Computational Intelligence
Lecture 19 Identification Using Neural Networks

Farzaneh Abdollahi

Department of Electrical Engineering

Amirkabir University of Technology

Fall 2010

- Engineers desired to model the systems by mathematical models.
- This model can expressed by operator f from input space u into an output space y.
- **System Identification problem:** is finding $\hat{f}$ which approximates f in desired sense.
  - Identification of static systems: A typical example is pattern recognition:
    - Compact sets $u_i \in \mathcal{R}^n$ are mapped into elements $y_i \in \mathcal{R}^m$ in the output
  - Identification of dynamic systems: The operator f is implicitly defined by I/O pairs of time function $u(t), y(t), t \in [0, T]$ or in discrete time:

$$y(k+1) = f(y(k), y(k-1), ..., y(k-n), u(k), ..., u(k-m)), \tag{1}$$

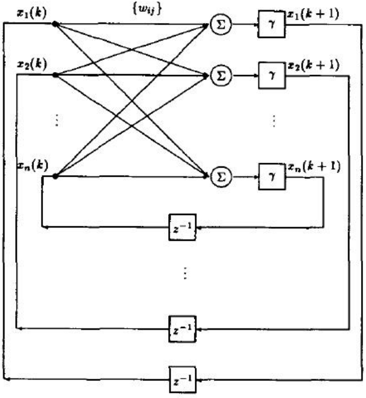
- In both cases the objective to determine $\hat{f}$ is

$$\|\hat{y} - y\| = \|\hat{f} - f\| \leq \epsilon, \text{for some desired} \epsilon > 0.$$

- Behavior of systems in practice are mostly described by dynamical models.
- $\therefore$ Identification of dynamical systems is desired in this lecture.
- In identification problem, it is always assumed that the system is stable

Representation of Dynamical Systems by Neural Networks

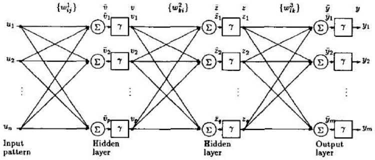
1. Using Dynamic Networks: Time-Delay Neural Networks (TDNN) [1] , Recurrent networks such as Hopfield:
   - Consists of a single layer network N_1, included in feedback configuration and a time delay
   - Can represent discrete-time dynamical system as : $x(k+1) = N_1[x(k)], \quad x(0) = x_0$
   - If N_1 is suitably chosen, the solution of the NN converge to the same equilibrium point of the system.
   - In continuous-time, the feedback path has a diagonal transfer matrix with $1/(s-\alpha)$ in diagonal.
   - $\therefore$ the system is represented by $\dot{x} = \alpha x + N_1[x] + I$



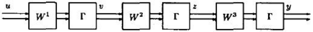
The Hopfield network.

Representation of Dynamical Systems by Neural Networks

2. Using Static Networks: Providing the dynamics out of the network and apply static networks such a multilayer networks (MLN).
   - Consists of an input layer, output layer and at least one hidden layer
   - In fig. there are two hidden layers with three weight matrices W_1, W_2 and W_3 and a diagonal nonlinear operator Γ with activation function elements.
   - Each layer of the network can be represented by $N_i[u] = \Gamma[W_i u]$.
   - The I/O mapping of MLN can be represented by $y = N[u] = \Gamma[W_3 \Gamma[W_2 \Gamma[W_1 u]]] = N_3 N_2 N_1[u]$
   - The weights W_i are adjusted s.t minimize a suitable function of the error between the network output y and desired output y_d.



A three layer neural network.



A block diagram representation of a three layer network.

Using Static Networks

- The *universal approximation theorem* stated in [2] shows that a three layers NN with a backpropagation training algorithm has the potential of behaving as a universal approximator
- **Universal Approximation Theorem:** *Given any* $\epsilon > 0$ *and any* $\mathcal{L}_2$ *function* $f : [0,1]^n \in \mathcal{R}^n \rightarrow \mathcal{R}^m$, *there exists a three-layer backpropagation network that can approximate* f *within* ϵ *mean-square error accuracy.*

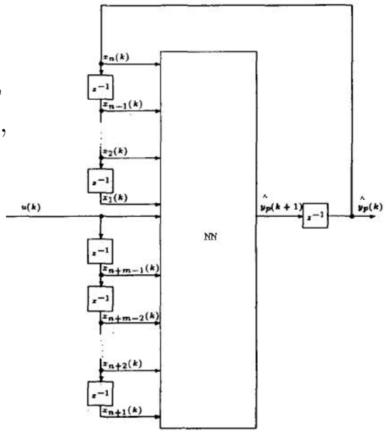
Using Static Networks

- Providing dynamical terms to inject to static networks:
  1. Tap-Delay-Lines (TDL): Consider (1) for identification

$$y(k+1) = f(y(k), y(k-1), ..., y(k-n), u(k), ..., u(k-m)),$$

    - Dynamical terms $u(k-j), y(k-i)$ for $i = 1, ..., n, j = 1, ..., m$ is made by delay elements out of the network and injected to the network as input.
    - The static network is employed to approximate the function f
    - $\therefore$ The model provided by the network will be

$$\hat{y}(k+1) = \hat{f}(\hat{y}(k), \hat{y}(k-1), ..., \hat{y}(k-n), u(k), ..., u(k-m)),$$



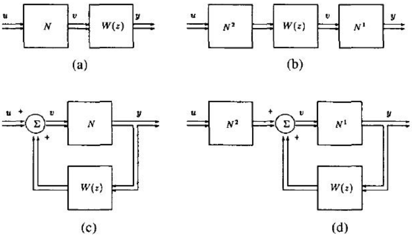
Using Static Networks

2 Filtering

- in continuous-time networks the delay operator can be shown by integrator.
- The dynamical model can be represented by an MLN , $N_1[.]$, + a transfer matrix of linear function, $W(s)$.
- For example:

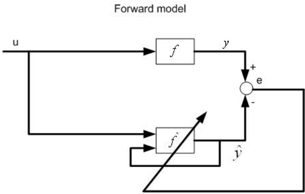
$$\dot{x}(t) = f(x, u) \pm Ax,$$

- where A is Hurwitz. Define $g(x, u) = f(x, u) - Ax$
- $\dot{x} = g(x, u) + Ax$
- Fig, shows 4 configurations using filter.



Neural Networks Identification Model

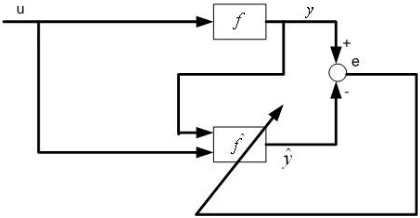
- Two principles of identification problems:
  1. Identification model
  2. Method of adjusting its parameters based on identification error $e(t)$
- **Identification Model**
  1. Direct modeling:
     - it is applicable for control, monitoring, simulation, signal processing
     - The objective: output of NN $\hat{y}$ converge to output of the system $y(k)$
     - $\therefore$ the signal of target is output of the system
     - Identification error $e = y(k) - \hat{y}(k)$ can be used for training.
     - The NN can be a MLN training with BP, such that minimizes the identification error.
     - The structure of identification shown in Fig named **Parallel Model**



Direct Modeling

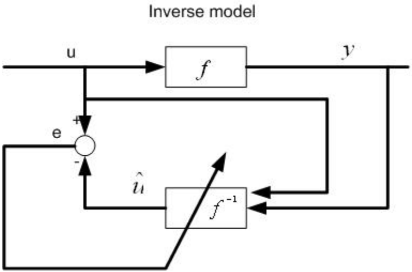
- Drawback of parallel model: There is a feedback in this model which some times makes convergence difficult or even impossible.

2. **Series-Parallel Model**
    - In this model the output of system is fed to the NN



Inverse Modeling

- It is employed for the control techniques which require inverse dynamic
- Objective is finding f^{-1}, i.e., $y \rightarrow u$
- Input of the plant is target, u
- Error identification is defined $e = u - \hat{u}$



Example 1: Using Filtering

- Consider the nonlinear system

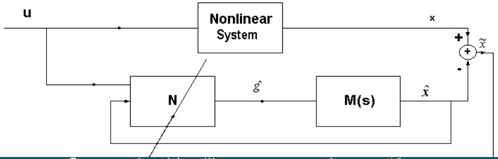
$$\dot{x} = f(x,u) \qquad (2)$$

  - $u \in R^m$: input vector, $x \in R^n$: state vector, $f(.)$: an **unknown** function.
  - Open loop system is stable.
  - Objective: Identifying f
- Define filter:
  - Adding Ax to and subtracting from (2), where A is an **arbitrary** Hurwitz matrix

$$\dot{x} = Ax + g(x,u) \qquad (3)$$

where $g(x,u) = f(x,u) - Ax$.

- Corresponding to the Hurwitz matrix A, $M(s) := (sI - A)^{-1}$ is an $n \times n$ matrix whose elements are stable transfer functions.



- The model for identification purposes:

$$\dot{\hat{x}} = A\hat{x} + \hat{g}(\hat{x}, u)$$

- The identification scheme is based on the *parallel* configuration
  - The states of the model are fed to the input of the neural network.
  - an MLP with at least three layers can represent the nonlinear function g as:

$$g(x,u) = W\sigma(V\bar{x}) + \epsilon(x)$$

- W and V are the ideal but **unknown** weight matrices
- $\bar{x} = [x \quad u]^T$, $\epsilon(x) \leq \epsilon_N$ is the neural network's bounded approximation error,
- $\sigma(.)$ is the transfer function of the hidden neurons that is usually considered as a sigmoidal function:

$$\sigma_i(V_i\bar{x}) = \frac{2}{1 + exp^{-2V_i\bar{x}}} - 1$$

- where V_i is the *ith* row of V,
- $\sigma_i(V_i\bar{x})$ is the *ith* element of $\sigma(V\bar{x})$.

- g can be approximated by NN as

$$\hat{g}(\hat{x}, u) = \hat{W}\sigma(\hat{V}\hat{\bar{x}})$$

- The identifier is then given by

$$\dot{\hat{x}}(t) = A\hat{x} + \hat{W}\sigma(\hat{V}\hat{\bar{x}})$$

- the error dynamics:

$$\dot{\tilde{x}}(t) = A\tilde{x} + \tilde{W}\sigma(\hat{V}\hat{\bar{x}}) + w(t)$$

  - $\tilde{x} = x - \hat{x}$: identification error
  - $\tilde{W} = W - \hat{W}$, $w(t) = W[\sigma(V\bar{x}) - \sigma(\hat{V}\hat{\bar{x}})] + \epsilon(x)$ is a bounded disturbance term, i.e, $\|w(t)\| \leq \bar{w}$ for some pos. const. $\bar{w}$, due to the sigmoidal function.
- Objective function $J = \frac{1}{2}(\tilde{x}^T\tilde{x})$

- **Training:**
  - Updating weights:

$$\dot{\hat{W}} = -\eta_1\left(\frac{\partial J}{\partial \hat{W}}\right) - \rho_1\|\tilde{x}\|\hat{W}$$

$$\dot{\hat{V}} = -\eta_2\left(\frac{\partial J}{\partial \hat{V}}\right) - \rho_2\|\tilde{x}\|\hat{V}$$

  - Therefore:

$$net_{\hat{v}} = \hat{V}\hat{\bar{x}}$$

$$net_{\hat{w}} = \hat{W}\sigma(\hat{V}\hat{\bar{x}}).$$

  - $\frac{\partial J}{\partial \hat{W}}$ and $\frac{\partial J}{\partial \hat{V}}$ can be computed according to

$$\frac{\partial J}{\partial \hat{W}} = \frac{\partial J}{\partial net_{\hat{w}}} \cdot \frac{\partial net_{\hat{w}}}{\partial \hat{W}}$$

$$\frac{\partial J}{\partial \hat{V}} = \frac{\partial J}{\partial net_{\hat{v}}} \cdot \frac{\partial net_{\hat{v}}}{\partial \hat{V}}$$

$$\frac{\partial J}{\partial net_{\hat{w}}} = \frac{\partial J}{\partial \tilde{x}} \cdot \frac{\partial \tilde{x}}{\partial \hat{x}} \cdot \frac{\partial \hat{x}}{\partial net_{\hat{w}}} = \tilde{x}^T \cdot \frac{\partial \hat{x}}{\partial net_{\hat{w}}}$$

$$\frac{\partial J}{\partial net_{\hat{v}}} = \frac{\partial J}{\partial \tilde{x}} \cdot \frac{\partial \tilde{x}}{\partial \hat{x}} \cdot \frac{\partial \hat{x}}{\partial net_{\hat{v}}} = \tilde{x}^T \cdot \frac{\partial \hat{x}}{\partial net_{\hat{v}}}$$

- and

$$\frac{\partial net_{\hat{w}}}{\partial \hat{W}} = \sigma(\hat{V}\hat{\hat{x}})$$

$$\frac{\partial net_{\hat{v}}}{\partial \hat{V}} = \hat{\hat{x}}$$

$$\frac{\partial \dot{\hat{x}}(t)}{\partial net_{\hat{w}}} = A\frac{\partial \hat{x}}{\partial net_{\hat{w}}} + \frac{\partial \hat{g}}{\partial net_{\hat{w}}}$$

$$\frac{\partial \dot{\hat{x}}(t)}{\partial net_{\hat{v}}} = A\frac{\partial \hat{x}}{\partial net_{\hat{v}}} + \frac{\partial \hat{g}}{\partial net_{\hat{v}}}.$$

- Which is dynamic BP. Modify BP algorithm s.t. the static approximations of $\frac{\partial \hat{x}}{\partial net_{\hat{w}}}$ and $\frac{\partial \hat{x}}{\partial net_{\hat{v}}}$ $(\dot{\hat{x}} = 0)$

- Thus,

$$\frac{\partial \hat{x}}{\partial net_{\hat{w}}} = -A^{-1}$$
$$\frac{\partial \hat{x}}{\partial net_{\hat{v}}} = -A^{-1}\hat{W}(I - \Lambda(\hat{V}\hat{\bar{x}}))$$

where

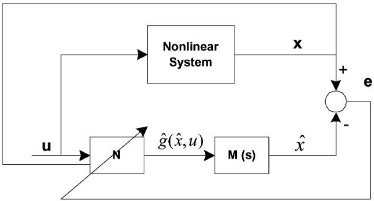
$$\Lambda(\hat{V}\hat{\bar{x}}) = diag\{\sigma_i^2(\hat{V}_i\hat{\bar{x}})\}, i = 1, 2, ..., m.$$

- Finally

$$\begin{aligned} \dot{\hat{W}} &= -\eta_1(\tilde{x}^T A^{-1})^T(\sigma(\hat{V}\hat{\bar{x}}))^T \\ &- \rho_1\|\tilde{x}\|\hat{W} \\ \dot{\hat{V}} &= -\eta_2(\tilde{x}^T A^{-1}\hat{W}(I - \Lambda(\hat{V}\hat{\bar{x}})))^T\hat{\bar{x}}^T \\ &- \rho_2\|\tilde{x}\|\hat{V} \end{aligned}$$

- $\tilde{W} = W - \hat{W}$ and $\tilde{V} = V - \hat{V}$,
- It can be shown that $\tilde{x}$, $\tilde{W}$, and $\tilde{V} \in L_\infty$
- The estimation error and the weights error are all ultimately bounded [3].

- **Series-Parallel Identifier**
  - The function g can be approximated by$\hat{g}(x,u) = \hat{W}\sigma(\hat{V}\bar{x})$
  - Only $\hat{\bar{x}}$ is changed to $\bar{x}$.
  - The error dynamics $\dot{\tilde{x}}(t) = A\tilde{x} + \tilde{W}\sigma(\hat{V}\bar{x}) + w(t)$ where $w(t) = W[\sigma(V\bar{x}) - \sigma(\hat{V}\bar{x})] + \epsilon(x)$
  - only definition of $w(t)$ is changed.
  - Applying this change, the rest remains the same



Case Study: Simulation Results on SSRMS

- The Space Station Remote Manipulator System (SSRMS) is a 7 DoF robot which has 7 revolute joints and two long flexible links (booms).
- The SSRMS have no uniform mass and stiffness distributions. Most of its masses are concentrated at the joints, and the joint structural flexibilities contribute a major portion of the overall arm flexibility.
- Dynamics of a flexible–link manipulator

$$M(q)\ddot{q} + h(q,\dot{q}) + Kq + F\dot{q} = u$$

  - $u = [\tau^T \ 0_{1\times m}]^T$, $q = [\theta^T \ \delta^T]^T$,
  - θ is the $n \times 1$ vector of joint variables
  - δ is the $m \times 1$ vector of deflection variables
  - $h = [h_1(q,\dot{q}) \ h_2(q,\dot{q})]^T$: including gravity, Coriolis, and centrifugal forces;
  - M is the mass matrix,
  - $K = \begin{bmatrix} 0_{n\times n} & 0_{n\times m} \\ 0_{m\times n} & K_{m\times m} \end{bmatrix}$ is the stiffness matrix,
  - $F = diag\{F_1, F_2\}$: the viscous friction at the hub and in the structure,
  - τ: input torque.

Case Study: Simulation Results on SSRMS

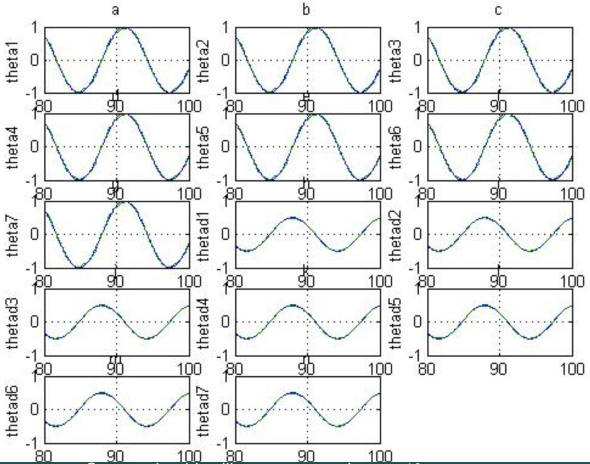
http://english.sohu.com/20050729/n226492517.shtml

Case Study: Simulation Results on SSRMS

- A joint PD control is applied to stabilize the closed-loop system $\rightsquigarrow$ boundedness of the signal $x(t)$ is assured.
- For a two link flexible manipulator
  - $x = [\theta_1 ... \ \theta_7 \ \dot{\theta}_1 ... \dot{\theta}_7 \ \delta_{11} \ \delta_{12} \ \delta_{21} \ \delta_{22} \ \dot{\delta}_{11} \ \dot{\delta}_{12} \ \dot{\delta}_{21} \ \dot{\delta}_{22}]^T$
  - The input: $u = [\tau_1, \ ..., \ \tau_7]$
  - A is defined as $A = -2I \in \mathcal{R}^{22\times 22}$
  - Reference trajectory: $sin(t)$
- The identifier:
  - Series-parallel
  - A three-layer NN network: 29 neurons in the input layer, 20 neurons in the hidden layer, and 22 neurons in the output layer.
  - The 22 outputs correspond to
    - 7 joint positions
    - 7 joint velocities
    - 4 in-plane deflection variables
    - 4 out-of plane deflection variables
  - The learning rates and damping factors: $\eta_1 = \eta_2 = 0.1, \ \rho_1 = \rho_2 = 0.001.$

Case Study: Simulation Results on SSRMS

- Simulation results for the SSRMS: (a-g) The joint positions, and (h-n) the joint velocities.



References

- A. Yazdizadeh and K. Khorasani, "Nonlinear system identification using adaptive time delay neural netowrks," in *World Congress in Neural Networks*, 1996.
- R. H. Nielsen, "Theory of the backpropagation neural network," in *International Joint Conf. on Neural Networks*, pp. 593–605, 1989.
- H. A. Talebi, F. Abdollahi, R. V. Patel and K. Khorasani, *Neural Network-based State Estimation of Nonlinear Systems: Application to Fault Detection and Isolation*. Springer, 2009.
- K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. on Neural Networks,* vol. 1, no. 1, pp. 4–27, March 1990.
- J. B. Cabrera and K. S. Narendra, "Issues in the application ofneural networks for tracking based on inverse control," *IEEE Trans. on Automatic Control,* vol. 44, no. 11, pp. 2007–2025, Nov. 1999.
- S. Mukhopadhyay and K. S. Narendra, "Disturbance rejection in nonlinear systems using neural networks," *IEEE Trans. on Neural Networks,* vol. 4, no. 1, pp. 63–72, Jan 1993.