Estimating Monotonic Functions and Their Bounds

Herbert Kay and Lyle H. Ungar
Chemical Engineering and CIS Depts., University of Pennsylvania, Philadelphia, PA 19104

A function estimator MSQUID is presented for fitting and bounding noisy data that are known to be monotonic. MSQUID augments a "backpropagation" neural network model with a set of constraints which restricts the model to monotonic functions. Model parameters are estimated using nonlinear, constrained optimization at little increase in computation over standard neural networks. It is proven that MSQUID can estimate any monotonic function and produces more accurate estimates than unconstrained optimization. These monotonic functions and their confidence bounds can be used in many fault detection and diagnosis systems.

Introduction

System identification traditionally assumes that exact equational forms are known and then estimates parameters. In contrast, the recent extensive use of neural networks makes (in the limit of large networks) no assumption of functional form. Often, one has some intermediate level of knowledge, such as monotonicity of a function. A reaction may be known to increase with temperatures or a flow rate increase with pressure even if the precise functional form is unknown.

In this article, we present MSQUID, a Monotonic Semi-QUantitative system IDentification method for defining and searching a model space composed of neural networks constrained to be monotonic functions. (MSQUID is based on our earlier work first presented in Kay and Ungar (1993).) A number of fault detection and diagnosis systems are based on knowing the approximate bounds on the functions which constitute the process models (Dvorak and Kuipers, 1989; Trave-Massuyes and Milne, 1999; Vinson and Ungar, 1995). In addition to providing a general method for fitting functions using weak prior knowledge, MSQUID provides an automated way of finding such bounds which does not require assuming a parametric form. It also forms the basis for the SQUID system (Key et al., 1999) that refines ODE model spaces.

Techniques for estimating a functional relationship between an output variable y and the input vector $\boldsymbol{x}$ typically assume that there is some deterministic function g and some random variate ϵ such that $y = g(\boldsymbol{x}) + \epsilon$, where ϵ is a normally distributed, mean zero random variable with variance σ^2 that represents measurement error and stochastic variations. The estimate is computed by using a parameterized fitting function $f(\boldsymbol{x};\boldsymbol{\theta})$ (which defines a model space in terms of the unknown $\boldsymbol{\theta}$) and then using regression analysis to determine the values $\hat{\boldsymbol{\theta}}$ such that $f(\boldsymbol{x};\hat{\boldsymbol{\theta}}) \approx g(\boldsymbol{x})$.

Traditional regression methods require knowledge of the form of the estimation function f. For instance, we may know that body weight is linearly related to amount of body fat. We may then determine that $f(\text{weight};\theta) = \text{weight}\cdot\theta$ is an appropriate model. Neural network methods have been developed for cases where no information about f is known. This may be the case if f models a complex process whose physics are poorly understood. Such networks are known to be capable of representing any functional relationship given a large enough network. In this article, we consider the case where some intermediate level of knowledge about f is known. In particular, we are interested in cases where we have knowledge of the monotonicity of f in terms of the signs of $(\partial f/\partial x_k)$ where $x_k \in \boldsymbol{x}$. For example, we might know that outflow from a tank monotonically increases with tank pressure. This type of knowledge is prevalent in qualitative descriptions of systems, so it makes sense to take advantage of it.

Since the estimate is based on a finite set of data, it is not possible for it to be exact. We therefore require our estimate to have an associated confidence measure which takes into account the uncertainty introduced by the finite sample size. For our semiquantitative representation, we are therefore interested in deriving an envelope that bounds all possible functions that could have generated the data-stream with some probability p.

This article describes MSQUID, a method for estimating and computing bounding envelopes for multivariate functions based on a set of data and knowledge of the monotonicity of

Correspondence concerning this article should be addressed to L. H. Unger.

the functional relationship. First, our method is described for computing an estimate $f(x;\hat{\theta}) \approx g(x)$ based on a neural network that is constrained to produce only monotonic functions. Second, our method is described for computing a bounding envelope, which is based on linearizing the estimation function and then using F statistics to compute a simultaneous confidence band. Third, several examples are presented of function fitting and its use in semiquantitative simulation using QSIM. Next, the performance of MSQUID is analyzed with respect to the number of units in the hidden layer, noise level, and complexity of the monotonic function. Finally, we discuss related work in neural networks and nonparametric analysis and describe some applications of this method.

Bounded monotonic functions are a key element of the semiquantitative representation used by simulators such as Q2 (Kuipers and Berleant, 1988), Q3 (Berleant and Kuipers, 1997), and Nsim (Kay and Kuipers, 1993) which predict behaviors from models that are incompletely specified. Bounded functions are also used, among other places, in VLSI simulation (Zukowski, 1986). To date, the bounds for such functions have been derived in an *ad hoc* manner. The work described in this article provides a systematic method for finding these functional bounds. It is particularly appropriate for semiquantitative monitoring and diagnosis systems (Dvorak and Kuipers, 1989; Rinner and Kuipers, 1999; TraveMassuyes and Milne, 1999; Vinson and Ungar, 1995) for chemical plants or devices such as turbine engines because process data is readily available in such applications.

Computing the Estimate

Computing the estimate of g requires that we make some assumptions about the nature of deterministic and stochastic portions of the model. We assume that the relationship between y and x is $y = g(x) + \epsilon$ where ϵ is a normally distributed random variable with mean 0 and variance σ^2. Other assumptions, such as different noise probability distributions or multiplicative rather than additive noise coupling could be made. The above model, however, is fairly general and it permits us to use powerful regression techniques for the computation of the estimate and its envelope. For situations where variance is not uniform, we can use variance stabilization techniques to transform the problem so that it has a constant variance.

In traditional regression analysis, the modeler supplies a function $f(x;\theta)$ together with a dataset to a least-squares algorithm which determines the optimal values for θ so that $f(x;\hat{\theta}) \approx g(x)$. The estimated value of y is then $\hat{y} = f(x;\hat{\theta})$. In our case, however, the only information available about f is the signs of its n partial derivatives $(\partial f/\partial x_k)$ where $x_k \in x$, so no explicit equation for f can be assumed. One way to work without an explicit form for f is to use a neural net function estimator. Figure 1 illustrates a network for determining $\hat{y}$ given a set of inputs x. The network has three layers. The input layer contains one node for each element x_k and one bias node set to a constant value of 1. The hidden layer consists of a set of n_h nodes which are connected to each input variable as well as to the bias input. The output layer consists of a single node which is connected to all the hidden nodes as well as to another bias value which is fixed at 1. (The bias terms permit the estimation function to shift the center of the sigmoid.) All nodes use sigmoidal basis functions and all connections are weighted. In our notation, $w_{i[i,j]}$ represents the connection from input x_i to hidden node j and $w_{o[j,1]}$ represents the connection from hidden node j to the output layer. (The weight $w_{i[n_x+1,j]}$ represents the connection to the input bias and the weight $w_{o[n_h+1,1]}$ represents the connection from the hidden layer bias node to the output node.) This network represents the function

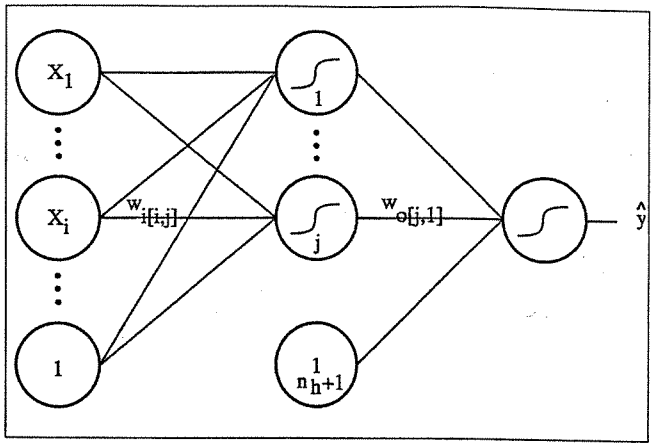

Figure 1. Neural net-based function estimator.

This three-layer net computes the function $\hat{y} = \sigma(\sum_{j=1}^{n_h}[w_{o[j,1]}\sigma(\sum_{i=1}^{n} w_{i[i,j]}x_i + w_{i[n+1,j]})] + w_{o[n_h+1,1]})$.

$$\hat{y} = \sigma(s + w_{o[n_h+1,1]})$$

$$s = \sum_{j=1}^{n_h}\left[w_{o[j,1]}\sigma\left(\sum_{i=1}^{n} w_{i[i,j]}x_i + w_{i[n+1,j]}\right)\right]$$

where $\sigma(x)$ is the sigmoidal function $(1-e^{-x})/(1+e^{-x})$. We can compute the weights by solving the nonlinear least squares problem

$$\min_{w} \sum_{i} (y_i - \hat{y}_i)^2$$

where $\hat{y}_i = f(x_i;w)$ and w is a vector of all weights. Cybenko (1989) and others have shown that with a large enough number of hidden units, any continuous function may be approximated by a network of this form.

One drawback of using this estimation function is that it can over-fit the given data. This results in the estimate following the random variate ϵ as well as the deterministic part of the model which means that we get a poor approximation of g. Many regularization methods can be used to reduce overfitting at, of course, the cost of introducing bias. We therefore reduce the scope of possible functions to include only monotonic functions. To do this, note that if f is monotonically increasing in x_k, then $(\partial f/\partial x_k)$ must be positive. By constraining the weights, we can force this derivative to positive for all x, insuring that the resulting function is mono-

tonic. The derivative in question is

$$\frac{\partial f}{\partial x_k} = \acute{\sigma}\left(s + w_{o[n_h+1,1]}\right)\cdot\frac{\partial s}{\partial x_k}$$

$$\frac{\partial s}{\partial x_k} = \sum_{j=1}^{n_h} w_{o[j,1]}\sigma'\left(\sum_{i=1}^{n}\left(w_{i[i,j]}x_i\right) + w_{i[k,j]}\right)w_{i[k,j]}$$

Since $\acute{\sigma}$ is positive for all values of its domain, $(\partial f/\partial x_k)$ will be positive if $(\partial s/\partial x_k)$ is positive. This will be the case if, for each k in $[1\ldots n_x]$,

$$\forall_{1\le j\le n_h} w_{o[j,1]}\cdot w_{i[k,j]} \ge 0. \qquad (1)$$

If partial derivative is negative, then the inequality is reversed. While this result holds for any k, for the remainder of this article we restrict our interest to single input models (that is, $k=1$). The Appendix contains a proof that a network constrained in this manner is capable of representing any monotonic function.

This set of additional constraints on weights causes the network to produce only monotonic functions. We may determine an estimate $\hat{w}$ for the weights by solving the problem

$$\min_{w} \sum_{i} (y_i - \hat{y}_i)^2$$

$$\text{subject to} \quad \forall_{1\le j\le n_h} w_{o[j,1]}\cdot w_{i[1,j]} \ge 0. \qquad (2)$$

These n_h constraints transform the least squares problem into a constrained nonlinear optimization problem. While more difficult to solve than the unconstrained problem, there are still a number of methods based on numerical methods. In our work we use the OPT algorithm (Biegler, 1985: Biegler and Cuthrell, 1985).

To compute the estimate, we need to find the best value for n_h. We determine n_h by repeatedly solving the optimization problem for increasing values of n_h starting at 1 and continuing until there is no improvement in the sample standard error

$$\frac{\Sigma_i (y_i - \hat{y}_i)^2}{n-p}$$

where the $p = n_w - n_h$ in the denominator accounts for reducing the degrees of freedom by one for each hidden unit since each hidden unit adds a constraint to the optimization problem.

Since this type of regression method is strongly susceptible to scaling problems, we prescale the data so that each independent variable falls in the range [0,1] and the dependent variable falls within the range [0,0.8]. (The upper value is less than 1.0 so that the output sigmoid does not saturate.) For monotonically decreasing functions, we perform a further transformation that maps each y value to $0.8 - y$ thus making it a monotonically increasing function.

Figure 2 shows the result of using the above method to estimate a fit to a data-stream derived from a quadratic function with noise from a normal distribution with $\sigma^2 = 4$. Note that the estimate is in fact monotonic and does not follow the noise.

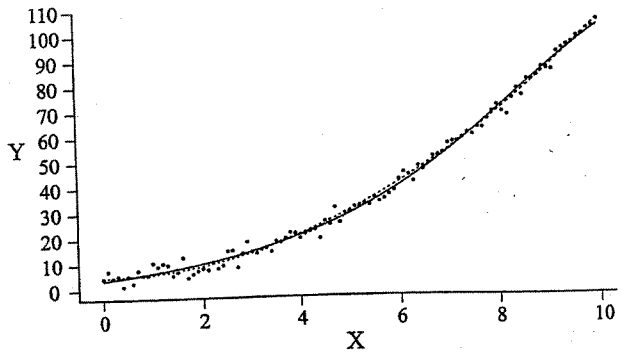

Figure 2. Fitting a neural net estimator to a data stream of 100 points.

The data was generated by adding noise with a variance of 4 to the curve $y = x^2 + 5$ (shown as a dashed line). The estimated function is shown as a solid line. There are two hidden nodes.

Computing the Envelope

The estimate computed in the previous section is affected by the sample dataset. Since this dataset is of finite size, the estimate generated from it will not be precisely correct. In this section, we describe our method for bounding our estimate with an envelope that captures the uncertainty introduced by using a finite set of data.

The typical confidence estimate used in regression analysis is the confidence interval, which is defined at a point $\boldsymbol{x}$ as $P(f(\boldsymbol{x}) - b_x \le g(\boldsymbol{x}) \le f(\boldsymbol{x}) + b_x) = 1 - \alpha$ where b_x depends on $\boldsymbol{x}$. The confidence interval is a point probability measure since it expresses the uncertainty of the estimated value $f(\boldsymbol{x})$ at a single point $\boldsymbol{x}$ in the domain. Since we wish our envelope to bound all possible functions, we require that our confidence interval holds simultaneously at all points in the domain. This measure (called a *confidence band*) can be easily computed for linear regression models. Since our model is nonlinear, we use a linearization of f to form an approximate confidence band.

Assume that we have a linear model $f(\boldsymbol{x};\boldsymbol{\beta}) = \boldsymbol{x}^T\boldsymbol{\beta}$ where $\boldsymbol{\beta}$ is a parameter vector of length p and that $\hat{\boldsymbol{\beta}}$ is our estimate of the parameters. If we represent the data-stream as a matrix $[\boldsymbol{Y}|\boldsymbol{X}]$ where the ith row represents a single sample data-point $(y_i, \boldsymbol{x}_i)$, it can be shown (Bates and Watts, 1988) that the $1-\alpha$ confidence band for f is

$$\boldsymbol{x}^T\hat{\boldsymbol{\beta}} \pm s\sqrt{pF(\alpha; p, n-p)}\,\|\,\boldsymbol{x}^T\boldsymbol{R}^{-1}\,\|$$

where s^2 is the sample standard error $(\|\,\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}\,\|^2)/(n-p)$, F is the α quantile of the F-statistic with p and $n-p$ degrees of freedom, and $\boldsymbol{R}$ is the square portion of the QR

decomposition of the array of sample inputs

$$X\left(X = Q\begin{bmatrix} R \\ 0 \end{bmatrix}\right).$$

Geometrically, the columns of X form a p-dimensional linear subspace called the *expectation surface* in which the solution must lie. The least square computation finds the point on the expectation surface that is closest to Y.

To use this result, we must linearize the nonlinear problem as follows. Note that the entry x_{np} of X is $(\partial x_n^T/\partial \beta_p)$. Using this, we linearize our estimator by defining a matrix V such that $v_{np} = \partial f(x_n;\hat{w})/\partial w_p$ and a vector v such that $v_p = (\partial f(x;\hat{w})/\partial w_p$. The envelope is then defined by

$$f(x;\hat{w}) \pm s\sqrt{pF(\alpha;p,n-p)}\,\|\,v^T R_v^{-1}\,\|$$

where

$$V = Q_v\begin{bmatrix} R_v \\ 0 \end{bmatrix}.$$

The linearization can be viewed as defining a plane which is tangent to the true expectation surface (which is not a linear subspace) at $\hat{w}$. Assuming that $\hat{w}$ is close to the exact value for w, the linearization will hold near this point on the plane.

Because the linearization is only an approximation, this estimate does not provide an exact $1-\alpha$ confidence band for f. However, the result is approximately correct and depends on the degree of nonlinearity of the expectation surface (Bates and Watts, 1988).

We use the LINPACK subroutine DQRDC to compute the QR decomposition of V. This method performs the decomposition using pivoting so that R_v is both upper triangular and has diagonal elements whose magnitude decreases going down the diagonal. With this form, we can easily recognize the case where the linearized matrix V may not have full rank (that is, only $q < p$ columns are linearly independent). This in turn means that R_v has zeros in all diagonals past column q. In such cases, the product $v^T R_v^{-1}$ may not have a solution. To handle this problem, we simply reduce the size of R_v by using only the upper left $q \times q$ corner. We must then also reduce the size of v so that it does not contain partials with respect to w_i where $i > q$. This is justified, since if V forms a p-dimensional basis for the linearized expectation surface, and its rank is only q, then the w_i where $i > q$ may be ignored since their value is arbitrary.

The confidence band is designed to cover all possible curves that fit the data with a probability of $1-\alpha$. With traditional regression, this confidence band is the most precise description we can achieve given the general form of f. Using monotonicity information, however, we can further refine our prediction to derive a tighter envelope by ruling out portions of the curves that could not be monotonic. Consider the confidence band shown in Figure 3. If we know that the underlying function is monotonic we may remove the shaded regions from the prediction since no monotonic function within the confidence band could pass through these regions. Note that this final step could not be performed if we used point confidence intervals.

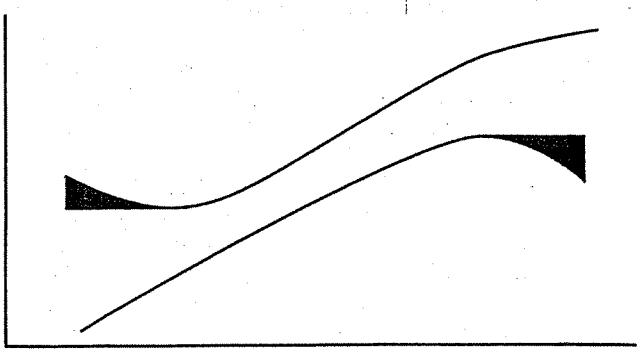

Figure 3. Reducing the size of a confidence band using monotonicity information.

If we know that the underlying function is monotonically increasing, we can rule out the shaded areas of the figure as part of the envelope.

Examples

This section consists of several simple examples of function and one a real-world application of using MSQUID to estimate stream-flow given level. It also briefly describes the use of MSQUID to learn SemiQuantitative Differential Equations (SQDEs) from data.

Simple Examples

We have applied our envelope method to noisy datasets whose underlying functions are linear, quadratic, square root, exponential, and have sign changes in the second derivative. In order to give a feeling for the form of the envelopes generated, we present several of these test cases. A complete summary is provided in the next section. Each result is for a univariate monotonically increasing model function $f(x)$. Figure 4 shows the estimate and envelope for a set of 100 samples drawn from the function $y = 0.5x + 5$ with additive noise with variance 4. Note that the envelope expands at the ends since there is less data there to constrain the estimate. The envelopes here are pointwise estimates and hence are not constrained to be monotonic. Simultaneous confidence bounds

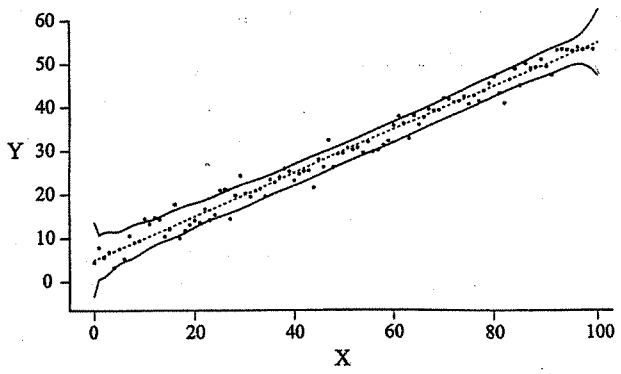

Figure 4. 95% envelope (solid lines) for the linear function $y = 0.5x + 5$ (dashed line).

The estimator has 3 hidden nodes. Sample variance is 4.520.

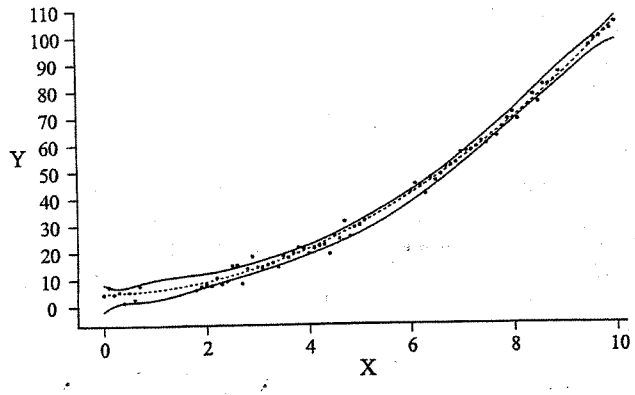

Figure 5. 95% envelope (solid lines) for the quadratic function $y = x^2 + 5$ (dashed line).

The estimator has two hidden nodes. Sample variance is 4.227.

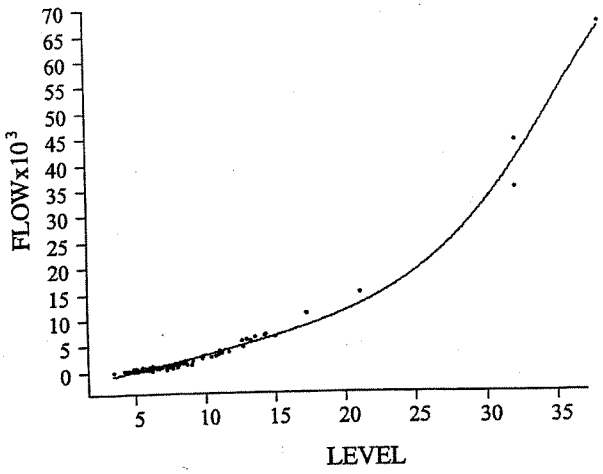

Figure 6. MSQUID to fit the flow vs. level data for the Bay City gaging station.

The standard error is 907. While the fit is reasonably good, it appears to miss the center of the main cluster between levels 4 and 15 and falls below the points near *level* = 20.

could be used and monotonicity imposed on them as illustrated in Figure 3.

The second example shown in Figure 5 is that of a quadratic function with noise of variance 4. This dataset demonstrates the effect of nonuniform sampling. In areas where there is little data, the envelope bulges outward to compensate. Note also that the upper portion of the envelope is narrowest at the far end of the plot. This is due to bias in the estimation function which assumes that the curve will "heel over" just past the end of the dataset. For this reason, the computed envelope is valid only over the range of the given data.

Application: Determining Stream Flow vs. Level Curves

The control of reservoir levels is of great importance to water management authorities. Part of the task is determining the amount of inflow into a watershed from the different streams that feed it. These measurements are particularly important during flood conditions so that potential reservoir overflow can be predicted (and prevented) well in advance of its actual occurrence.

Unfortunately, measuring the flow of a stream is a manual and time-consuming task involving the lowering of a flow gauge into the stream at various points on a cross-section of the stream. On the other hand, the measurement of level is quite easy, and is often automatically sampled by data acquisition equipment that immediately relays the information back to a central office. Thus, an experimentally determined curve that relates level to flow would provide a means for quickly ascertaining stream-flow given only level.

One difficulty with producing such a curve is that finding a parametric model for a particular stream can be difficult since the shape of the stream-bed cross-section (which depends on effects such as silting, vegetation, and physical obstructions) must be accounted for. Typical solutions to this problem are (1) to use a hand-drawn curve (which is time-consuming and requires expertise) or (2) to use a logarithmic approximation under the belief that "everything looks like a line on log-log paper" (which often results in too simplistic a model).

While it is difficult to find a good parametric model, the knowledge that flow is a monotonic function of level makes constructing the stream-flow versus level curve an ideal application for MSQUID. To test MSQUID on this task, we collected stream-flow versus level datasets compiled by the Lower Colorado River Authority (LCRA) for three streams in the Austin area and fit them using MSQUID with three hidden units. Figure 6 shows the results of fitting the data for the Bay City gaging station. The standard error is 907, and an analysis of the residuals shows that 95% of the data falls within two standard errors of the estimate, indicating a reasonable fit.

On more careful examination, however, we see that the fit could be improved especially at levels below 10 (where the curve passes above much of the data) and near level = 20 (where the curve passes below the data). The reason why MSQUID has produced a curve with a poor fit in these regions is because the data is not sampled uniformly in the dependent variable. Instead, the data is clustered about particular values of level. Clustering can lead to a bad fit because local properties of the data may be swamped by the effect of far-away clusters. Clustering is particularly common in stream-flow data since most measurements are made at "normal" water levels whereas it takes floods and droughts to produce data for high and low levels.

One way to overcome this fitting error is to treat each cluster as a separate fitting problem and to run MSQUID on each region separately. One can then interpolate between the clusters using a function that maintains monotonicity. (This approach is in essence with what someone using a French Curve would do—fit each cluster and then interpolate smoothly between them.) Figure 7 demonstrates the resulting fit using such a method. The Bay City measurements can be split into three clusters representing the large mass of points in [4, 15] together with two "arms" which contain points to the left and right of the main cluster. Fitting each of these clusters separately yields a standard deviation of 690 resulting in a 25% better fit. (To ensure a minimum number of

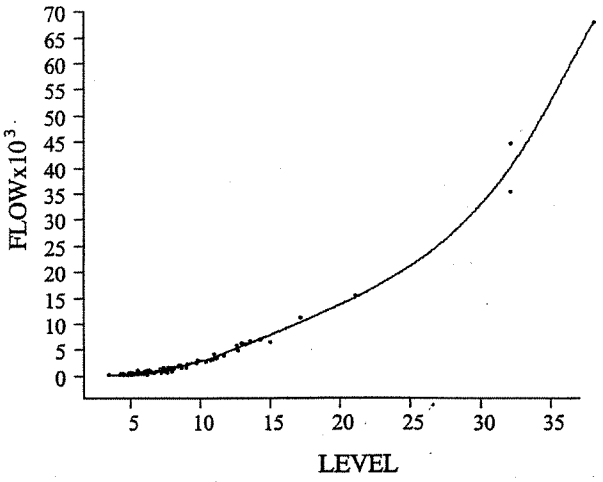

Figure 7. Clustering version of MSQUID to fit the same data as in Figure 6.

By using different MSQUID functions for the central cluster and its arms, the fit is improved. The standard error in this case is 690.

points in a cluster, we "borrow" points from the main cluster as needed. This also improves the interpolation between clusters).

Application: SQDE Learning

One of the motivations for developing MSQUID is as a basis for learning semiquantitative models. Qualitative Process Theory, as implemented in simulators such as QSIM (Kuipers, 1986, 1994) give a formal language for describing the monotonic relations MSQUID requires. Simulators such as SQSIM (Kay, 1998) support simulations based on qualitative models augmented with numerical information about the monotonic functions and parameters. Normally, these numerical envelopes are *ad hoc* in that they are hand-derived by the modeler. By applying the MSQUID envelope method to a data stream from the system, we can construct these envelopes in a more principled way. See Kay et al. (1999) for examples.

Analysis

Experiments comparing MSQUID and UNCMND (Kahaner, Moler, and Nash, 1989), a fast, readily available, unconstrained optimizer, using the same neural network structure confirm what one would expect: when three hidden nodes are sufficient to approximate the data, MSQUID gives equally accurate models with 3 and 8 hidden nodes, while unconstrained optimization overfits. For noisy data, this effect can be dramatic.

Discussion

Related work

This work is related to several approaches to function estimation using neural networks. Cybenko (1989), among many others, has shown that any continuous function can be represented with a neural net that is similar to one that we use. With our method, we assume *a priori* that the true function is monotonic. This assumption restricts our attention to a smaller class of functions which means that less data is needed to compute a reasonable estimate. Additionally, we compute a confidence measure on our estimate in the form of an envelope.

The VI-NET method (Leonard et al., 1992) also uses a neural network for computing estimates of general functions. It is notable in that it provides a confidence measure on its estimate and can determine when it is being asked to extrapolate. By using radial basis functions instead to sigmoidal ones, it is able to handle different variances across the function. This is especially useful in applications where there is no a priori information about g. Because it allows nonconstant variances, it would be difficult to get a VI-NET to return simultaneous confidence bands, which are important since we wish to bound all curves that could have generated the data-stream. In contrast, our approach can only handle fixed-variance problems, but since variance will often track either x or y, we can make monotonicity assumptions about σ^2 if it should prove necessary. Under such circumstances, variance stabilization techniques (Draper and Smith, 1981) should prove useful in fixing the variance.

Other researchers have also computed prediction intervals for neural networks, typically assuming constant variance (de Veaux et al., 1999). As in VI-NET, this work does not make any assumptions about the monotonicity of the functions. Closer in spirit is the extensive work on hybrid networks (Psichogios and Ungar, 1992; de Veaux et al., 1999), in which neural networks are embedded into algebraic or differential equations. Hybrid Nets, like MSQUID, use prior knowledge to constrain the space of functions being estimated. They differ in the form of constraint being used. It might, in fact, be desirable to use both forms of constraint together.

This work is also related to monotonic function estimation (Joerding and Meador, 1991; Hellerstein, 1990; Kruskal, 1964), particularly the NIMF estimator (Hellerstein, 1990) which also determines envelopes for multivariate monotonic functions. NIMF allows for a more general noise model (zero mean, *symmetrically* distributed) and uses a nonparametric statistical method for determining point confidence bounds. These bounds, however, are much weaker than those derived with methods that first compute an estimate. This is not surprising, since we are assuming more about the noise than NIMF does. Finally, NIMF produces point bounds only, and it is unclear how these bounds could be made into confidence bands of reasonable width.

Conclusions

This article has described a method for computing envelopes for functions described solely by monotonicity information and a stream of data. It improves over existing methods for function estimation by providing a simultaneous confidence band that encloses all functions that could have generated the data-stream. Using monotonicity information provides several benefits to function estimation:

- Less data are required to obtain a reasonably precise model.

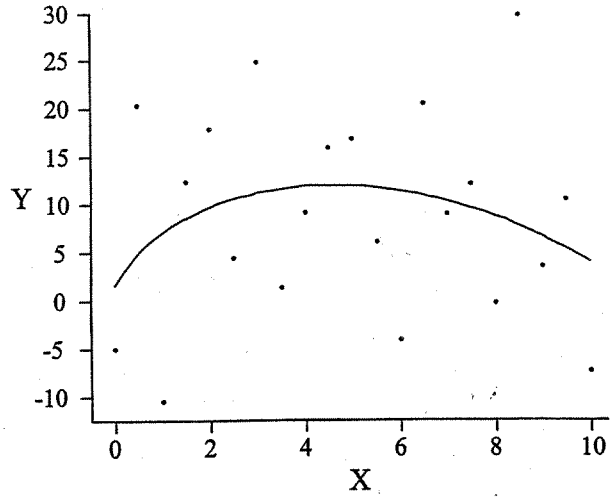

Figure 8. Example of overfitting with an unconstrained optimizer.

- Problems with overfitting are reduced.
- When combined with simultaneous confidence bands, portions of the band can be eliminated, thus further improving model precision.

While the class of monotonic functions may seem restrictive for modeling continuous systems, we can represent many nonmonotonic functions as compositions of monotonic ones. For example, in a system of two cascaded tanks, the level of water in the lower tank is typically a nonmonotonic function of time. The flow into the lower tank, however, is the composition of a monotonically increasing function of the amount of water in the upper tank and a monotonically decreasing function of the water in the lower tank. By using semiquantitative simulation methods, we can represent this composition and thus simulate systems with nonmonotonic behaviors.

Use of low order polynomials provide a potential alternative to MSQUID for single dimensional problems such as we have mostly used to illustrate the method. Unfortunately, even quadratics violate monotonicity, and as can be seen looking at the fit in Figure 8 (actually a neural net fit to a monotonic model plus noise), a quadratic can give a deceptively good fit to data which is monotonic if it contains substantial noise. One could extend the concept of MSQUID to use higher order polynomials (cubic, quartic, quintic), with monotonicity constraints. This should work reasonably well for low dimensional problems although it will not scale well to higher dimensional problems.

Our method for deriving bounds for monotonic functions also plays a key role in the construction of semiquantitative models, especially for monitoring and diagnosis tasks where process data are readily available. Because the precision of the resulting envelopes is a function of the amount of data used to compute them, the MSQUID bounding method provides a systematic way of shifting the precision of a semiquantitative model along a continuum from pure qualitative to exact quantitative models.

Acknowledgments

Dr. Herbert Kay unfortunately passed away in 1997. He left a significant body of great scientific work (www.cs.utexas.edu/users/bert/). We feel deprived of his unfulfilled promise of further contributions to the world.

This work was part of Herbert Kay's doctoral dissertation in the Computer Science Dept., University of Texas at Austin, supervised by Dr. Benjamin Kuipers. This research was supported in part by NSF grants CTS-9504407, IRI-8905494, IRI-9017047, and IRI-9216584, by NASA grants NAG2-507, NAG 9-512, NCC 2-760, NAG 9-665, and NAG 2-944, and by the Texas Advanced Research Program under grants no. 003658-242 and 003658-347.

Notation

n = number of observations
n_x = dimension of x
n_h = number of hidden units in the network
n_w = number of weights in the network (a total of $(n_x+2)n_h+1$)
p = degrees of freedom $(n_w - n_h)$
x, y = domain and range of the function $[y = g(x)]$
$w_{i[i,j]}$ = weight from x_i to hidden unit j
$w_{o[j,1]}$ = weight from hidden unit j to the output
w = vector of all weights
$\hat{w}$ = estimated weight vector
$\hat{y}$ = estimate of y
σ^2 = variance

Literature Cited

Bates, D. M., and D. G. Watts, *Nonlinear Regression and Its Applications*, Wiley, New York (1988).

Berleant, D., and B. Kuipers, "Qualitative and Quantitative Simulations: Bridging the Gap," *Artificial Intelligence*, **95**, 215 (1997).

Biegeler, L. T., and J. E. Cuthrell, "Improved Infeasible Path Optimization for Sequential Modular Simulators: II. The Optimization Algorithm," *Comp. and Chem. Eng.*, **9**, 257 (1985).

Biegler, L. T., "Improved Infeasible Path Optimization for Sequential Modular Simulators: I. The Interface," *Comp. and Chem. Eng.*, **9**, 245 (1985).

Cem Say, A. C., and S. Kuru, "Qualitative System Identification: Deriving Structure from Behavior," *Artificial Intelligence*, **83**, 75 (May 1996).

Cybenko, G., "Approximation by Superpositions of Sigmoidal Functions," *Mathematics of Control, Signals, and Systems*, **2**, 303 (1989).

De Veaux, R. D., R. Bain, and L. H. Ungar, "Hybrid Neutral Network Models for Environmental Process Control," *Environmetrics*, **10**, 225 (1999).

De Veaux, R., J. Schumi, J. Schweinsberg, D. Shellington, and L. H. Ungar, "Prediction Intervals for Neural Networks via Nonlinear Regression," *Technometrics*, **40**, 273 (1998).

Draper, N. R., and H. Smith, *Applied Regression Analysis*, 2nd ed., Wiley, New York (1981).

Dvorak, D. L., and B. Kuipers, "Model-Based Monitoring of Dynamic Systems," *Proc. Int. Joint Conf. on Artificial Intelligence*, 1238 (1989).

Hellerstein, J., "Obtaining Quantitative Predictions from Monotone Relationships," *Proc. National Conf. on Artificial Intelligence* (AAAI-90), 388 (1990).

Joerding, W. H., and J. L. Meador, "Encoding a Priori Information in Feedforward Networks," *Neural Networks*, **4**, 847 (1991).

Kahaner, D., C. Moler, and S. Nash, *Numerical Methods and Software*, Prentice-Hall, Englewood Cliffs, NJ (1989).

Kay, H., "Monitoring and Diagnosis of Multi-Tank Flows Using Qualitative Reasoning," Master's Thesis," The Univ. of Texas at Austin (1991).

Kay, H., "Refining Imprecise Models and Their Behaviors," PhD Thesis, The Univ. of Texas at Austin (1996).

Kay, H., "SQSIM: A Simulator for Imprecise ODE Models," *Comp. and Chem. Eng.*, **23**, 27 (1998).

Kay, H., and B. Kuipers, "Numerical Behavior Envelopes for Qualitative Models," *Proc. of Eleventh National Conf. on Artificial Intelligence* (AAAI-93), published by MIT Press, Cambridge, MA, 606 (1993).

Kay, H., and L. H. Ungar, "Deriving Monotonic Function Envelopes from Observations," *The Seventh International Workshop on Qualitative Reasoning about Physical Systems*, Orcas Island, WA, 117 (1993).

Kay, H., B. Rinner, and B. Kuipers, "Semi-Quantitative System Identification," *University of Texas at Austin Technical Report*, Number AI99-279 (1999).
Kruskal, J. B., "Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis," *Psychometrika*, **29**, 1 (1964).
Kuipers, B., and D. Berleant, "Using Incomplete Quantitative Knowledge in Qualitative Reasoning," *Proc. Nat. Conf. on Artificial Intelligence (AAAI-88)*, published by Morgan Kaufman, 324 (1988).
Kuipers, B., "Qualitative Simulation," *Artificial Intelligence*, **29**, 289 (1986).
Kuipers, B., *Qualitative Reasoning—Modeling and Simulation with Incomplete Knowledge*, MIT Press, Cambridge, MA (1994).
Leonard, J. A., M. A. Kramer, and L. H. Ungar, "Using Radial Basis Functions to Approximate a Function and Its Error Bounds," *IEEE Trans. of Neural Networks*, **3**, 624 (1992).
Psichogios, D. C., and L. H. Ungar, "A Hybrid Neural Network—First Principles Approach to Process Modeling," *AIChE J.*, **38**, 1499 (1992).
Rinner, B., and B. Kuipers, "Monitoring Piecewise Continuous Behaviors by Refining Trackers and their Models," *Proc. of Int. Joint Conf. on Artificial Intelligence*, Stockholm, Sweden (1999).
TraveMassuyes, R. Milne, "Gas-Turbine Condition Monitoring Using Qualitative Model-Based Diagnosis," *IEEE Expert*, **12**, 22 (1997).
Vinson, J. M., and L. H. Ungar, "Dynamic Process Monitoring and Fault Diagnosis with Qualitative Models," *IEEE Trans. on Man, Machines and Cybernetics*, **25**, 181 (1995).
Zukowski, C. A., *The Bounding Approach to VLSI Circuit Simulation*, Kluwer Academic Publishers, Boston (1986).

Appendix: MSQUID Can Produce Any Monotonic Function

In this section, we show that the set of functions that MSQUID can represent includes the set of all continuous monotonic functions. For simplicity, we consider only the case of a one dimensional monotonic function (that is, function is $y = f(x)$ where x is scalar). The proof would follow directly from [?] except that, for reasons of computational efficiency, our constraint that guarantees monotonicity is overly strong. To see this, note that the derivative of y is

$$\frac{d\hat{y}}{dt} = \sigma'(z + w_{o[n+1,1]}) \sum_{j=1}^{n_h} w_{o[j,1]} \sigma'(w_{i[1,j]}t + w_{i[2,j]}) w_{i[1,j]}$$

$$z = \sum_{j=1}^{n_h} w_{o[j,1]} \sigma(w_{i[1,j]}t + w_{i[2,j]})$$

To ensure monotonicity, this *sum* must be greater than or equal to zero. To simplify the fitting process, however, MSQUID uses the constraint

$$\min_{w} \sum_{i} (y_i - \hat{y}_i)^2$$

$$\text{subject to } \forall_{1 \le j \le n_h} w_{o[j,1]} \cdot w_{i[1,j]} \ge 0.$$

which requires only that each *term* of the equation must be greater than or equal to zero. Intuitively therefore, MSQUID requires that each sigmoid in the composition must be monotonically increasing which is not the case of the more general constraint.

For clarity, the proof below uses a slightly different definition of the sigmoid function and the network does not include a sigmoid at the final output of the network.

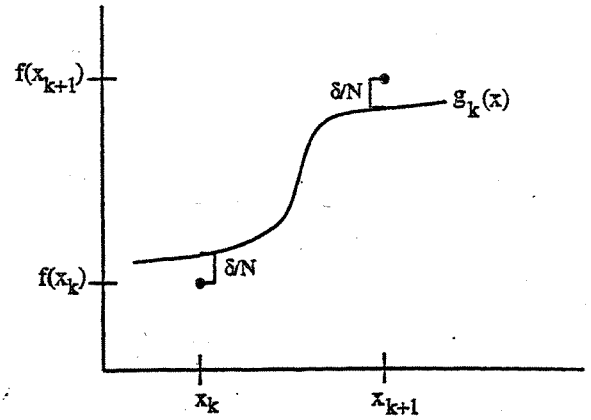

Figure A1. Geometric view of the proof.

Theorem 1. *Let σ be sigmoidal function such that $\sigma(x) \to 1$ as $x \to \infty$ and $\sigma(x) \to 0$ as $x \to -\infty$. Then given any $f(x)$ which is monotonic over some region I and $\epsilon > 0$ there is a finite sum of the form $y(x) = \sum_{j=1}^{N} a_j \sigma(b_j x + c_j)$ where $\forall_j a_j b_j \ge 0$ for which $|y(x) - f(x)| < \epsilon$ for all $x \in I$.*

Proof. Let $\epsilon = \gamma + \delta$ where δ and γ are nonzero. Select N by partitioning I such that $\forall_{0 \le k < N} f(x_{k+1}) - f(x_k) < \gamma$, that is, f is subdivided so that there is no more than γ difference between any two contiguous points. We prove that MSQUID can still produce any continuous monotonic function as follows (refer to Figure A1):

Assume that at x_0 the value of f is $f(x_0)$. Let $y_0(x) = f(x_0)$. Now let $d_k = f(x_{k+1}) - f(x_k)$. Define $g_k(x) = d_k \sigma\{bx - [x_k + (h_k/2)]\}$ where b is chosen so that $g_k(x) \le \delta/N$ for $x \le x_k$ and $d_k - g_k(x) \le \delta/N$ for $x \ge x_k$. Then define $y_{k+1}(x) = y_k(x) + g_k(x)$.

We wish to show that

$$|y_k(x) - f(x)| \le k\frac{\delta}{N} + \gamma \qquad \text{(A1)}$$

for all $x \in [x_0, x_k]$. Clearly this is true for $k = 0$. For $k = 1$ we have

$$|y_1(x) - f(x)| = |y_0(x) + g_0(x) - f(x)|$$
$$= |y_0(x_0) - f(x_0) + g_0(x) + e_1(x)|$$

where $e_k(x) \le d_k$ for all $x \in [x_k, x_{k+1}]$. The first two terms inside the absolute value are identical while the third term is less than δ/N and the final term is less than γ. Thus

$$|y_1(x) - f(x)| \le \frac{\delta}{N} + \gamma.$$

If we assume that Eq. A1 is true for k then for $k+1$ we have

$$|y_{k+1}(x) - f(x)| = |y_k(x) + g_k(x) - f(x)|.$$

We split this into two cases—$x \in [x_0, x_k]$ and $x \in [x_k, x_{k+1}]$.

For $x \in [x_0, x_k]$ we have $g_k(x) \le \delta/N$ and so

$$|y_k(x) + g_k(x) - f(x)| \le k\frac{\delta}{N} + \gamma + \frac{\delta}{N} = (k+1)\frac{\delta}{N} + \gamma.$$

For $x \in [x_k, x_{k+1}]$, let $d_k = g_{l,k} + g_{m,k} + g_{u,k}$ where $g_{l,k}$ and $g_{u,k}$ are each less than δ/N. These terms represent the approximation error of the sigmoid over $[x_x, x_{k+1}]$.

$$\begin{aligned}|y_k(x) + g_k(x) - f(x)| &= |y_k(x_k) + \mu_k + g_l + g_m - f(x_k) - d_k| \\ &= |y_k(x_k) - f(x_k)| + |\mu_k + g_l + g_m - d_k|\end{aligned}$$

where $\mu_k = y_k(x) - y_k(x_k)$ is the amount that y_k increases over the same interval. Note that this term must be less than δ/N. We know that $g_l + g_m - d_k = -g_u$ and therefore

$$|y_k(x) + g_k(x) - f(x)| \le k\frac{\delta}{N} + \gamma + |\mu_k - g_u|$$

Now since $\mu_k \le \delta/N$ and $g_u \le \delta/N$ the third term of the inequality must be smaller than δ/N which proves the statement for $k+1$.

Letting $k = N$ gives us that

$$|y_N(x) - f(x)| \le N\frac{\delta}{N} + \gamma = \epsilon$$

over I. Setting $y(x) = y_N(x)$ and the proof is complete.

Manuscript received May 13, 1999, and revision received Apr. 27, 2000.