Industrial applications of neural networks, F. Fogelman, P. Gallinari, eds (World Scientific, 1996)

KNOWLEDGE-BASED NEURAL MODELING: PRINCIPLES AND INDUSTRIAL APPLICATIONS

J.L. PLOIX
NETRAL S.A.
14, rue Verdi
92130 ISSY LES MOULINEAUX

G. DREYFUS
Ecole Supérieure de Physique et de Chimie Industrielles
Laboratoire d'Electronique
10, rue Vauquelin
75005 PARIS

A methodology for designing semi-physical neural models is presented. Starting from a mathematical model of the process, a recurrent neural network is constructed, and some of its weights are adjusted from the measurements by training. The application of this methodology to the modeling of an industrial distillation column, designed for early fault detection, is described.

The use of neural networks as black-box models of dynamical processes has been more and more widespread during the past years. It is a conceptually straightforward consequence of the universal approximation property of neural nets, and it has proved very useful in many applications. However, the black-box character of these models has often been criticized, since it is indeed wasteful to throw away all the information that can be gathered from a physico-chemical analysis of the process, even though this information may be incomplete or insufficiently accurate. In the present paper, we introduce a methodology, termed *knowledge-based neural modeling*, which allows us to take advantage of the available mathematical knowledge, while retaining the flexibility of neural black-box models. The principles of knowledge-base neural modeling are first be presented; it is illustrated by the simulation of an industrial distillation column.

1. Principles of knowledge-based neural modeling

When modeling a dynamical process, two situations may be encountered:

- the mathematical knowledge on the process is very inaccurate, or possibly non-existent: in such a case, the only possibility is black-box modeling. The use of neural networks for such purposes has been investigated extensively: the optimal predictor structures and the optimal training algorithms under various noise assumptions (NARX models, NARMAX models, Output Error models, etc.) have been derived[1] and applied to various processes, simulated and real;

- a state-space model can be derived from a physical analysis of the process; this model is reasonably accurate, but the approximations made in the derivation of the equations, the insufficient knowledge of some phenomena, or the uncertainties on the numerical values of the parameters of the model, make it unsuitable for the purpose that it should serve. The key idea of the present paper is that, despite its shortcomings, such a model can be used as a basis for designing an accurate neural model.

Assume that the state-space model is of the form:

$$\frac{dx}{dt} = f\left[x(t), u(t)\right]$$
$$y(t) = g\left[x(t)\right]$$

where f and g are known analytically. In a typical chemical engineering unit, the number of state variables would be on the order of one (or a few) hundred.

The state equations can be discretized to

$$x(k+1) = x(k) + f\left[x(k), u(k)\right]$$
$$y\left(k+1\right) = g\left[x\left(k+1\right)\right] \qquad (1)$$

by Euler's method (other discretization techniques can be used as well). If two feedforward neural networks can be trained to approximate functions f and g, then a network such as shown on Figure 1 obeys the same discrete-time equation as the model.

Since $x(k)$ is a vector, f is a vector too; therefore, instead of using a single network for approximating the whole vector f, it is generally advantageous to use different networks for different components f_i of f. Several situations may arise:

- function f_i (known analytically) can readily be computed: it can therefore be put simply into a "neural" form; this is a purely formal step, which is just intended to ease the implementation of the whole model as a neural network ;
- the computation of function f_i (known analytically) is time-consuming: for instance, one has :

  $$\frac{\mathrm{d}x_i}{\mathrm{d}t} = f_i\left[x(k), T\left[x(k), u(k)\right], u(k)\right]$$
  $$\text{with } \Gamma\left\{x(k), T[x(k), u(k)], u(k)\right\} = 0$$

  where Γ is known analytically; the computation of the value of $x_i(k+1)$ requires solving the second equation at each time step. In such a case, it may be advantageous to generate a set of representative sequences by solving numerically the above equations, and to use these sequences for training and validating neural network #i. Since a properly designed neural network uses a very small number of neurons, the time necessary for a trained network to compute $x_i(k+1)$ can be smaller than the time necessary for solving the above equations by several orders of magnitude. The same considerations apply to function g.
- function f_i is known with very poor accuracy: it can be implemented purely in a black-box fashion.

Thus, at the end of this step, one has a neural network which performs exactly as well - or as badly - as the state-space model.

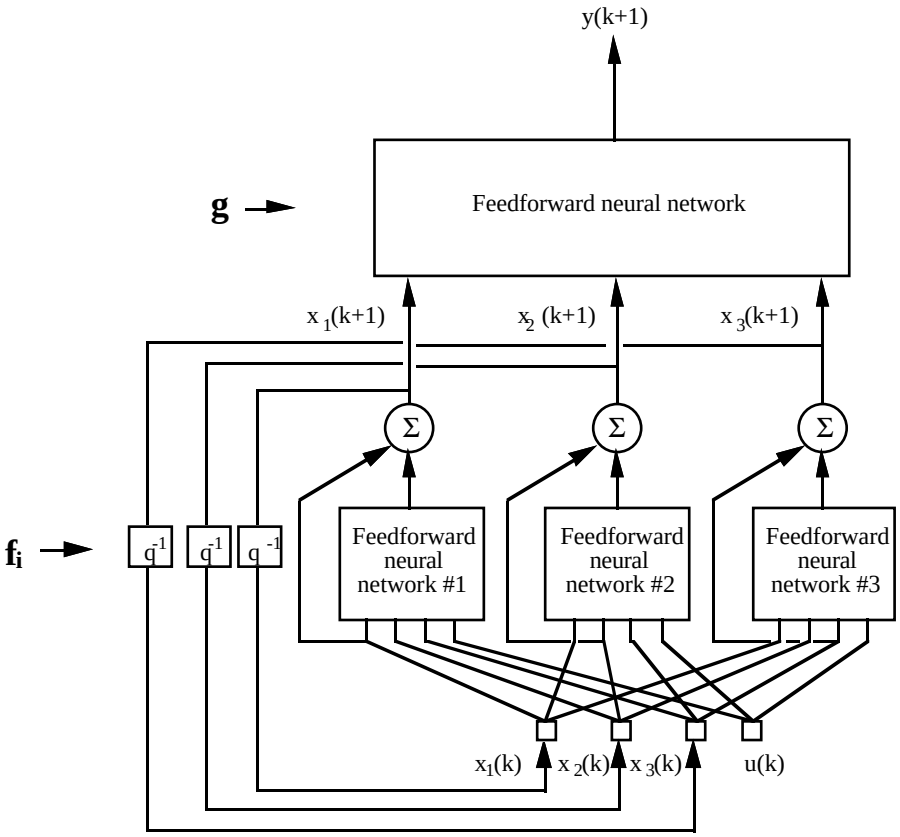

Figure 1 A third-order network obeying equations (1)

As a final step, the knowledge-based neural model is trained with sequences measured on the process itself. In this step, not all weights are adjustable: since most weights of the network have a physical meaning, those which are known to be accurate and not to require any adjustment are kept fixed during training. The only adjustable weights are the weights of the black-box networks (if any), and the weights whose values are not known accurately from theory.

One of the benefits of this procedure is that it allows the designer to divide the global task into smaller subtasks, which greatly facilitates the analysis of the behaviour of the network.

2. An industrial application

We illustrate the above methodology by the real-time simulation of a distillation column which is a part of a steam-cracking unit. The purpose of the model is early anomaly detection: if an accurate model of the normal operation of the column is available in real time, anomaly detection can be performed by detecting statistically significant differences between the predictions of the model and the actual measurements performed on the column.

The distillation column under consideration processes a mixture of three main species, with impurities up to 9 %. The measurements taken on-site are the temperatures of eight trays in the column, and the mole fractions of the components at the top and at the bottom of the column. Therefore, there are only 10 measurable variables. The state variables of the process are the mole fractions of two species at each tray; the dynamics of the whole column can be described, as a first approximation, by a set of 51 coupled non-linear differential equations[2, 3] each of them involving 2 state variables, thereby combining into a model with 102 state variables. These equations could be solved numerically, but the required accuracy necessitates the use of much more complex models, which cannot be solved in real time on personal computers. Therefore, the use of a knowledge-based neural model is an attractive alternative to a pure knowledge model.

In a simple model, each tray is described by the following differential equations:

$$M_i \dot{x}_{a,i} = L\, x_{a,i-1} + V\, y_{a,i+1} - L\, x_{a,i} - V\, y_{a,i} \quad ,$$

$$M_i \dot{x}_{b,i} = L\, x_{b,i-1} + V\, y_{b,i+1} - L\, x_{b,i} - V\, y_{b,i} \quad ,$$

where $x_{a,i}$ is the mole fraction of component a in the liquid phase at tray i, $y_{a,i}$ is the mole fraction of component a in the vapour phase at tray i, M_i is the mass of material held at tray i, V and L are the vapour and liquid flows, assumed (in this simple model) to be constant throughout the column. The mole fraction of a given component in the liquid phase is related to its mole fraction in the vapour phase by the equations of thermodynamics:

$$y_{a,i} = G_a\left(P, x_{a,i}, x_{b,i}\right)$$

$$y_{b,i} = G_b\left(P, x_{a,i}, x_{b,i}\right) \quad ,$$

where P is the pressure, and where G_a and G_b are known functions.

Since the measurable variables are temperatures, the relation between the temperature and the mole fractions at tray i must be taken into account :

$$T_i = \Theta\left(P,\ x_{a,i},\ x_{b,i}\right)$$

Applying the methodology described in the previous section, each tray was modeled by two small neural networks (3 hidden neurons each) implementing the right-hand side of the discretized non-linear equations of the process. The external inputs of the network that models tray i are the control inputs P, L, V, and the mole fractions of components a and b at trays i-1 and i+1, at time t ($x_{a,i-1}(t)$, $x_{b,i-1}(t)$, $x_{a,i+1}(t)$, $x_{b,i+1}(t)$); the state inputs of each tray model are the mole fractions of components a and b at time t ($x_{a,i}(t)$, $x_{a,i}(t)$. The outputs of each tray network are the mole fractions of components a and b at tray i at time t+1 ($x_{a,i}(t+1)$, $x_{a,i-1}(t+1)$.

These networks were stacked (with shared weights) so as to build a complete model of the column. The resulting canonical form[1] of the recurrent network is shown on Figure 3; it has 9 inputs, 10 outputs, 2 of which (mole fractions) are state variables and 8 of which (temperatures) are related to state variables by non-linear relations. No measurement can be performed on the other state variables. Due to weight sharing, and to many weights being fixed by the physics of the process, the network has 32 adjustable weights only, although the total number of neurons is in excess of one thousand. Algorithms used for training recurrent networks for dynamical process identification, when some state variables are not measured (thus cannot be assigned desired values during training), have been described previously[1, 4].

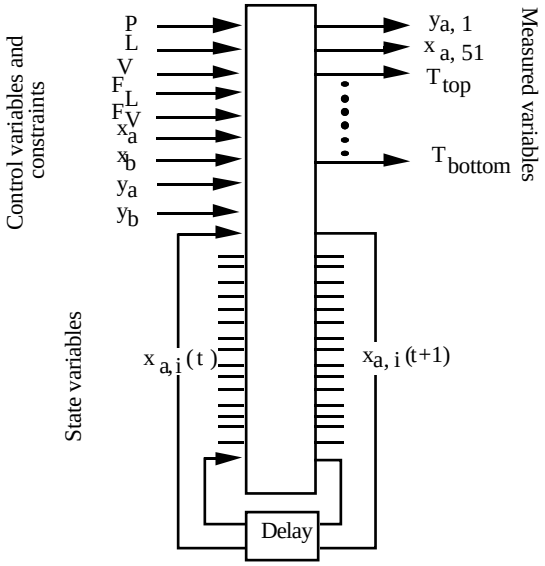

Figure 3 The canonical form of the neural model

This model is implemented on a standard PC 486 running at 66 MHz. One minute of computation allows the simulation of six minutes of real operation of the column, which is faster, by at least one order of magnitude, than dynamical models of equivalent accuracy. Detailed results have been reported in a previous paper[5].

3. Conclusion

The present paper introduces an original neural modeling methodology which allows the designer of a model of a non-linear dynamical process to take advantage of the existing mathematical knowledge while retaining the flexibility of neural black-box modeling. This methodology thus circumvents the frequently criticized black-box character of neural net models, and opens the way to the modeling of large, complex non-linear dynamical systems in industry. Similar concepts have been applied to the automatic control of an autonomous four-wheel-drive vehicle[6].

Acknowledgments

The authors are very grateful to Jean-Pierre Corriou, who made this work possible, and to Léon Personnaz and Isabelle Rivals for helpful comments and suggestions.

References:

1. O. Nerrand, P. Roussel-Ragot, L. Personnaz, G. Dreyfus, S. Marcos, Neural Computation 5, 165 (1993).

2 M. Rovaglio, E. Ranzi, G. Biardi, T. Faravelli, Computers Chem. Engng. 14, 871 (1990).

3 L. Lang, E.D. Gilles, Computers Chem. Engng. 14, 1297 (1990).

4 O. Nerrand, D. Urbani, P. Roussel-Ragot, L. Personnaz, G. Dreyfus, IEEE Transactions on Neural Networks 5, 178 (1994).

5 J.-L. Ploix, G. Dreyfus, J.-P. Corriou, D. Pascal, in *Neural networks and their applications*, ed. J. Hérault (1994).

6 I. Rivals, D. Canas, L. Personnaz, G. Dreyfus, IEEE Conference on Intelligent Vehicles (Paris, 1994).