Nonlinear modelling and control

Johan Suykens

K.U. Leuven, ESAT-STADIUS
Kasteelpark Arenberg 10
B-3001 Leuven (Heverlee), Belgium
Email: johan.suykens@esat.kuleuven.be
http://www.esat.kuleuven.be/stadius

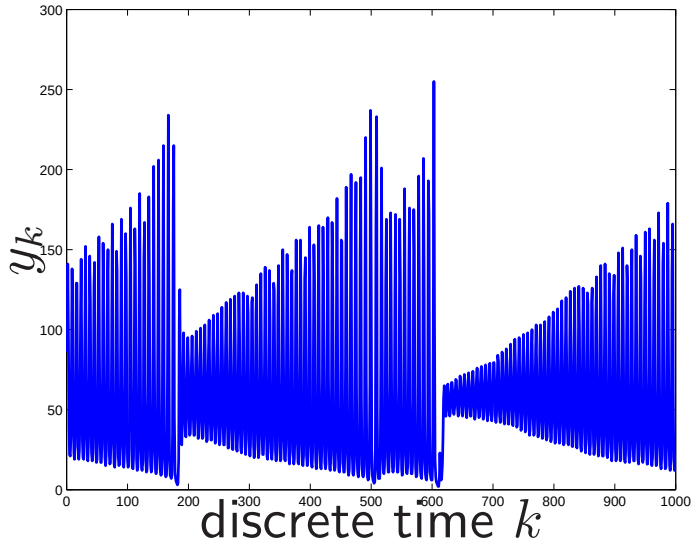
Lecture 8

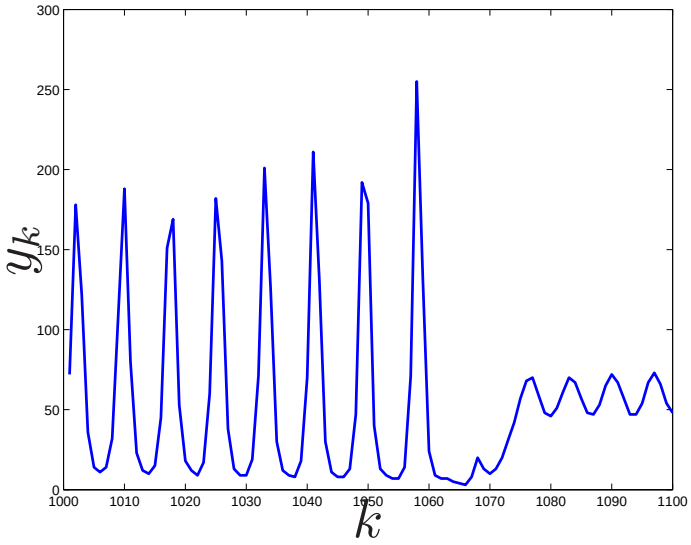
Overview

- Time-series prediction
- Nonlinear system identification
- Model structures: input-output models, state space models
- Dynamic backpropagation
- Neural networks for control

Time-series prediction (1)

- Laser data of Santa Fe time series prediction competition
  www-psych.stanford.edu/∼andreas/Time-Series/SantaFe.html





(left) training set (given 1000 data points)
(right) test data (100 points ahead to be predicted in the future)

Time-series prediction (2)

- Model structure:

$$\hat{y}_{k+1} = f(y_k, y_{k-1}, ..., y_{k-p})$$

where f is e.g. parameterized by a MLP with one hidden layer:

$$f(y_{k|k-p}) = w^T \tanh(V\, y_{k|k-p} + \beta)$$

with $y_{k|k-p} = [y_k, y_{k-1}, ..., y_{k-p}]^T$.

- Notation:

y_k	measured data
$\hat{y}_k$	estimated value
w	output weights
V	hidden layer matrix
β	bias term vector

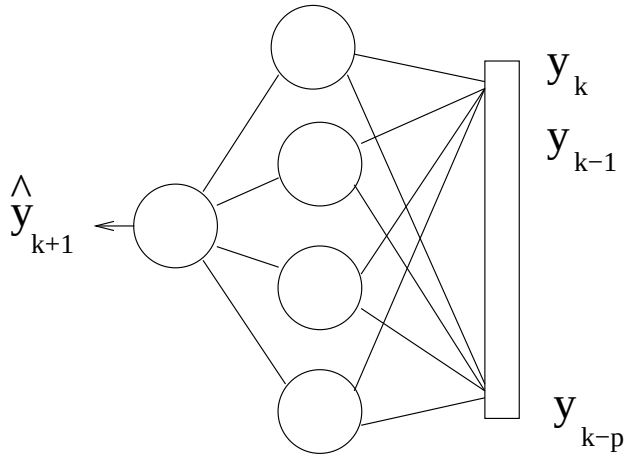
Time-series prediction (3)

- Training error to be minimized (N given data points):

$$\min_{w,V,\beta} \quad \frac{1}{2N} \sum_{k=p+1}^{p+N} (y_{k+1} - \hat{y}_{k+1})^2$$

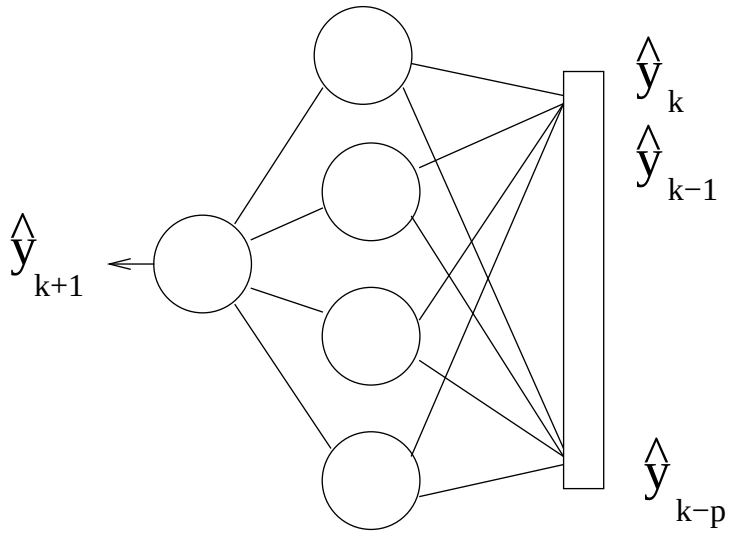
  Note that $y_k = \hat{y}_k + e_k$.

- Training as a feedforward network (e.g. by backpropagation):



Time-series prediction (4)

- Iterative prediction as a recurrent network



- In order to predict several points ahead in time, the estimated values $\hat{y}_k$ are used when iteratively predicting over time.

Time-series prediction (5)

- Model selection issues:

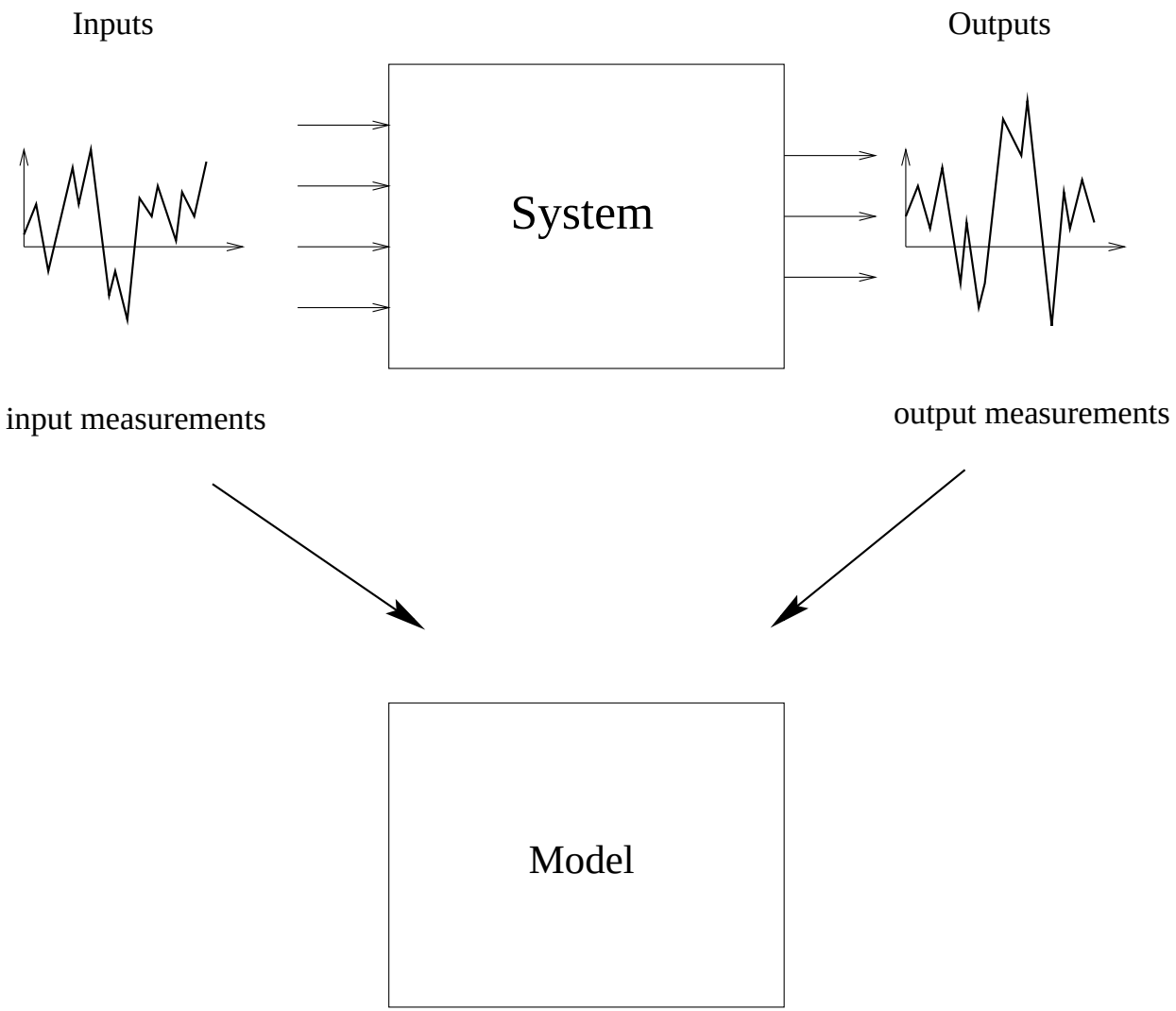
  number of lags p
  number of hidden nodes
  choice of the regularization constant (in case a weight decay term is used in the objective)

- A validation set can be used for this purpose:
  one partitions the data then in a training, validation and test set part

- A more sophisticated method is 10-fold cross-validation

- Important message: avoid overfitting!

System identification (1)



System identification (2)

- **White box models:**

  The model equations are obtained by applying physical laws (e.g. Newton's laws). The model parameters that ones estimates have a physical meaning.

- **Black box models:**

  The aim is to obtain a predictive model that is able to establish the relation between inputs and outputs of the system. The estimated parameters do not have any physical meaning.

- **Grey box models:**

  Contains aspects of both white box and black box models (e.g. one relies on a physical model and one only represents an unknown nonlinear part in a black-box way)

System identification (3)

Linear time-invariant Input/Output model structure

General family of linear model structures

Five polynomials $A(q), B(q), C(q), D(q), F(q)$

Generalized linear model structure:

$$A(q)y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t)$$

Special cases of commonly used models:

ARX :	$A(q)y(t) = B(q)u(t) + e(t)$
ARMAX :	$A(q)y(t) = B(q)u(t) + C(q)e(t)$
Output Error :	$y(t) = \frac{B(q)}{F(q)}u(t) + e(t)$
Box-Jenkins :	$y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t)$

System identification (4)

Nonlinear I/O model structures

- NARX (Nonlinear ARX):

$$\hat{y}_k = f(y_{k-1}, y_{k-2}, ..., y_{k-n_y}, u_{k-1}, u_{k-2}, ..., u_{k-n_u})$$

- NARMAX (Nonlinear ARMAX):

$$\begin{aligned} y_k = f(\quad & y_{k-1}, y_{k-2}, ..., y_{k-n_y}, u_{k-1}, u_{k-2}, ..., u_{k-n_u}, \\ & \epsilon_{k-1}, \epsilon_{k-2}, ..., \epsilon_{k-n_\epsilon}) + \epsilon_k \end{aligned}$$

- NOE (Nonlinear output error model):

$$\hat{y}_k = f(\hat{y}_{k-1}, \hat{y}_{k-2}, ..., \hat{y}_{k-n_y}, u_{k-1}, u_{k-2}, ..., u_{k-n_u})$$

with error $y_k - \hat{y}_k = \epsilon_k$, estimated output $\hat{y}_k$ and measured output y_k

System identification (5)

State space models - Deterministic case:

- Linear state space model

$$\begin{cases} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{cases}$$

with state vector $x_k \in \mathbb{R}^n$, input vector $u_k \in \mathbb{R}^m$ and output vector $y_k \in \mathbb{R}^l$.

- Nonlinear state space model

$$\begin{cases} x_{k+1} &= f(x_k, u_k) \\ y_k &= g(x_k) \end{cases}$$

System identification (6)

State space models - with stochastic part:

- Linear state space model

$$\begin{cases} x_{k+1} &= Ax_k + Bu_k + w_k \\ y_k &= Cx_k + v_k \end{cases}$$

  where w_k is the process noise and v_k is the observation (or measurement) noise.

- Nonlinear state space model

$$\begin{cases} x_{k+1} &= f(x_k, u_k, w_k) \\ y_k &= g(x_k, v_k) \end{cases}$$

System identification (7)

Parameterizations by neural nets - I/O models:

$$\hat{y}_{k+1} = f(z_{k|k-p}), \qquad z_{k|k-p} = [y_k, y_{k-1}, ..., y_{k-p}, u_k, u_{k-1}, ..., u_{k-p}]^T$$

where f can be parameterized by feedforward neural networks

1. MLP with one hidden layer:

$$\hat{y}_{k+1} = w^T \sigma(V z_{k|k-p} + \beta)$$

2. MLP with two hidden layers:

$$\hat{y}_{k+1} = w^T \sigma(V_2 \sigma(V_1 z_{k|k-p} + \beta_1) + \beta_2)$$

3. Radial basis functions (RBF):

$$\hat{y}_{k+1} = \sum_{i=1}^{n_h} w_i \, h(\|z_{k|k-p} - c_i\|)$$

System identification (8)

Parameterizations by neural nets - state-space models:

$$\begin{cases} \hat{x}_{k+1} &= f(\hat{x}_k, u_k) \\ \hat{y}_k &= g(\hat{x}_k, u_k) \end{cases}$$

where f and g can be parameterized e.g. by

1. MLP with one hidden layer (recurrent network):

$$\begin{cases} \hat{x}_{k+1} &= W_{AB}\,\sigma(V_A\hat{x}_k + V_B u_k + \beta_{AB}) \\ \hat{y}_k &= W_{CD}\,\sigma(V_C\hat{x}_k + V_D u_k + \beta_{CD}) \end{cases}$$

2. MLP with two hidden layers (recurrent network):

$$\begin{cases} \hat{x}_{k+1} &= W_{AB}\,\sigma(V_{AB}\,\sigma(V_A\hat{x}_k + V_B u_k + \beta_{AB}^{(1)}) + \beta_{AB}^{(2)}) \\ \hat{y}_k &= W_{CD}\,\sigma(V_{CD}\,\sigma(V_C\hat{x}_k + V_D u_k + \beta_{CD}^{(1)}) + \beta_{CD}^{(2)}) \end{cases}$$

Dynamic backpropagation (1)

- State space model

$$\dot{x}(t) = f[x(t); \alpha], \quad x(t_0) = x_0$$

  where $x(t) \in \mathbb{R}^n$ and α some scalar parameter to be adjusted.

- Performance index (cost function):

$$J(\alpha) = \frac{1}{T} \int_0^T [x(t; \alpha) - x_d(t)]^2 dt$$

  where $x_d(t)$ is a desired reference trajectory.

- Gradient:

$$\frac{\partial J(\alpha)}{\partial \alpha} = \frac{1}{T} \int_0^T 2[x(t; \alpha) - x_d(t)] \frac{\partial x(t)}{\partial \alpha} \, dt$$

  Question: how to find $\partial x(t)/\partial \alpha$?

Dynamic backpropagation (2)

- Sensitivity model:

$$\frac{\partial \dot{x}(t)}{\partial \alpha} = f_x(t)\,\frac{\partial x(t)}{\partial \alpha} + f_\alpha(t), \quad \frac{\partial x(t_0)}{\partial \alpha} = 0$$

  with Jacobian $f_x(t)$ and $f_\alpha(t)$ evaluated around the nominal values.

- Feedforward networks: Backprogagation (BP)
  Recurrent networks: Dynamic backpropagation [Narendra, 1991]

- Computation of the gradient of a cost function defined on a recurrent network is more complicated. It involves a sensitivity model, which is another dynamical system derived from the model (another approach: Werbos' backpropagation through time)

Dynamic backpropagation (3)

- Neural state space model

$$\begin{cases} \hat{x}_{k+1} &= W_{AB} \tanh(V_A \hat{x}_k + V_B u_k + \beta_{AB}) \; ; \; \hat{x}_0 = x_0 \\ \hat{y}_k &= W_{CD} \tanh(V_C \hat{x}_k + V_D u_k + \beta_{CD}) \end{cases}$$

  given a training set of N input/output data.

- Prediction error algorithm with cost function

$$\min_{\theta} J(\theta) = \frac{1}{2N} \sum_{k=1}^{N} (y_k - \hat{y}_k(\theta))^2$$

  with unknown parameter vector
  $\theta = [W_{AB}(:); V_A(:); V_B(:); \beta_{AB}; W_{CD}(:); V_C(:); V_D(:); \beta_{CD}]$.

Dynamic backpropagation (4)

- Consider the model as

$$\begin{cases} \hat{x}_{k+1} &= \Phi(\hat{x}_k, u_k; \alpha) \ ; \ \hat{x}_0 = x_0 \text{ given} \\ \hat{y}_k &= \Psi(\hat{x}_k, u_k; \beta) \end{cases}$$

  with α, β elements of parameter vector θ.

- Sensitivity model for dynamic backpropagation:

$$\begin{cases} \frac{\partial \hat{x}_{k+1}}{\partial \alpha} &= \frac{\partial \Phi}{\partial \hat{x}_k} \frac{\partial \hat{x}_k}{\partial \alpha} + \frac{\partial \Phi}{\partial \alpha} \\ \frac{\partial \hat{y}_k}{\partial \alpha} &= \frac{\partial \Psi}{\partial \hat{x}_k} \frac{\partial \hat{x}_k}{\partial \alpha} \\ \frac{\partial \hat{y}_k}{\partial \beta} &= \frac{\partial \Psi}{\partial \beta} \end{cases}$$

Examples (1)

Load Forecasting in Electric Power Systems

- Data: hourly temperature and load data for Seattle/Tacoma area in the period Nov 1 1988 - Jan 30 1989 [Park et al., 1991]
- Case 1: peak load and total load of the day

$$\begin{aligned} \text{Input NN} &= [T_1(k)\ T_2(k)\ T_3(k)]^T \\ \text{Output NN} &= L(k) \end{aligned}$$

  with T_1, T_2, T_3 the average, peak and lowest temperature at day k and $L(k)$ the peak or total load at day k.
- Case 2: hourly load

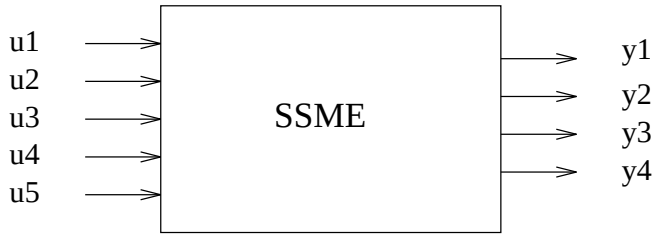
$$\begin{aligned} \text{Input NN} &= [k\ L(k-2)\ L(k-1)\ T(k-2)\ T(k-1)\ \hat{T}(k)]^T \\ \text{Output NN} &= L(k) \end{aligned}$$

  with k the hour of the predicted load and $L(k), T(k), \hat{T}(k)$ the load, temperature and predicted temperature at hour k.

Examples (2)

Space shuttle main engine (SSME) [Saravanan et al., 1993]

System:



Inputs $(u_1, ..., u_5)$: rotary motion of five valves

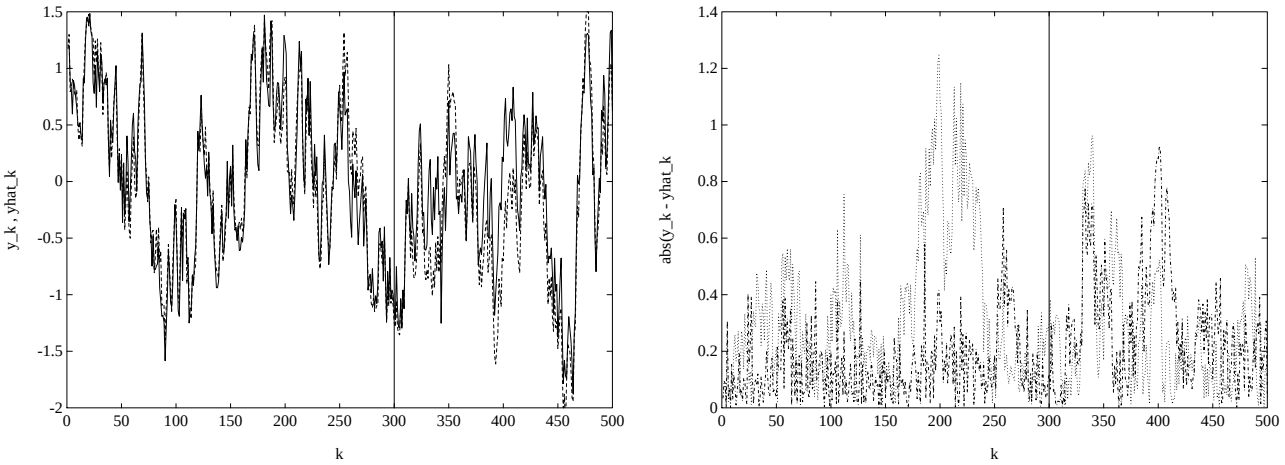
Outputs $(y_1, ..., y_4)$: chamber pressure, mixture ratio, speed of high pressure fuel turbine, speed of high pressure oxidizer turbine

A complete nonlinear dynamic model for simulation was already available, but due to its complexity difficult to use for real time applications. Neural nets are therefore suitable thanks to their parallel architecture.

Examples (3)

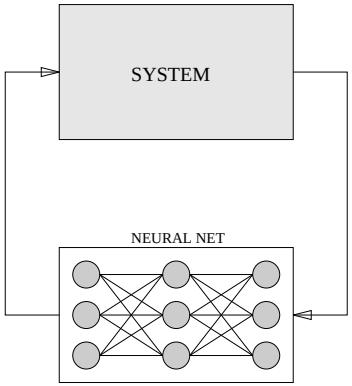
Identification of a glass furnace [Suykens et al., 1992]

- System:
  3 inputs: 2 heating inputs and 1 cooling input
  6 outputs: temperatures in a cross section of the furnace.
- Identification result using neural state space models:
  improvement with respect to linear model



(left) full line: original data, dashed line: estimated output by NN
(right) dotted line: error for linear model, dashdot line: error for NN
(data before the vertical line were used for training)

Neural control (1)

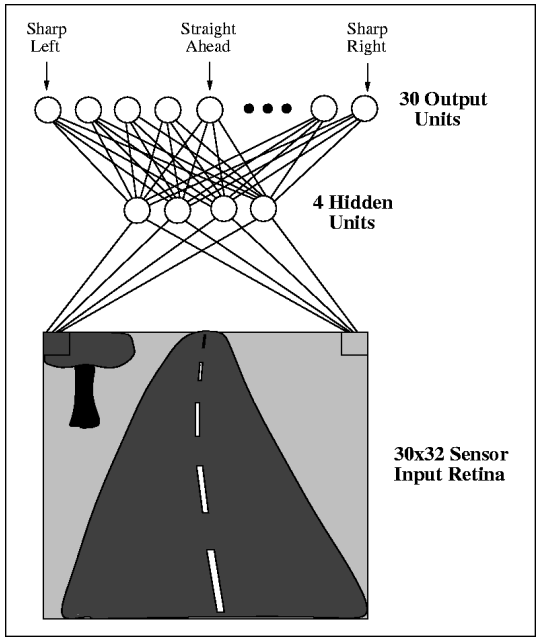


Different approaches are possible:

- Mimic an expert by collecting input/output data from the expert (e.g. learning to automatically drive a car (e.g. ALVINN system))
- Model-based control of a system: first estimate a (neural network) model and then design a neural controller based on the estimated model (e.g. back-propagation through the plant method)
- Control a system without making use of a model for the system (e.g. using reinforcement learning methods)

Neural control (2)

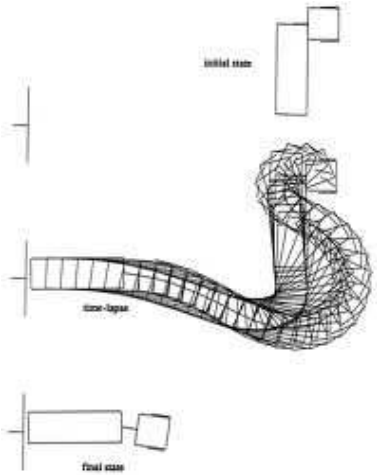
ALVINN: Autonomous Land Vehicle In a Neural Network



First learning to drive with a driver on-board of the car, then use the neural network instead of the driver for autonomously driving [Pomerleau, 1989].

Neural control (3)

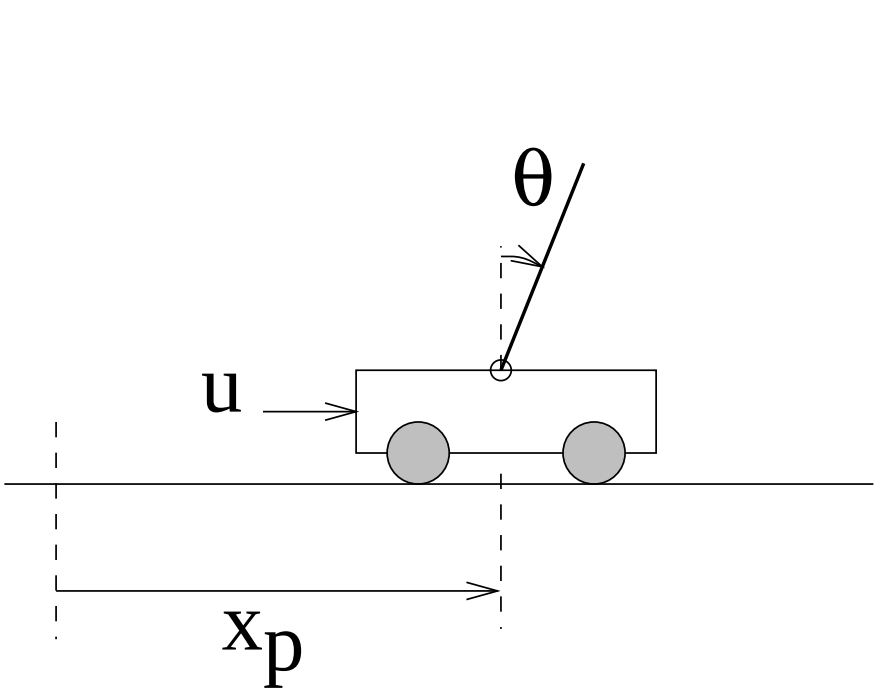
Steering a trailer truck while backing up to a loading dock from different initial positions [Nguyen & Widrow, 1990]

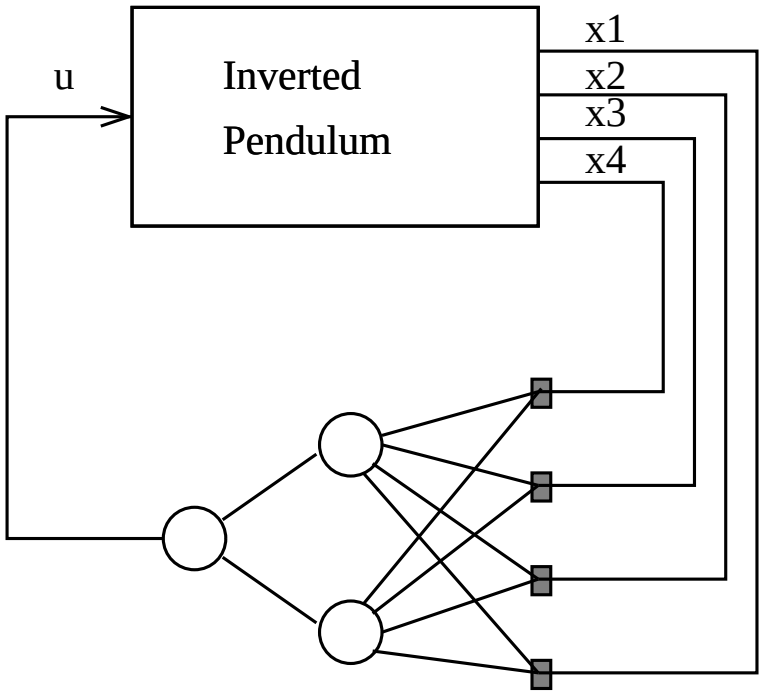


Neural control strategy:

1. Use NN emulator (identifier) for equations of motion
2. Parametrize both plant and controller by multilayer feedforward NNs
3. Consider a 'chain' of NN emulator and NN controller through time from initial to final state
4. Backpropagate the error of objective function through the NN

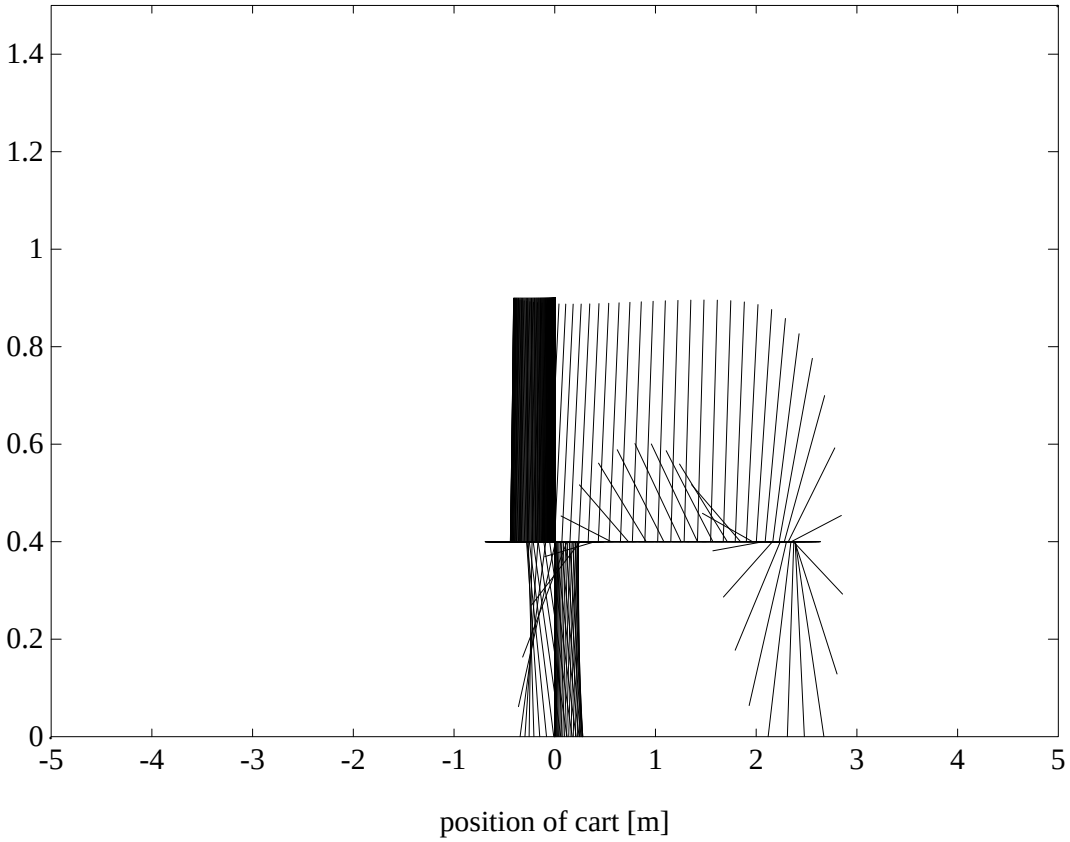
Neural control (4)



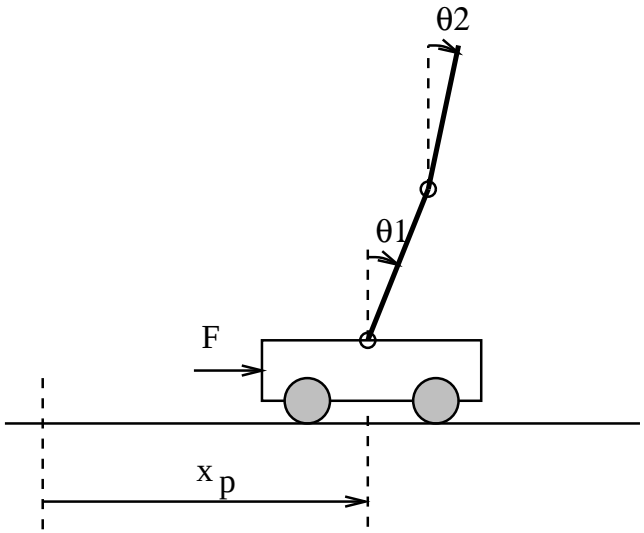


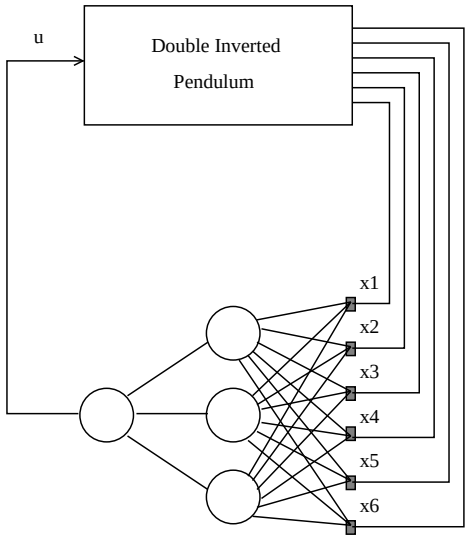
Swinging up an inverted pendulum using a multilayer perceptron [Suykens et al., 1994]

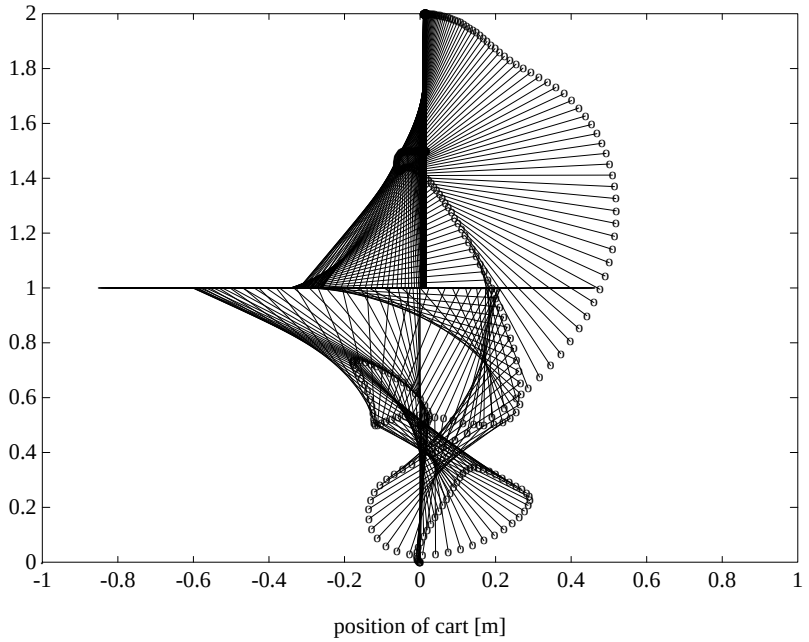
Neural control (5)



Neural control (6)







Swinging up a double inverted pendulum using a multilayer perceptron [Suykens et al., 1993]

Additional optional material (1)

- Weigend A.S., Gershenfeld N.A., "Results of the time series prediction competition at the Santa Fe institute", *IEEE International Conference on Neural Networks*, pp. 1786-1793, 1993.
- Narendra K.S., Parthasarathy K., "Identification and control of dynamical systems using neural networks", *IEEE Transactions on Neural Networks*, Vol.1, No.1, pp.4-27, 1990.
- Narendra K.S., Parthasarathy K., "Gradient methods for the optimization of dynamical systems containing neural networks", *IEEE Transactions on Neural Networks*, Vol.2, No.2, pp.252-262, 1991.
- Sjöberg J., Zhang Q., Ljung L., Benveniste A., Delyon B., Glorennec P.-Y., Hjalmarsson H., Juditsky A., "Nonlinear black-box modeling in system identification: a unified overview", *Automatica*, **31**(12), 1691-1724, 1995.
- Park D.C., El-Sharkawi M.A., Marks II R.J., Atlas L.E., Damborg M.J., "Electric load forecasting using an artificial neural network", *IEEE Trans. Power Syst.*, vol.6, no.2, May 1991, pp.442-449.
- Saravanan N., Duyar A., Guo T.-H., Merrill W.C., "Modeling Space Shuttle main engine using feed-forward neural networks", *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 4, p. 641-648, 1994

Additional optional material (2)

- Pomerleau D., "Efficient Training of Artificial Neural Networks for Autonomous Navigation", *Neural Computation*, Vol. 3, No. 1, 1991, pp. 88-97.
- Werbos P., "Backpropagation through time: what it does and how to do it", *Proceedings of the IEEE*, 78 (10), pp.1150-1560, 1990.
- Nguyen D., Widrow B., "Neural networks for self-learning control systems", *IEEE Control Systems Magazine*, 10(3), pp.18-23, 1990.
- Suykens J.A.K., De Moor B., Vandewalle J., "Static and dynamic stabilizing neural controllers applicable to transition between equilibrium points", *Neural Networks*, vol. 7, no. 5, 1994, pp. 819-831.
- http://www.youtube.com/
  "The Robot Champion of DARPA's Urban challenge"
  "DARPA Grand Challenge: Stanford"
  "Stanford wins driverless car race"