A Data Approach Alternative at System Identification and Modeling Using the Self-organizing Associative Memory (SAM) System

Wei Kang Tsai, Wei-min Chiu, and Hon-Mun Lee
Electrical and Computer Engineering, University of California, Irvine
wtsai@ece.uci.edu

Abstract

We introduce a data-based approach alternative to the rule-based parameter approach toward system identification. Motivated by the design-intensive problem of the parameter approach, the Self-organizing Associative Memory (SAM) system seeks to represent the system using a subset of stored training data. We surmise that knowledge is association between memorized objects, not memorized rules. We postulate that only novel and distinct data should be organized into memory, while familiar data may be reproduced to an acceptable degree of accuracy by association between memorized data. The concept is materialized in several computational formats and tested on four different test cases. Results indicate that this data approach has high accuracy, relatively design-free, and requires only one pass of the training data to train.

1. Introduction

System identification, the discipline of reconstructing static and dynamic behaviors of unknown systems, has increasingly been done using adaptive and soft-computing methods such as modern control theory, neural networks and adaptive fuzzy logic systems. The design of these intelligent systems may be classified as the *parameter approach* which involves the definition of a set of parameters to represent the system identifier, and the optimization of that set of parameters to produce a "learned" system model.

The neural network and other basis-function methods have performed as robust and highly adaptive system models. Indeed, the popular Multi-Layer Perceptron (MLP) has been shown by Kolmogorov to be an universal approximator – given the right network architecture [5]. However, the neural network, as a parameter approach, is also highly design-intensive. In defining a neural network, many design parameters come into consideration: initialization, activation function, weighting function, bias, learning step size, and the training method. Perhaps the most tasking aspect of neural network design is its architecture: often designed from experience, sometimes by node "pruning" or "growing" such as the cascade correlation method [4]. Furthermore, the training of the MLP by way of descent methods such as the popular back error propagation or extended Kalman filter [7] requires numerous passes of the training data through the system model. And when the MLP eventually identifies a system, it is exceptionally difficult for the designer to extract the knowledge "learned" from the system, for the knowledge acquired in the MLP is really a global behavior encoded in its individual basis functions (neuron nodes) which serve as the system's memory units.

In cases where there is insufficient information about the system to be modeled, we cannot really make assumptions on the complexity and size of the system by arbitrarily specifying and optimizing some set of system parameters to adequately model the system. The design-intensive nature of adaptive parameter-based systems motivates a departure: an alternative, *data approach*. In this paper we propose a novel Self-organizing Associative Memory (SAM) model for identification of nonlinear dynamical systems.

SAM takes a fresh approach of buying fast learning speed with larger memory use. As the semiconductor industry makes continuous progress to produce cheaper and faster memory logic chips, it becomes extremely attractive to use larger sets of memory in exchange for faster training. The concept of SAM is based on two premises. The first is the assumption that rules are not memorized; rather, they are induced from association between memorized objects. Secondly, only novel and distinct objects are memorized, objects which yield knowledge that cannot otherwise be obtained through any association between already-memorized objects.

2. Self-organizing Associative Memory

The SAM has a simple procedure: (1) present the system with a new input point (2) the new input evokes some association between its closest neighboring points (3) should the association produce a corresponding output that is reasonably close to the desired output then

0-7803-4859-1/98 $10.00©1998 IEEE

the new input-output data pair is familiar and deemed redundant (4) if the association of the stored neighboring points yields an output insufficiently accurate then the new data pair is novel and is memorized. So the novelty of the training data – not the frequency of its occurrence – determines whether or not it is memorized.

The SAM model may be as simple or complex as the system it identifies. When SAM encounters a complex system, it stores numerous data and become large and complex. Conversely, for a system as simple as a line in two dimensions, SAM is simple, storing only two relevant data points. From the procedure of SAM we see that in the very first pass of the training data SAM will memorize each and every important data and ignore the redundant ones, making subsequent passes through the same set unnecessary.

In a review of related research we have found Memory-Based Reasoning method detailed (MBR) in [11] and [9] to be similar to SAM in philosophy. The MBR model is also based on a set of stored data and rules, some form of matching of a new input stimulus with "similar" stored data, and interpolation between them. MBR has also been employed successfully in pattern classification problems [2] and control [10], often drawing from a database of rules. SAM, on the other hand, takes the idea further structurally and computationally, conforming the idea to an adaptive system identification model with different variations of local interpolative functions.

Looking at its weaknesses, we see that SAM system takes more time to produce an output than would, for example, the MLP. For the SAM, searching for the proper neighborhood for interpolation is part of the storage and retrieval processes. In every neural network architecture some form of searching is involved. For example, ART1, ART2, and LVQ2 [3] all include explicit searching as a part of the network operation. Even for the MLP, searching is implicitly done via the corporate computation performed at all the nodes with the activation function serving as discriminators.

There are, however, many strengths of SAM. Because of the local interpolation method of SAM, a local value of a function is only dependent on a small (stored) subset of training samples. Hence, the actual recall time outside of the search for the neighborhood locale is minimal. The time it takes for the SAM to "learn" the training data is also short, given that all SAM does for learning is to selectively memorize and index certain training data. Secondly, due to the interpolation approach, the SAM system directly stores the training data (prototypes), enabling SAM to do knowledge level processing very simply. Since the SAM system contains actual, physical data in its memory, it encodes its knowledge "transparently," making rule extraction more realizable.

3. Computation models of SAM

For the application of system identification, the nonlinear dynamical system may typically be described in the NARX (Nonlinear Auto-Regression with eXogenous inputs) model [1]:

$$y(k) = f\big(y(k-1),\cdots,y(k-p),u(k),\cdots u(k-q+1)\big) \quad (1)$$

where $y(k) \in \Re^m$, $u(k) \in \Re^r$, k is a discrete time index, and $f(\cdot)$ a general vector-valued nonlinear function of multiple variables, or the typical state-state model:

$$x(k+1) = f\big(x(k),u(k)\big), \quad y(k) = h\big(x(k),u(k)\big) \quad (2)$$

where the state $x(k) \in \Re^n$ and $h(\cdot)$ is again a general vector-valued nonlinear function of multiple variables. We adopt the NARX model, with some controllability and observability conditions, with the assumption that the state of the system in (2), $x(k)$, may be reconstructed from p past outputs, $y(k-1),\cdots,y(k-p)$ and q past inputs $u(k),\cdots,u(k-q+1)$, an assumption adapted from Narendra and Parthasarathy [6].

Though we have developed several models of computation for the SAM, Figure 1 illustrates the general procedure of SAM. Given an input, SAM will search in the input-space of the stored data for the closest neighbors and generalize these neighboring data into a neighborhood or local transfer function, which maps the input x onto output y'. Should the percentage error ε between the desired output y and the predicted output y' of the given input exceed some error threshold ε_1, the given input and its corresponding desired output are stored.

$$\varepsilon = \frac{\|y - y'\|_2}{\min\{\|y\|_2, \|y'\|_2\}} \quad (3)$$

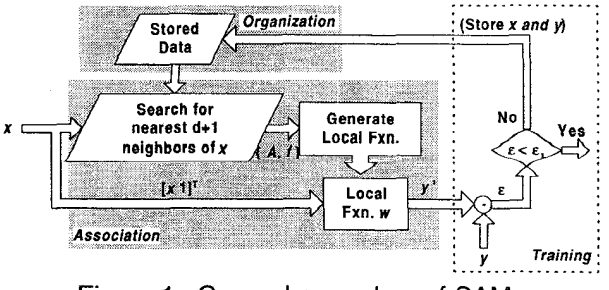

Figure 1. General procedure of SAM

3.1. Linear SAM using overlapping local linear approximation

The approximation method for the input-output function of the NARX model in the linear rendition of SAM is a set of locally-linear approximations that are overlapped in the global view. The safest local approximation relating a set of points given a d-dimensional input space is a $(d+1)$-dimensional hyperplane. Given the input $x \in \Re^d$, the linear SAM gathers $d+1$ closest stored neighbors in the input-space x_i^T and their corresponding outputs y_i^T into a matrix augmented by a '1' as reference for the hyperplane:

$$\begin{bmatrix} 1 & x_1^1 & \cdots & x_1^d \\ 1 & x_2^1 & \cdots & x_2^d \\ & \vdots & & \\ 1 & x_{d+1}^1 & \cdots & x_{d+1}^d \end{bmatrix} \begin{bmatrix} w^0 \\ w^1 \\ \vdots \\ w^d \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{d+1} \end{bmatrix}, \text{ or } Aw = f, \quad (4)$$

$w' = A^+ f = (A^T A)^{-1} A^T f$, and

$$y' = \begin{bmatrix} 1 & x_1 & \cdots & x_d \end{bmatrix} w' \quad (5)$$

the hyperplane being the weight matrix w in $(d+1)$-space. Given that sometimes the matrix may be singular or near-singular and to allow the freedom for over-determined systems, we use the least squares solution pseudoinverse [8] with singular-value decomposition for singular cases in equation (5). The hyperplane w then maps the augmented input $[1\ x^T]^T$ onto the predicted output y': $y' = [1\ x^T]w$.

3.2. Nonlinear SAM using weighted pseudoinverse

Working on the same principle of generalizing a set of stored data in a neighborhood into local functions, SAM may use nonlinear interpolation methods for the local functions as well. Having attempted various methods such as natural splines and local fuzzy-inferencing functions, we have decided on a simple and effective way of introducing nonlinearity – the weighted pseudoinverse: a modification to the least squares cost function of a positive-definite symmetric weighting matrix W which may be decomposed into its square roots Ψ and figured into the least squares cost function [8]:

$$W = \Psi^T \Psi \quad (6)$$

$$J(w) = (f - Aw)^T W (f - Aw) \quad (7)$$

$$= [(\Psi f) - (\Psi A)w]^T [(\Psi f) - (\Psi A)w]$$

Let $\bar{f} = \Psi f$ and let $\bar{A} = \Psi A$, then

$$\bar{w}' = \bar{A}^+ \bar{f} \text{ and } y' = \begin{bmatrix} 1 & x^T \end{bmatrix} \bar{w}' \quad (8)$$

The weighting function Ψ may be a diagonal matrix for which the diagonal elements indicate a confidence for their corresponding stored data. The weighted pseudoinverse exhibits some smoothing properties of local multi-dimensional splines, and may be a first step toward extracting rules from the data. Section 4.2 lists some examples of nonlinear weighting functions and their test results.

3.3. logic SAM using weighted linear combination

In cases where SAM is to learn a logical system where the system outputs are confined to the set {0,1}, the pseudoinverse method may predict values outside of the [0, 1] range because it also extrapolates. A way to confine the predicted output y' in the range [0, 1] is to take the collection of the $d+1$ nearest stored data in equation (4), weight f with some weighting vector v and normalize the weights:

$$y' = \frac{v^T f}{\sum v_i}, \text{ where } v^T = \begin{bmatrix} v_1 & v_2 & \cdots & v_{d+1} \end{bmatrix} \quad (9)$$

Since all elements of f are in the set {0, 1}, and the weighting is normalized, y' is bounded by [0, 1]. Section 4.3. will demonstrate the results of this method.

4. Problems and results

4.1. Example 1: MIMO nonlinear system identification

A Multi-Input and Multi-Output (MIMO) nonlinear system is described by the following set of state-space equations:

$$x_1(k) = .5\sqrt{x_1(k-1)} + .3x_2(k-1)x_3(k-1) + .2u_1(k) \quad (10)$$

$$x_2(k) = .5\sqrt{x_2(k-1)} + .3x_1(k-1)x_3(k-1) + .2u_1(k)$$

$$x_3(k) = .5\sqrt{x_3(k-1)} + .3x_1(k-1)x_2(k-1) + .2u_2(k)$$

$$y_1(k) = .5[x_1(k) + x_2(k) + x_3(k)], \quad y_2(k) = 2[x_1(k)]^2 \quad (11)$$

A Linear SAM (as in section 3.1) is trained with the following data sets: all 25 possible combinations of steps starting from 0 and ending at 0, 0.125, 0.25, 0.375, 0.5. Also, 25 positive and 25 negative ramps with the same starting and ending points are included in the training data set. Each training set contains 100 points. The total number of training data is 7500. The Linear SAM is then tested with two input sequences: one containing two pulses of magnitude 0.3 offset by 10 time steps, and the other is two steps of magnitude 0.3 and 0.2 with Gaussian noise of 0.1 standard deviation superimposed on them. SAM's prediction results for the pulse and the Gaussian noise sequences are shown in Figures 2a and 2b, respectively. The prediction errors for testing of the system with the pulsed input and the Gaussian noise inputs are 1.7073% and 3.1864%, respectively.

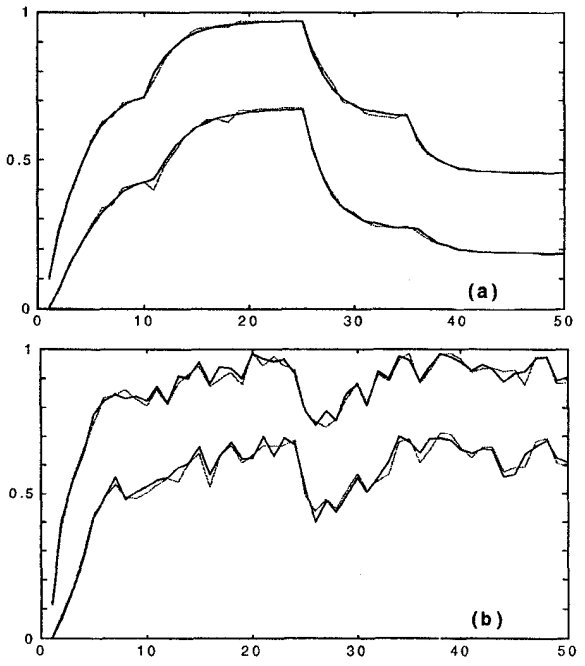

Figure 2. MIMO system testing outputs (a) with pulse inputs (b) with Gaussian noise inputs

4.2. Example 2: tenth-order boiler model

Using the Linear SAM described in section 3.1, we attempt to identify a 10^{th} order boiler model using a 2^{nd} order NARX model as in equation (1). That is, given only a ramp-step signal $u(k)$ driving the boiler model and its output signal $y(k)$, we seek to build a system identifier taking the set $[u(k)\ y(k\text{-}1)\ y(k\text{-}2)]^T$ as inputs and $y'(k)$ as the predicted output $y'(k) = f(u(k), y(k-1), y(k-2))$. Two sets of 90 data from the boiler model were used to train a Linear SAM system, an adjusted 12-5-1 MLP neural network, and a radial basis function (RBF) neural network. A third set of 90 data is used to test the three system identifiers.

Both SAM and the RBF were able to imitate the system dynamics well on-line given the desired output feedbacks $y(k\text{-}1)$ and $y(k\text{-}2)$ of the test set (Figure 3a). But suppose the boiler model is off-line, e.g. the desired feedback $y(k)$ are not available and all we have are our past predictions. Then $[u(k)\ y'(k\text{-}1)\ y'(k\text{-}2)]^T$ would be the input for the test set and

$$y'(k) = f(u(k), y'(k-1), y'(k-2)),$$

i.e. the entire system model dynamics would require *highly* accurate predictions in order to stay on track. Figure 3b compares the three systems. While all three systems strayed from the correct response after the 20^{th} time step, the linear SAM, having stored 37 of the 180 training set data, appears to perform better than the MLP and the RBF.

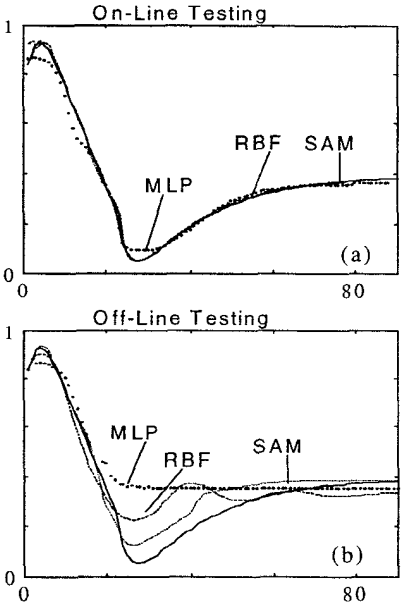

Figure 3. Testing of different system identifiers

We take the problem further by implementing a Nonlinear SAM as discussed in section 3.2. Five different weighting functions of the Euclidean distance between the given new input and inputs from the recalled stored data set are put on the diagonal of the weighting matrix Ψ. The test is done *off-line* to illustrate the high accuracy of the nonlinear SAM, and the five results are similar in accuracy. Note that these off-line testing cases using the nonlinear SAM with 5 different weighted pseudoinverses (Figure 4) are significantly more accurate than those using the linear SAM (Figure 3b). Table 1 lists the results of off-line testing using Nonlinear SAM with different determinants of the matrix Ψ for the weighted pseudoinverse. The Nonlinear SAM is able to store significantly more data without over-training

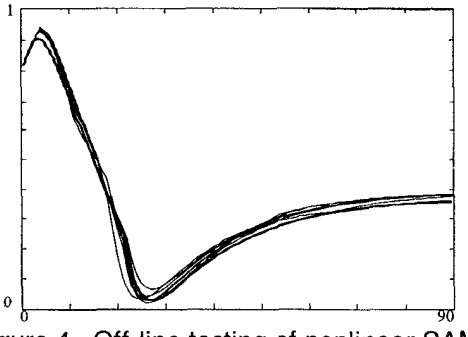

Figure 4. Off-line testing of nonlinear SAM's

Table 1. Results of off-line testing for SAM with different weighted pseudoinverses

Weighting function	error	ε_1	# stored
$W_i = \|x - \bar{x}_i\|_2^{-1}$	0.9601	0.03%	67
$W_i = \|x - \bar{x}_i\|_1^{-1}$	0.9791	0.04%	64
$W_i = \|x - \bar{x}_i\|_{\inf}^{-1}$	1.1644	0.05%	60
$W_i = \frac{\|x - \bar{x}_i\|_1}{\|x - \bar{x}_i\|_{\inf}} - 1$	1.8845	0.05%	53
$W_i = \exp\left(-\|x - \bar{x}_i\|_2 \sigma^{-1}\right)$	1.0473	0.01%	91

4.3. Example 3: logic problem

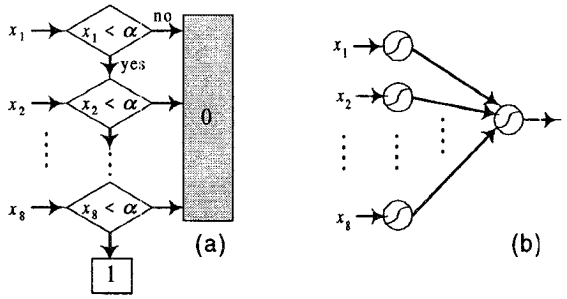

Figure 5. Eight logic gates in series: (a) flow chart, (b) the 8-1 MLP

Given a sample logic system in Figure 5a of eight logic gates in series, where an input $x \in \Re^8$ has to pass through all 8 gates in order to output a 'yes' or '1' (or else it will be 'no' or '0'), a set of 2000 testing data and another set of 2000 testing data are generated such that it is equally probable to pass or fail at each of the 8 gates. This implies that of the 2000 training data there would be approximately $2000*2^{-8}$ or about 8 data out of the set of 2000 with an output of '1', the rest are '0.' With 99.6% of the training set being 0's, the challenge here is to "learn" the 0.4% of the data, the '1's. This is a common situation where the distinctness of the data is a more important property in identifying the system than is the frequency of occurrence of the data.

SAM's performance in the 2000-data test set is compared with a 8-1 neural network with sigmoid activation functions in the input layer (Figure 5b), designed specifically for a logic system: 8 discriminators at input and one at the output. The results are telling: the Logic SAM has identified and stored all 8 of the '1' data out of 2000 in the training set to be novel. The well-tuned 8-1 MLP treated the '1' data like the rest of the '0' data and did not classify *any* point in the testing set as "1" until its 30th cycle. The logic SAM, however, correctly identified 3 out of the 8 ones in the testing data set with just one pass through the training set.

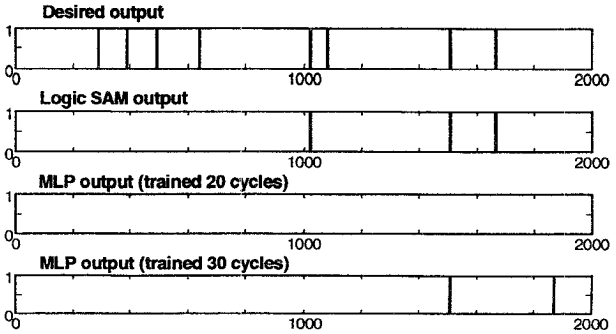

Figure 6. Outputs of the testing set for the logic system

4.4. Example 4: next-day stock trend prediction

SAM is employed in the difficult task of predicting the next-day stock trends. A way to do that would be to first predict the percentage change of the price between today and tomorrow, and use its sign to determine the next-day market trend. The desired training output may be formulated as in equation (12), where $p(n)$ is the price on day n, and $\%p(n+1)$ is the percentage change in the price between day n+1 and n:

$$\%\, p(n+1) = \frac{p(n+1) - p(n)}{p(n)} \tag{12}$$

$$X(n) = \frac{[p_{close}(n) - p_{low}(n)] - [p_{high}(n) - p_{close}(n)]}{p_{high}(n) - p_{low}(n)} \tag{13}$$

From the wide array of combinations of stock market technical indicators, we chose as input a relevant "X-indicator" in (13), which indicates the position of the close price relative to the high and low prices of the day.

An experiment is set up to model next-day market trend predictions of the New York Stock Exchange closing price over 3,081 days (April 1980 through June 1995). A linear SAM system is set up to begin learning the real NYSE data from day 1 and predict the next-day trend each day until day 3,081, making one pass through the data while predicting at each day the stock trend of the next day. To regulate the randomness, the input and the desired output training sets are quantized into different levels in the domain (-1,+1).

Since the market trend prediction may be either "up" or "down," the profit generated since day 1 may be calculated by having the investor buy on "up" predictions and sell on "down" predictions:

$$pr(k) = [1 + ud(k)\%\, p(k)] \cdot pr(k-1), \tag{14}$$

where $pr(k)$ is the profit at day k, ud(k) is the market trend up/down prediction made on day k-1, and $\%p(k)$ is

the percentage price change from equation (12). The results of the SAM is compared with that of using the day-before *X*-indicator and Buy and Hold (B&H), the case where the investor invests everyday.

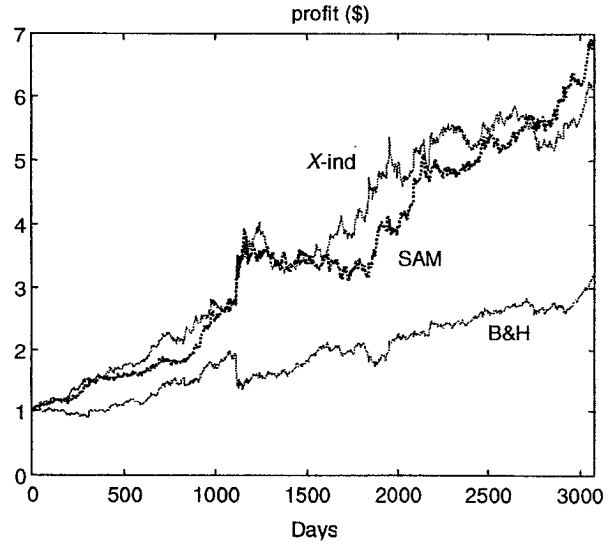

Figure 7. Plot of profits over 3,081 days starting at $1.00 on day one

Table 2. Next-day market trend prediction

Prediction method	% Correct predictions	% "Up" prediction
Buy and Hold	54.30%	100%
SAM training on the X-indicator	52.65%	56.52%
X-indicator	52.00%	56.06%

Figure 7 illustrates SAM's improvement over the *X*-indicator prediction in terms of profit and percentage correct stock market trend prediction. Even though B&H has a higher percentage correct prediction, the accumulated profit from B&H is significantly lower than those of *X*-ind and SAM, indicating that *X*-ind and SAM are more responsive and more accurate in days of large stock price changes. Although SAM has not excelled the *X*-ind by a wide margin, it is consistent over 3,081 days (15 years) of stock data and is a viable tool to improve upon good stock market indicators in the difficult task of stock market prediction.

5. Conclusion

In this paper we have presented the Self-organizing Associative Memory (SAM) system, a departure from the parametric systems for identification of unknown systems that assumes that objects – not rules – constitute the knowledge-forming memory by association, and that only novel and distinct objects need to be organized into memory.

The Linear SAM with locally linear approximation and especially with the nonlinear weighted pseudoinverse SAM achieves high accuracy in the case of off-line test for system identification. Furthermore, the logic problem demonstrates that SAM is apt in identifying unknown systems given sparse or unevenly-distributed data. In the abundant-data and high noise identification case of 3,081 days or 15 years of real stock market data, SAM serves to improve upon a stock trend indicator.

In all, the data approach allows SAM to trade memory for high accuracy and fast, one-pass, learning of the training data. Further studies may result in more versatile nonlinear local functions and possible fuzzy rule or knowledge extraction from the stored data.

6. References

[1] Chen, S. and S.A. Billings, "Representations of non-linear systems: the NARMAX model," *Int'l Journal of Control*, vol. 49, no. 3 (1989) pp. 1013-1032.

[2] Creecy, R.H., B.M. Masand, S.J. Smith, D.L. Waltz, "Trading Mips and Memory for Knowledge Engineering," *Communications of the ACM*, vol. 35, no. 8 (Aug. 1992) 48-64.

[3] Hertz, J., A. Krogh and R.G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.

[4] Hoehfeld, M. and S.E. Fahlman, "Learning with limited numerical precision using the cascade-correlation algorithm," *IEEE Trans. Neural Networks* vol. 3, no. 4 (July, 1992):602

[5] Lippmann, P.R., "An introduction to computing with neural nets," *ASSP Magazine*, vol. 4, no. 2, pp. 4-18, Apr. 1987.

[6] Narendra, K.S. and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Trans. Neural Networks*, vol. 1, no. 1 (Mar. 1990).

[7] Ruck, D.W., S. K. Rogers, M. Kabrisky, P. S. Maybeck, and M. E. Oxley, "Comparative analysis of backpropagation and the extended Kalman filter for training multilayer perceptrons," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 14, no. 6, pp. 686-691, June 1992.

[8] Santina, M., A. R. Stubberud, and G. H. Hostetter. *Digital Control Systems 2nd ed.* Fort Worth: Saunders, 1994.

[9] Salzberg, S. and A.L. Delcher, "Best-Case Results for Nearest-Neighbor Learning." *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 6 (June 1995) 599-608.

[10] Schaal, S. and C.G. Atkeson, "Robot Juggling: Implementation of Memory-Based Learning," *IEEE Control Systems*, vol. 14, no. 1 (Feb. 1994) 57-71.

[11] Waltz, D., "On Reasoning From Data," *ACM Computing Surveys*, vol. 27, no. 3 (Sep. 1995) 356-9.