Proceedings of the American Control Conference
Albuquerque, New Mexico June 1997
0-7803-3832-4/97/$10.00 © 1997 AACC

SOFT SENSING BASED ON ARTIFICIAL NEURAL NETWORK

Yingxu Yang and Tianyou Chai
Research Center of Automation
Northeastern University 309#
Shenyang 110006, Liaoning, P R China

Abstract Soft-sensing or inferential estimation has long been considered a potent tool to deal with the conflict between small control interval and large sampling interval existing in a wide variety of industrial processes. To extend the soft-sensing from linear system to nonlinear case, we propose a nonlinear soft-sensor on the basis of multi-step predication using recurrent neural network and a novel alternating training method especially suitable for slowly sampled primary output. The nonlinear soft-sensor has been demonstrated by simulation results to be able to produce qualified estimation with good convergence speed.

I. INTRODUCTION

In many process control systems there exist a formidable difficulty concerning the large sampling rate of the controlled output, i.e. ,the primary output, due to measurement device limitations. Often, these primary outputs are employed as feedback signals for process control or for other control actions. Their use, however, can cause major and prolonged deviations from set points since disturbance effects remain undetected in between the long sample periods. And these effects cannot be sufficiently overcome by existing advanced controller algorithms and can lead to unsatisfactory, or even unacceptable control system performance. Typical cases occur in product control of distillation columns, biomass concentration control of fermentation process and Kappa number control of pulp digesters.

A traditional way to combat this difficulty is to incorporate secondary outputs, i.e. ,the easily measured variables which are related to the primary output in a certain way into control system. The application studies and publications in this field may date back to 1970's when Brosilow and coworkers first proposed a controller design method termed inferential control [1]. Since then, a large amount of effort and time has been devoted to this field, among them, M.T. Guilandoust and coworkers presented adaptive inferential control algorithm in [2]and [3]concerning standard state-space-based estimator and input-output-based estimator. M.T.Tham and coworkers discussed multirate and multivariable self-tuning control and its application in distillation column study in [4]. T.Mejdell and S.Skogestad summarized several output estimation method in [5]. All this work has been done with the postulate that the controlled plant is linear or at least functioning adjacently at the working point which make it possible to be linearized reasonably. For most real processes, however, especially those chemical industrial processes such as biomass reactor or pulp digester , there exist strong nonlineraity among manipulate inputs, primary outputs, secondary outputs and other relating variables, therefore, the presented method may not achieve good performance, due to their inability to deal with nonlinearity. It was this dilemma that stimulate the research work of nonlinear inferential control.

Inferential control is largely based on the output inferential estimation obtained by finding the relation between primary output and secondary output, hence 'inferential'. Clearly, here you see the output estimator has taken the place of measurement device such as chromatography which is prohibitively expensive. That is using output estimator as sensor rather than the other commonly reckoned apparatus, hence, soft-sensing.

As for nonlinear system, how to get the output estimation remains a main obstacle, fewer work has been done comparing with the linear case. A typical method to obtain nonlinear output estimation is to employ Extended Kalman Filter using detailed nonlinear dynamic model of the plant[6] whose performance may vary with the extent of match between model and plant. Other work employ Artificial Neural Network as a useful tool to deal with the model-based estimator's poor robustness, for more detail, reader may refer to [7, 8, 9]. But there is still no guarantee of perfect convergence speed, avoidance of local minima and most of all, the best use of information available.

In this paper , we propose a nonlinear soft-sensor based on multi-step predication of the output using recurrent neural network with main effort to combat the difficulties mentioned above.

II. MUTISTEP PREDICTIVE SOFT-SENSOR

The soft-sensing problem can be stated as follows (for simplicity, we will employ SISO system to demonstrate, and all the description can be easily extended to MIMO case). In a control process, the

primary output y is generated by the controlled plant at every sample interval T, but mainly due to the limitation of measurement device, the sampling value of y can only be obtained at interval NT. There are other variables which can be sampled at interval T and have more or less relation to y, these are secondary outputs v_i $(i = 1,2,\cdots,p)$. Our objective here is to use easily measured v_i and manipulated input u to estimate unmeasured y, as shown below.

$$y_{unmeasured} = f(u, v_{measurable}) \qquad (2.1)$$

with $\{v_{measurable}\} = \{v_1, v_2, \cdots, v_p\}$.

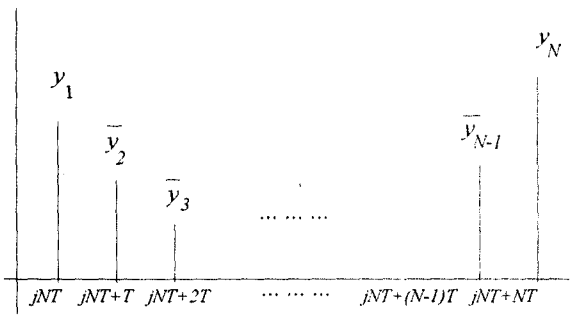

Fig. 2.1 Insert estimated primary output

We may view the idea in another way as shown in Fig 2.1, i.e., with the fixed primary output y_{jNT} obtained every other NT, our work is to insert or predict estimated primary outputs $\bar{y}$ between y_{jNT} correctly enough to approximate the real output generated by plant, and for a single slow sampling internal NT, the work is to predict $\bar{y}$ at the instant of $jNT+T, jNT+2T, \cdots, jNT+(N-1)T$.

Suppose that the plant can be described by the equation:

$$y(k+1) = f(y(k), y(k-1), \cdots, y(k-n+1), u(k), u(k-1), \cdots, u(k-m+1)) \qquad (2.2)$$

with $f(\cdot)$ as a nonlinear function. This function $f(\cdot)$ can be taken as a one-step predictor, since we get $y(k+1)$ using the present and past values of y and u. Could we get multi-step predictor the same way?

Here we will utilize generalized prediction as our tool to accomplish the task. In [10], Clarke proposed GPC method first to predict the outputs within prediction horizon then manipulate control input to let them follow the reference trajectory. We incorporate the first part into our work.

For a nonlinear system,a similar multi-step predictor is obtained by the following deduction:

$$\begin{aligned} y(k+1) &= f(y(k),\cdots,y(k-n+1);u(k),\cdots,u(k-m+1)) \\ &= f(y(k),\cdots,y(k-n+1);(k-1)+\Delta u(k),\cdots,u(k-m+1)) \\ &= F_1(y(k),\cdots,y(k-n+1);u(k-1),\cdots,u(k-m+1);\Delta u(k)) \end{aligned}$$

……

$$\begin{aligned} y(k+j) &= f(y(k-1+j),\cdots,y(k-n+j);u(k+j-1),\cdots,u(k-m+j)) \\ &= f(F_{j-1}(\cdots,\cdots,\cdots),\cdots,u(k-1)+\sum_{i=0}^{j-1}u(k+i),\cdots) \\ &= F_j(y(k),\cdots,y(k-n+1);u(k-1),\cdots,u(k-m+1); \\ &\quad \Delta u(k),\cdots,\Delta u(k+j-1)) \\ &\quad (j = 1,2,\cdots,N) \end{aligned} \qquad (2.3)$$

Finally we get :

$$Y^N{}_k = F(Y_k, U_k, \Delta U^N{}_k)$$

$$Y^N{}_k = (y(k+1),\cdots,y(k+N))'_N$$

$$Y_k = (y(k),\cdots,y(k-n+1)'_n$$

$$U_k = (u(k-1),\cdots,u(k-m+1))'_{m-1}$$

$$\Delta U^N{}_k = (\Delta u(k),\cdots,\Delta u(k+N-1))'_N$$

$$F = (F_1(\cdots,\cdots,\cdots),\ldots,F_N(\cdots,\cdots,\cdots))_N)$$

and $F(.)$ is the N-step predicator to be built.

To realilze such a complicated function as $F(.)$, using ANN seems to be desirable. However, according to (2.3), we must supply $\{y(k),\cdots,y(k-n+1)\}$ as part of the inputs to ANN to obtain $y(k+j)$, and among them, $\{y(k-1),\cdots,y(k-n+1)\}$ are last time outputs from ANN, i.e. , they should be marked as $\{\bar{y}(k-1),\cdots,\bar{y}(k-n+1)\}$ and therefore (2.3) has to be changed into

$$\begin{aligned} y(k+j) = F(&y(k), y(k-1),\cdots,y(k-n+1); \\ &u(k-1),\cdots,u(k-m+1);\Delta u(k),\cdots,\Delta u(k+j-1)) \\ &j = 1,2,\cdots,N \end{aligned} \qquad (2.4)$$

It has been demonstrated by Narandra and co-worker [11] that a parallel identification model like (2.4) is not preferable to generate stable laws, since even if the plant is bounded-input bounded-output stable, there is no guarantee that the parameters will converge or that the output error will tend to zero. And in spite of two decades of work, conditions under which the parallel model parameters will converge even in linear case are at present unknown.

To overcome this difficulty, we resort to (2.1), where

$$y_{unmeasured} = f(u, \{v_{measurable}\}).$$

For simplicity, assume there is only one secondary output, and we suppose

$$y(k) = f(u(k-1),\ldots,u(k-m);v(k-1),\ldots,v(k-q)) \quad (2.5)$$

Substitute $\{\bar{y}(k-1),\cdots,\bar{y}(k-n+1)\}$ by (2.4) will obtain the changed form of (2.4)

$$y(k+j) = G_j(y(k);v(k-2),\cdots,v(k-q-n+1);\ u(k-2),\cdots,u(k-m-n+1);\Delta u(k),\cdots\Delta u(k+j-1)) \quad j=1,2,\cdots,N \quad (2.6)$$

Remark 1: As has been shown in (2.6), for every NT, we will get $y(k+j)(j=1,\cdots,N-1)$ between two slowly sampled output values just as shown in Fig. 2.1, and with G_j well chosen, $\bar{y}_j$ will approximate y_j adequately.

Remark 2: $\{\Delta u(k),\cdots,\Delta u(k+j-1)\}(1,2,\cdots,N)$ has been used as part of the network inputs, which makes difference from GPC whose future manipulated input is unknown and could only be obtained through minimizing the cost function.As for inferential estimation, the interval between two sampled instants is not actually 'future', hence $\{\Delta u(k),\cdots,\Delta u(k+j-1)\}(1,2,\cdots,N)$ is known and what we are supposed to do is just to estimate the already generated output. Furthermore, the use of $\{\Delta u(k),\cdots,\Delta u(k+j-1)\}(1,2,\cdots,N)$ demonstrates the proposed algorithm's ability to have a better use of available information.

Remark 3: The direct obtaining of relationship G_j is considered time and computation consuming and if possible, finally result in very complicated form. Here, we try to employ Artificial Neural Network as a powerful tool to realize G_j.

III. RECURRENT NEURAL NETWORK IN SOFT-SENSING

Our work requires a system that will store and update context information, i.e., information computed from the past inputs and useful to produce desired outputs. Recurrent neural networks are well suited for this task because they have an internal state that can represent context information. The cycles in the graph of a recurrent network allow it to keep information about past inputs for an amount of time that is not fixed a priori, but rather depends on its weights and on the input data. In contrast , static networks (i.e., with no recurrent connection), even if they include delays (such as time delay neural networks [12]), have a finite impulse response and can't store a bit of information for an indefinite time.

3.1 Inner Recurrent Network and Its Training Method

The topology of a typical Inner recurrent Network is shown in Fig 3.1. It has three layers--input layer, hidden layer and output layer with input and output nodes linear while hidden nodes nonlinear.

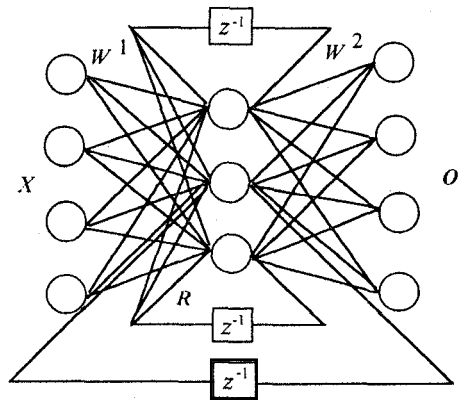

Fig. 3.1 Inner Recurrent neural network

For inner Recurrent Network, only hidden nodes have feedbacks with the corresponding weights denoted R. It's dynamics can be described as

$$O(k) = w^1 h(k)$$
$$h(k) = f(w^2 x(k) + Rh(k-1) - T)$$

Since there are a number of approaches to train recurrent network by gradient-based algorithms[13], it's not necessary for us to present them in detail. Instead, our concentration will be on some novel reforms.

To speed up the training process, we consider to use Le's Conjugate Gradient method for unconstrained optimization[14]. Le's Conjugate gradient method is a type of conjugate gradient method with dynamic optimization of step size $a^{(q)}$. In each iteration q , the vector of interconnection weights is improved by $w^{(q+1)} = w^{(q)} + a^{(q)} s^{(q)}$ where $a^{(q)}$ is chosen to minimized w along the search direction $s^{(q)}$. The algorithm generates n mutually conjugate directions and minimizes a positive definite quadratic function of dimension m in at most n steps, where n is the size of the vector w . The sequence of search direction $s^{(q)}$ are formed by linear combinations of the current steepest descent direction and the previous search direction.

3.2 Alternating Training

Take a closer look at (2.6)

$$y(k+j) = G_j(y(k); v(k-2), \cdots, v(k-q-n+1);$$
$$u(k-2), \cdots, u(k-m-n+1); \Delta u(k), \cdots \Delta u(k+j-1))$$

for each training step, N output $\bar{y}_1(k), \bar{y}_2(k), \cdots, \bar{y}_N(k)$ will be produced, but only one desired output $y_N(k)$ will be available to be compared with. To have a better use of $y_N(k)$, for every N steps, at the first step, compute the error $E = y_N(k) - \bar{y}_N(k)$ and modify the linking weights, then , at the next step, let $E = y_N(k) - \bar{y}_{N-1}(k)$ and make another modification of the weights. Follow the same procedure until $E = y_N(k) - \bar{y}_1(k)$ has been used. Thus in every N steps $y_N(k)$ has been compared with $\bar{y}$ for N times and the weights have been modified for N times rather than just one time. This alternating training method is supposed to be able to result in much higher convergence speed than those batch comparison methods.

IV. SIMULATION RESULTS AND DISCUSSION

To demonstrate the effectiveness of our algorithm, three examples have been incorporated.

Example 1.

$$y(k+1) = 0.5\sin(y(k)) + 0.4\cos(u(k))$$
$$y(k) = 0.7\sqrt{v(k-1)} - 0.4u(k-1)^3$$

with $n = 1, m = 1, q = 1, N = 4$, and we use 5-15-4 Inner Recurrent Network with activation function $f(x) = 1 - e^x / 1 + e^x$, learning rate $\eta = 0.1$, $a^{(1)} = 0.85$, the result is shown in Fig.4.1 .

Example 2.

$$y(k+1) = 0.8\sin(y(k-1)) + \sqrt[3]{u(k)}$$
$$y(k) = \sqrt{v(k-1)} + 0.6u(k-1)v(k-2) + 0.3u(k-2)$$

with $n = 1, m = 2, q = 2, N = 3$. and we use 6-20-3 Inner Recurrent Network with learning rate $\eta = 0.15$, the result is shown in Fig.4.2.

Example 3.

$$y(k+1) = 0.2y(k)^2 - 0.5y(k-1)u(k)$$
$$y(k) = 0.7\sqrt{v(k-1)} + 0.3u(k-1)$$

with $n = 1, m = 1, q = 1, N = 2$, and we use 5-20-2 Inner Recurrent Network with learning rate $\eta = 0.15$, the result is shown in Fig. 4.3.

It is demonstrated by the simulation results that our algorithm can drive the network to produce estimated primary output approximate to the real output well enough. To show its good convergence speed , see Fig. 4.4 ,where you may notice that the error generated by standard B-P algorithm decreases much more slowly than by our alternating training algorithm with Le's Conjugate method after the error has reached a certain level .

However, there is a conflict that to eliminate local minima, as demonstrated in Theorem, we have to incorporate enough inputs, it's the increase in the number of inputs that will increase the training time, and as the sampling rate decreases, the same happens. But this would be of concern only when on-line weight adaptation is needed. Ideally this should not be required if a representative non-linear neural-based process model can be found.

As you may notice that for all of the three examples , we use three layer networks , that is mainly because higher order systems map high order nonlinearities of system, which could be regarded as noise.

V. CONCLUSIONS

The soft-sensor proposed in this paper may be used to extend inferential estimation from linear process to nonlinear case.And we borrow some idea from Generalized Predictive Control on which a multi-step estimator is presented. Finally neural network has been used to realize the predictor since its structure and parameters are unknown and if really known , very complicated. A novel learning algorithm especially suitable for inferential estimation has been used to train the network.But we still have a postulate that the order of the plant is known without which the inputs to network will be uncertain and that may result in bad estimation.A condition under which the given cost function may be local minima free has also been proposed though may lead to large number of inputs. Future work should be on how to eliminate the knowledge of plant order and to avoid local minima adequately.

REFERENCES

1. B. Joseph and C.B. Brosilow. Inferential Control of Processes. AIChE Journal , Vol. 24, No. 3, pp. 485-508, 1978.

2. M.T. Guilandoust, A.J. Morris and M.T. Tham. Adaptive Estimation in Inferential Process Control. 1987 ACC, pp. 1743-1748.

3. M.T. Guilandoust, A.J. Morris and M.T.Tham. Adaptive Inferential Control. Proc. IEE Pt.d. Vol. 134, No. 3, pp.171-179, 1987.
4. M.T. Tham, F.Vagi, A.J. Morris,and R.K.Wood. Multivariable and Multirate Self-tuning Control: A Distillation Column Case Study. Proc. IEE Pt.d, Vol. 138, No. 1, pp. 9-24, 1991.
5. T. Mejdell and S. Skogestad. Output Estimation Using Multiple Secondary Measurements: High-Purity Distillation. AIChE Journal, Vol.39, No.10, pp.1642-1653, 1993.
6. J.H.Lee and A.K.Datta. Nonlinear Inferential Control of Pulp Digesters. AIChe Journal , Vol.40, No.1, pp. 50-64, 1994.
7. A.J. Willis, C. Di Massimo, G.A.Montague, M.T. Tham, A.J. Morris. Artificial Neural Network in Process Engineering. Proc. IEE Pt.d, Vol.138, No.3, pp. 256-266,1991.
8. A.G.Hofland, A.J. Morris and G.A. Motague. Radial Basis Function Networks Applied to Process Control. 1992 ACC, pp.480-484.
9. M.N. Karim and S.L. Bivera. Application of Neural Networks in Bioprocess State Estimation. 1992 ACC, pp. 495-499.
10. D.W. Clarke, C. Mohtadi and P.S. Tuffs. Generalized Predictive Control --- Part I . The Basic Algorithm. Automatica, vol.23, No.2, pp.137-148, 1987.
11. G.S. Narandra and K. Parthasarathy. Identification and Control of Dynamical Systems Using Neural Networks. IEEE Trans. NN, Vol.1, No.1, pp.4 -27, 1990.
12. K.J. Lang and G.E. Hinton. The Development of the Time-delay Neural Network Architecture for Speech Recognition. Technical Report CMU-CS-88-152, Carnegie-Mellon University, 1988.
13. O.Nerrand, P.Roussel-Ragot, D.Urbani, L.Personnaz, and G. Dreyfus. TrainingRecurrent Neural Networks: Why and How? An Illustration in Dynamical Process Modelling. IEEE Trans. NN, Vol.5, No.2, pp.178-184, 1995.
14. D.Le. A Fast and Robust Unconstrained Optimization Method Requiring Minimm Storage. Math. Prog. , 32, pp.41-68, 1985.
15. M. Bianchini, M. Gori, and M. Maggini. On the Problem of Local Minima in Recurrent Neural Networks. IEEE Trans. NN, Vol.5, No.2, pp.167-177, 1995.

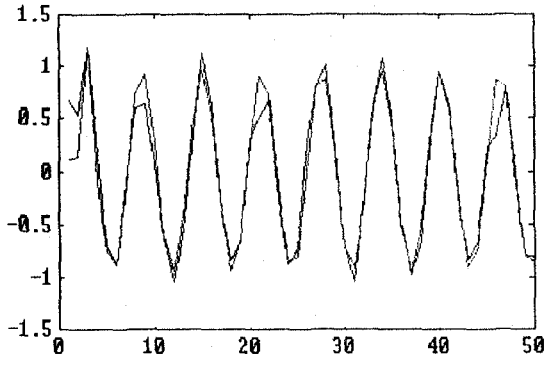

Fig. 4.1 Estimated output with IRNN 5-15-4

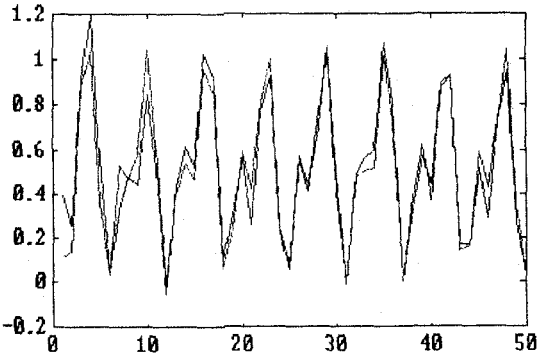

Fig. 4.2 Estimated output with IRNN 6-20-3

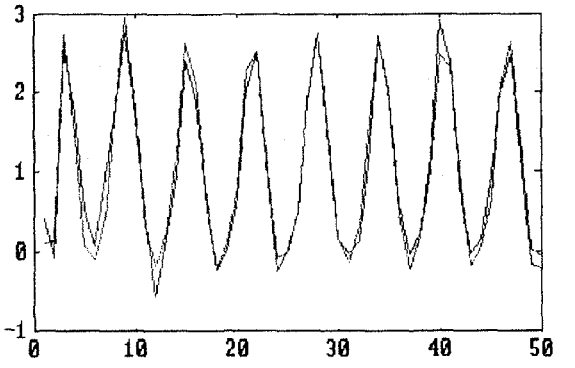

Fig. 4.3 Estimated output with IRNN 5-20-2

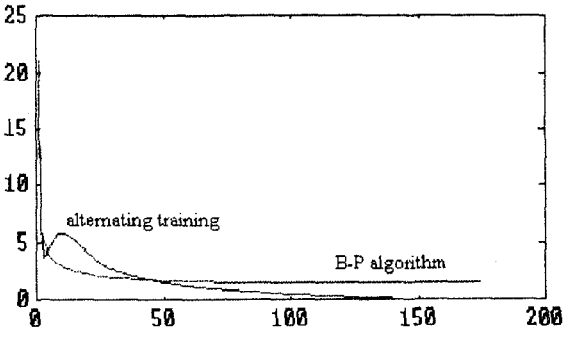

Fig. 4.4 ISE against iterations