performance using squared error. Our objective is to find a set of weights $w^* \in \arg\min_w \int (y - f(x,w))^2 \nu(dy, dx)$. We alert the reader that randomness will be eventually dropped, so that $y = g(x)$. We define our general approach for the case of noisy data, considering the special case of noiseless data when appropriate.

3 Derivation of the Method

3.1 Integrated Mean Squared Error

We consider a (measurable) nonlinear least-squares estimator

$$\hat{w}_n \in \arg\min_w \frac{1}{n} \sum_{i=1}^{n} (Y_i - f(X_i, w))^2.$$

Given appropriate conditions, $\hat{w}_n$ displays a strong form of stochastic convergence to optimal weights w^*, in that $\hat{w}_n \to w^*$ a.s.-P. (White, 1989, Theorem 1). One of these conditions dictates that $\{Z_i\}$ be independent and identically distributed (i.i.d.) random variables. Here, we violate this condition. Given $z^n = (z_i, i = 1, \ldots, n)$, we choose x_{n+1} according to some specific criterion that will depend upon z^n. Therefore, the z_i are obtained from random variables that are neither independent, nor identically distributed. Nevertheless, we expect that if $\{x_i\}$ is properly chosen, and if $\{Y_i\}$ is appropriately behaved, than a similar convergence result can be established.

A criterion for selecting training examples should work well in conjunction with the criterion used for learning (here, squared error.) We take as a suitable choice of criterion the Integrated Mean Squared Error (IMSE) [Box and Draper, 1987], [Khuri and Cornell, 1987]. Define l_n as the learning rule giving the realization w_n of $\hat{w}_n$ as a function of the realizations x^n, y^n:

$$w_n = l_n(x^n, y^n) \in \arg\min_w \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2.$$

For given x^n, the mean squared error (MSE) of the network output at input x is [Khuri and Cornell, 1987, p. 209]

$$\begin{aligned} M(x|x^n) &= \int (g(x) - f(x, l_n(x^n, y^n)))^2 \gamma^n(dy^n|x^n) \\ &= E[(g(x) - f(x, \hat{w}_n))^2 | X^n = x^n], \end{aligned} \tag{1}$$

where $\gamma^n(\cdot\,|\,x^n)$ is the conditional probability of Y^n given $X^n = x^n$. For convenience in what follows, we write $E(\cdot\,|\,x^n)$ in place of $E(\cdot\,|\,X^n = x^n)$, and we assume implicitly that the indicated expectations exist and are finite. Integrating MSE over the input space gives IMSE:

$$IMSE(x^n) = \int E[(g(x) - f(x, \hat{w}_n))^2 | x^n] \mu(dx), \tag{2}$$

where μ is an appropriately chosen measure on $(\mathbf{X}, \mathcal{B}(\mathbf{X}))$.

to [Fedorov, 1972], [Ash and Hedayat, 1978], [Box and Draper, 1987], [Khuri and Cornell, 1987], [Cooray-Wijesinha and Khuri, 1987], [Myers et.al., 1989], [Seber and Wild, 1989].

1.4 Results

The IMSE criterion developed below is intuitively pleasing and straightforward to compute. The approach is to sequentially add examples to the training set, training the network in between selection of successive examples. Heuristically, we select a training example maximizing the decrement in IMSE (equation (6) below) that would result from adding the new example to the training set and training upon the resulting training set. Preliminary experience with the technique indicates that for best results a stringent optimization should be performed between selection of successive examples.

In our experience with the sampling criterion on simple mappings, it provided a reasonable evaluation of prospective training examples. The selection criterion reliably selected training sets that allowed the learning rule to converge upon the desired function after training on only a small percentage of the total number of available examples. The selection criterion performed consistently over the course of several experiments, and so we expect these results to be repeatable.

The empirical results obtained from these one-dimensional mappings do well to corroborate the theory and demonstrate the behavior of the technique, but alone are insufficient to recommend this technique over random sampling for general purpose use. However, the criterion does show promise for use in some practical settings. Based upon the satisfying behavior of the criterion upon simple mappings, we do recommend the technique be tested for use with data-driven learning techniques in more challenging problem domains (for examples, see [Platt,1990], [White, 1990].) The technique should also be useful for reducing the training set size in domains where it is possible to average out noise by repetitive sampling.

2 Definition of the Learning Task

We consider the learning task of determining the relationship between two (possibly random) vectors X and Y, where X takes values in $\mathbf{X} \subset \Re^r$, and Y takes values in $\mathbf{Y} \subset \Re^s$, for integers r and s. We refer to $\mathbf{X}$ as the *input space*. For concreteness we take X to represent a vector of input variables and Y a vector of target variables. For simplicity, let $s = 1$. The learning task is thus one of training a neural network with r inputs and one output.

We adhere to the framework provided by White (1989). We draw n observations on X, $x^n = (x_1, \ldots, x_n)$, along with n corresponding observations on Y, $y^n = (y_1, \ldots, y_n)$. For convenience let $Z = (X', Y')'$, and $z_i = (x_i', y_i')'$, where the prime denotes transpose. We denote our *sample*, or, *training set*, as $z^n = (z_1, \ldots, z_n)$. We suppose that the measurement process could be continued indefinitely, to obtain a sequence of observations $\{z_i\} = (z_i,\ i = 1,2,\ldots)$, as a realization of the random sequence $\{Z_i = (X_i', Y_i')'\}$, where X_i and Y_i have the distributions of X and Y respectively for all $i = 1,2,\ldots$.

In our simulations we will have complete control in determining x_i. We define a (fixed) "environmental" probability law μ that describes the relative frequencies with which different values for x_i are set; $\mu : \mathcal{B}(\Re^r) \mapsto [0,1]$. $\mathcal{B}(\Re^r)$ is the Borel σ-algebra generated by the open sets of $\Re^r$.

If y_i can be measured with perfect precision, and there is no inherent uncertainty, we model the relationship between X and Y as an exact functional relationship between x_i and y_i, such that $y_i = g(x_i)$, for some mapping $g : \Re^r \mapsto \Re$. Otherwise, if the measurement of y_i is subject to inherent uncertainty or measurement errors, we express a *probabilistic* relationship between x_i and y_i. In this case we view x_i and y_i as a realization of jointly distributed random variables X_i and Y_i, and write $Z_i = (X_i', Y_i')'$. Assuming X_i and Y_i have the same joint probability law as X and Y, we drop the subscript i when convenient. The joint behavior of X and Y is described by a joint probability law ν. We may decompose ν into a environmental probability law μ and a conditional probability law γ that describes the relative frequency of occurrence of Y given X. For $A \in \mathcal{B}(\Re), B \in \mathcal{B}(\Re^r)$,

1.2 Approach

Heuristically, we wish to select observations that in some sense provide high information density. However, this choice should also take into account the current state of the network, so that selected examples are not only informative with respect to the desired function, but useful to the network given its ability to learn from the examples. Where random sampling requires a large number of examples to provide a sufficient density of observations where they are most necessary, we hope that active selection will determine a sampling distribution that reduces the data requirement.

There are two possibly conflicting objectives we wish to attain. The first is to maximize accuracy of fit, that is, to reduce the bias (misspecification) of the network model. The second objective is to minimize unreliability in the presence of noise (network variance.) We make these notions precise below. First, we choose a general sampling criterion quantifying both these considerations. Then, for the purposes of this prefatorial investigation into the utility of this general sampling criterion, we consider the special case in which the data is noiseless. While the result we obtain here considers the noiseless learning task exclusively, we choose the general sampling criterion in anticipation of extending this work to noisy data. Considering a special case of this sampling criterion, we derive a selection criterion for a class of feedforward network architectures applicable to the case of noiseless data, and illustrate its behavior upon one-dimensional mappings. We then present a method for utilizing our criterion during learning. Finally, to examine the effect of the selection criterion upon the sampling distribution for a training set of a particular size, we provide a comparison of the network performance obtained by using a training set obtained by the selection criterion with typical results obtained when examples are selected randomly.

1.3 Previous Work

Active sampling is referred to in various forms throughout the literature. In a loose sense, we can think of a training set obtained in this way as analogous to a set of "exemplars," similar to the notion of "extreme points" employed by Cover for classification tasks [Cover, 1965]. In computational learning theory, deduction programs can make queries to an oracle [Valiant, 1984] [Gasarch and Pleszkoch, 1989] [Hellerstein and Karpinski, 1989] [Kurtz and Smith, 1989]. In the connectionist literature, Ahmad and Omohundro consider a related approach in which a network is trained by *selective attention.* Hwang et.al. present a query-based approach for neural network learning of classification tasks, that generates training examples in the vicinity of classification boundaries [Hwang, et.al., 1990]. In contrast, our approach selects a new training example by estimating its potential utility in comparison with other available (candidate) examples. In this sense, the method we present is not a query-based approach. That is, the input space is not actively sampled by the learning device as in a query-based approach; rather, available examples are evaluated for their potential worth. Our approach is similar to that of Hwang et.al. in that examples are chosen according to information derived from a partially trained network. However, we consider a more general learning task, viz., estimating an unknown functional mapping.

The topic of selecting training examples for the purpose of estimating an unknown mapping has been studied extensively in the statistics and econometrics literature [Müller, 1984], [Faraway, 1990], [Box and Draper, 1987], [Khuri and Cornell, 1987]. If it is known that the function to be learned is reasonably smooth, a selection criterion exists for choosing examples for a kernel-based smoothing approach [Faraway, 1990]. Faraway uses the IMSE criterion (defined below) to select training examples sequentially for a nonparametric regression. We adopt this general technique, and adapt it to training multilayer feedforward networks. Although our criterion is in fact a computationally convenient approximation to the IMSE, we shall for simplicity refer to it also as IMSE. We do not make any a priori assumptions regarding smoothness of the mapping. The method presented here is nonparametric in the sense that the complexity of the network need not be pre-specified. Parameterization of the network function is necessary to evaluate training examples, but modifying the number of network parameters (weights) between successive training examples is allowed.

We draw upon theory of *response surface methodology*, *optimal experiments* and *sequential design* in working with the IMSE sampling criterion. For review, the interested reader may refer

Active selection of training examples for network learning in noiseless environments.

Mark Plutowski
Department of Computer Science and Engineering
University of California, San Diego

Halbert White
Institute for Neural Computation and Department of Economics
University of California, San Diego

February 1991
Technical Report No. CS91-180

Abstract

We derive a method for *actively selecting* examples to be used in estimating an unknown mapping with a multilayer feedforward network architecture. Active selection chooses from among a set of available examples an example which, when added to the previous set of training examples and learned, maximizes the decrement of network error over the input space. In practice, this amounts to incrementally growing the training set as necessary to achieve the desired level of accuracy.

The objective is to minimize the data requirement of learning. Towards this end, we choose a general criterion for selecting training examples that works well in conjunction with the criterion used for learning, here, least squares. Examples are chosen to minimize *Integrated Mean Square Error* (IMSE). IMSE embodies the effects of bias (misspecification of the network model) and variance (sampling variation due to noise). We consider a special case of IMSE, *Integrated Squared Bias* (ISB) to derive a selection criterion (ΔISB) which we maximize to select new training examples. ΔISB is applicable whenever sampling variation due to noise can be ignored. We conclude with graphical illustrations of the method, and demonstrate its use during network training.

1 Introduction and Overview

1.1 Statement of the Problem

The task is to train multilayer networks to estimate functional mappings, by using a learning rule to set the network weights, given a stream of training examples. Rather than passively accepting examples from some data generating process, we utilize information derived from a network trained on previous examples to actively select subsequent examples in the stream. By a judicious choice on the order of presentation of training examples we hope to improve learning performance and reduce computational costs, relative to random sampling. Such an ordering on available examples would be valuable for use with data driven learning techniques, in which network complexity required to achieve functional interpolation is directly related to the number of examples. It may also facilitate training in data abundant domains by reducing the size of the training set.

Active selection of training examples for network learning in noiseless environments.

Mark Plutowski*
Department of Computer Science and Engineering
University of California, San Diego

Halbert White
Institute for Neural Computation and Department of Economics
University of California, San Diego

February 1991

Technical Report No. CS91-180

Abstract

We derive a method for *actively selecting* examples to be used in estimating an unknown mapping with a multilayer feedforward network architecture. Active selection chooses from among a set of available examples an example which, when added to the previous set of training examples and learned, maximizes the decrement of network error over the input space. In practice, this amounts to incrementally growing the training set as necessary to achieve the desired level of accuracy.

The objective is to minimize the data requirement of learning. Towards this end, we choose a general criterion for selecting training examples that works well in conjunction with the criterion used for learning, here, least squares. Examples are chosen to minimize *Integrated Mean Square Error* (IMSE). IMSE embodies the effects of bias (misspecification of the network model) and variance (sampling variation due to noise). We consider a special case of IMSE, *Integrated Squared Bias* (ISB) to derive a selection criterion (ΔISB) which we maximize to select new training examples. ΔISB is applicable whenever sampling variation due to noise can be ignored. We conclude with graphical illustrations of the method, and demonstrate its use during network training.

*The author thanks the following people: Duane Desieno, for his insightful observations; Garrison Cottrell and Robert Hecht-Nielsen, for helpful advice; Bill Hart, David Pintsov and L. Steven Biafore, for numerous provocative conversations; and Timur Ash, for use of his excellent simulation code.