NEURAL NETWORKS FOR SYSTEM MODELING

Gábor Horváth

Budapest University of Technology and Economics

Dept. Measurement and Information Systems

Budapest, Hungary

Copyright © Gábor Horváth

The slides are based on the NATO ASI presentation (NIMIA) in Crema Italy, 2002

Outline

- Introduction
- System identification: a short overview
  - Classical results
  - Black box modeling
- Neural networks architectures
  - An overview
  - Neural networks for system modeling
- Applications

Introduction

- The goal of this course:

  to show why and how neural networks can be applied for system identification

  - Basic concepts and definitions of system identification
    - classical identification methods
    - different approaches in system identification
  - Neural networks
    - classical neural network architectures
    - support vector machines
    - modular neural architectures
  - The questions of the practical applications, answers based on a real industrial modeling task (case study)

System identification

System identification: a short overview

- Modeling
- Identification
  - Model structure selection
  - Model parameter estimation
- Non-parametric identification
  - Using general model structure
- Black-box modeling
  - Input-output modeling, the description of the behaviour of a system

Modeling

- What is a model?
- Why we need models?
- What models can be built?
- How to build models?

Modeling

- What is a model?
  - Some (formal) description of a system, a separable part of the world.

    Represents essential aspects of a system
  - Main features:
    - All models are imperfect. Only some aspects are taken into consideration, while many other aspects are neglected.
    - Easier to work with models than with the real systems
  - Key concepts: *separation, selection, parsimony*

Modeling

- **Separation:**
  - the boundaries of the system have to be defined.
  - system is separated from all other parts of the world
- **Selection:**

  Only certain aspects are taken into consideration e.g.
  - information relation, interactions
  - energy interactions
- **Parsimony:**

  It is desirable to use as simple model as possible
  - Occam's razor (William of Ockham or Occam) 14th Century English philosopher)

  The most likely hypothesis is the simplest one that is consistent with all observations

  The simpler of two theories, two models is to be preferred.

Modeling

- **Why do we need models?**
  - To understand the world around (or its defined part)
  - To simulate a system
    - to predict the behaviour of the system (prediction, forecasting),
    - to determine faults and the cause of misoperations, fault diagnosis, error detection,
    - to control the system to obtain prescribed behaviour,
    - to increase observability: to estimate such parameters which are not directly observable (indirect measurement),
    - system optimization.
  - Using a model
    - we can avoid making real experiments,
    - we do not disturb the operation of the real system,
    - more safe then working with the real system,
    - etc...

Modeling

- **What models can be built?**
  - Approaches
    - functional models
      - parts and its connections based on the functional role in the system
    - physical models
      - based on physical laws, analogies (e.g. electrical analog circuit model of a mechanical system)
    - mathematical models
      - mathematical expressions (algebraic, differential equations, logic functions, finite-state machines, etc.)

Modeling

- **What models can be built?**
  - A priori information
    - physical models, “first principle” models
      use laws of nature
    - models based on observations (experiments)
      the real physical system is required for
      obtaining observations
  - Aspects
    - structural models
    - input-output (behavioral) models

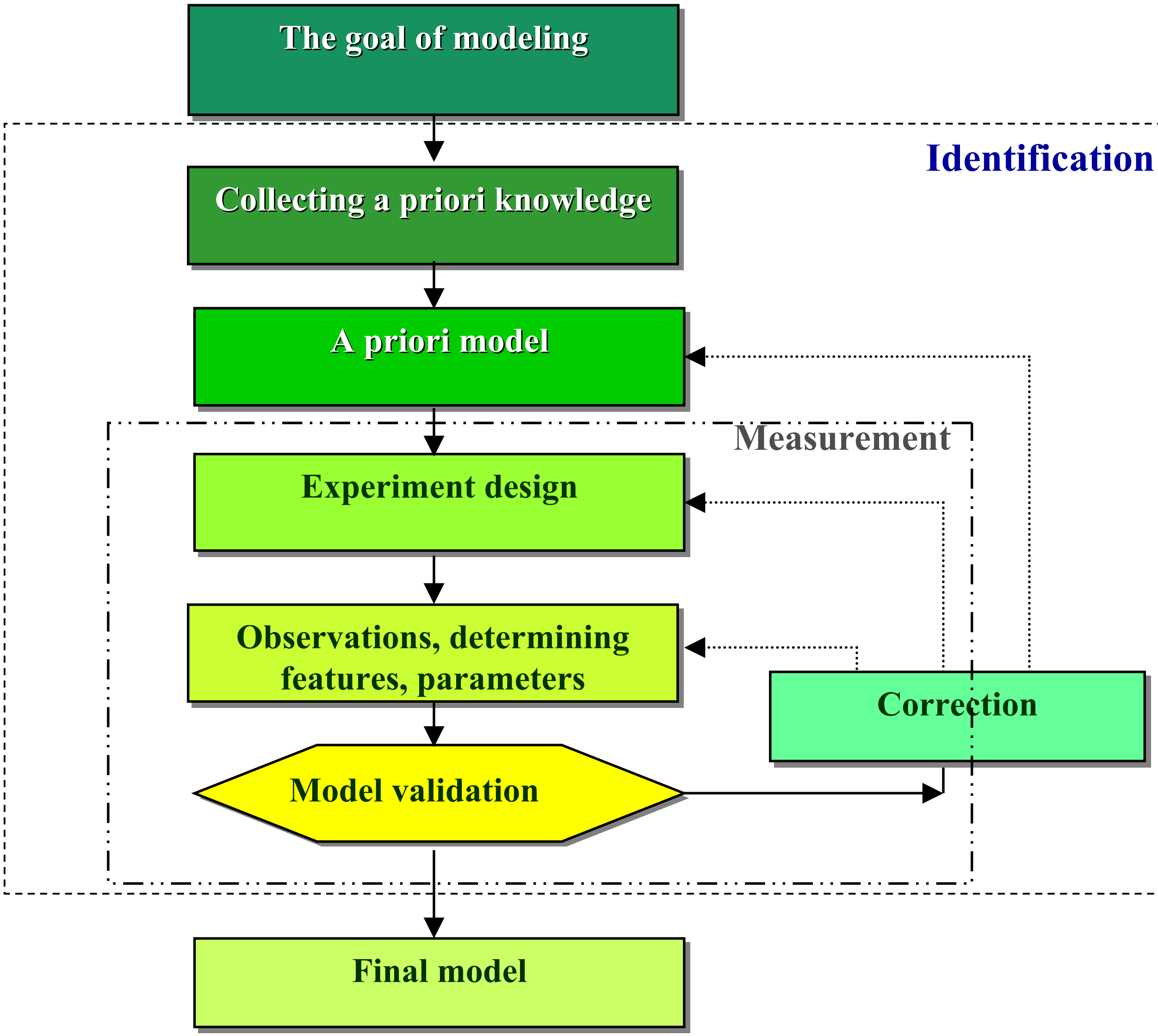
Identification

- **What is identification?**
  - Identification is the process of deriving a (mathematical) model of a system using observed data

Measurements

- Empirical process
  - to obtain experimental data (observations),
    - **primary information collection,** or
    - **to obtain additional information to the a priori one.**
  - to use the experimental data for obtaining (determining) the free parameters (features) of a model.
  - to validate the model

Identification (measurement)



Model classes

- Based on the system characteristics
- Based on the modeling approach
- Based on the a priori information

Model classes

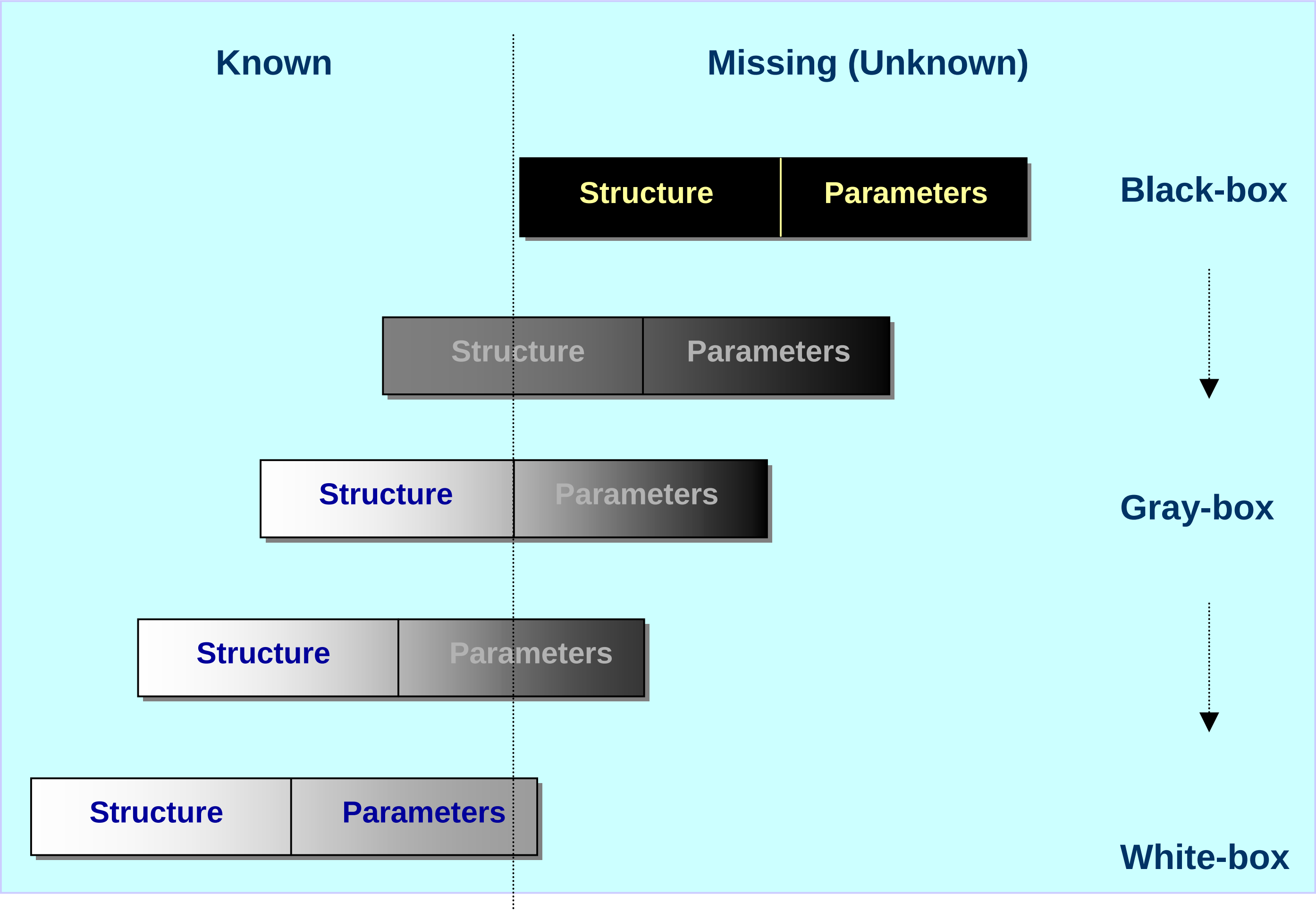
- Based on the system characteristics
  - Static – dynamic
  - Deterministic – stochastic
  - Continuous-time – discrete-time
  - Lumped parameter – distributed parameter
  - Linear – non-linear
  - Time invariant – time variant
  - …

Model classes

- Based on the modeling approach
  - parametric
    - known model structure
    - limited number of unknown parameters
  - nonparametric
    - no definite model structure
    - described in many points (frequency characteristics, impulse response)
  - semi-parametric
    - general class of functional forms are allowed
    - the number of parameters can be increased independently of the size of the data

Model classes

- Based on the a priori information (physical insight)
  - white-box
  - gray-box
  - black-box



Identification

- Main steps
  - collect information
  - model set selection
  - experiment design and data collection
  - determine model parameters (estimation)
  - model validation

Identification

- Collect information
  - physical insight (a priori information)

    understanding the *physical behaviour*
  - only observations or experiments can be designed
  - application
    - what operating conditions
      - one operating point
      - a large range of different conditions
    - what purpose
      - scientific

        basic research
      - engineering

        to study the behavior of a system,
        to detect faults,
        to design control systems,
        etc.

Identification

- Model set selection
  - static – dynamic
  - linear – non-linear
  - non-linear
    - linear - in - the - parameters
    - non-linear - in - the - parameters
  - white-box – black-box
  - parametric – non-parametric

Identification

- Model structure selection
  - known model structure (available a priori information)
  - no physical insights, general model structure
    - general rule: always use as simple model as possible (Occam's razor)
      - linear
      - feed-forward

        •
        •
        •

Experiment design and data collection

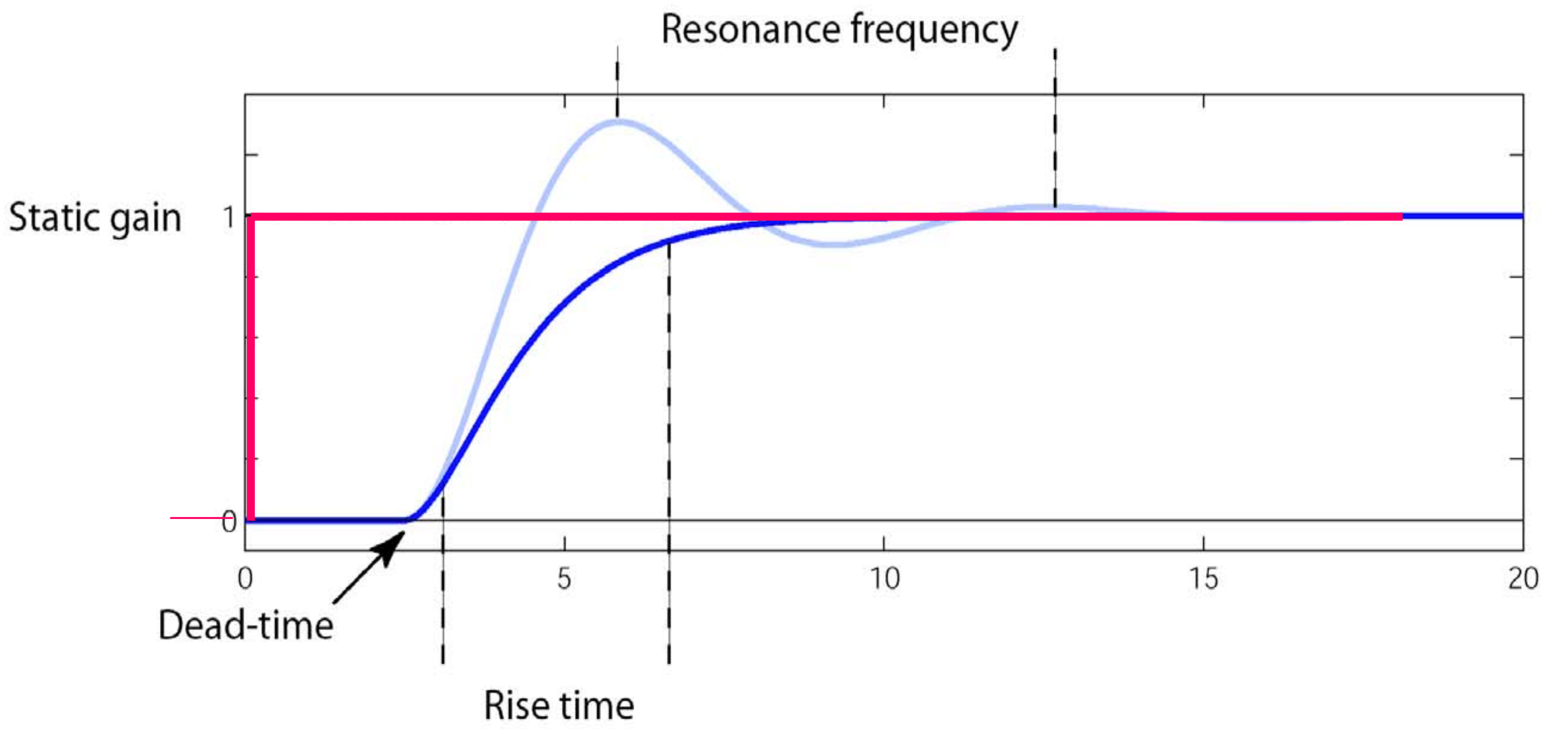
- Excitation
  - input signal selection
  - design of excitation
    - time domain or frequency domain identification (random signal, multi-sine excitation, impulse response, frequency characteristics)
    - persistent excitation
- Measurement of input-output data
  - no possibility to design excitation signal
    - noisy data, missing data, distorted data
    - non-representing data

Excitation

- Step function
- Random signal (autoregressive moving average (ARMA) process)
- Pseudorandom binary sequence
- Multisine

Excitation

- Step function



Excitation

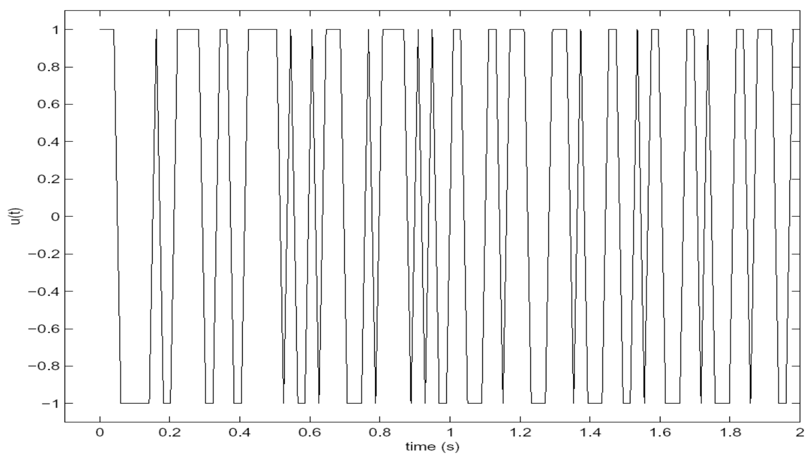
- Random signal (autoregressive moving average (ARMA) process)
  - obtained by filtering white noise
  - filter is selected according to the desired frequency characteristic
  - an ARMA(p,q) process can be characterized
    - in time domain
    - in lag (correlation) domain
    - in frequency domain

Excitation

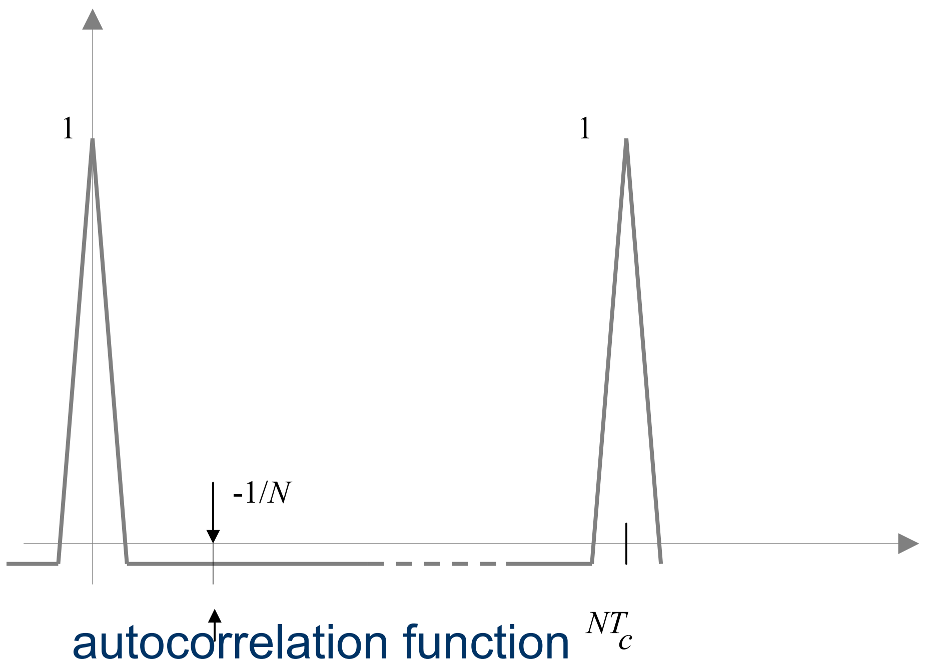
- Pseudorandom binary sequence
  - The signal switches between two levels with given probability

$$u(k+1)=\begin{cases} u(k) & \text{with probability } p \\ -u(k) & \text{with probability } 1-p \end{cases}$$

  - Frequency characteristics depend on the probability *p*
  - Example



time function

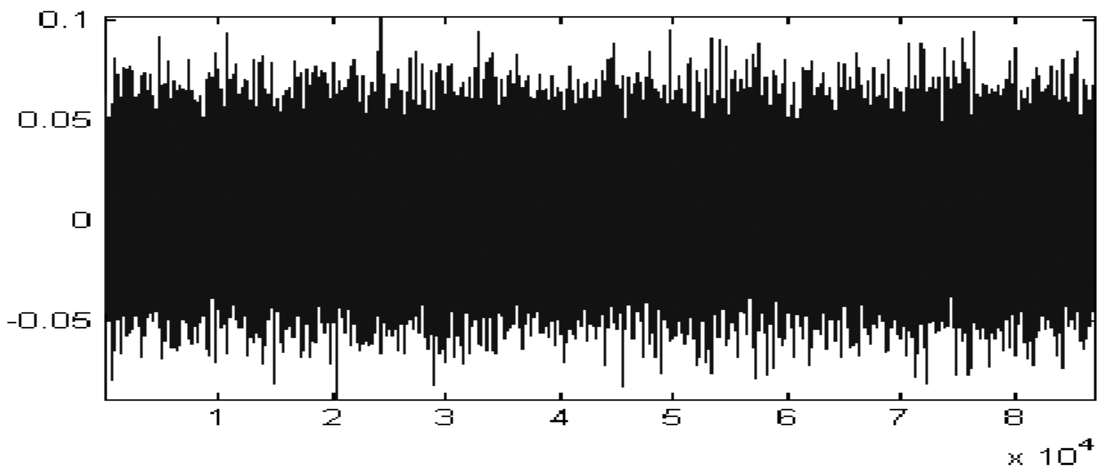


autocorrelation function

Excitation

- Multisine $$u(k)=\sum_{k=1}^{K} U_k \cos\left(2\pi \frac{k}{N} f_{\max} + \varphi(k)\right)$$
  - where $f_{\max}$ is the maximum frequency of the excitation signal, K is the number of frequency components
- Crest factor $$CF = \frac{\max(|u(t)|)}{u_{rms}(t)}$$

  minimizing CF with the selection of φ phases

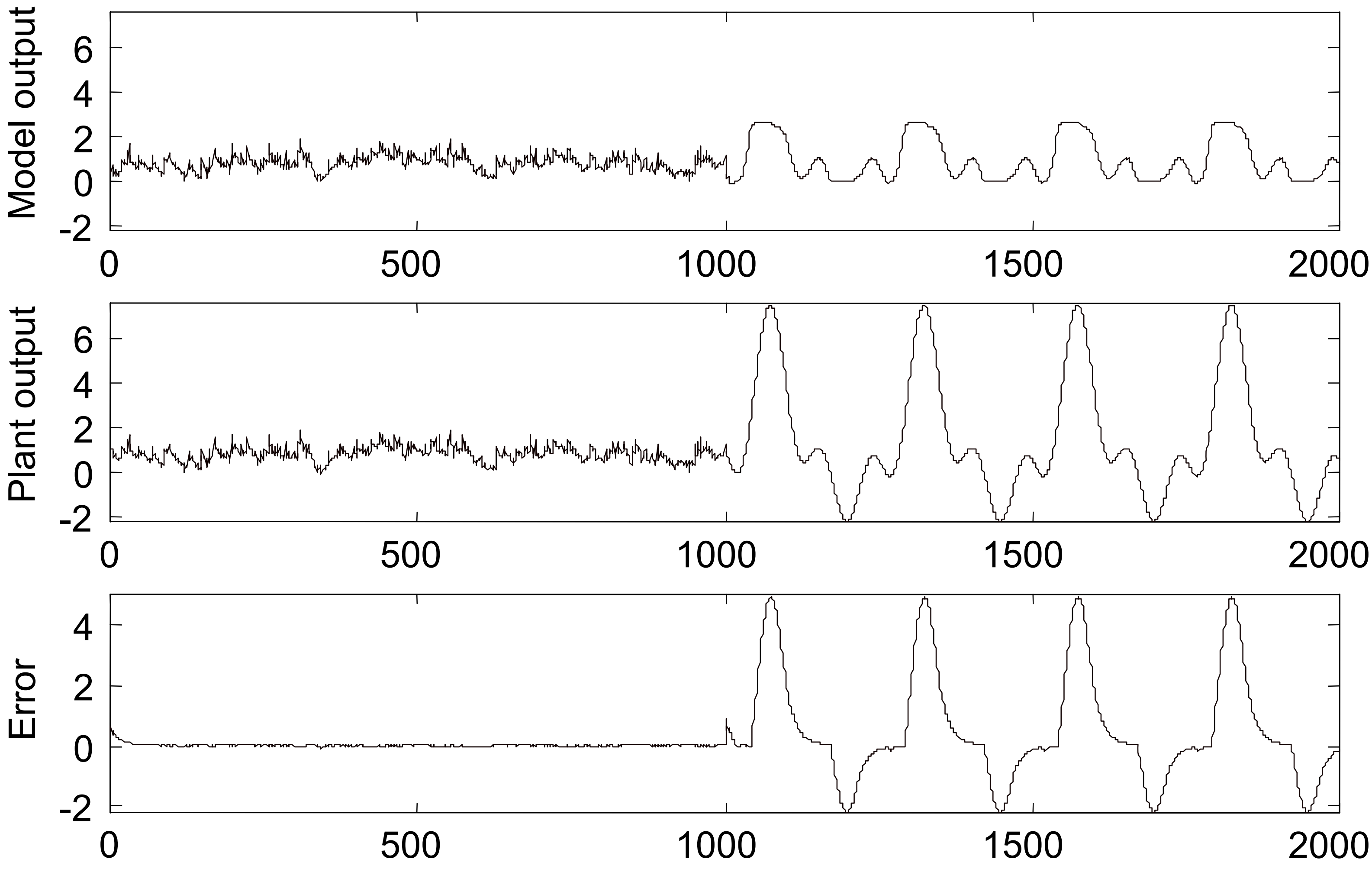


Multisine with minimal crest factor

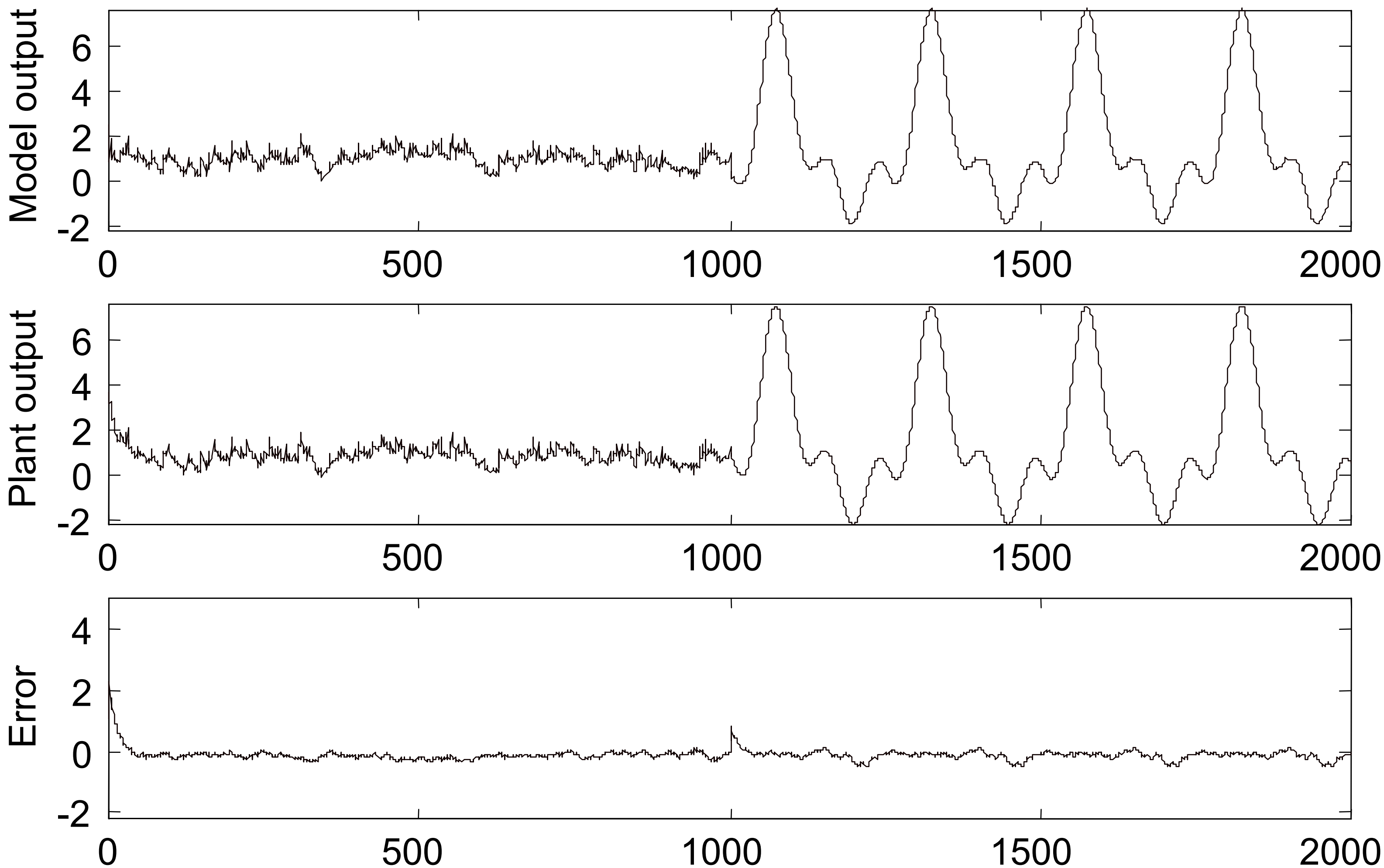
Excitation

- Persistent excitation
  - The excitation signal must be „rich” enough to excite all modes of the system
  - Mathematical formulation of persistent excitation
- For linear systems
  - Input signal should excite all frequencies, amplitude not so important
- For nonlinear systems
  - Input signal should excite all frequencies and amplitudes
  - Input signal should sample the full regressor space

The role of excitation: small excitation signal
(nonlinear system identification)



The role of excitation: large excitation signal
(nonlinear system identification)



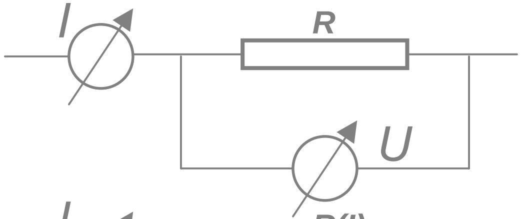
Modeling (some examples)

- Resistor modeling
- Model of a duct (an anti-noise problem)
- Model of a steel converter (model of a complex industrial process)
- Model of a signal (time series modeling)

Modeling (example)

- Resistor modeling
  - the goal of modeling: to get a description of a physical system (electrical component)
  - parametric model
    - linear model
    - constant parameter

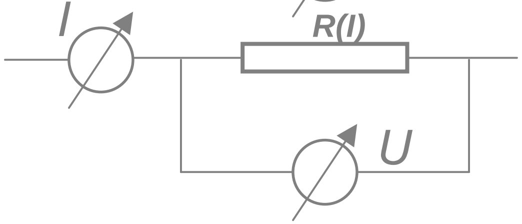
$$U = RI$$



DC

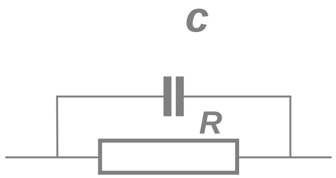
- variant model

$$U = R(I)I$$



- frequency dependent

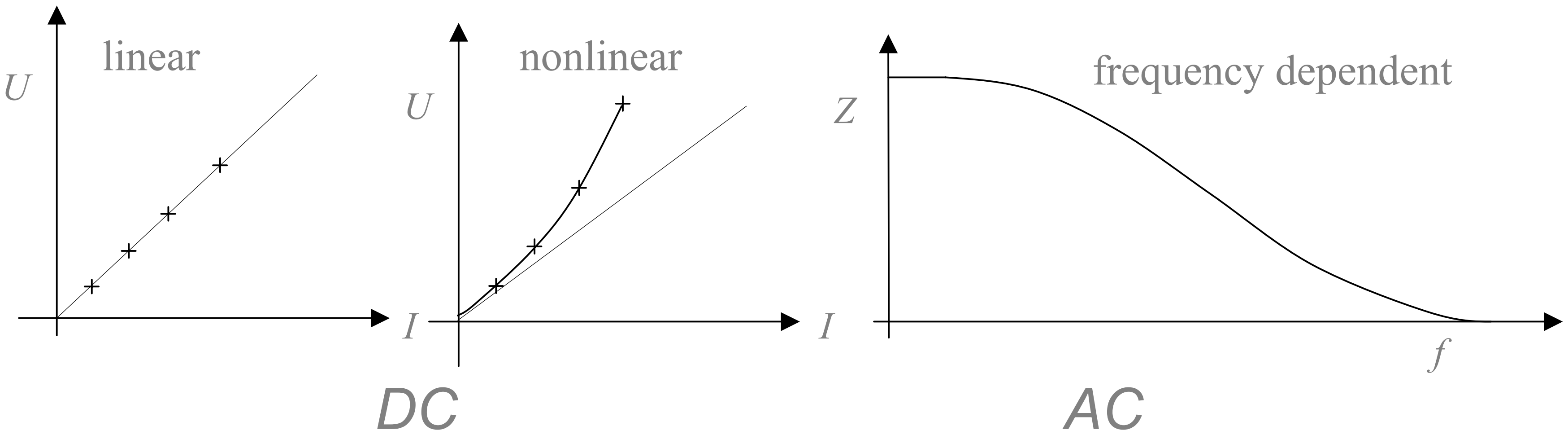
$$U(f) = Z(f)I(f) \qquad Z(f) = \frac{U(f)}{I(f)} \qquad Z(f) = \frac{R}{j\,2\pi\, f\,R\,C + 1}$$



AC

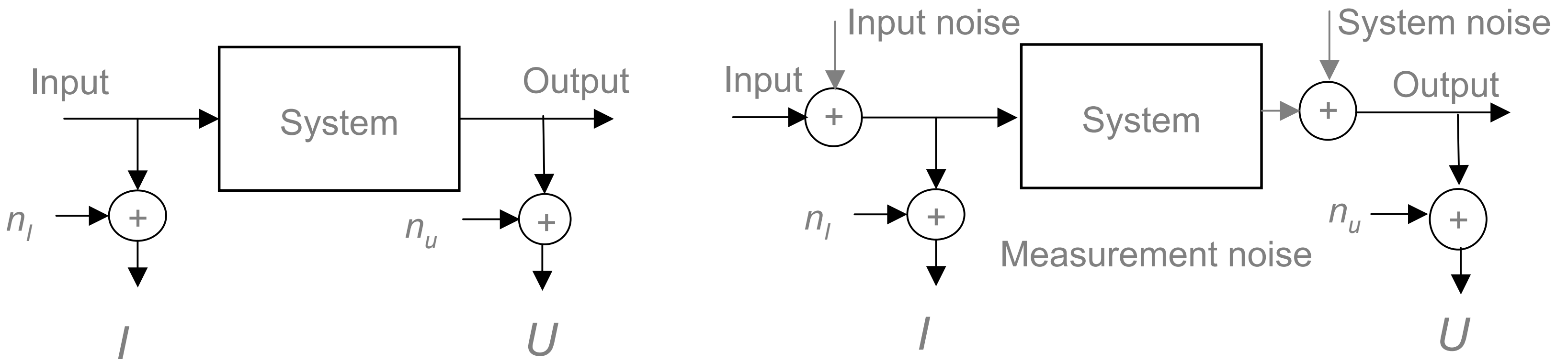
Modeling (example)

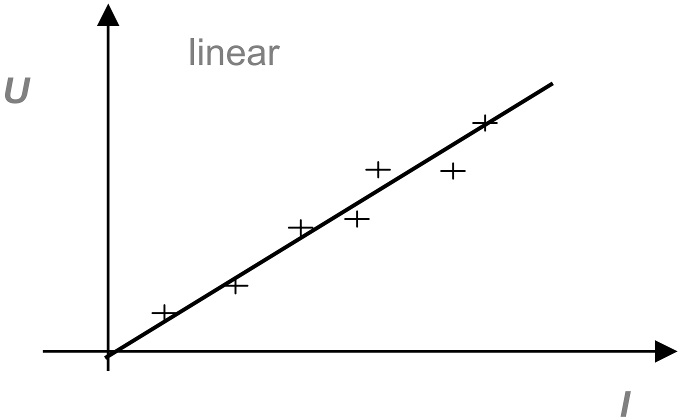
- Resistor modeling
  - nonparametric model



Modeling (example)

- Resistor modeling
  - parameter estimation based on noisy measurements

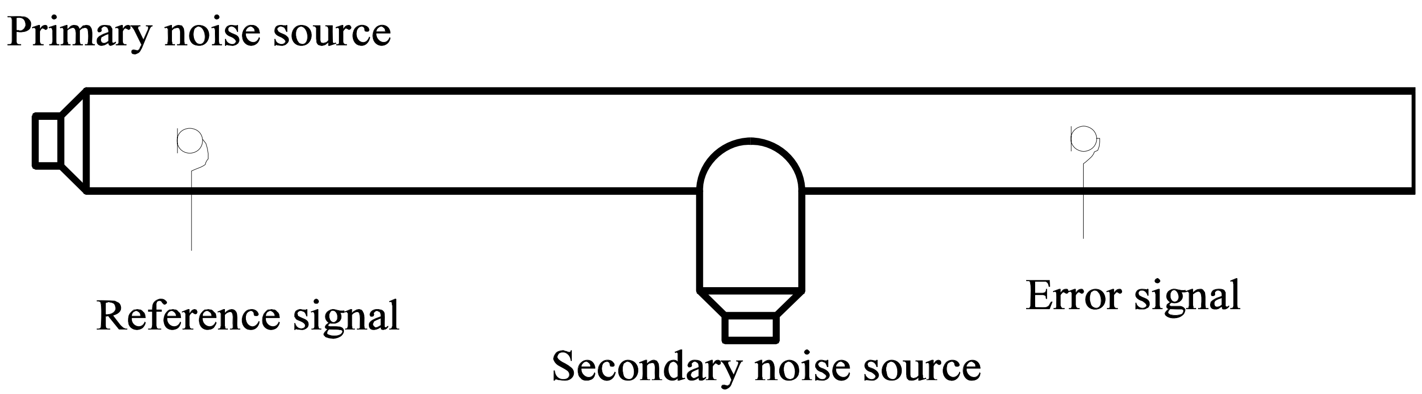




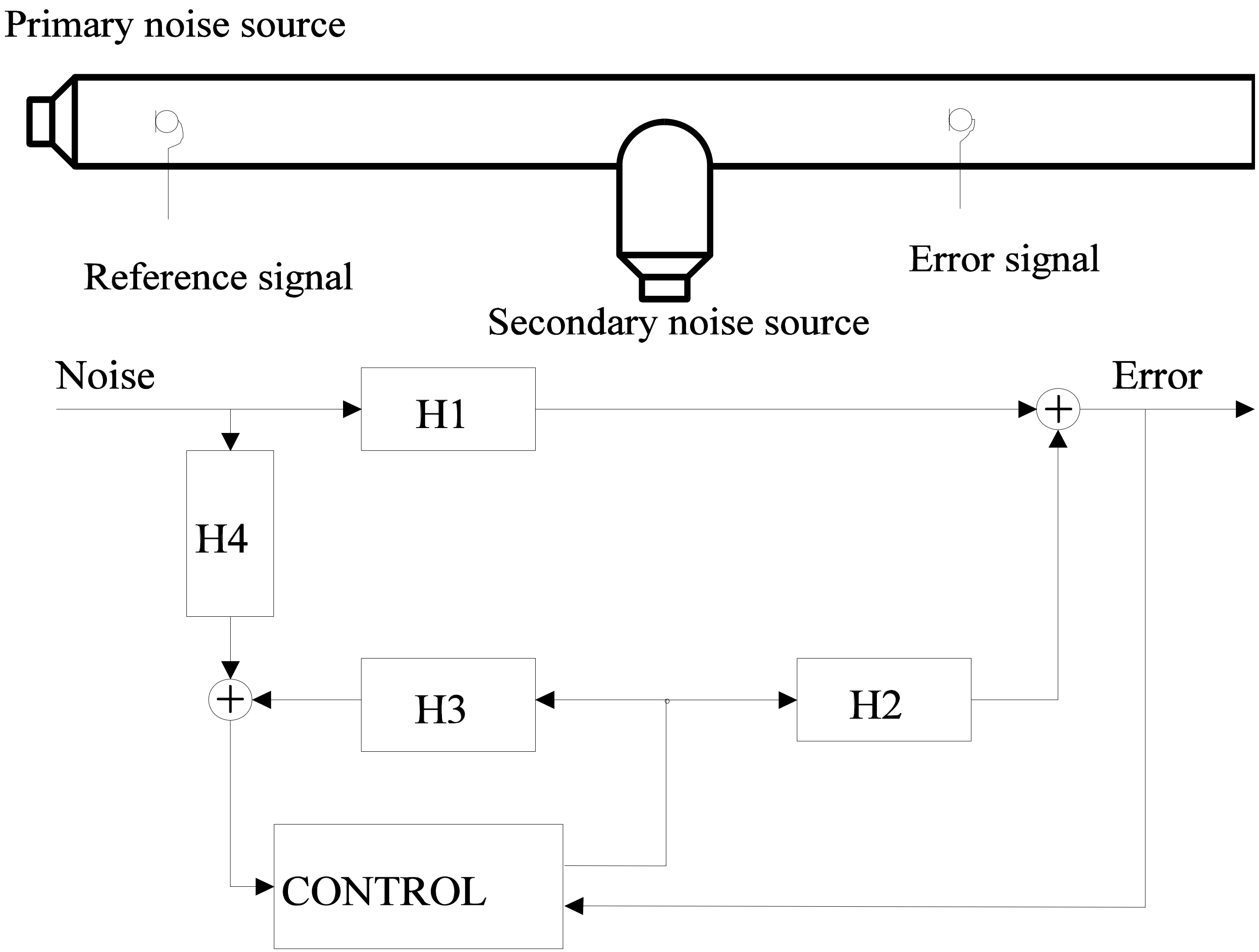
Modeling (example)

- Model of a duct
  - the goal of modeling: to design a controller for noise compensation.

    active noise control problem



Modeling (example)

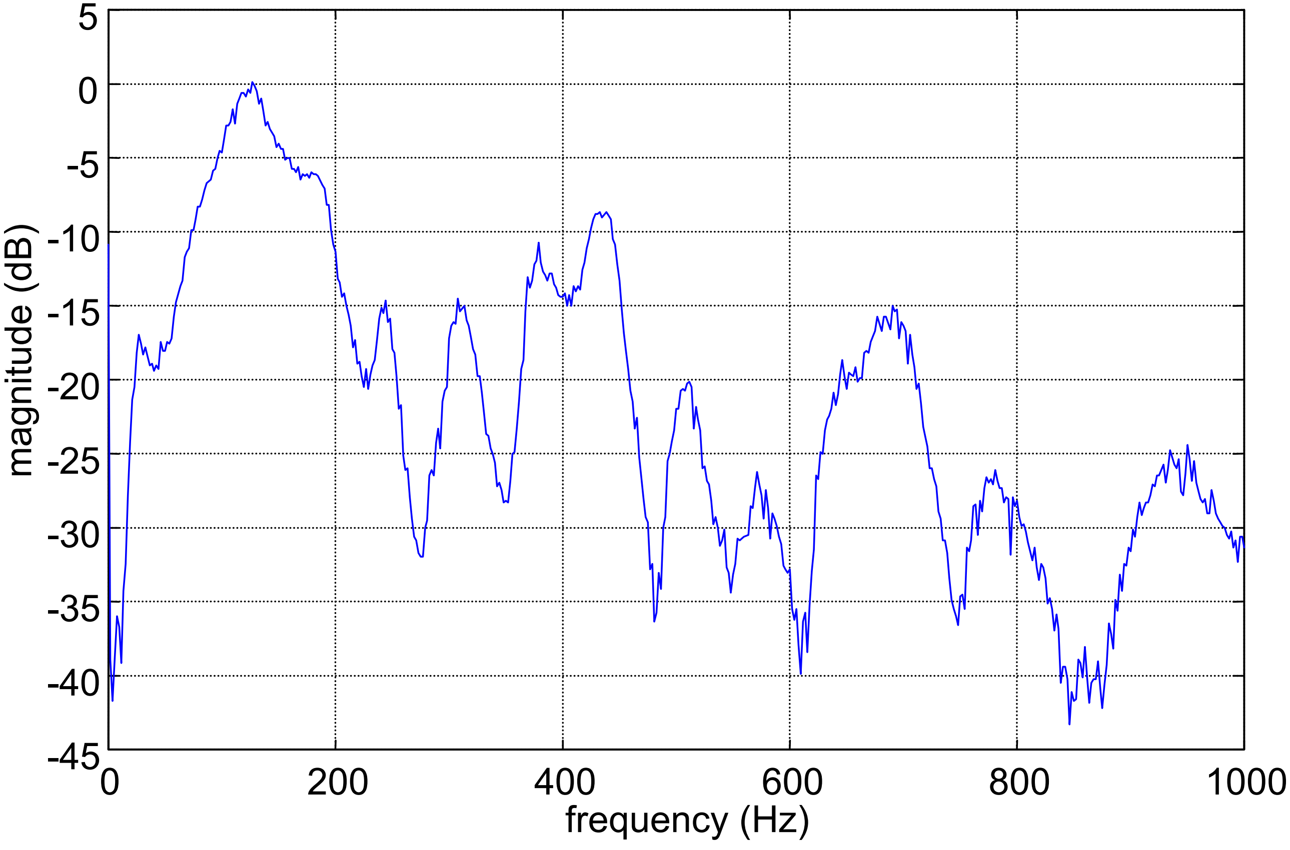


Modeling (example)

- Model of a duct
  - physical modeling: general knowledge about acoustical effects; propagation of sound, etc.
  - no physical insight. *Input:* sound pressure, *output:* sound pressure
  - what signals: stochastic or deterministic: periodic, non-periodic, combined, etc.
  - what frequency range
  - time invariant or not
  - fixed solution, adaptive solution. Model structure is fixed, model parameters are estimated and adjusted: adaptive solution

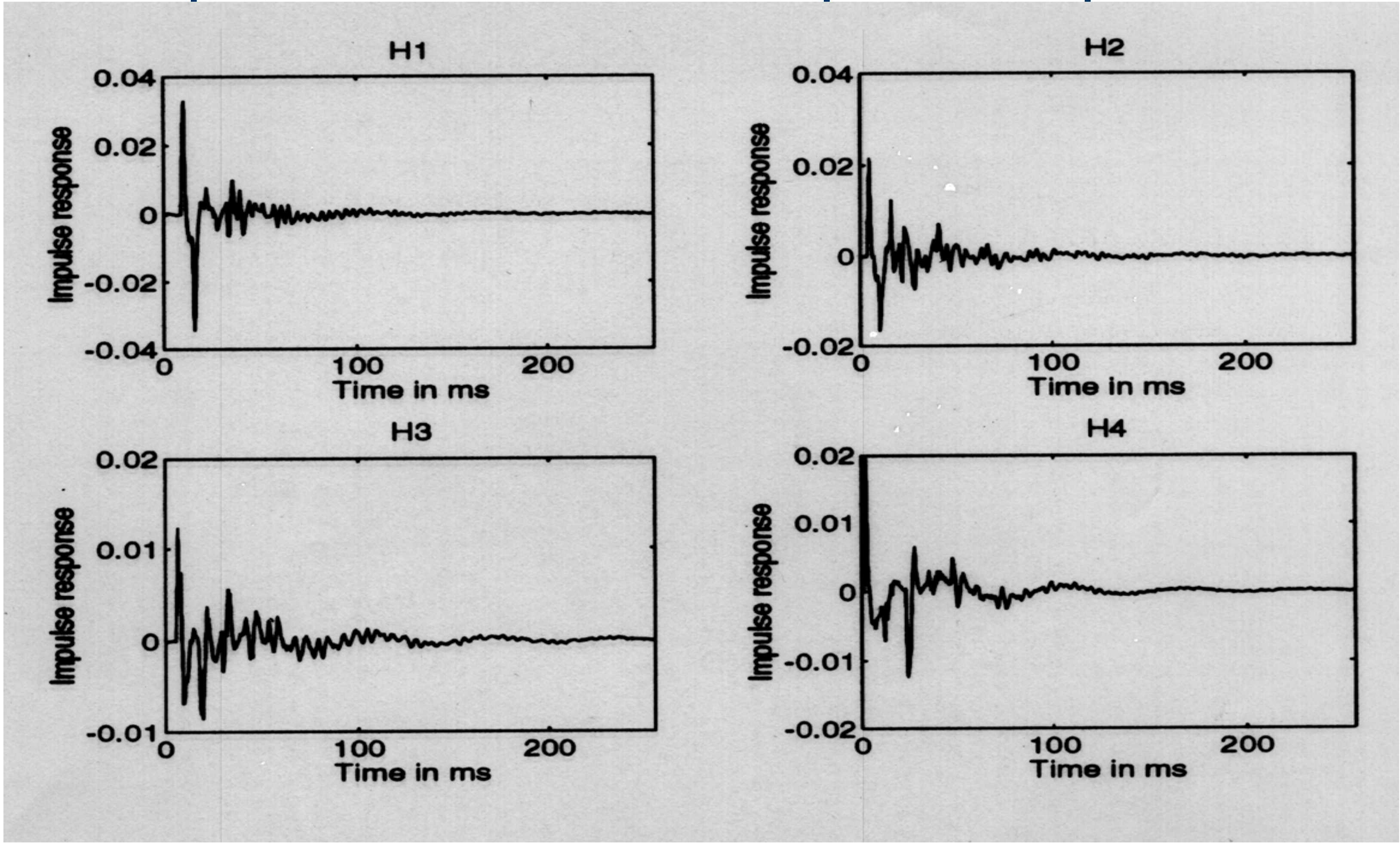
Modeling (example)

- Model of a duct
  - nonparametric model of the duct (H1)
  - FIR filter with 10-100 coefficients



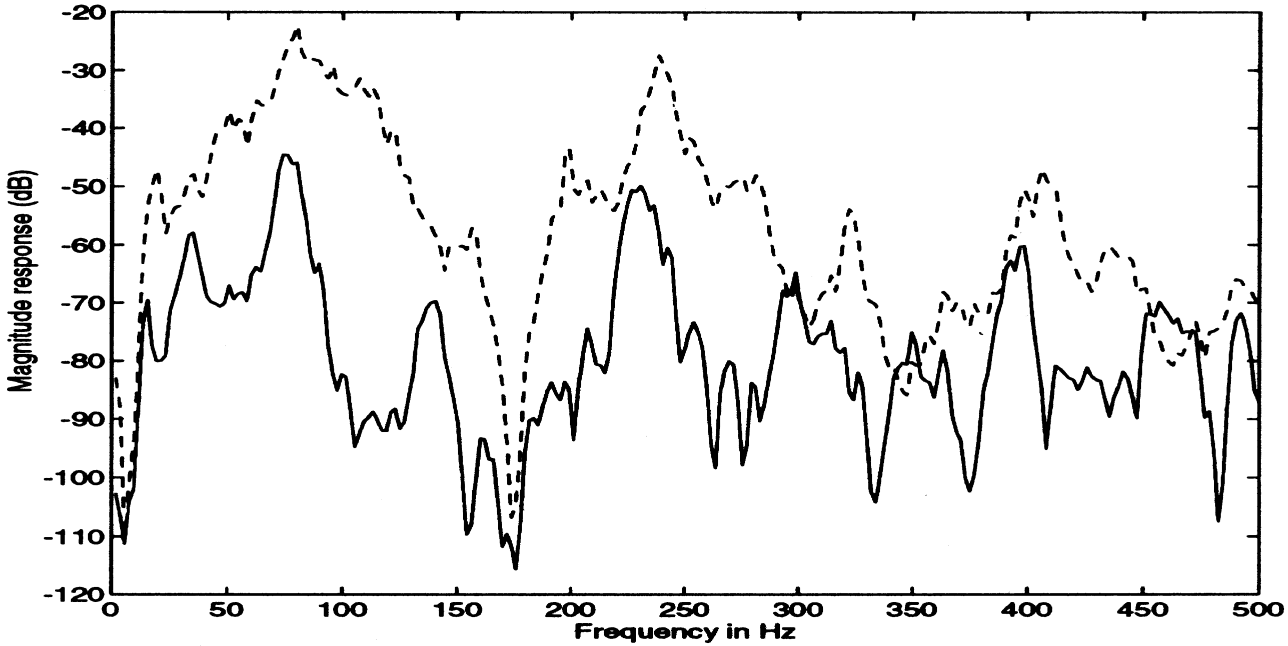
Modeling (example)

- Nonparametric models: impulse responses



Modeling (example)

- The effect of active noise compensation



Output error magnitude response;

noise excitation.

Frequency responses of the system without (dashed line) and with the application of adaptive controller (solid line).

Modeling (example)

- Model of a steel
  converter (LD converter)

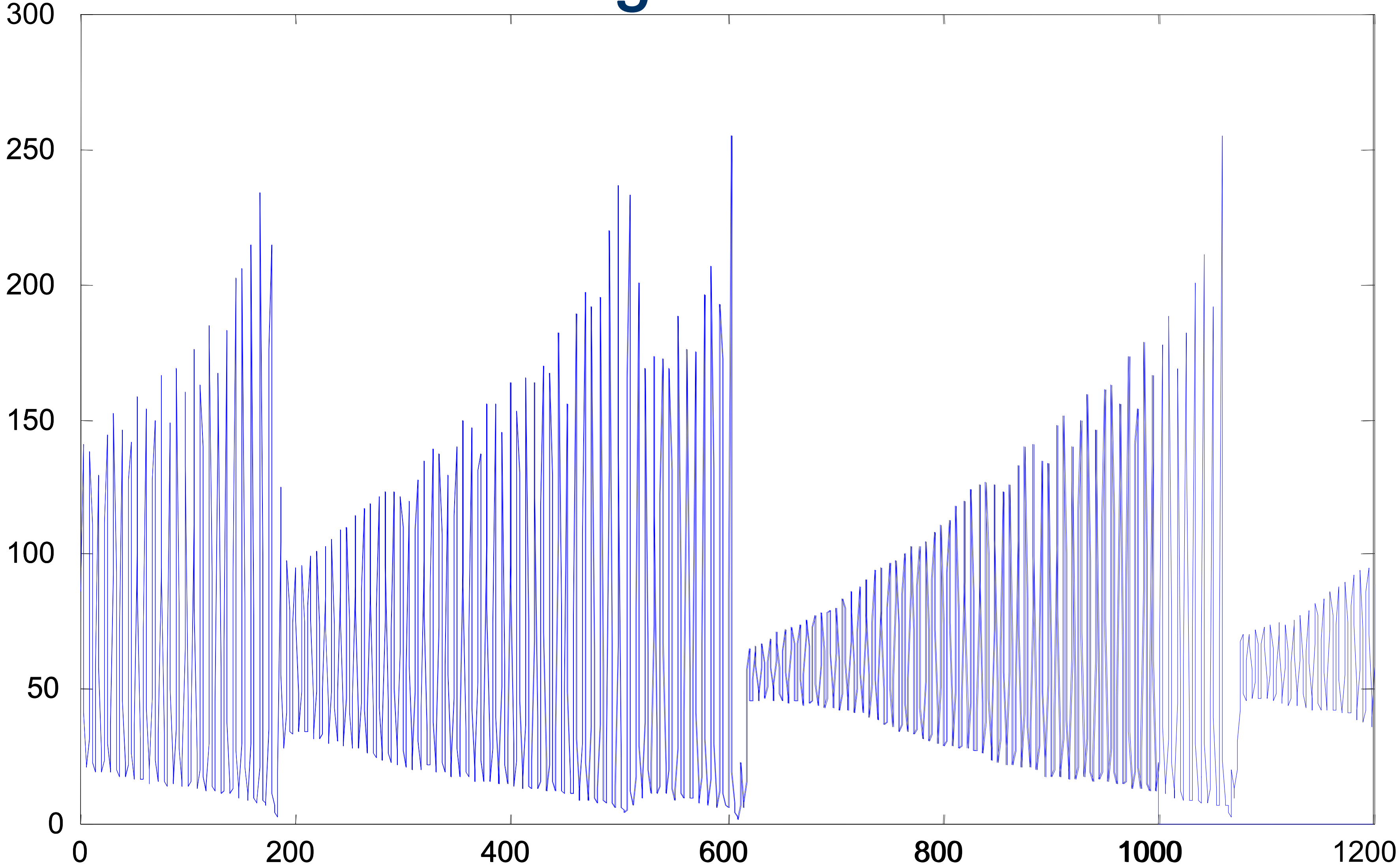
Modeling (example)

- Model of a steel converter (LD converter)
  - the goal of modeling: to control steel-making process to get predetermined quality steel
  - physical insight:
    - complex physical-chemical process with many inputs
    - heat balance, mass balance
    - many unmeasurable (input) variables (parameters)
  - no physical insight:
    - there are input-output measurement data
  - no possibility to design input signal, no possibility to cover the whole range of operation

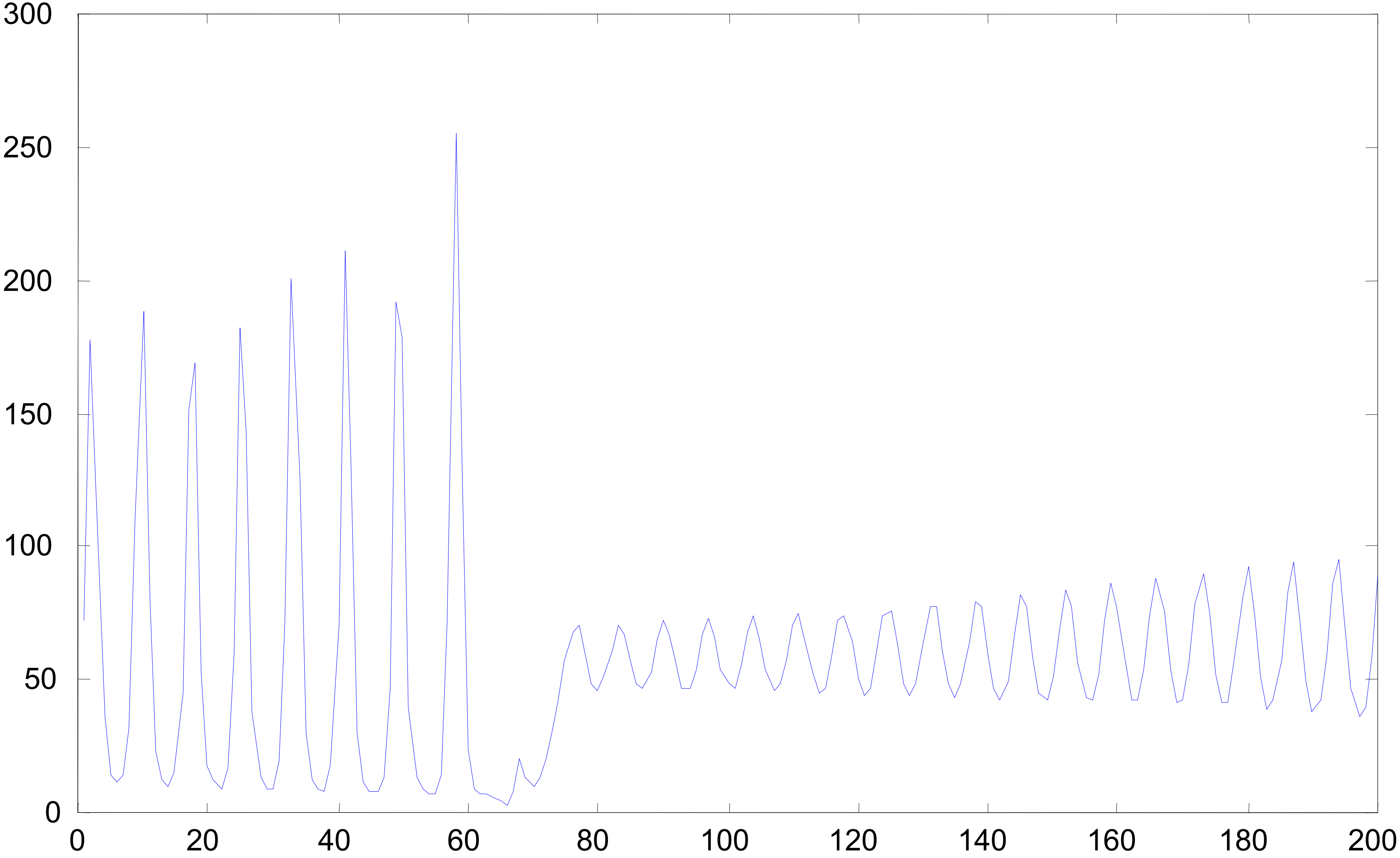
Modeling (example)

- Time series modeling
  - the goal of modeling: to predict the future behaviour of a signal (forecasting)
    - financial time series
    - physical phenomena e.g. sunspot activity
    - electrical load prediction
    - an interesting project: Santa Fe competition
    - etc.
  - signal modeling = system modeling

Time series modeling

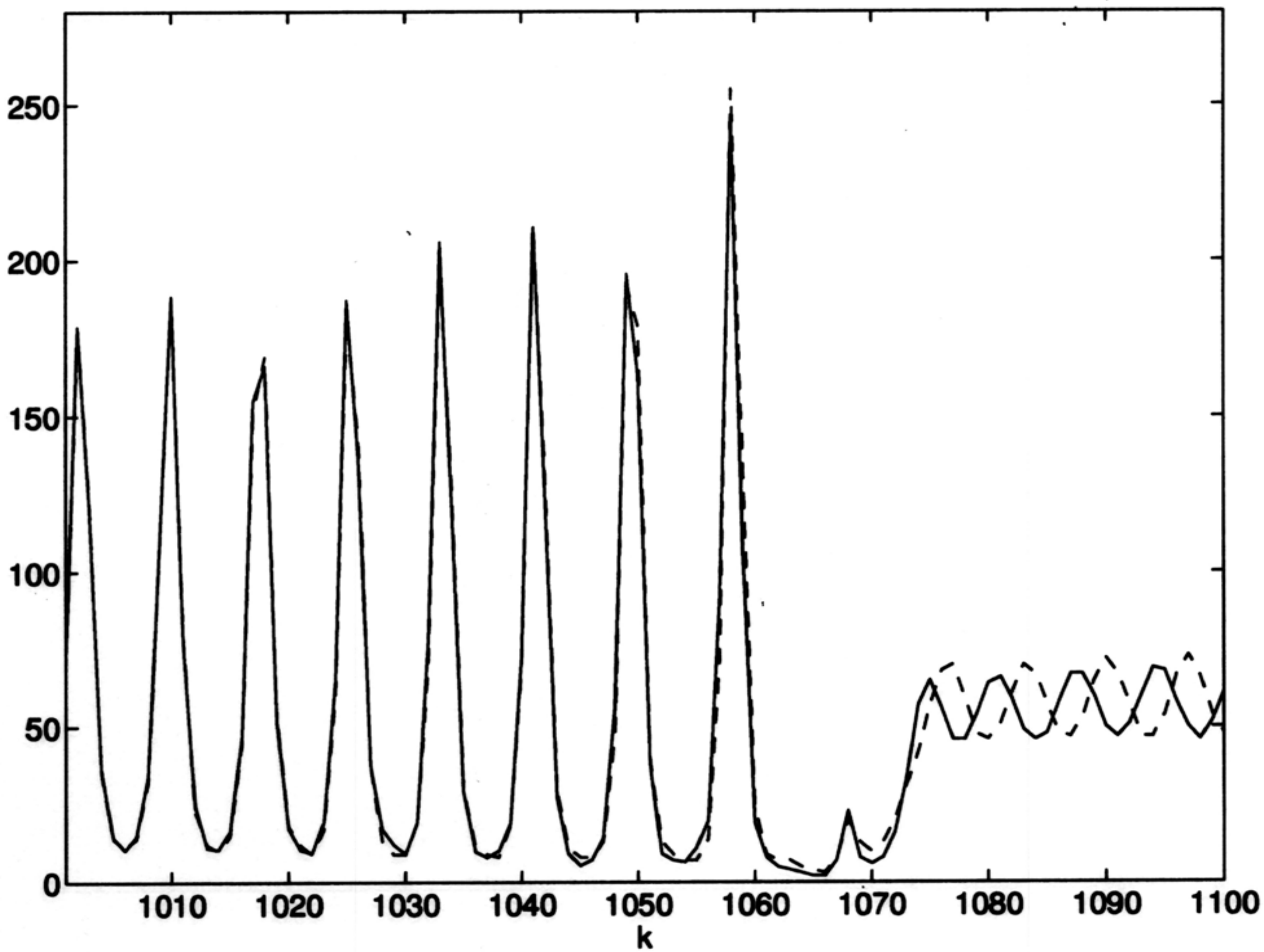


Time series modeling



Time series modeling

- Output of a neural model



References and further readings

Box, G.E.P and Jenkins, G.M: “Time Series Analysis: Forecasting and Control”, Revised Edition, Holden Day, 1976

Eykhoff, P. “System Identification, Parameter and State Estimation”, Wiley, New York, 1974.

Goodwin, G.C. and R. L. Payne, “Dynamic System Identification”, Academic Press, New York, 1977.

Horváth, G. “Neural Networks in Systems Identification”, (Chapter 4. in: S. Ablameyko, L. Goras, M. Gori and V. Piuri (Eds.) Neural Networks in Measurement Systems) NATO ASI, IOS Press, pp. 43-78. 2002.

Horváth, G., Dunay, R.: "Application of Neural Networks to Adaptive Filtering for Systems with External Feedback Paths." Proc. of The International Conferenace on Signal Processing Application and Technology. Vol. II. pp. 1222-1227. Dallas, Tx. 1994.

Ljung, L. “System Identification - Theory for the User”. Prentice-Hall, N.J. 2nd edition, 1999.

Pintelon R. and Schoukens, J. “System Identification. A Frequency Domain Approach”, IEEE Press, New York, 2001.

Pataki, B., Horváth, G., Strausz, Gy. and Talata, Zs. "Inverse Neural Modeling of a Linz-Donawitz Steel Converter" e & i Elektrotechnik und Informationstechnik, Vol. 117. No. 1. 2000. pp. 13-17.

Rissanen, J. “Modelling by Shortest Data Description”, Automatica, Vol. 14. pp. 465-471, 1978.

Sjöberg, J., Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.-Y. Glorennec, H. Hjalmarsson, and A. Juditsky: "Non-linear Black-box Modeling in System Identification: a Unified Overview", *Automatica*, 31:1691-1724, 1995.

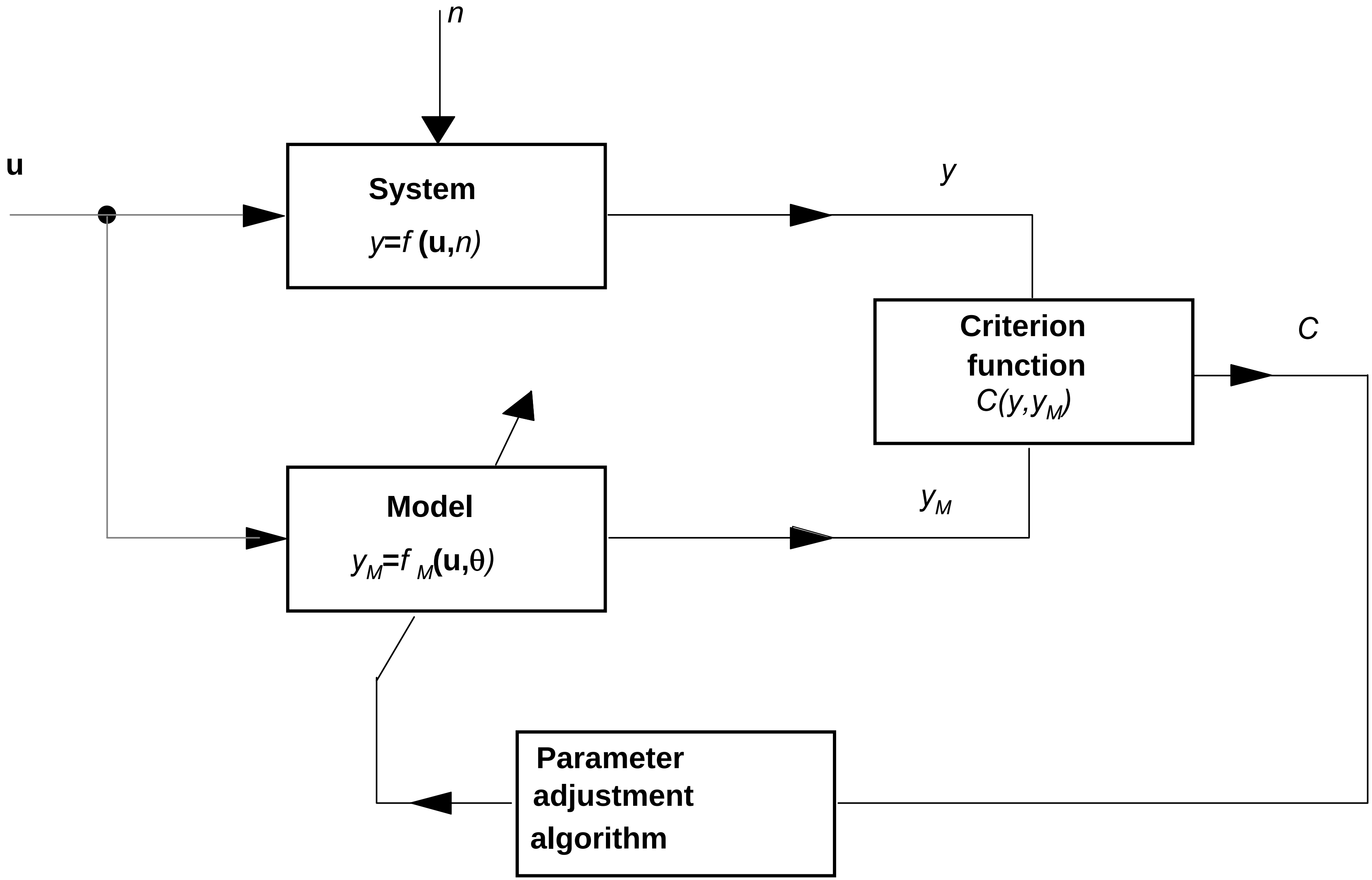
Söderström, T. and P. Stoica, “System Indentification”, Prentice Hall, Englewood Cliffs, NJ. 1989.

Weigend,. A.S and N.A Gershenfeld "Forecasting the Future and Understanding the Past" Vol.15. Santa Fe Institute Studies in the Science of Complexity, Reading, MA. Addison-Wesley, 1994.

Identification (linear systems)

- Parametric identification (parameter estimation)
  - LS estimation
  - ML estimation
  - Bayes estimation
- Nonparametric identification
  - Transient analysis
  - Correlation analysis
  - Frequency analysis

Parametric identification



Parametric identification

- Parameter estimation
  - linear system

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}(1)^T \\ \vdots \\ \mathbf{u}(N)^T \end{bmatrix}$$

$$y(i) = \mathbf{u}(i)^T \Theta + n(i) = \sum_{j=1}^{L} u_j(i)\Theta_j + n(i) \qquad i = 1,2,\ldots,N$$

$$\mathbf{y} = \mathbf{U}\Theta + \mathbf{n}$$

$$\mathbf{y}^T = \mathbf{y}_N^T = \left[y(1) \quad \cdots \quad y(N)\right]$$

  - linear-in-the parameter model

$$y_M(i) = \mathbf{u}(i)^T \hat{\Theta} = \sum_j u_j(i)\hat{\Theta}_j \qquad \mathbf{y}_M = \mathbf{U}\hat{\Theta}$$

  - criterion (loss) function

$$\varepsilon(\hat{\Theta}) = \mathbf{y} - \mathbf{y}_M\left(\hat{\Theta}\right) \qquad V\left(\hat{\Theta}\right) = V(\varepsilon(\hat{\Theta})) = V\left(\mathbf{y} - \mathbf{y}_M\right) = V\left(\mathbf{y} - \mathbf{y}_M\left(\hat{\Theta}\right)\right)$$

Parametric identification

- LS estimation

quadratic loss function

$$V(\hat{\boldsymbol{\Theta}}) = \frac{1}{2}\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon} = \frac{1}{2}\sum_{i=1}^{N}\varepsilon(i)^2 =$$

$$\frac{1}{2}\sum_{i=1}^{N}\left(y(i)-\mathbf{u}(i)^T\hat{\boldsymbol{\Theta}}\right)\left(y(i)-\mathbf{u}(i)^T\hat{\boldsymbol{\Theta}}\right) = \frac{1}{2}\left(\mathbf{y}_N - \mathbf{U}\hat{\boldsymbol{\Theta}}\right)^T\left(\mathbf{y}_N - \mathbf{U}\hat{\boldsymbol{\Theta}}\right)$$

LS estimate

$$\hat{\boldsymbol{\Theta}}_{LS} = \arg\min_{\hat{\boldsymbol{\Theta}}} V(\hat{\boldsymbol{\Theta}}) \qquad \frac{\partial V(\hat{\boldsymbol{\Theta}})}{\partial\hat{\boldsymbol{\Theta}}} = 0$$

$$\hat{\boldsymbol{\Theta}}_{LS} = (\mathbf{U}_N^T\mathbf{U}_N)^{-1}\mathbf{U}_N^T\mathbf{y}_N$$

Parametric identification

- Weighted LS estimation
  - weighted quadratic loss function

$$V(\hat{\mathbf{\Theta}})=\frac{1}{2}\sum_{i=1}^{N}\varepsilon(i)^2=\frac{1}{2}\sum_{i,k=1}^{N}\left(y(i)-\mathbf{u}(i)^T\hat{\mathbf{\Theta}}\right)q_{ik}\left(y(k)-\mathbf{u}(k)^T\hat{\mathbf{\Theta}}\right)=\frac{1}{2}\left(\mathbf{y}_N-\mathbf{U}\hat{\mathbf{\Theta}}\right)^T\mathbf{Q}\left(\mathbf{y}_N-\mathbf{U}\hat{\mathbf{\Theta}}\right)$$

weighted LS estimate

$$\hat{\mathbf{\Theta}}_{WLS}=(\mathbf{U}_N^T\mathbf{Q}\mathbf{U}_N)^{-1}\mathbf{U}_N^T\mathbf{Q}\mathbf{y}_N$$

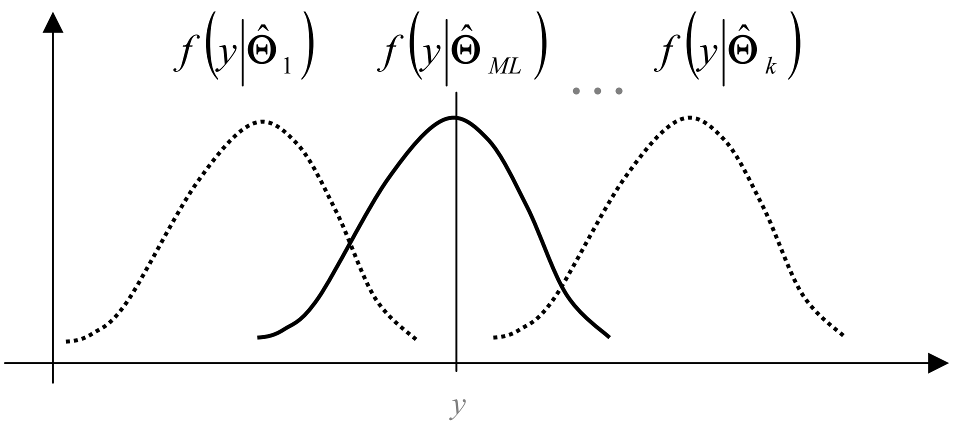
  - Gauss-Markov estimate (*BLUE=best linear unbiased estimate*)

$$\mathrm{E}\{\mathbf{n}\}=\mathbf{0}\qquad \mathrm{cov}[\mathbf{n}]=\Sigma\qquad \mathbf{Q}=\mathbf{\Sigma}^{-1}$$

$$\hat{\mathbf{\Theta}}_{WLS}=(\mathbf{U}_N^T\mathbf{\Sigma}^{-1}\mathbf{U}_N)^{-1}\mathbf{U}_N^T\mathbf{\Sigma}^{-1}\mathbf{y}_N$$

Parametric identification

- Maximum likelihood estimation
  - we select the estimate which makes the given observations most probable



  - likelihood function, log likelihood function

$$f(\mathbf{y}_N|\hat{\mathbf{\Theta}}) \qquad \log f(\mathbf{y}_N|\hat{\mathbf{\Theta}})$$

  - maximum likelihood estimate

$$\hat{\mathbf{\Theta}}_{ML} = \arg\max_{\hat{\Theta}} f(\mathbf{y}_N|\hat{\mathbf{\Theta}}) \qquad \frac{\partial}{\partial \hat{\mathbf{\Theta}}} \log f(\mathbf{y}_N|\hat{\mathbf{\Theta}}) = \mathbf{0}$$

Parametric identification

- Properties of ML estimates
  - consistency

$$\lim_{N\to\infty} P\left\{\left|\hat{\mathbf{\Theta}}_{ML(N)} - \mathbf{\Theta}\right| > \varepsilon\right\} = 0 \quad \text{for any} \quad \varepsilon > 0$$

  - asymptotic normality

$\hat{\mathbf{\Theta}}_{ML(N)}$ converges to a normal random variable as *N*→∞

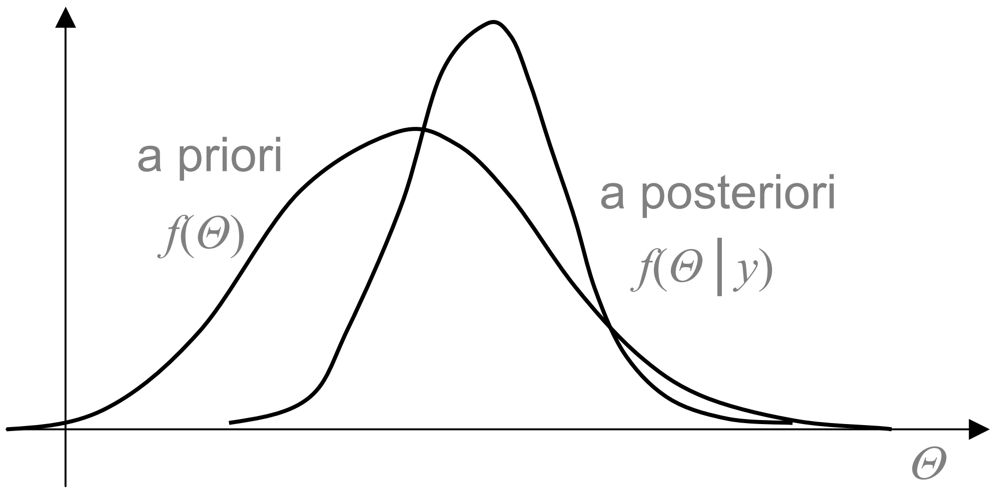
  - asymptotic efficiency: the variance reaches *Cramer-Rao* lower bound

$$\lim_{N\to\infty} \operatorname{var}(\hat{\mathbf{\Theta}}_{ML(N)} - \mathbf{\Theta}) = -\left(E\left\{\frac{\partial^2 \ln f\left(y\middle|\mathbf{\Theta}\right)}{\partial \mathbf{\Theta}^2}\right\}\right)^{-1}$$

  - Gauss-Markov if $f(\mathbf{y}_N | \hat{\mathbf{\Theta}})$ Gaussian

Parametric identification

- Bayes estimation
  - the parameter Θ is a random variable with known pdf

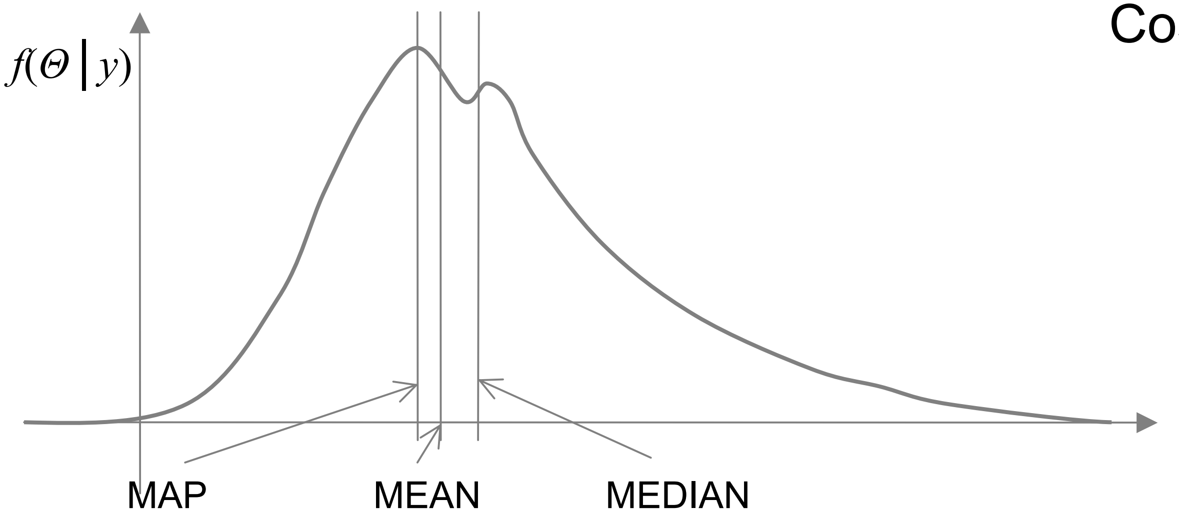


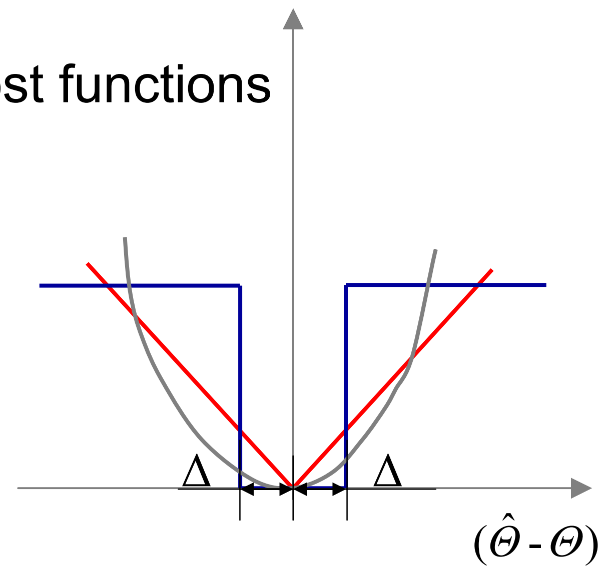
the loss function $V_B(\hat{\boldsymbol{\Theta}}) = \int C(\hat{\boldsymbol{\Theta}} | \boldsymbol{\Theta}) f(\boldsymbol{\Theta} | \mathbf{y}) d\boldsymbol{\Theta}$

  - Bayes estimate $\hat{\boldsymbol{\Theta}}_B = \arg\min_{\hat{\boldsymbol{\Theta}}} \int C(\hat{\boldsymbol{\Theta}} | \boldsymbol{\Theta}) f(\boldsymbol{\Theta} | \mathbf{y}) d\boldsymbol{\Theta}$

Parametric identification

- Bayes estimation with different cost functions
  - median $$C\left(\hat{\Theta}\middle|\Theta\right) = \left|\hat{\Theta} - \Theta\right|$$
  - MAP $$C\left(\hat{\Theta}\middle|\Theta\right) = \begin{cases} \text{Const} & \text{if } \left|\hat{\Theta} - \Theta\right| \le \Delta \\ 0 & \text{otherwise} \end{cases}$$
  - mean $$C\left(\hat{\Theta}\middle|\Theta\right) = \left\|\hat{\Theta} - \Theta\right\|^2$$





Parametric identification

- Recursive estimations

  - $\hat{\mathbf{\Theta}}(k)$ is estimated from $\{y(i)\}_{i=1}^{k-1}$
  - $y(k)$ is predicted as $y_M(k) = \mathbf{u}(k)^T \hat{\mathbf{\Theta}}$
  - the error $e(k) = y(k) - y_M(k)$ is determined
  - update the estimate $\hat{\mathbf{\Theta}}(k+1)$ from $\hat{\mathbf{\Theta}}(k)$ and $e(k)$

Parametric identification

- Recursive estimations
  - least mean square *LMS*

$$\hat{\mathbf{\Theta}}(k+1)=\hat{\mathbf{\Theta}}(k)+\mu(k)\varepsilon(k)\mathbf{u}(k)$$

  - the simplest gradient-based iterative algorithm
  - it has important role in neural network training

Parametric identification

- Recursive estimations
  - recursive least square *RLS*

$$\hat{\mathbf{\Theta}}(k+1)=\hat{\mathbf{\Theta}}(k)+\mathbf{K}(k+1)\varepsilon(k)$$

$$\mathbf{K}(k+1)=\mathbf{P}(k)\mathbf{U}(k+1)\left[\mathbf{I}+\mathbf{U}(k+1)\mathbf{P}(k)\mathbf{U}^{T}(k+1)\right]^{-1}$$

$$\mathbf{P}(k+1)=\mathbf{P}(k)-\mathbf{P}(k)\mathbf{U}^{T}(k+1)\left[\mathbf{I}+\mathbf{U}(k+1)\mathbf{P}(k)\mathbf{U}^{T}(k+1)\right]^{-1}\mathbf{U}(k+1)\mathbf{P}(k)$$

where $\mathbf{P}(k)$ is defined as $\mathbf{P}(k)=\left[\mathbf{U}(k)^{T}\mathbf{U}(k)\right]^{-1}$

$\mathbf{K}(k)$ changes the search direction from instantenous gradient direction

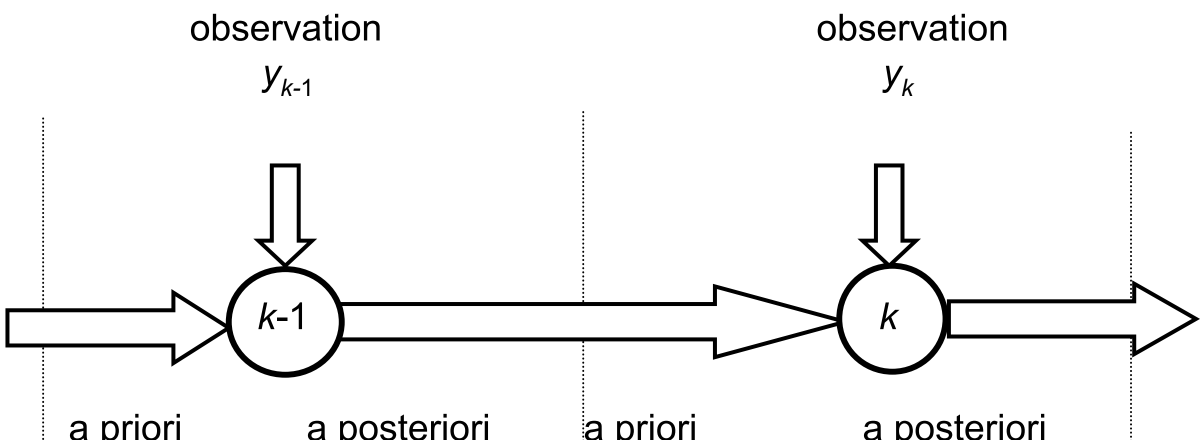
Parametric identification

- Recursive estimations
  - recursive Bayes a posteriori df $f(\boldsymbol{\Theta}|y)$

$$f(\boldsymbol{\Theta}|\mathbf{y}_1)=\frac{f(\mathbf{y}_1|\boldsymbol{\Theta})f(\boldsymbol{\Theta})}{\int_{-\infty}^{+\infty} f(\mathbf{y}_1|\boldsymbol{\Theta})f(\boldsymbol{\Theta})d\boldsymbol{\Theta}}$$

$$f(\boldsymbol{\Theta}|\mathbf{y}_1,\mathbf{y}_2)=\frac{f(\mathbf{y}_2|\mathbf{y}_1,\boldsymbol{\Theta})f(\mathbf{y}_1,\boldsymbol{\Theta})}{\int_{-\infty}^{+\infty} f(\mathbf{y}_2|\mathbf{y}_1,\boldsymbol{\Theta})f(\mathbf{y}_1,\boldsymbol{\Theta})d\boldsymbol{\Theta}}$$

$$f(\boldsymbol{\Theta}|\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_k)=\frac{f(\mathbf{y}_k|\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_{k-1},\boldsymbol{\Theta})f(\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_{k-1},\boldsymbol{\Theta})}{\int_{-\infty}^{+\infty} f(\mathbf{y}_2|\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_{k-1},\boldsymbol{\Theta})f(\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_{k-1},\boldsymbol{\Theta})d\boldsymbol{\Theta}}$$



Parametric identification

- Parameter estimation
  - Least square *less a priori information*
  - Maximum Likelihood

    conditional probability density f. $f(\mathbf{y}_N|\hat{\mathbf{\Theta}})$
  - Bayes *most a priori information*

    a priori probability density f. $f(\mathbf{\Theta})$

    conditional probability density f. $f(\mathbf{y}_N|\hat{\mathbf{\Theta}})$

    cost function $C\left(\hat{\mathbf{\Theta}}|\mathbf{\Theta}\right)$

Non-parametric identification

- Frequency-domain analysis
  - frequency characteristic, frequency response
  - spectral analysis
- Time-domain analysis
  - impulse response
  - step response
  - correlation analysis
- These approaches are for linear dynamical systems

Non-parametric identification (frequency domain)

- Secial input signals
  - sinusoid
  - multisine

$$u(t) = \sum_{k=1}^{K} U_k e^{j\left(2\pi \frac{k}{N} f_{\max} + \varphi(k)\right)}$$

where $f_{\max}$ is the maximum frequency of the excitation signal
K is the number of frequency components

crest factor $$CF = \frac{\max\left(\left|u(t)\right|\right)}{u_{rms}(t)}$$

minimizing CF with the selection of φ phases

Non-parametric identification (frequency domain)

- Frequency response
  - Power density spectrum, periodogram
  - Calculation of periodogram
  - Effect of finite registration length
  - Windowing (smoothing)

References and further readings

Eykhoff, P. System Identification, Parameter and State Estimation, Wiley, New York, 1974.

Ljung, L. ”System Identification - Theory for the User” Prentice-Hall, N.J. 2nd edition, 1999.

Goodwin, G.C. and R.L. Payne, Dynamic System Identification, Academic Press, New York, 1977.

Rissanen, J. “Stochastic Complexity in Statistical Inquiry”, Series in Computer Science”. Vol. 15 World Scientific, 1989.

Sage, A.P. and J.L. Melsa, Estimation Theory with Application to Communications and Control, McGraw-Hill, New York, 1971.

Pintelon, R. and J. Schoukens, System Identification. A Frequency Domain Approach, IEEE Press, New York, 2001.

Söderström, T. and P. Stoica, System Indentification, Prentice Hall, Englewood Cliffs, NJ. 1989.

Van Trees, H.L. Detection Estimation and Modulation Theory, Part I. Wiley, New York, 1968.

Black box modeling

Black-box modeling

- **Why do we use black-box models?**
  - the lack of physical insight: *physical modeling is not possible*
  - the physical knowledge is too complex, there are mathematical difficulties; physical modeling is *possible in principle* but *not possible in practice*
  - there is no need for physical modeling, (only the behaviour of the system should be modeled)
  - black-box modeling may be much simpler

Black-box modeling

- Steps of black-box modeling
  - select a *model structure*
  - determine the *size* of the model (the *number of parameters*)
  - use *observed (measured) data* to adjust the model (estimate the *model order* - the number of parameters - and the *numerical values* of the parameters)
  - validate the resulted model

Black-box modeling

- Model structure selection

  Dynamic models: $y_M(k) = f(\Theta, \varphi(k))$ with $\varphi(k)$ regressor-vectors

- how to chose $\varphi(k)$ regressor-vectors?

  past inputs

$$\varphi(k) = [u(k-1), u(k-2), \ldots, u(k-N)]$$

  past inputs and outputs

$$\varphi(k) = [u(k-1), u(k-2), \ldots, u(k-N), y_M(k-1), y_M(k-2), \ldots, y_M(k-P)]$$

  past inputs and system outputs

$$\varphi(k) = [u(k-1), u(k-2), \ldots, u(k-N), y(k-1), y(k-2), \ldots, y(k-P)]$$

  past inputs, system outputs and errors

$$\varphi(k) = [u(k-1), \ldots, u(k-N), y(k-1), \ldots, y(k-P), \varepsilon(k-1), \ldots, \varepsilon(k-L)]$$

  past inputs, outputs and errors

$$\varphi(k) = [u(k-1), \ldots, u(k-N), y_M(k-1), \ldots, y_M(k-P), \varepsilon(k-1), \ldots, \varepsilon(k-L), \varepsilon_u(k-1), \ldots, \varepsilon_u(k-K)]$$

Black-box identification

- Linear dynamic model structures

FIR

$$y_M(k) = a_1 u(k-1) + a_2 u(k-2) + \ldots + a_N u(k-N)$$

ARX

$$y_M(k) = a_1 u(k-1) + \ldots + a_N u(k-N) + b_1 y(k-1) + \ldots + b_P y(k-P)$$

OE

$$y_M(k) = a_1 u(k-1) + \ldots + a_N u(k-N) + b_1 y_M(k-1) + \ldots + b_P y_M(k-P)$$

ARMAX

$$y_M(k) = a_1 u(k-1) + \ldots + a_N u(k-N) + b_1 y(k-1) + \ldots + b_P y(k-P) + c_1 \varepsilon(k-1) + \ldots + c_L \varepsilon(k-L)$$

BJ

$$y_M(k) = a_1 u(k-1) + \ldots + a_N u(k-N) + b_1 y(k-1) + \ldots + b_P y(k-P) +$$
$$+ c_1 \varepsilon(k-1) + \ldots + c_L \varepsilon(k-L) + d_1 \varepsilon_u(k-1) + \ldots + d_K \varepsilon_u(k-K)$$

$$\vdots$$

$$\mathbf{\Theta} = [a_1 a_2 \ldots a_N]^T$$

parameter vector $\mathbf{\Theta} = [a_1 a_2 \ldots a_N, b_1 b_2 \ldots b_P, c_1 c_2 \ldots c_L, d_1 d_2 \ldots d_K]^T$

Black-box identification

- Non-linear dynamic model structures

NFIR

$$y_M(k)=f\left(u(k-1),u(k-2),\,...,\,u(k-N)\right)$$

NARX

$$y_M(k)=f\left(u(k-1),\,...,u(k-N),\;y(k-1),\,...,y(k-P)\right)$$

NOE

$$y_M(k)=f\left(u(k-1),\;...,u(k-N),\,y_M(k-1),\,...,y_M(k-P)\right)$$

NARMAX

$$y_M(k)=f\left(u(k-1),\;...,u(k-N),\,y(k-1),\;...,\;y(k-P),\varepsilon(k-1),\,...,\varepsilon(k-L)\right)$$

NBJ

$$y_M(k)=f[u(k-1),...,u(k-N),y(k-1),...,y(k-P),\varepsilon(k-1),...,\varepsilon(k-L),\varepsilon_u(k-1),....,\varepsilon_u(k-K)]$$

Black-box identification

- How to choose nonlinear mapping?

$$y_M(k) = f(\boldsymbol{\Theta}, \boldsymbol{\varphi}(k))$$

  - linear-in-the-parameter models

$$y_M(k) = \sum_{j=1}^{n} \alpha_j f_j(\boldsymbol{\varphi}(k)) \qquad \boldsymbol{\Theta} = [\alpha_1 \alpha_2 \ldots \alpha_n]^T$$

  - nonlinear-in-the-parameters

$$y_M(k) = \sum_{j=1}^{n} \alpha_j f_j(\boldsymbol{\beta}_j, \boldsymbol{\varphi}(k)) \qquad \boldsymbol{\Theta} = [\alpha_1 \alpha_2 \ldots \alpha_n, \beta_1 \beta_2 \ldots \beta_n]^T$$

Black-box identification

- Model validation, model order selection
  - residual test
  - Information Criterion:
    - AIC Akaike Information Criterion
    - BIC Bayesian Information Criterion
    - NIC Network Information Criterion
    - etc.
  - Rissanen MDL (Minimum Description Length)
  - cross validation

Black-box identification

- Model validation: residual test

  residual: the difference between the model and the measured (system) output $\varepsilon(k) = \mathbf{y}(k) - \mathbf{y}_M(k)$

  - autocorrelation test:
    - are the residuals white (white noise process with mean 0)?
    - are residuals normally distributed?
    - are residuals symmetrically distributed?
  - cross correlation test:
    - are residuals uncorrelated with the previous inputs?

Black-box identification

- Model validation: residual test
  autocorrelation test:

$$\hat{C}_{\varepsilon\varepsilon}(\tau) = \frac{1}{N-\tau}\sum_{k=\tau+1}^{N}\varepsilon(k)\varepsilon(k-\tau)$$

$$\mathbf{r}_{\varepsilon\varepsilon} = \frac{1}{\hat{C}_{\varepsilon\varepsilon}(0)}\left(\hat{C}_{\varepsilon\varepsilon}(1) \ \ \ldots \ \ \hat{C}_{\varepsilon\varepsilon}(m)\right)^{T}$$

$$\sqrt{N}\mathbf{r}_{\varepsilon\varepsilon} \xrightarrow{\text{dist}} \mathcal{N}(0,\ \mathbf{I})$$

Black-box identification

- Model validation: residual test
  - cross-correlation test:

$$\hat{C}_{u\varepsilon}(\tau) = \frac{1}{N-\tau}\sum_{k=\tau+1}^{N}\varepsilon(k)u(k-\tau)$$

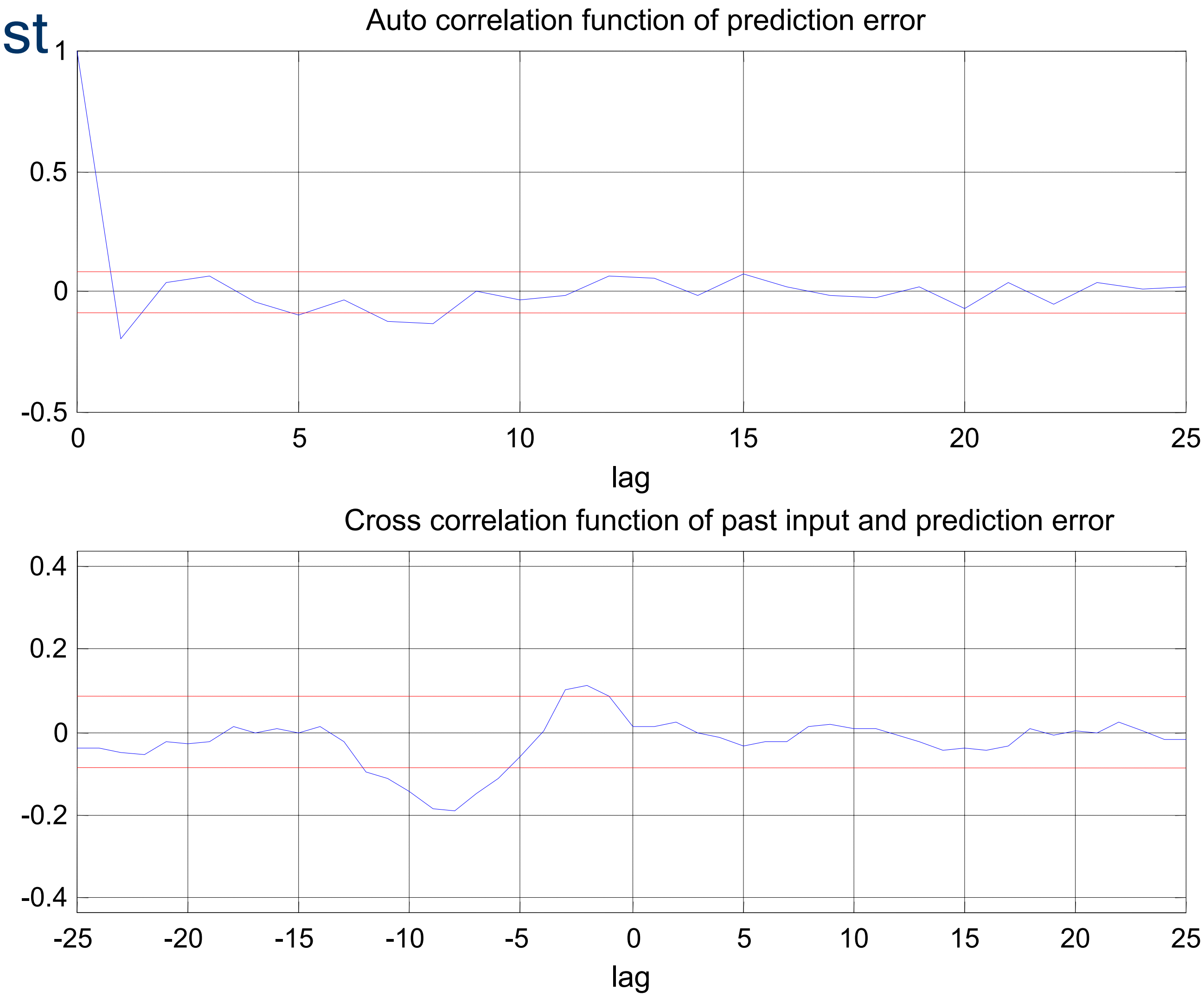
$$\mathbf{r}_{u\varepsilon}(m) = \frac{1}{\sqrt{\hat{C}_{u\varepsilon}(0)}}\left(\hat{C}_{u\varepsilon}(\tau+1)\ \ldots\ \hat{C}_{u\varepsilon}(\tau+m)\right)^{T}$$

$$\sqrt{N}\mathbf{r}_{u\varepsilon} \xrightarrow{\text{dist}} \mathcal{N}(0,\ \hat{\mathbf{R}}_{uu})$$

$$\hat{\mathbf{R}}_{uu} = \frac{1}{N-m}\sum_{k=m+1}^{N}\begin{bmatrix} u_{k-1} \\ \vdots \\ u_{k-m}\end{bmatrix}\begin{bmatrix} u_{k-1} & \cdots & u_{k-m}\end{bmatrix}$$

Black-box identification

- residual test



Black-box identification

- Model validation, model order selection
  - the importance of a priori knowledge (physical insight)
  - under- or over-parametrization
  - Occam's razor
  - variance-bias trade-off

Black-box identification

- Model validation, model order selection
  - criterions: noise term+penalty term
    - AIC:

$$\mathrm{AIC}(\hat{\Theta}) = (-2)\ \log\ (\text{max imum likelihood}) + 2p$$

$$\mathrm{AIC}(p) = (-2)\log L\left(\hat{\Theta}_N\right) + 2p$$

    - NIC network information criterion

      extension of AIC for neural networks

    - MDL $\mathrm{MDL}(p) = (-2)\log L(\hat{\Theta}_N) + \frac{p}{2}\log N + \frac{p}{2}\log\left\|\hat{\Theta}_N\right\|_M$

      p = number of parameters

      M = Fisher information matrix

Black-box identification

- Model validation, model order selection
  - cross validation
    - testing the model on new data (from the same problem)
    - leave out one cross validation
    - leave out *k* cross validation

Black-box identification

- Model validation, model order selection
  - variance-bias trade-off

    difference between the model and the real system

    - model class is not properly selected: *bias*
    - actual parameters of the model are not correct: *variance*

Black-box identification

- Model validation, model order selection
  - variance-bias trade-off

$$y(k) = f_o(\Theta, \varphi(k)) + n(k) \qquad n(k) \text{ white noise with variance } \sigma$$

$$V(\Theta) = \mathrm{E}\left\{\|\mathbf{y} - f(\Theta)\|^2\right\} = \boldsymbol{\sigma} + \mathrm{E}\left\{\left\| f_0(\Theta, \varphi(k)) - f\left(\hat{\Theta}, \varphi(k)\right)\right\|^2\right\}$$

$$E\{V(\Theta)\} = \boldsymbol{\sigma} + \mathrm{E}\left\{\left\| f_0(\Theta, \varphi(k)) - f\left(\hat{\Theta}, \varphi(k)\right)\right\|^2\right\}$$

$$\approx \boldsymbol{\sigma} + \mathrm{E}\left\{\left\| f_0(\Theta, \varphi(k)) - f\left(\Theta^*(m), \varphi(k)\right)\right\|^2\right\} + \mathrm{E}\left\{\left\| f\left(\Theta^*(m), \varphi(k)\right) - f\left(\hat{\Theta}, \varphi(k)\right)\right\|^2\right\}$$

noise bias variance

The order of the model (m) is the dimension of $\varphi(k)$.

The larger m the smaller bias and the larger variance

Black-box identification

- Model validation, model order selection
  - approaches
    - A sequence of models are used with **increasing** *m*

      Validation using cross validation or some criterion e.g. AIC, MDL, etc.
    - A complex model structure is used with a lot of parameters (over-parametrized model)

      Select important parameters
      - regularization
      - early stopping
      - pruning

Neural modeling

- Neural networks are (general) nonlinear black-box structures with “interesting” properties
  - general architecture
  - universal approximator
  - non-sensitive to over-parametrization
  - inherent regularization

Neural networks

- Why neural networks?
  - There are many other black-box modeling approaches: e.g. polynomial regression.
  - Difficulty: *curse of dimensionality*
  - In high-dimensional (N) problem and using M-th order polynomial the number of the independently adjustable parameters will grow as N^M.
  - To get a trained neural network with good generalization capability the dimension of the input space has significant effect on the size of required training data set.

Neural networks

- The advantages of neural approach
  - Neural nets (MLP) use basis functions to approximate nonlinear mappings, which depend on the function to be approximated.
  - This adaptive basis function set gives the possibility to decrease the number of free parameters in our general model structure.

Other black-box structures

- Wavelets
  - mother function (wavelet), dilation, translation
- Volterra series

$$y_M(k) = \sum_{l=0}^{\infty} g_l u(k-l) + \sum_{l=0}^{\infty}\sum_{s=0}^{\infty} g_{ls}\, u(k-l)\, u(k-s) + \sum_{l=0}^{\infty}\sum_{s=0}^{\infty}\sum_{r=0}^{\infty} g_{lsr}\, u(k-l)u(k-s)u(k-r) + \cdots$$

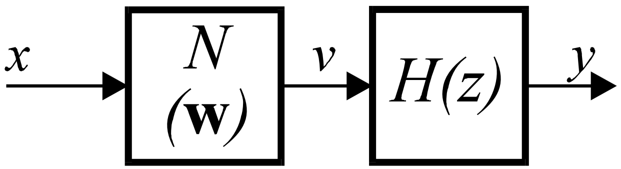
Volterra series can be applied succesfully for weakly nonlinear systems and impractical in strongly nonlinear systems

Other black-box structures

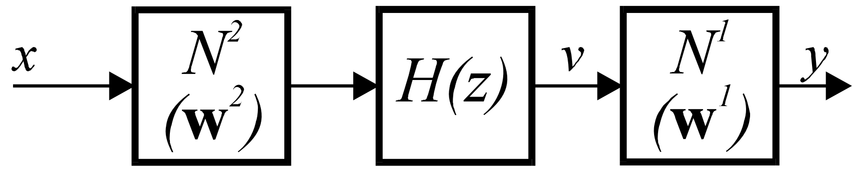
- Fuzzy models, fuzzy neural models
  - general nonlinear modeling approach
- Wiener, Hammerstein, Wiener-Hammerstein
  - dynamic linear system + static nonlinear
  - static nonlinear + dynamic linear system
  - dynamic linear system + static nonlinear + dynamic linear
- Narendra structures
  - other combined linear dynamic and nonlinear static systems

Combined models

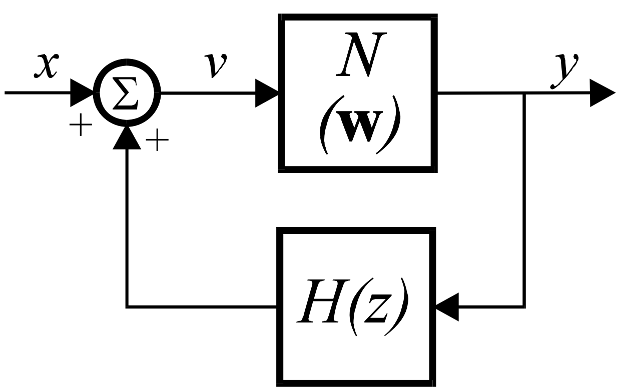
- Narendra structures



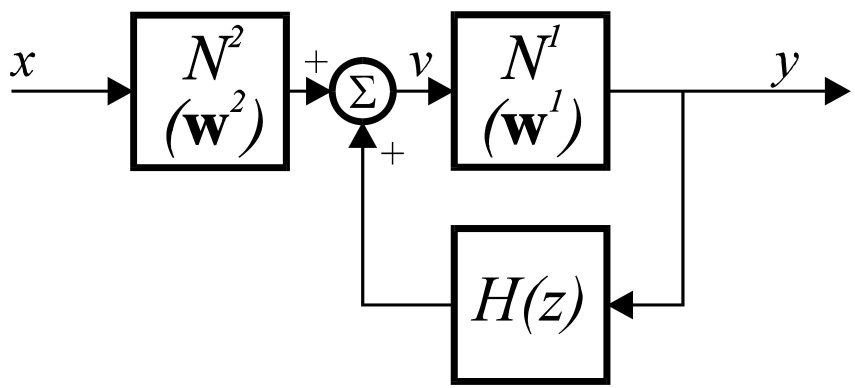
(a) modell



(b) model



(c) model



(d) model

References and further readings

Akaike, H. “Information Theory and an Extension of the Maximum Likelihood Principle” Second Intnl. Symposium on Information Theory. Akadémiai Kiadó, Budapest, pp. 267-281. 1972.

Akaike, H. “A New Look at the Statistical Model Identification” IEEE Trans. On Automatic Control, Vol. 19. No. 9. pp. 716-723. 1974.

Haykin, S.: "Neural Networks. A Comprehensive Foundation" Prentice Hall, N. J.1999.

L. Ljung, ”System Identification - Theory for the User” Prentice-Hall, N.J. 2nd edition, 1999.

Narendra, K. S. and Pathasarathy, K. "Identification and Control of Dynamical Systems Using Neural Networks," IEEE Trans. Neural Networks, Vol. 1. 1990. pp.

Noboru Murata, Shuji Yoshizawa and Shun-Ichi Amari “Network Information Criterion - Determining the Number of Hidden Units for an Artificial Neural Network Model” IEEE Trans. on Neural Networks, Vol. 5. No. 6. Pp. 865-871

Pataki, B., Horváth, G., Strausz, Gy. and Talata, Zs. "Inverse Neural Modeling of a Linz-Donawitz Steel Converter" e & i Elektrotechnik und Informationstechnik, Vol. 117. No. 1. 2000. pp. 13-17.

M.B. Priestley, “ Non-linear and Non-stationary Time Series Analysis” Academic Press, London, 1988.

Rissanen, J. “Stochastic Complexity in Statistical Inquiry”, Series in Computer Science”. Vol. 15 World Scientific, 1989.

J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.-Y. Glorennec, H. Hjalmarsson, and A. Juditsky: "Non-linear Black-box Modeling in System Identification: a Unified Overview", *Automatica*, 31:1691-1724, 1995.

A. S Weigend,. - N.A Gershenfeld "Forecasting the Future and Understanding the Past" Vol.15. Santa Fe Institute Studies in the Science of Complexity, Reading, MA. Addison-Wesley, 1994.

Neural networks

Outline

- Introduction
- Neural networks
  - elementary neurons
  - classical neural structures
  - general approach
  - computational capabilities of NNs
- Learning (parameter estimation)
  - supervised learning
  - unsupervised learning
  - analytic learning
- Support vector machines
  - SVM architectures
  - statistical learning theory
- General questions of network design
  - generalization
  - model selection
  - model validation

Neural networks

- Elementary neurons
  - linear combiner
  - basis-function neuron
- Classical neural architectures
  - feed-forward
  - feedback
- General approach
  - nonlinear function of regressors
  - linear combination of basis functions
- Computational capabilities of NNs
  - approximation of function
  - classification

Neural networks (a definition)

Neural networks are massively parallel distributed information processing systems, implemented in hardware or software form

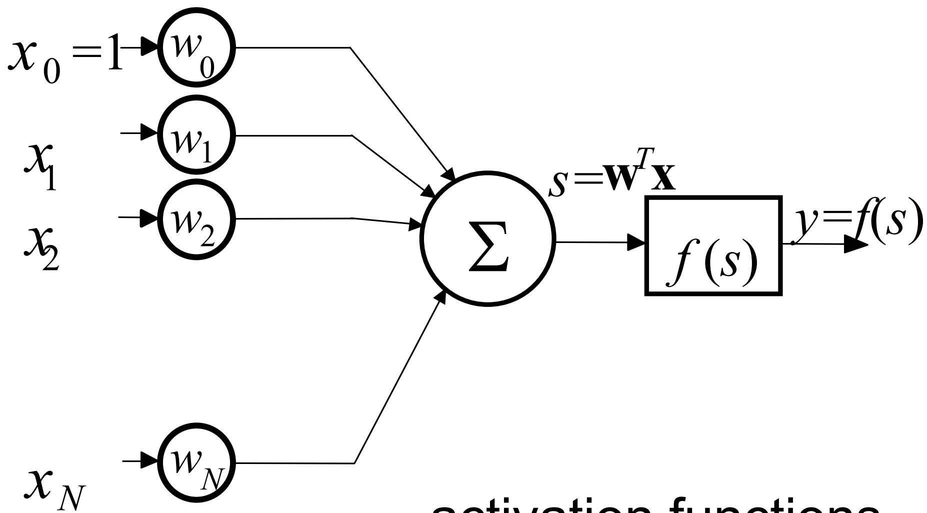
- made up of: a great number highly interconnected identical or similar simple processing units (*processing elements, neurons*) which are doing local processing, and are arranged in ordered topology,
- have *learning algorithm* to acquire knowledge from their environment, using examples
- have *recall algorithm* to use the learned knowledge

Neural networks (main features)

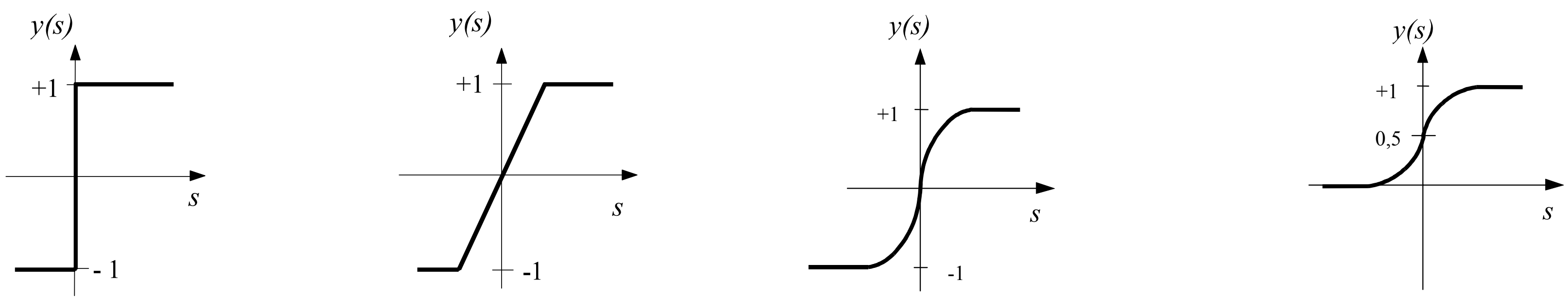
- Main features
  - complex nonlinear input-output mapping
  - adaptivity, learning capability
  - distributed architecture
  - fault tolerance
  - VLSI implementation
  - neurobiological analogy

The elementary neuron (1)

- Linear combiner with nonlinear activation function



activation functions



$$y=\begin{cases}+1 & s>0\\ -1 & s\leq 0\end{cases}$$

a.)

$$y=\begin{cases}+1 & s>1\\ s & -1\leq s\leq 1\\ -1 & s<-1\end{cases}$$

b.)

$$y=\frac{1-e^{-Ks}}{1+e^{-Ks}}\quad ; K>0$$

c.)

$$y=\frac{1}{1+e^{-Ks}}\quad ; K>0$$

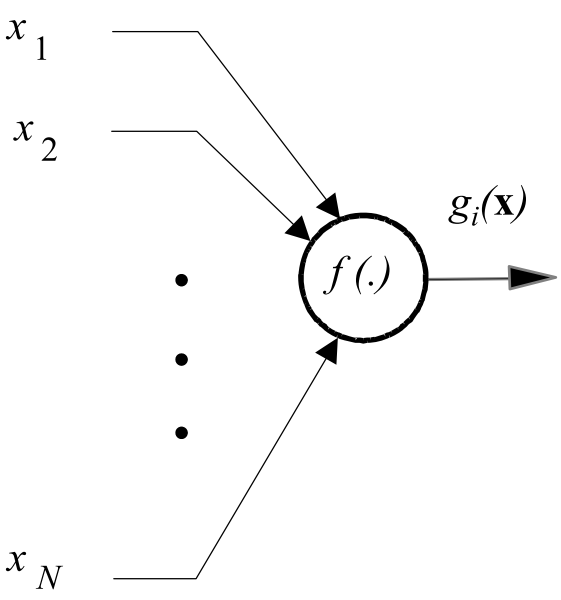
d.)

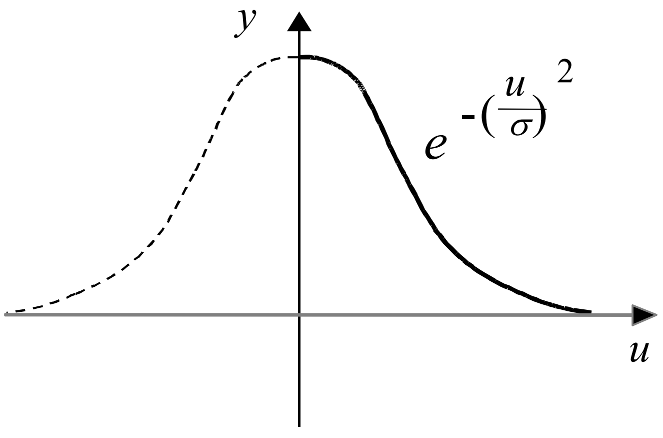
Elementary neuron (2)

- Neuron with basis function

$$y = \sum_i w_i g_i(\mathbf{x})$$

Basis functions $g_i(\mathbf{x}) = g\|\mathbf{x} - \mathbf{c}_i\|$ e.g. Gaussian





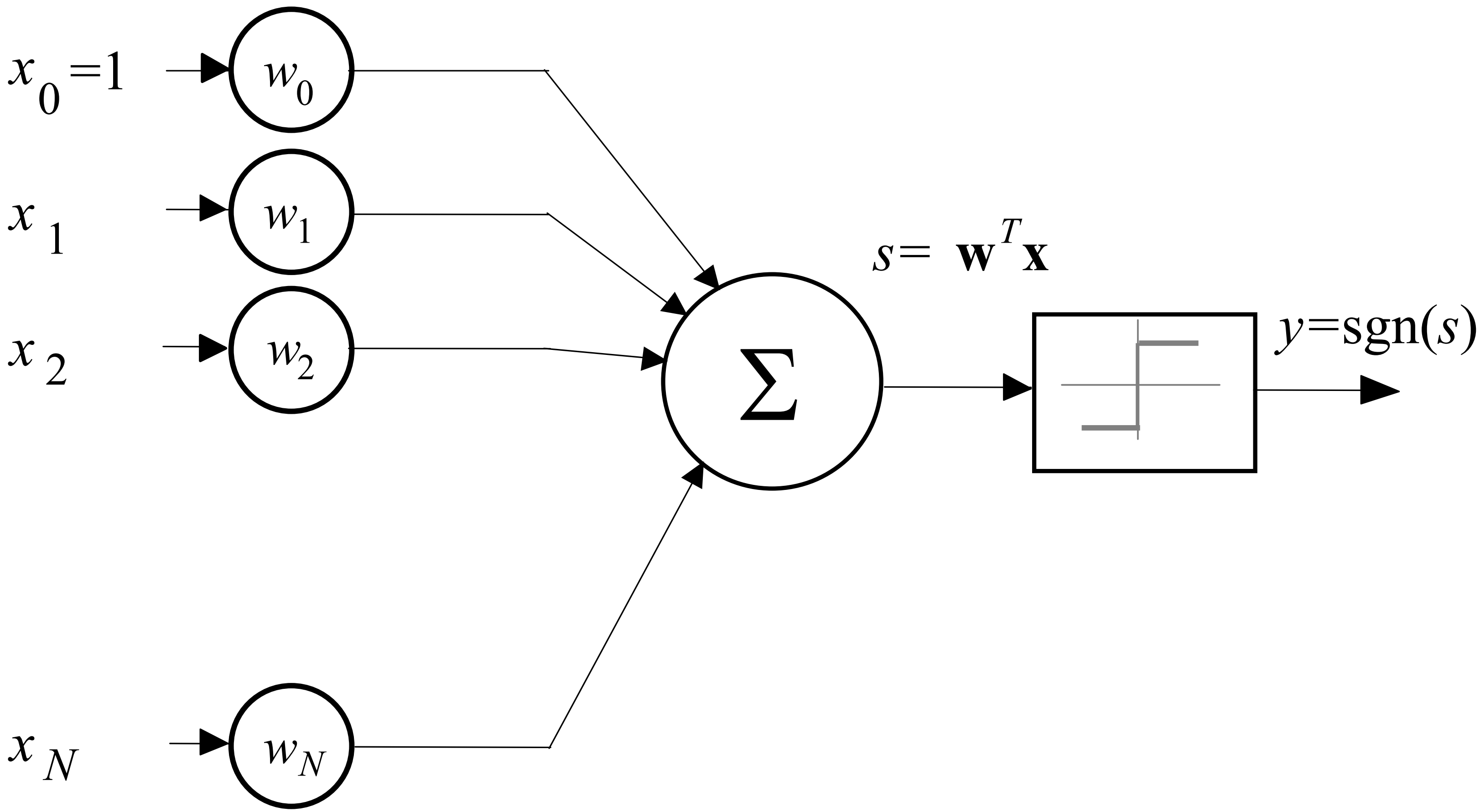
$$u = \sqrt{(x_1 - c_1)^2 + (x_2 - c_2)^2 + \ldots + (x_N - c_N)^2}$$

Classical neural networks

- static (no memory, feed-forward)
  - single layer networks
  - multi-layer networks
    - MLP
    - RBF
    - CMAC
- dynamic (memory or feedback)
  - feed-forward (storage elements)
  - feedback
    - local feedback
    - global feedback

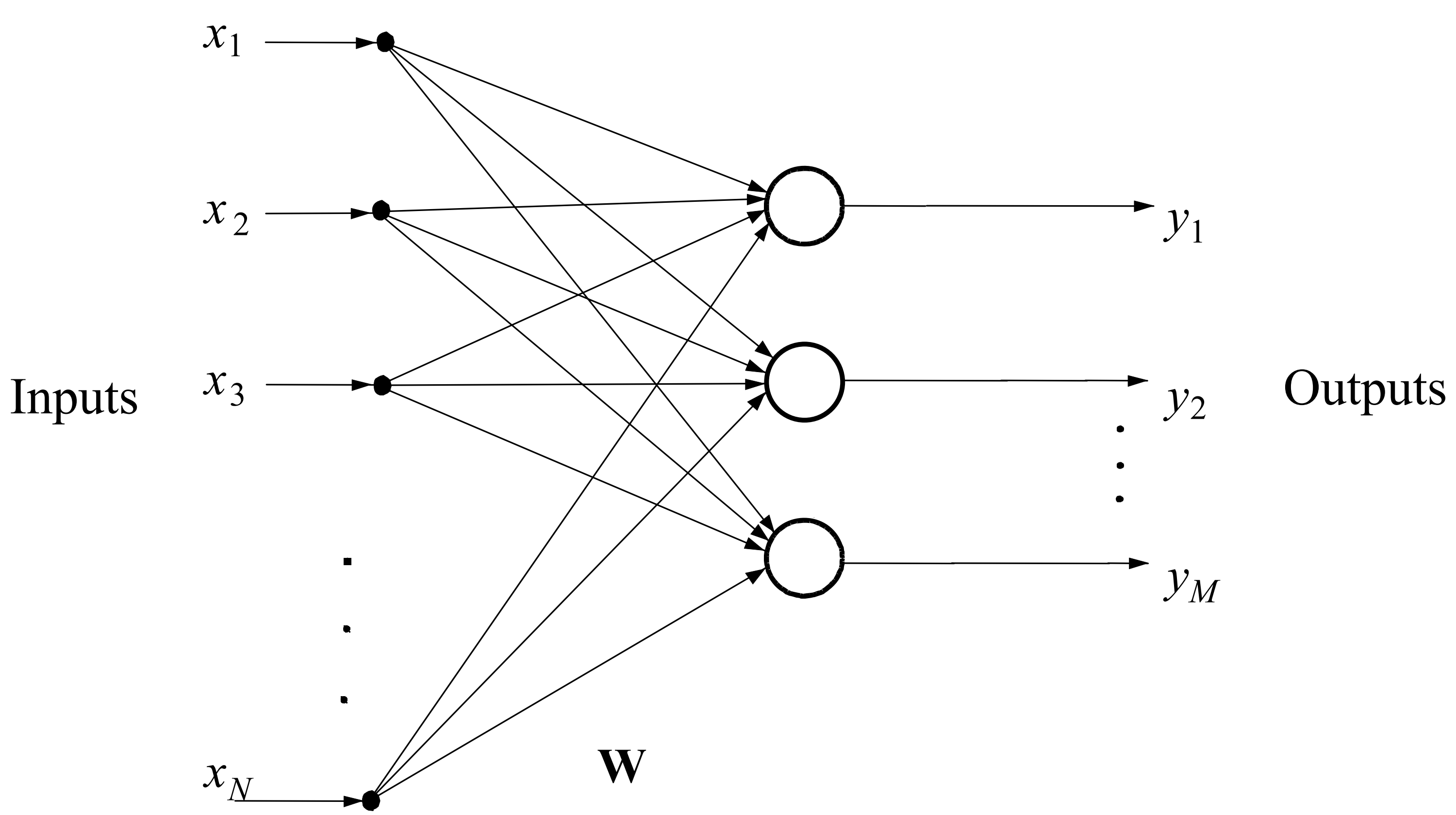
Feed-forward architecture

- Single layer network: Rosenblatt's perceptron



Feed-forward architecture

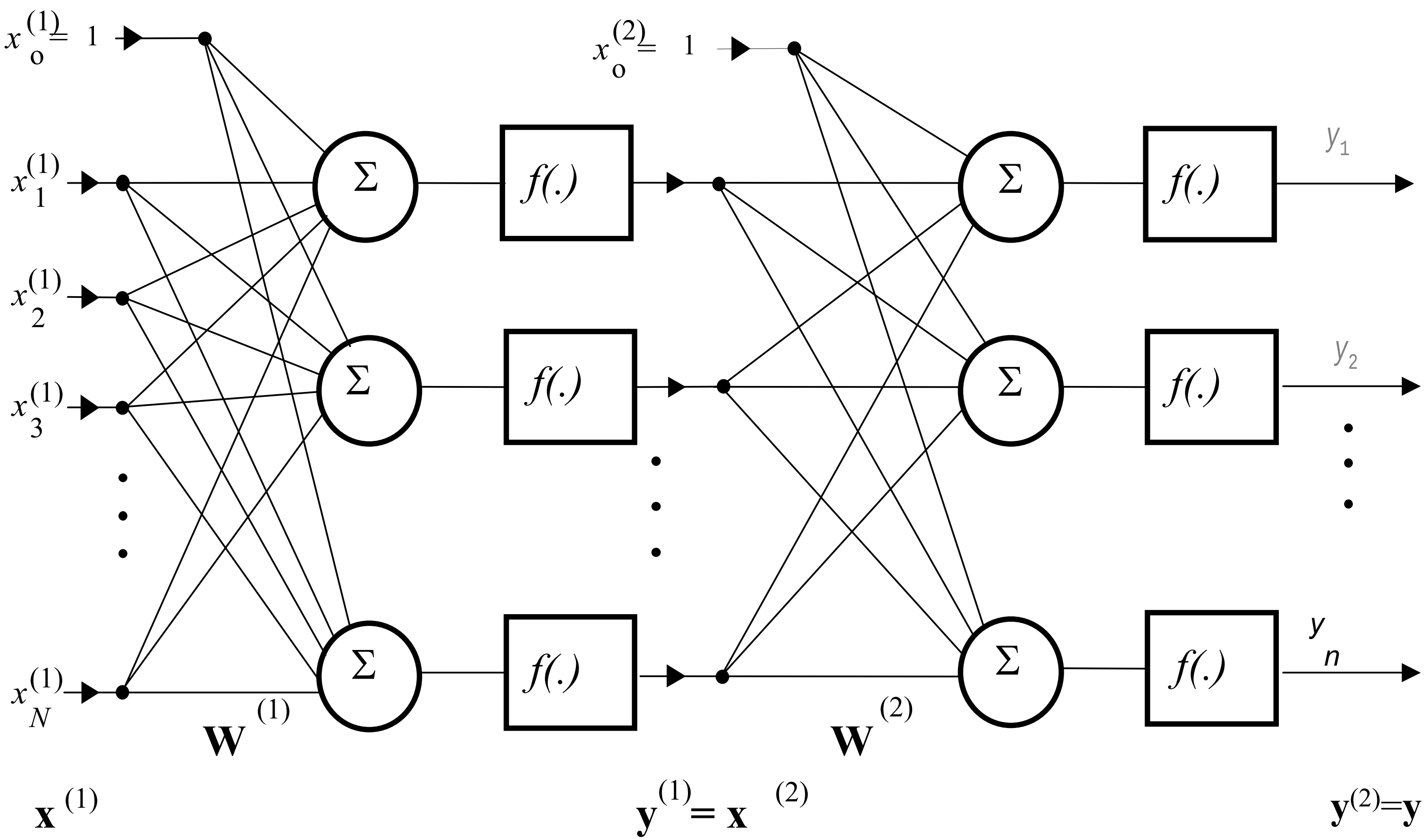
- Single layer network



Trainable parameters (weights)

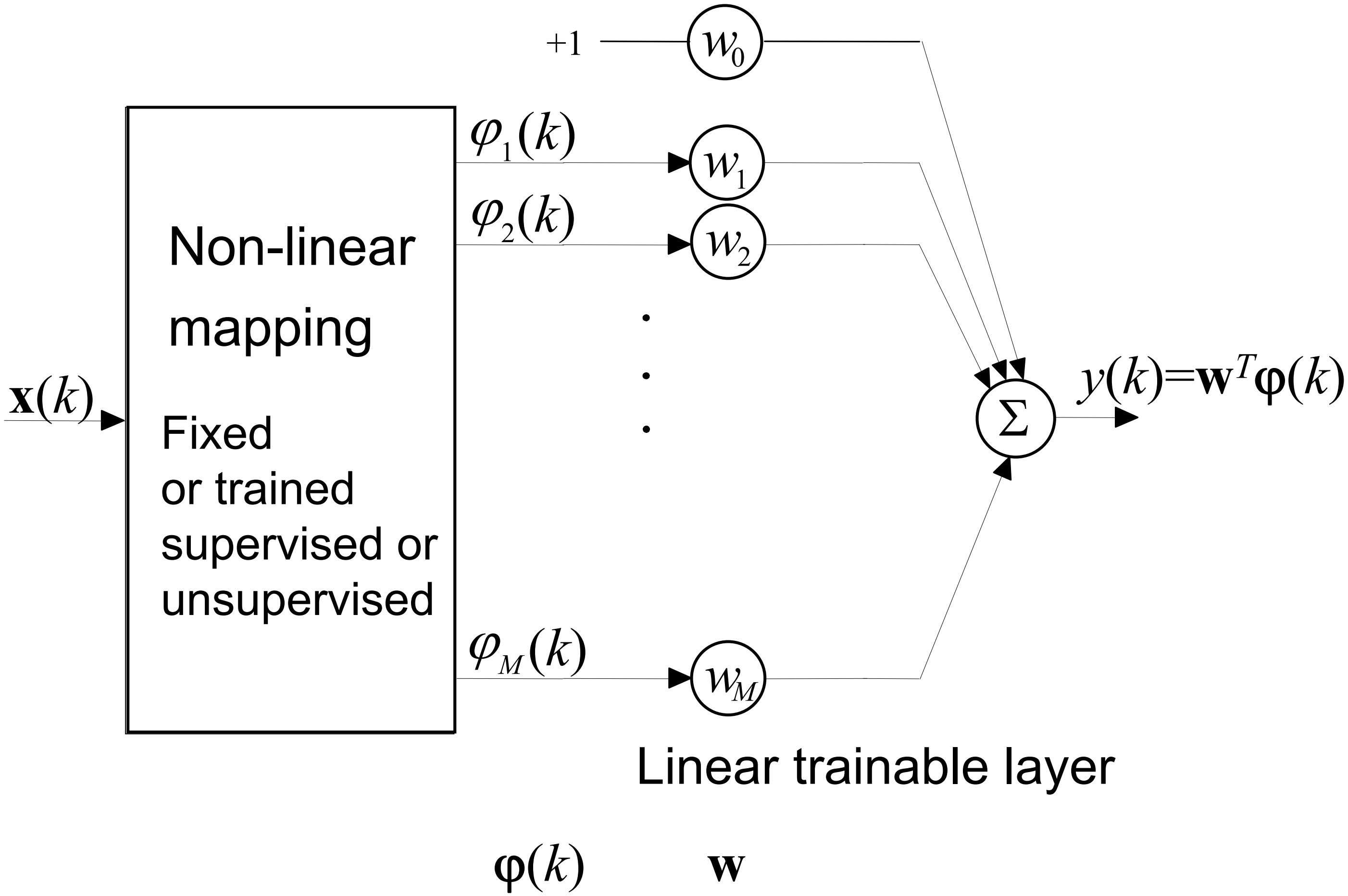
Feed-forward architecture

- Multi-layer network (static MLP network)



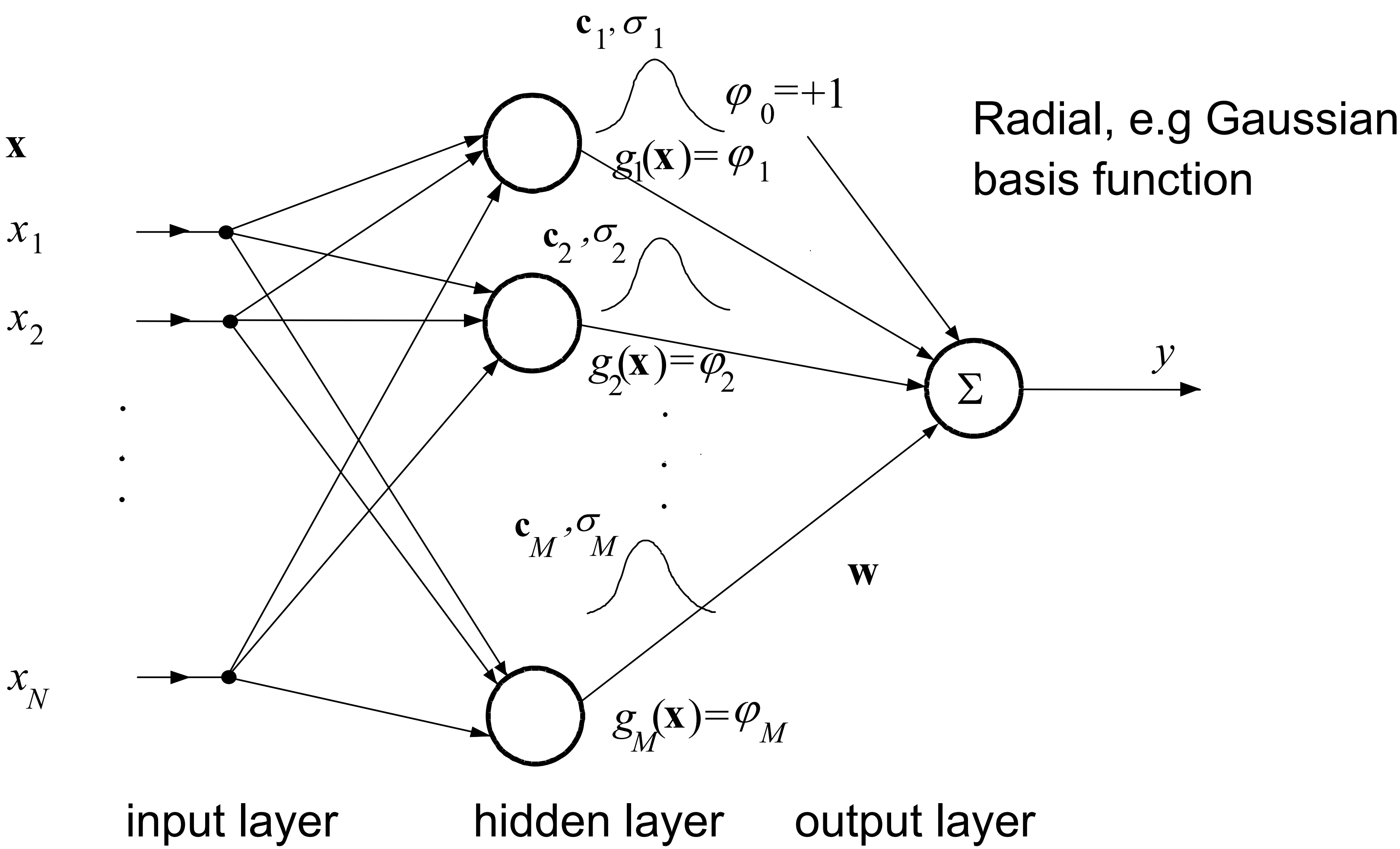
Feed-forward architecture

- Network with one trainable layer (basis function networks)



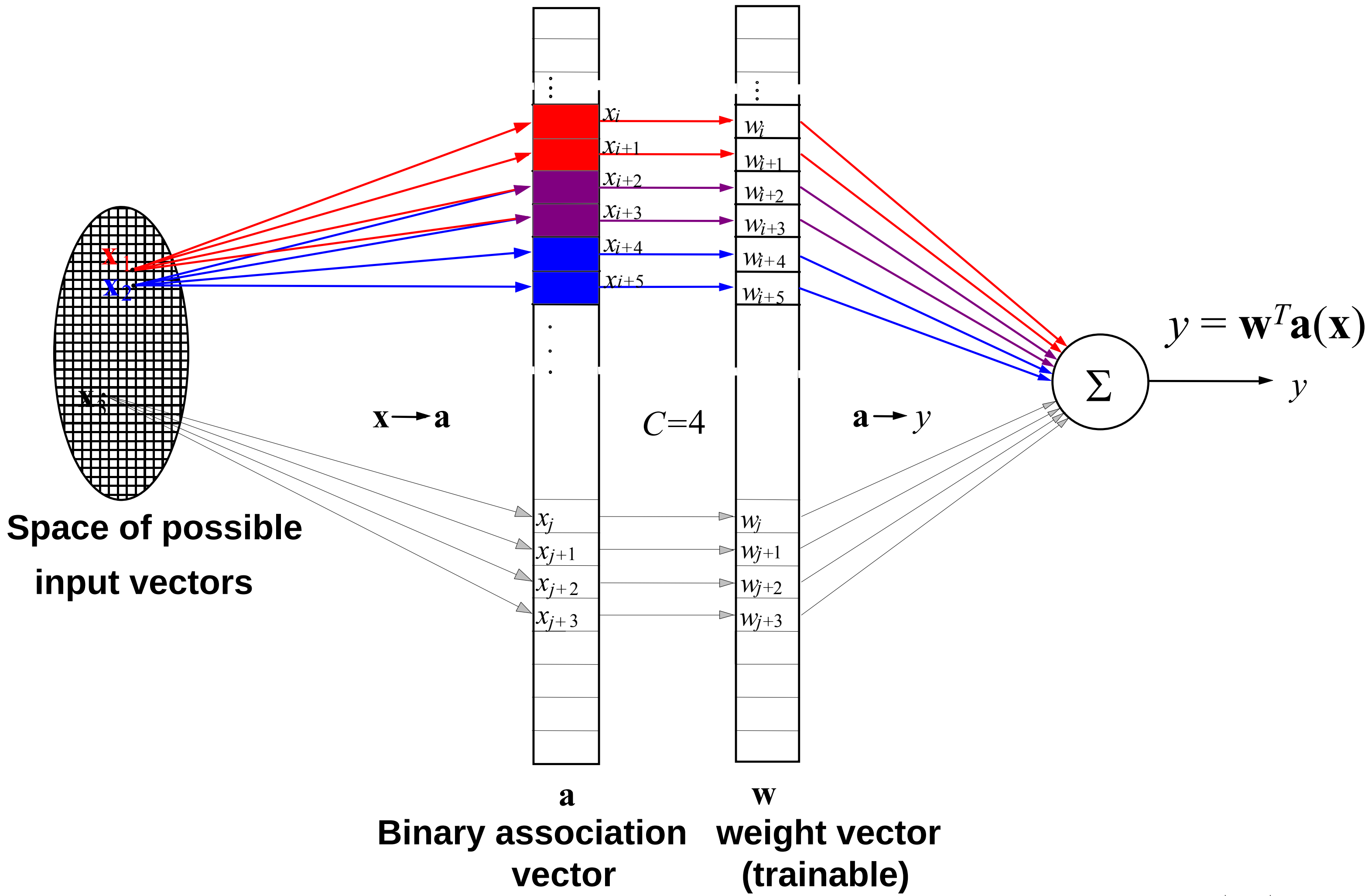
Radial basis function (RBF) network

- Network with one trainable layer



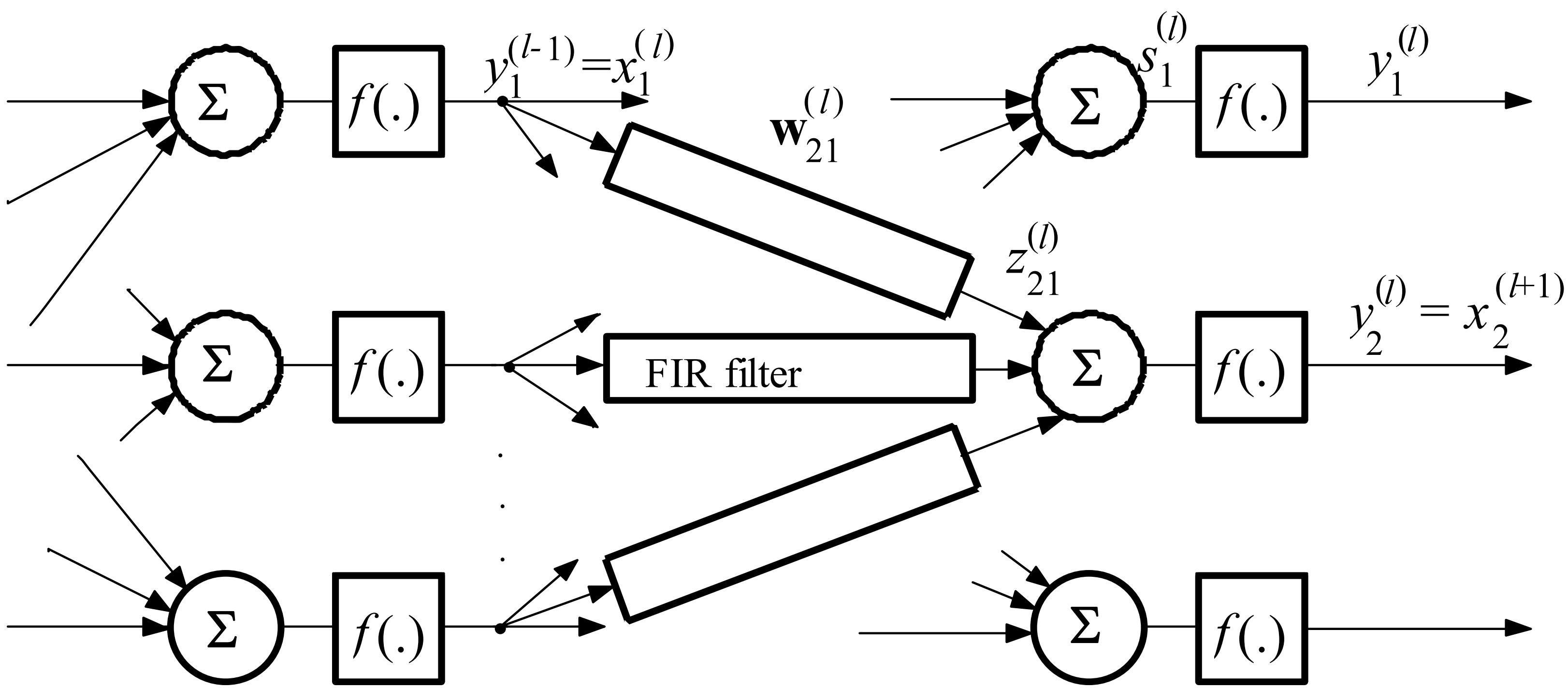
CMAC network

- Network with one trainable layer



Feed-forward architecture

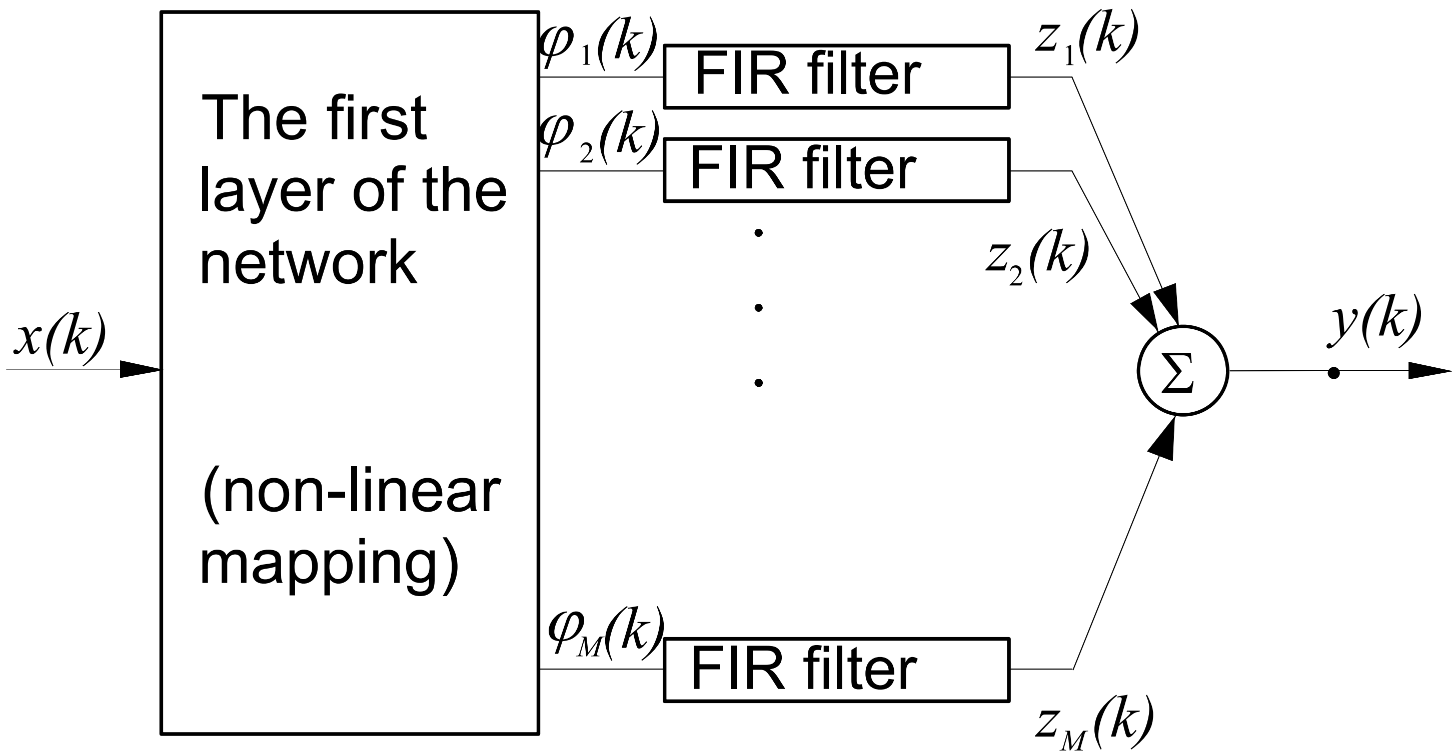
- Dynamic multi-layer network



l -th layer

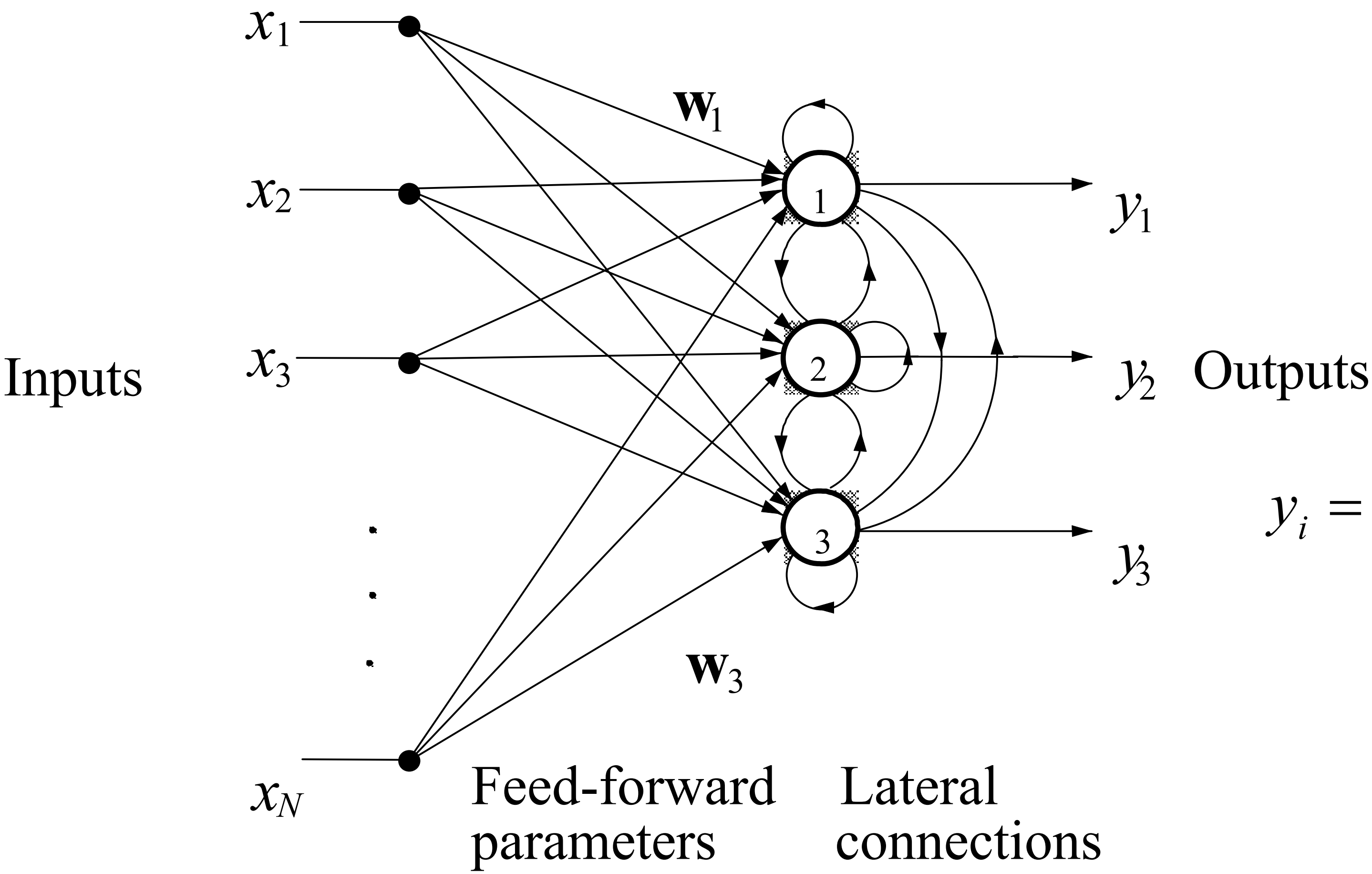
Feed-forward architecture

- Dynamic multi layer network (single trainable layer)



Feedback architecture

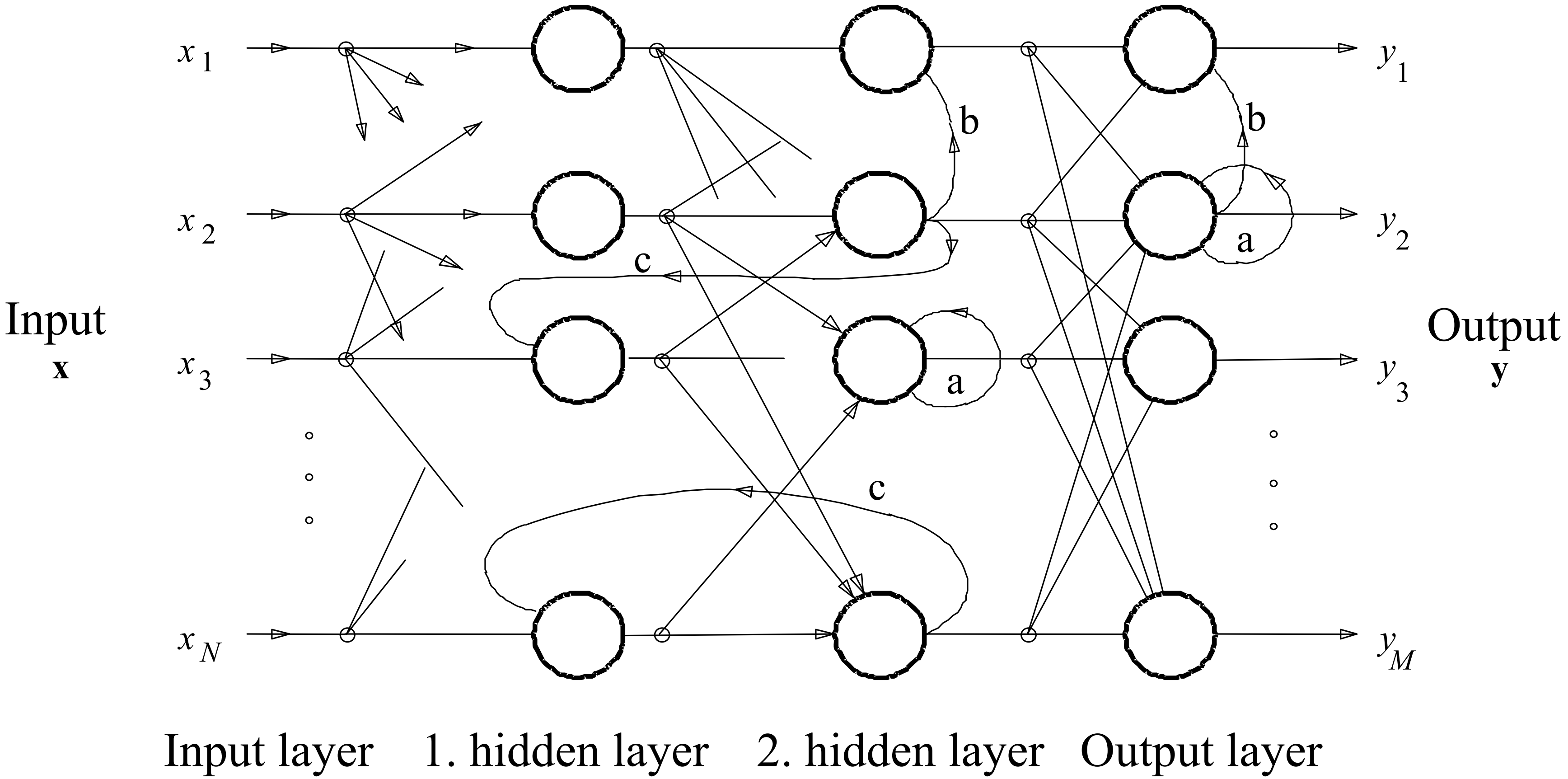
- Lateral feedback (single layer)



$$y_i = \sum_j w_{ij} x_j$$

Feedback architecture

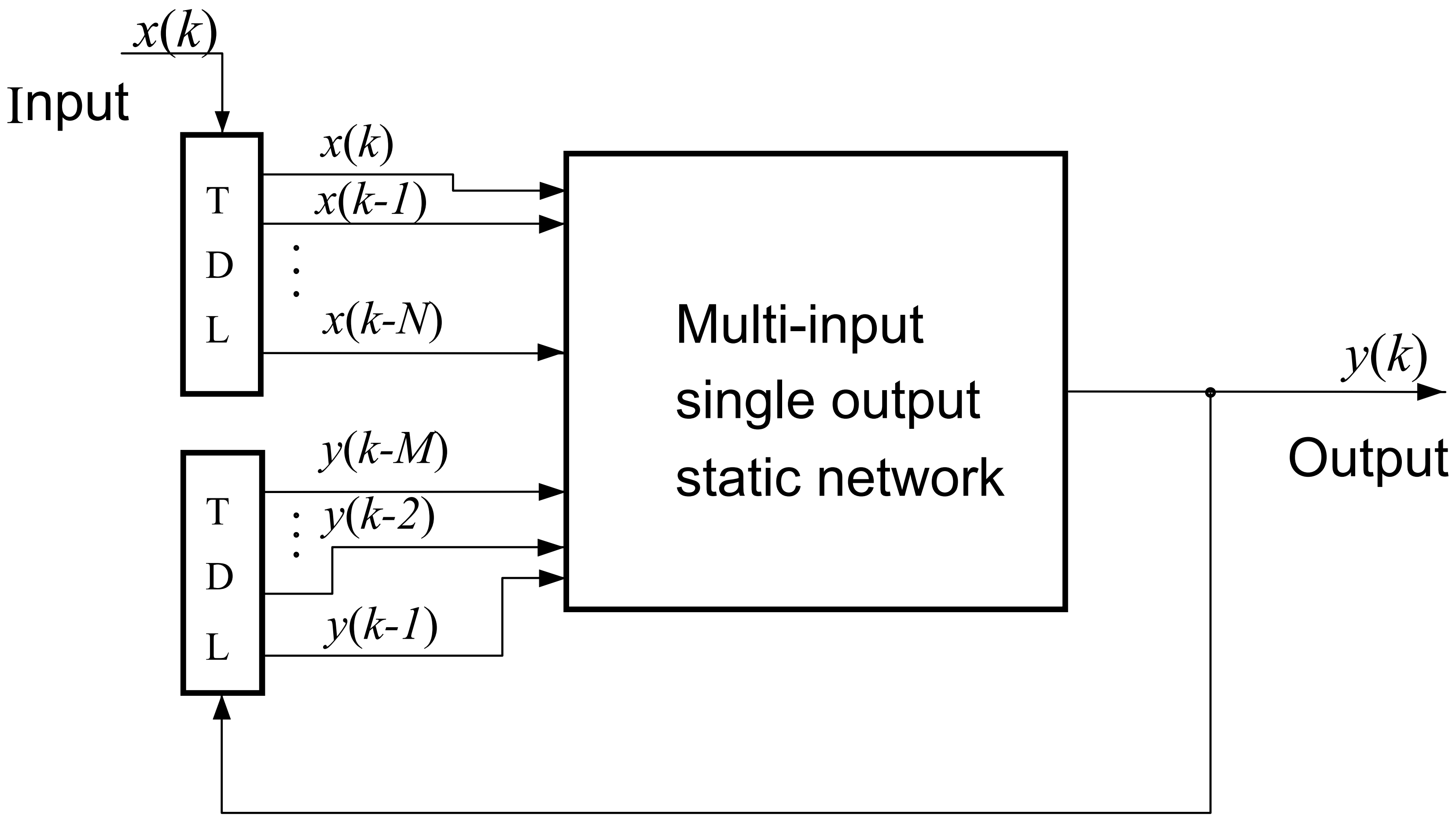
- Local feedback (MLP)



a.)self feedback, b.) lateral feedback, c.) feedback between layers

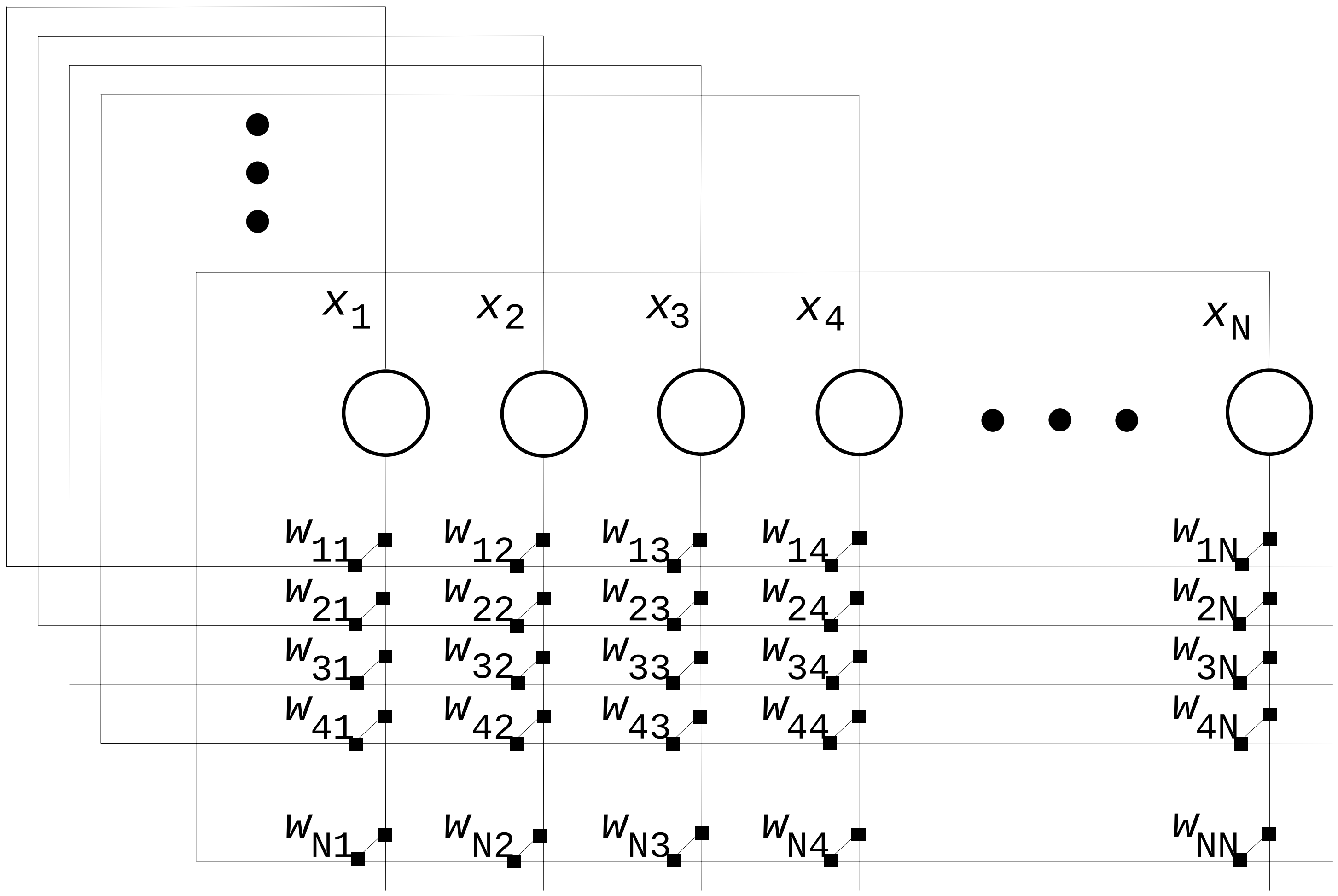
Feedback architecture

- Global feedback (sequential network)



Feedback architecture

- Hopfield network (global feedback)



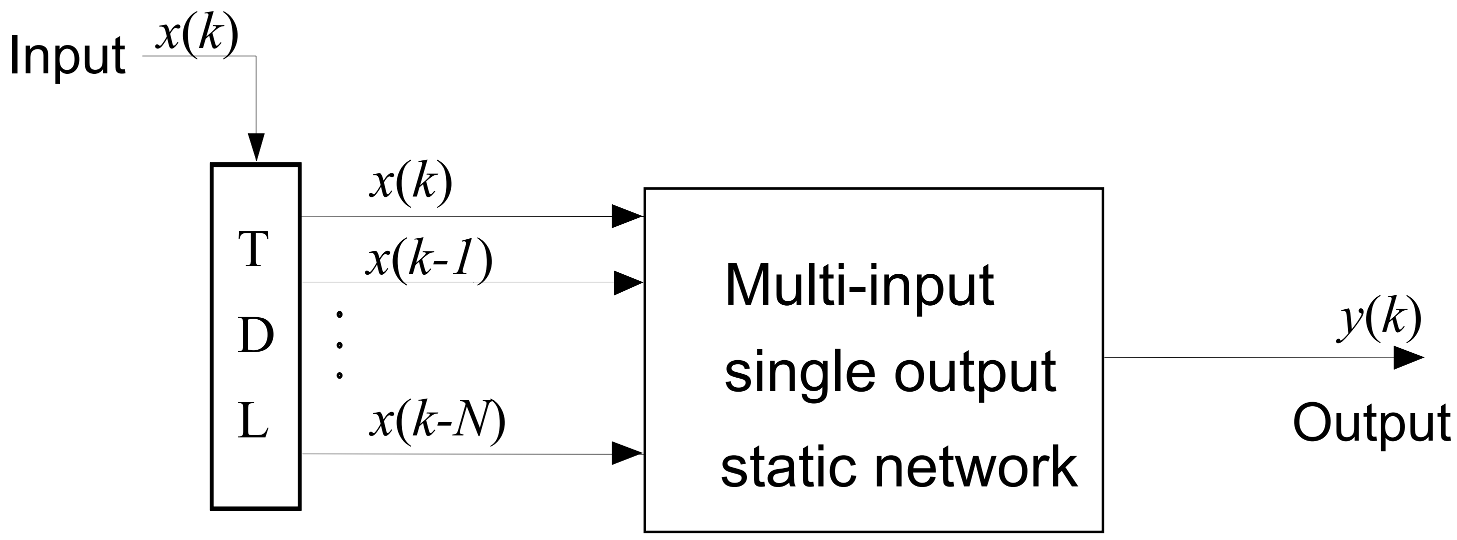
Basic neural network architectures

- Genaral approach
  - Regressors
    - current inputs (static networks)
    - current inputs and past outputs (dynamic networks)
    - past inputs and past outputs (dynamic networks)
  - Basis functions
    - non-linear-in-the-parameter network
    - linear-in-the-parameter networks

Basic neural network architectures

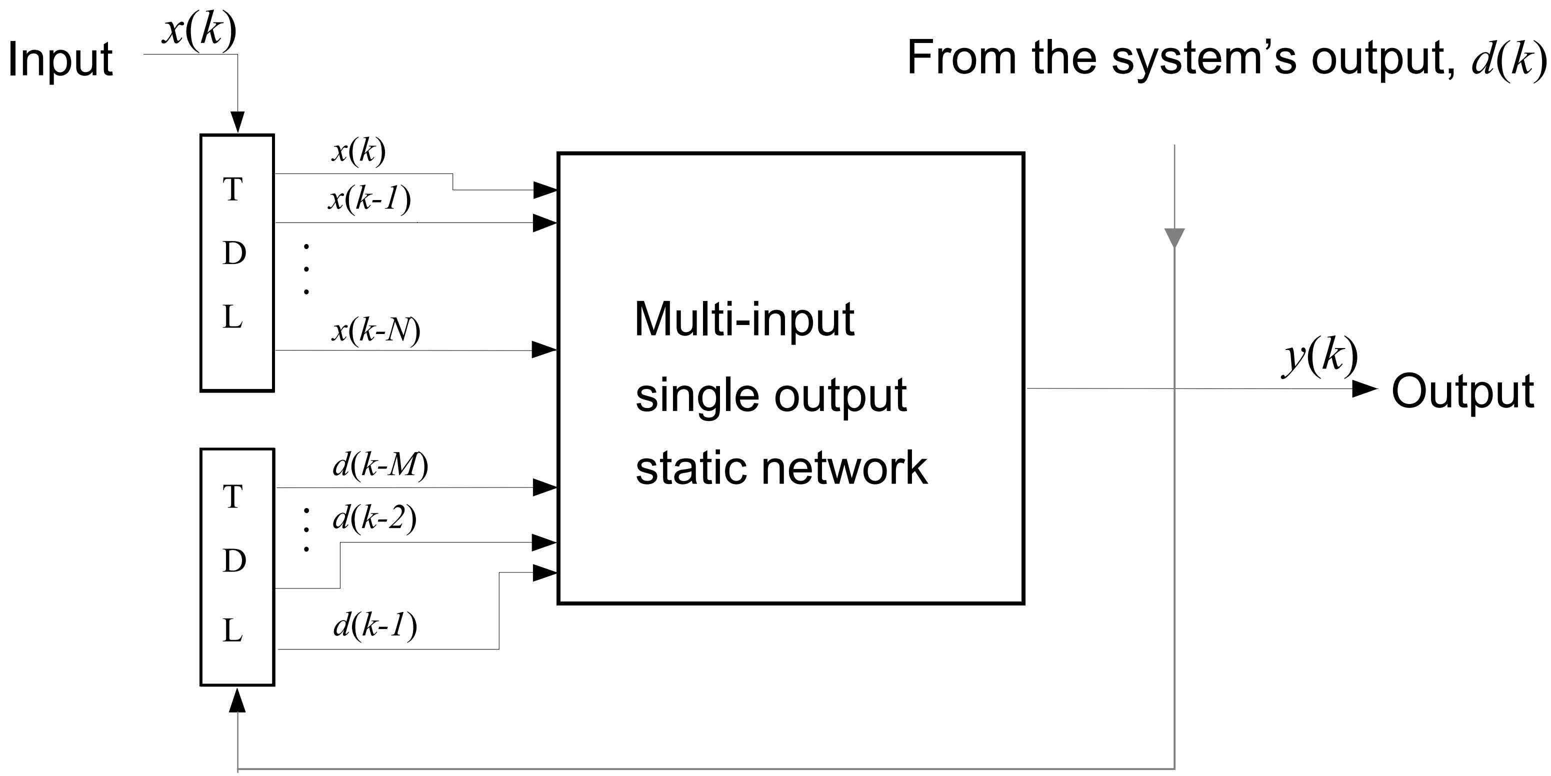
- Non-linear dynamic model structures based on regressor
  - NFIR

$$y(k) = f\left(x(k), x(k-1), \ \ldots, \ x(k-N) \right)$$



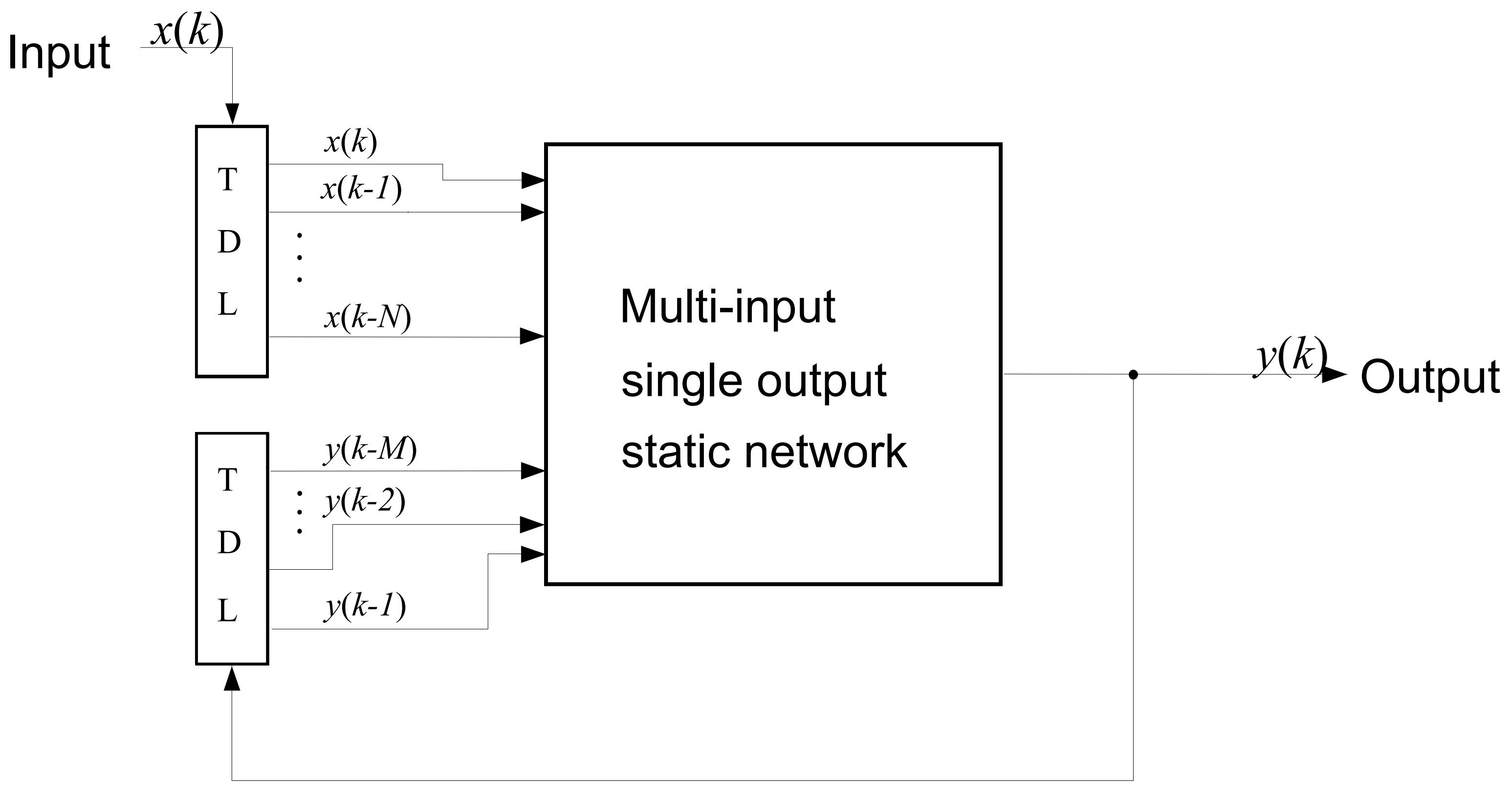
Basic neural network architectures

- Non-linear dynamic model structures based on regressor
  - NARX $$y(k) = f\left(x(k), \ldots, x(k-N), d(k-1), \ldots, d(k-M)\right)$$



Basic neural network architectures

- Non-linear dynamic model structures based on regressor
  - NOE $y(k)=f\left(x(k),\ \ldots,x(k-N),\ y(k-1),\ldots,y(k-M)\right)$



Basic neural network architectures

- Non-linear dynamic model structures based on regressor
  - NARMAX

$$y(k) = f\left(x(k), \; . \; . \; . \; , x(k-N), \; d(k-1), \; . \; . \; ., \; d(k-M), \varepsilon(k-1), \; . \; . \; ., \varepsilon(k-L)\right)$$

  - NJB

$$y(k) = f\left(x(k), \ldots, x(k-N), y(k-1), \ldots, y(k-M), \varepsilon(k-1), \ldots, \varepsilon(k-L), \varepsilon_x(k-1), \ldots, \varepsilon_x(k-K)\right)$$

  - NSS nonlinear state space representation

Basic neural network architectures

- Nonlinear function of regressor

$$y(k) = f(\mathbf{w}, \boldsymbol{\varphi}(k))$$

  - linear-in-the-parameter models (basis function models)

$$y(k) = \sum_{j=1}^{n} w_j f_j(\boldsymbol{\varphi}(k)) \qquad \mathbf{w} = [w_1 w_2 \ldots w_n]^T$$

  - nonlinear-in-the-parameter models

$$y(k) = \sum_{j=1}^{n} w_j^{(2)} f_j\left(\mathbf{w}^{(1)}, \boldsymbol{\varphi}(k)\right) \quad \mathbf{w} = \left[w_1^{(2)}\, w_2^{(2)} \ldots w_n^{(2)}, \mathbf{w}^{(1)}\right]^T$$

Basic neural network architectures

- Basis functions $f_j\left(\varphi(k)\right)$
  - MLP (with single nonlinear hidden layer)
    - sigmoidal basis function $sgm(s)=\dfrac{1}{1+e^{-Ks}}$

$$y(k)=\sum_{j=1}^{n} w_j^{(2)} f_j\left(\mathbf{w}^{(1)},\varphi(k)\right) \qquad f_j\left(\mathbf{w}^{(1)},\varphi(k)\right)=sgm(\varphi(k)^T\mathbf{w}_j^{(1)}+w_{j0}^{(1)})$$

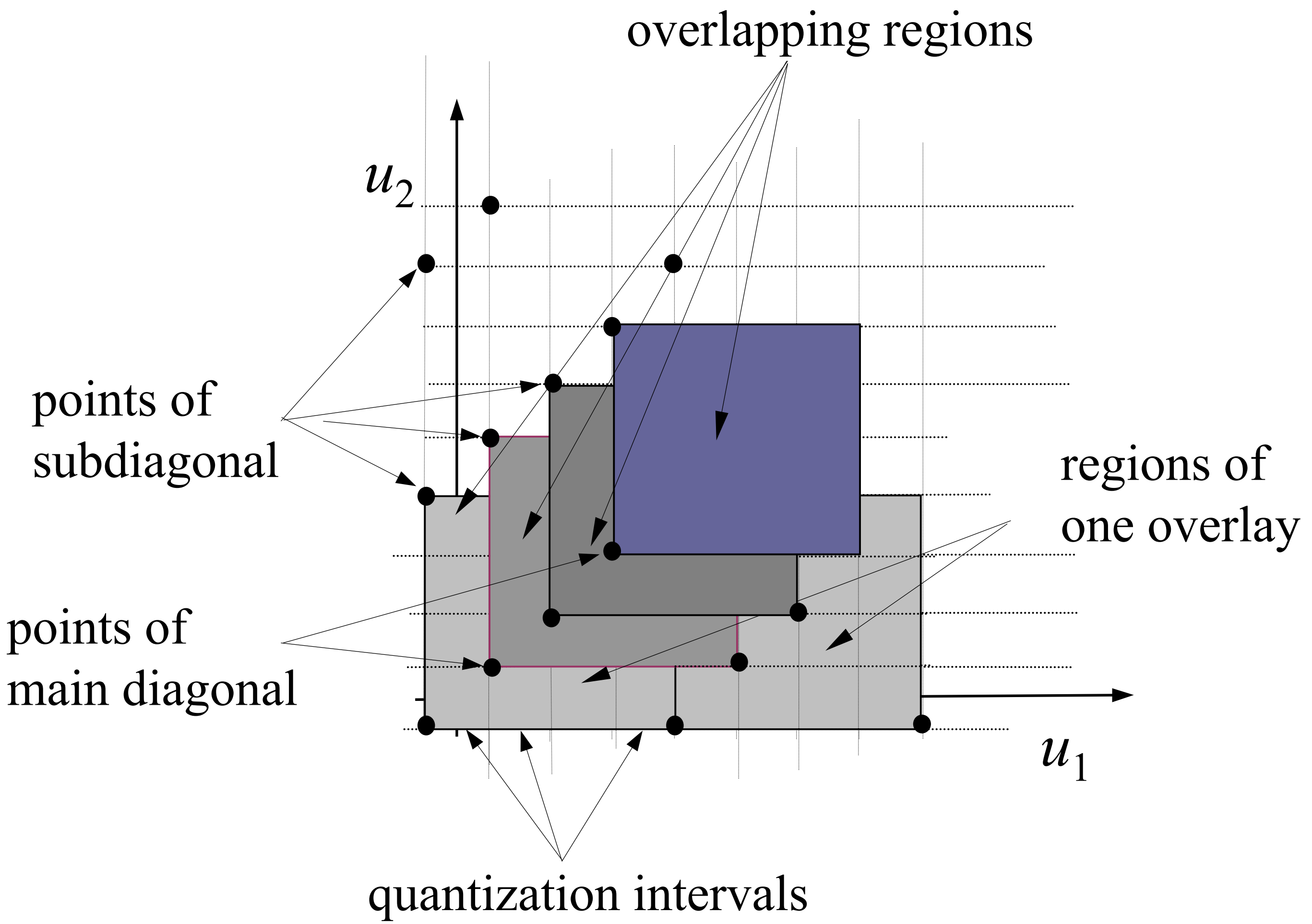
  - RBF (radial basis function, e.g. Gaussian)

$$y(k)=\sum_j w_j f_j(\varphi(k))=\sum_j w_j f\left(\left\|\varphi-\mathbf{c}_j\right\|\right) \qquad f\left(\varphi-\mathbf{c}_j\right)=\exp\left[-\left\|\varphi-\mathbf{c}_i\right\|^2/2\sigma_i^2\right]$$

  - CMAC (rectangular basis functions, splines)

Basic neural network architectures

- CMAC (rectangular basis functions)

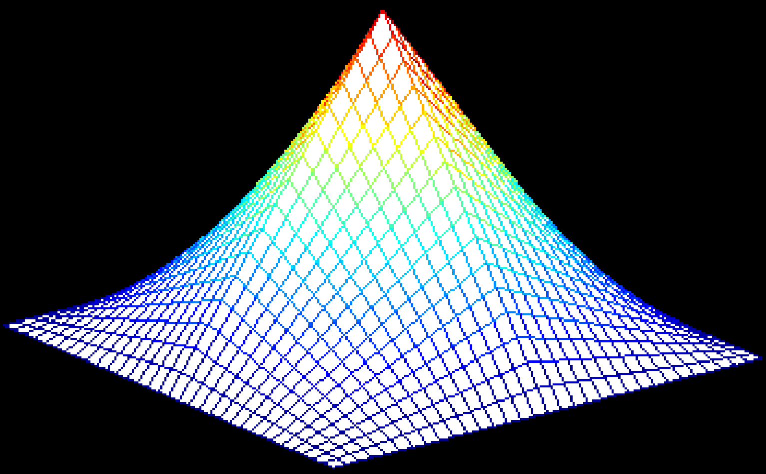


Basic neural network architectures

- General basis functions of compact support (higher-order CMAC)
- B-splines

  advantages

A two-dimensional basis function
with compact support:
tensor product of a second-order B-spline



Capability of networks

Capability of networks

- Function approximation
- Classification
- Association
- Clustering
- Data compression
- Significant component selection
- Optimization

Supervised learning network

Unsupervised learning network

Capability of networks

- Approximation of functions
  - Main statements: some FF neural nets (MLP, RBF) are universal approximators (in some sense)
  - Kolmogorov's Theorem (representation theory): any continuous real-valued N-variable function defined on $[0,1]^N$ can be represented using properly chosen functions of one variable (non constructive).

$$f(x_1, x_2, \ldots, x_N) = \sum_{q=0}^{2N} \phi_q \left(\sum_{p=1}^{N} \psi_{pq}(x_p) \right)$$

Capability of networks

- Approximation of function (MLP)
  - Arbitrary continuous function $f: R^N \rightarrow R$ on a compact subset **K** of R^N can be approximated to any desired degree of accuracy (maximal error) if and only if the activation function, $g(x)$ is non-constant, bounded, monoton increasing.
    (Hornik, Cybenko, **Funahashi**, Leshno, Kurkova, etc.)

$$\hat{f}(x_1,\ldots,x_N) = \sum_{i=1}^{M} c_i g\Big(\sum_{j=0}^{N} w_{ij} x_j\Big) \qquad ; \quad x_0 = 1$$

$$\max_{\mathbf{x}\in\mathbf{K}} \left| f(x_1,\ldots,x_N) - \hat{f}(x_1,\ldots,x_N) \right| < \varepsilon \qquad \varepsilon > 0$$

Capability of networks

- Approximation of function (MLP)
  - Arbitrary continuous function $f: R^N \rightarrow R$ on a compact subset of R^N can be approximated to any desired degree of accuracy (in the L_2 sense) if and only if the activation function is non-polynomial (Hornik, Cybenko, Funahashi, **Leshno**, Kurkova, etc.)

$$\hat{f}(x_1,\ldots,x_N) = \sum_{i=1}^{M} c_i g\left(\sum_{j=0}^{N} w_{ij} x_j\right), \qquad x_0 = 1$$

Capability of networks

- Classification
  - Perceptron: linear separation
  - MLP: universal classifier

$$f(\mathbf{x}) = j, \text{ iff } \quad x \in X^{(j)} \qquad f : K \rightarrow \{1,2,\ldots,k\}$$

$$K \text{ compact subset of } R^N$$

$$X^{(j)} \quad \mathrm{j} = 1,\ldots,k \quad \text{disjoint subsets of } K$$

$$K = \bigcup_{j=1}^{k} X^{(j)} \text{ and } X^{(j)} \bigcap X^{(j)} \text{ is empty if } i \neq j$$

Capability of networks

- Universal approximator (RBF)

  An arbitrary continuous function $f: R^N \to R$ on a compact subset K of R^N can be approximated to any desired degree of accuracy in the following form

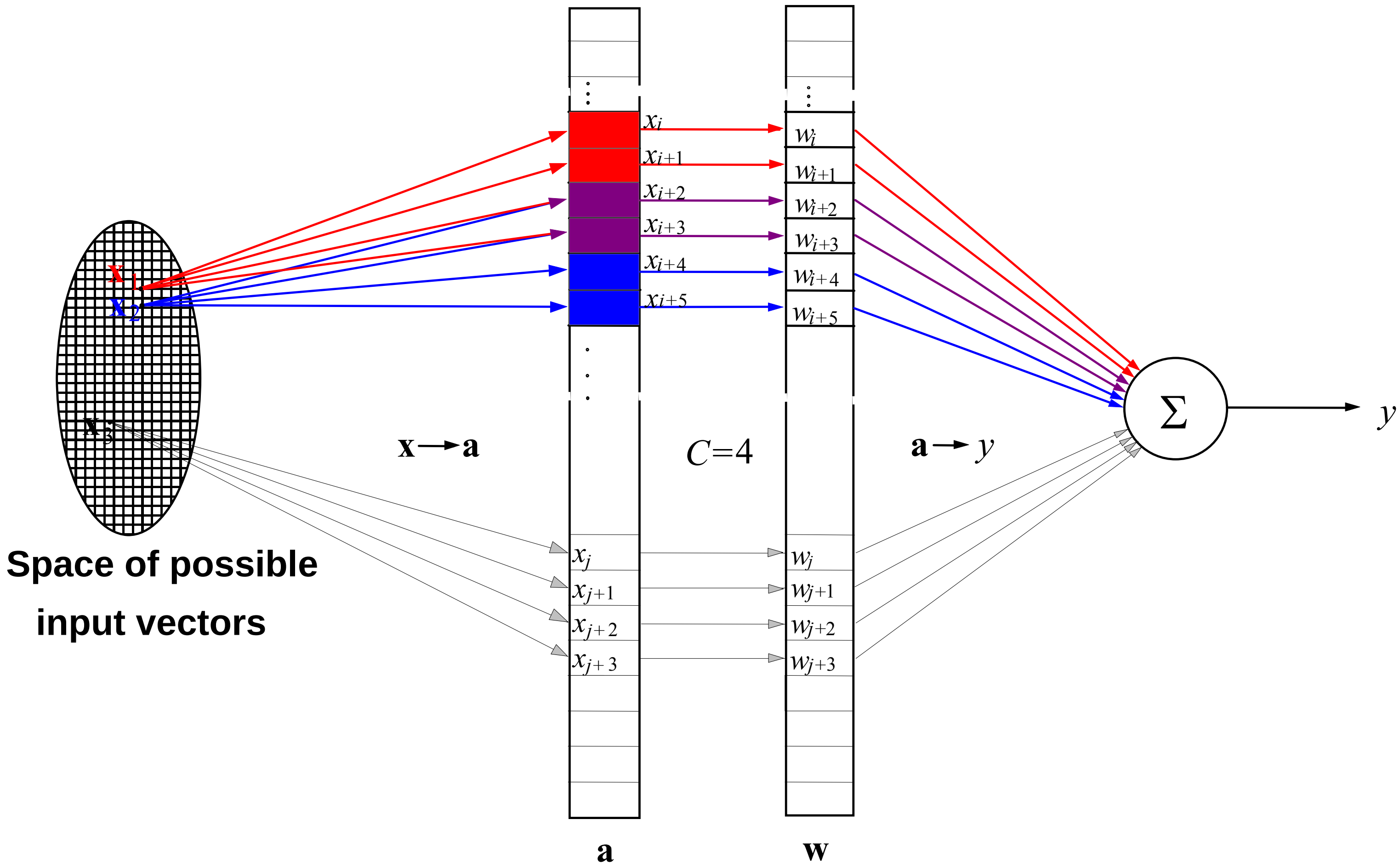
$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{M} w_i g\left(\frac{\mathbf{x} - \mathbf{c}_i}{\sigma_i}\right)$$

  if $g: R^N \to R$ is non-zero, continuous, integrable function.

Computational capability of the CMAC

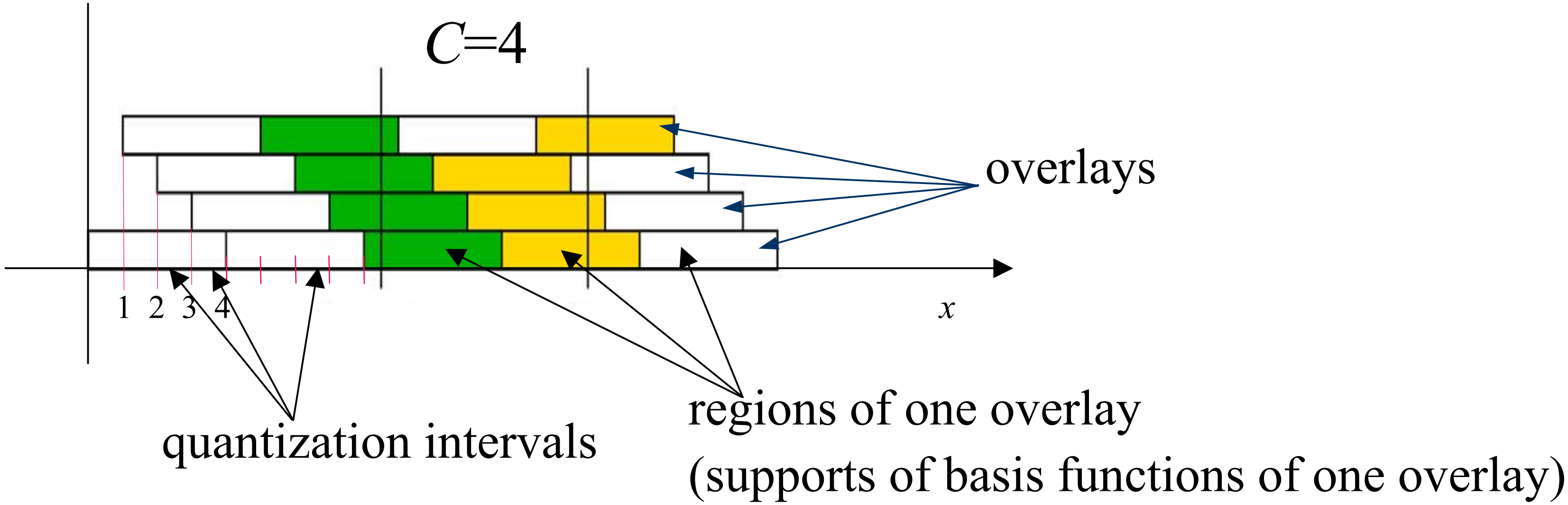
- The approximation capability of the Albus binary CMAC
- Single-dimensional (univariate) case
- Multi-dimensional (multivariate) case

Computational capability of the CMAC



Computational capability of the CMAC

- Arrangement of basis functions: uni-variate case



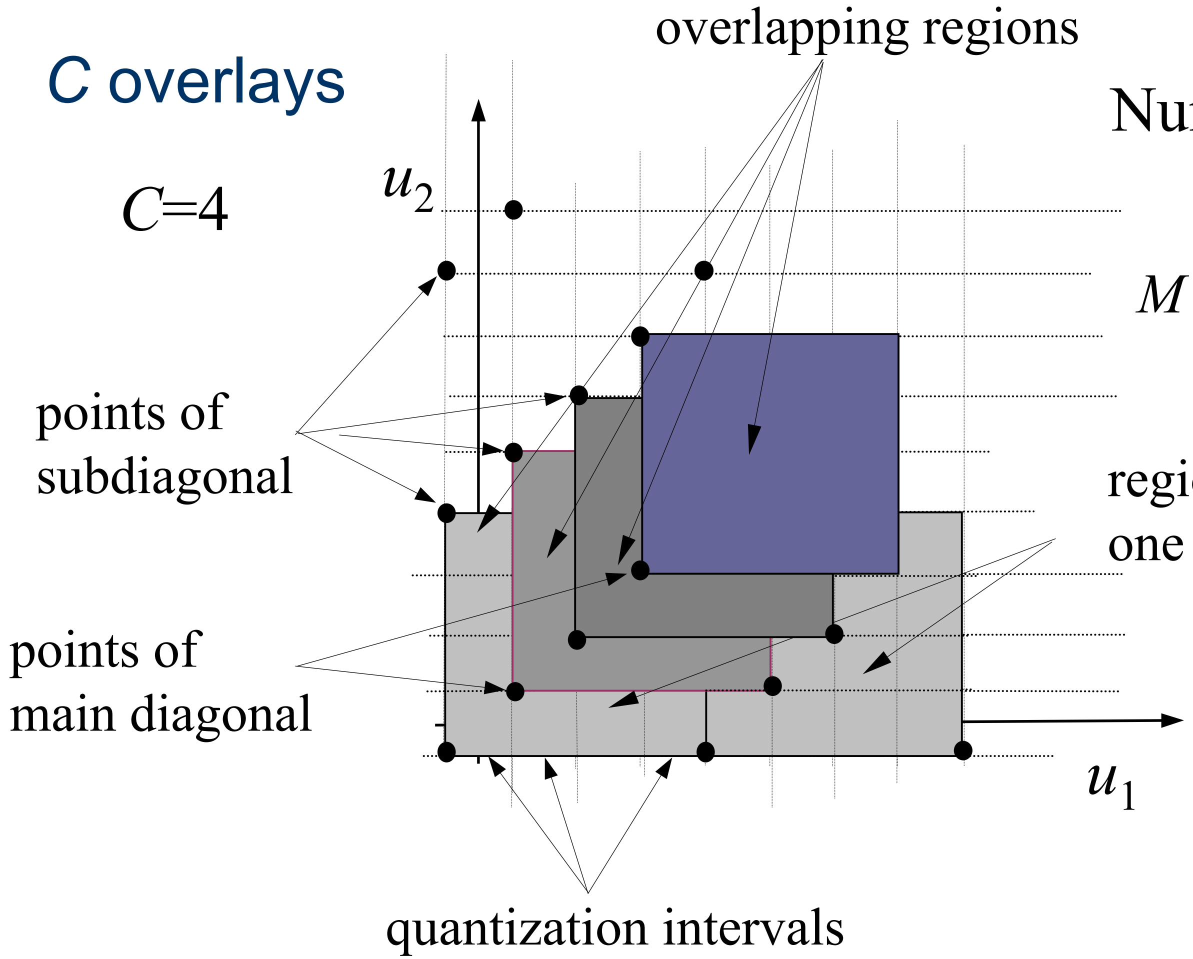
Number of basis functions: $M = R + C - 1$

Computational capability of the CMAC

- Arrangement of basis functions: multi-variate case

C overlays

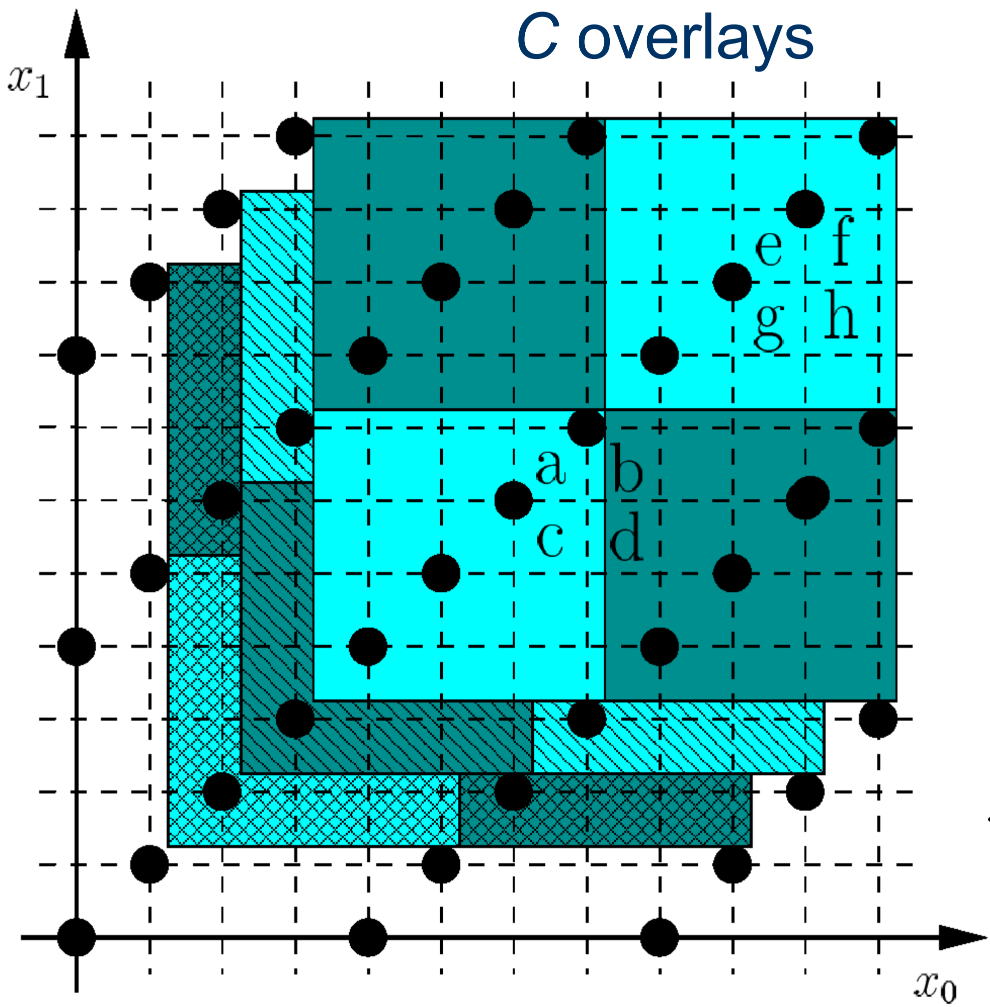
C=4



Number of basis functions

$$M = \left\lceil \frac{1}{C^{N-1}} \prod_{i=0}^{N-1} (R_i + C - 1) \right\rceil$$

CMAC approximation capability



Consistency equations:

$$f(\mathbf{a}) - f(\mathbf{b}) = f(\mathbf{c}) - f(\mathbf{d})$$

can model only additive functions

$$f(\mathbf{x}) = f(x_1, x_2, \ldots, x_N) = \sum_{i=1}^{N} f_i(x_i)$$

CMAC modeling capability

- One-dimensional case: can learn any training data set exactly
- Multi-dimensional case: can learn any training data set from the additive function set (consistency equations)

CMAC generalization capability

- Important parameters:

  C generalization parameter

  d_{train} distance between adjacent training data

- Interesting behavior

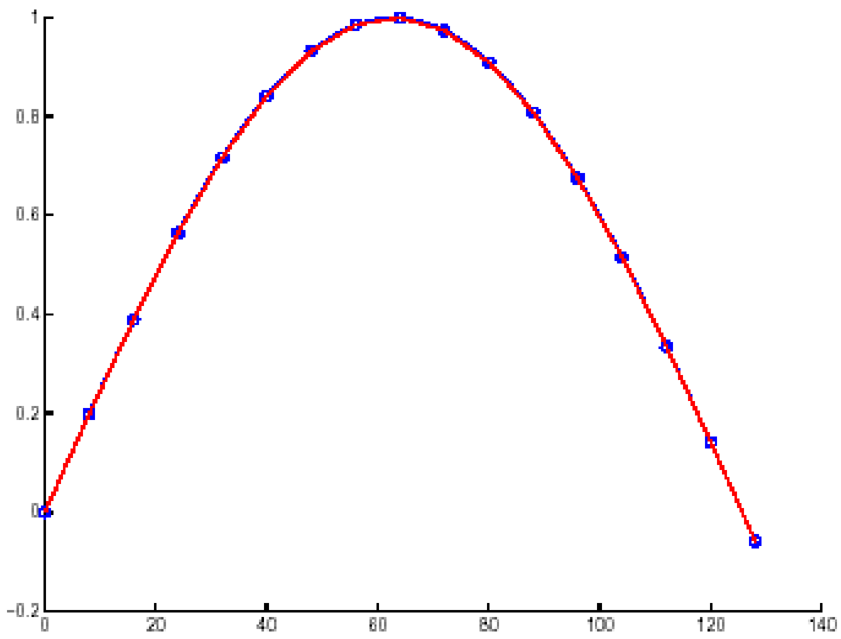
  $C = l * d_{train}$: linear interpolation between the training points

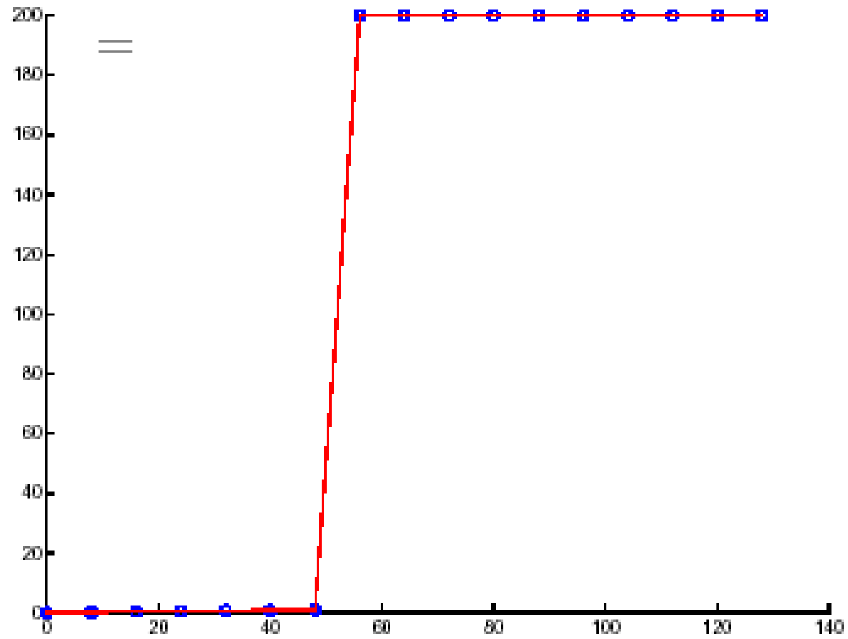
  $C \neq l * d_{train}$: significant generalization error

  non-smooth output

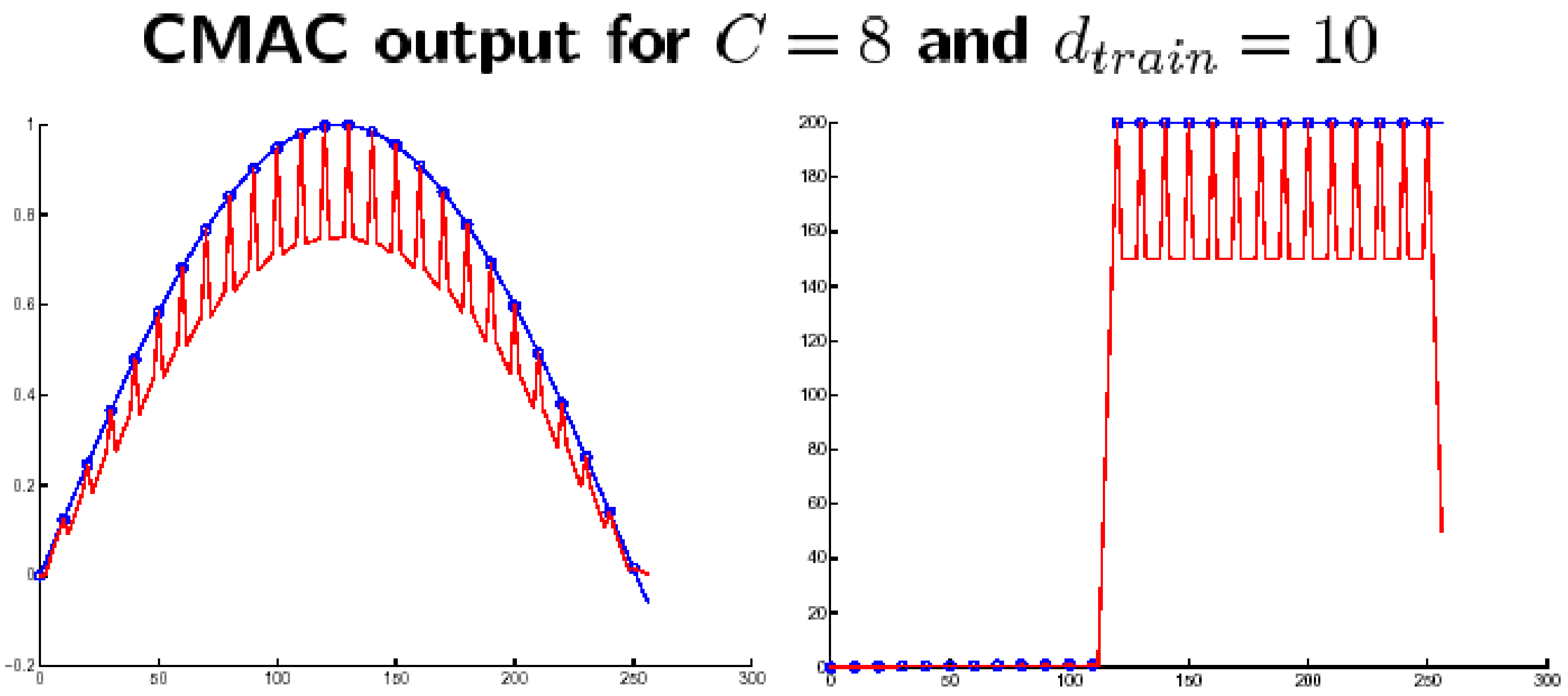
CMAC generalization error

CMAC output for $C = 8$ **and** $d_{train} = 8$ **param**



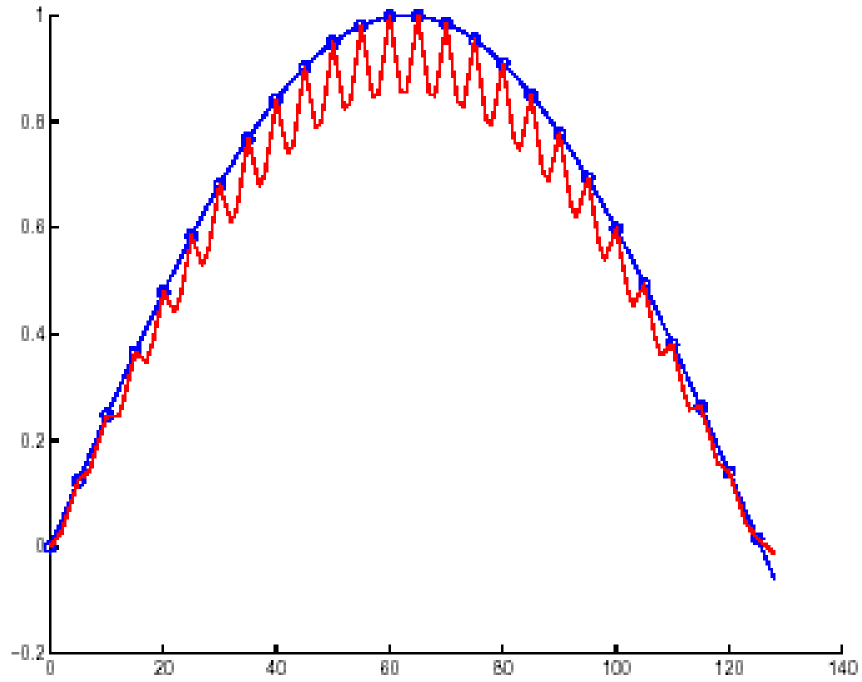


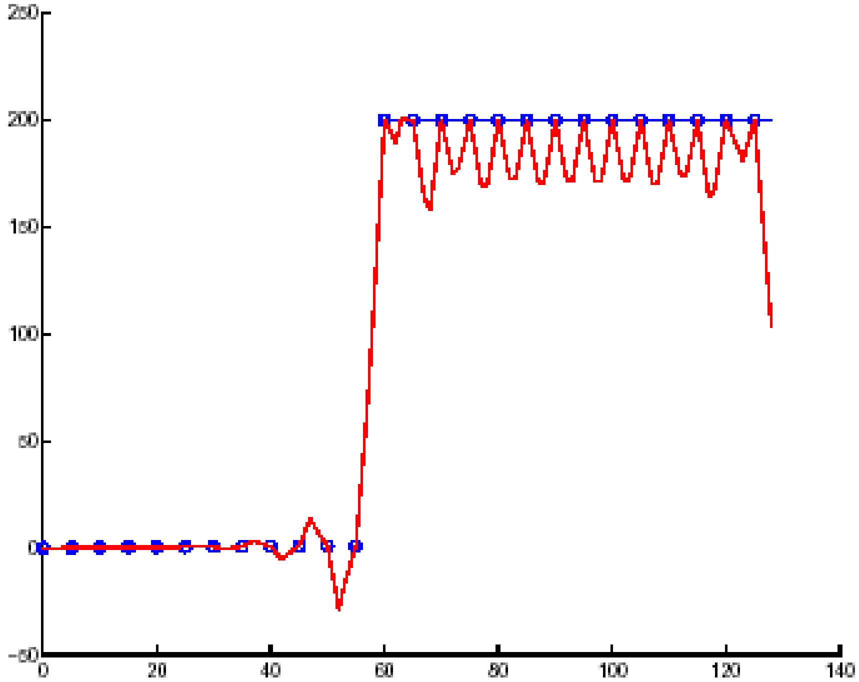
CMAC generalization error



CMAC generalization error

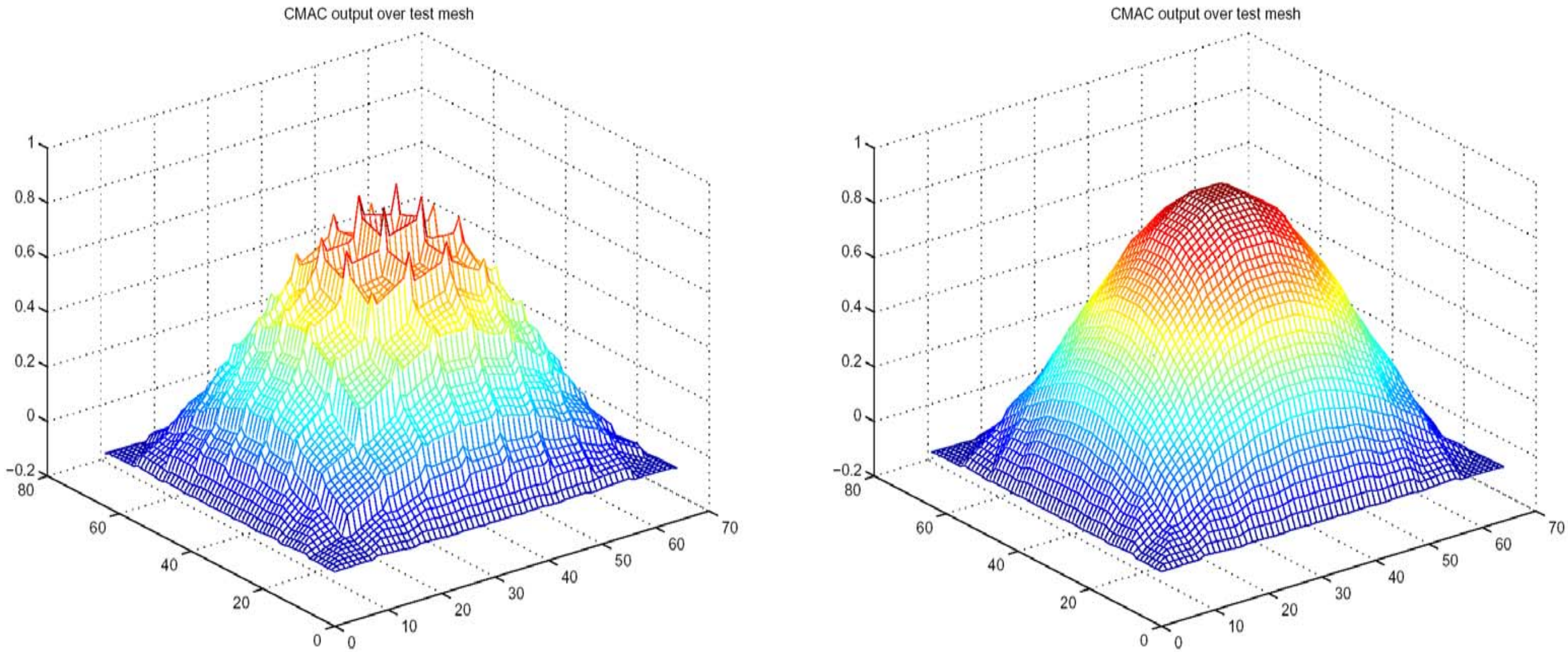
CMAC output for $C = 8$ and $d_{train} = 5$





CMAC generalization error

Multidimensional case

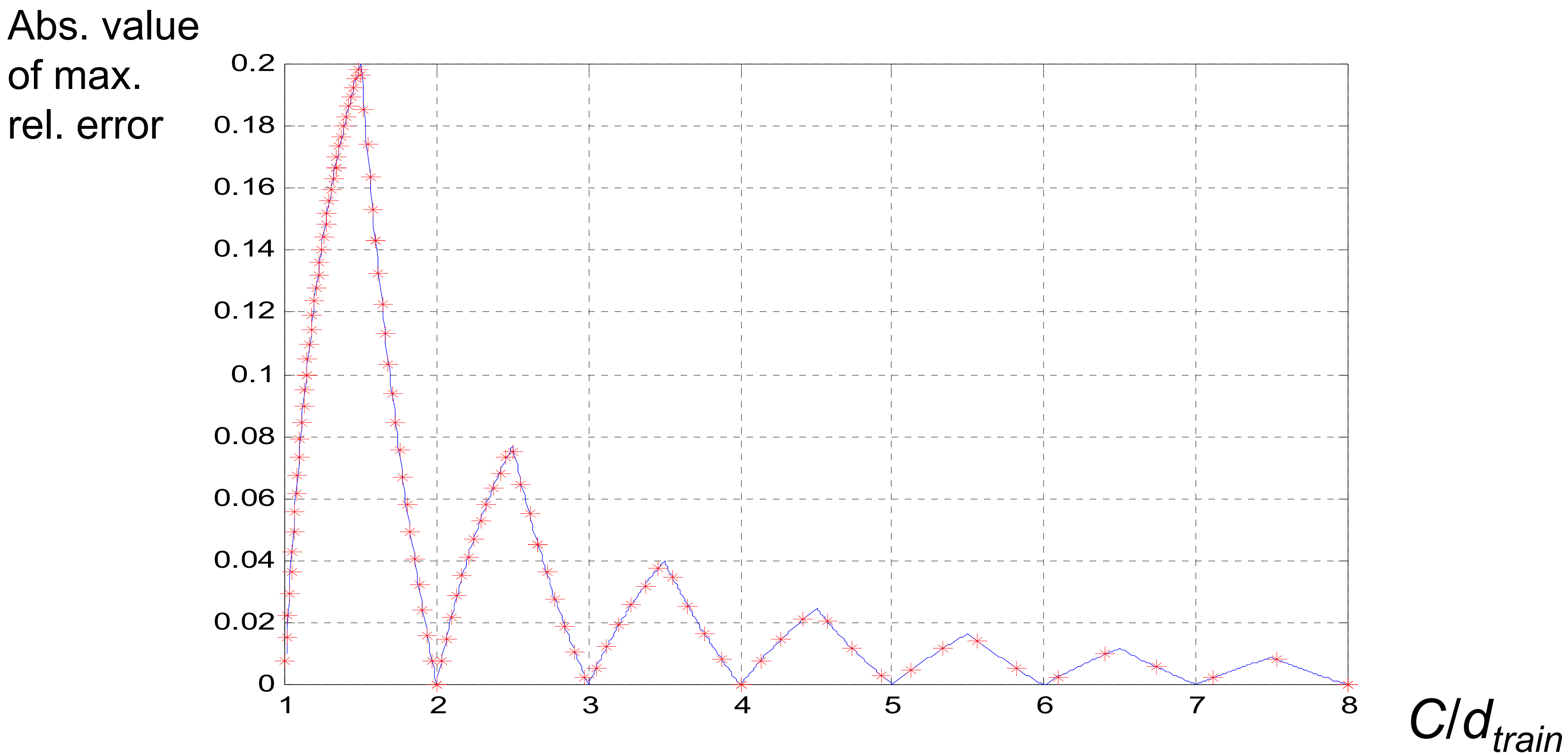


without   with

regularization

CMAC generalization error univariate case (max)

$$h = \frac{min\left\{C mod\ d_{train},\ \left(d_{train} - C mod\ d_{train}\right)\right\}}{\left(2\left\lfloor \frac{C}{d_{train}} \right\rfloor + 1\right) C - \left\lfloor \frac{C}{d_{train}} \right\rfloor \left(\left\lfloor \frac{C}{d_{train}} \right\rfloor + 1\right) d_{train}}$$



Application of networks (based on the capability)

- Regression: function approximation
  - modeling of static and dynamic systems, signal modeling, system identification
  - filtering, control, etc.
- Pattern association
  - association
    - autoassociation (similar input and output)
      (dimension reduction, data compression)
    - Heteroassociation (different input and output)
- Pattern recognition, clustering
  - classification

Application of networks (based on the capability)

- Optimization
  - optimization
- Data compression, dimension reduction
  - principal component analysis (PCA), linear networks
  - nonlinear PCA, non-linear networks
  - signal separation, BSS, independent component analysis (ICA).

Data compression, PCA networks

- Karhunen-Loève tranformation

$$\mathbf{y}=\mathbf{\Phi x} \quad \mathbf{\Phi}=[\varphi_1,\varphi_2,\ldots,\varphi_N]^T \quad \varphi_i^T\varphi_j=\delta_{ij}, \text{ further } \mathbf{\Phi}^T\mathbf{\Phi}=\mathbf{I}, \quad \rightarrow \mathbf{\Phi}^T=\mathbf{\Phi}^{-1}$$

$$\mathbf{x}=\sum_{i=1}^{N} y_i\varphi_i \qquad \hat{\mathbf{x}}=\sum_{i=1}^{M} y_i\varphi_i,\ M\le N$$

$$\varepsilon^2=\mathrm{E}\left\{\|\mathbf{x}-\hat{\mathbf{x}}\|^2\right\}=\mathrm{E}\left\{\left\|\sum_{i=1}^{N} y_i\varphi_i-\sum_{i=1}^{M} y_i\varphi_i\right\|^2\right\}=\sum_{i=M+1}^{N}\mathrm{E}\left\{(y_i)^2\right\}$$

$$\hat{\varepsilon}=\varepsilon^2-\sum_{i=M+1}^{N}\lambda_i\left(\varphi_i^T\varphi_i-1\right)=\sum_{i=M+1}^{N}\left[\varphi_i^T\mathbf{C}_{\mathbf{xx}}\varphi_i-\lambda_i\left(\varphi_i^T\varphi_i-1\right)\right] \qquad \mathbf{C}_{\mathbf{xx}}=E\left\{\mathbf{x}\mathbf{x}^T\right\}$$

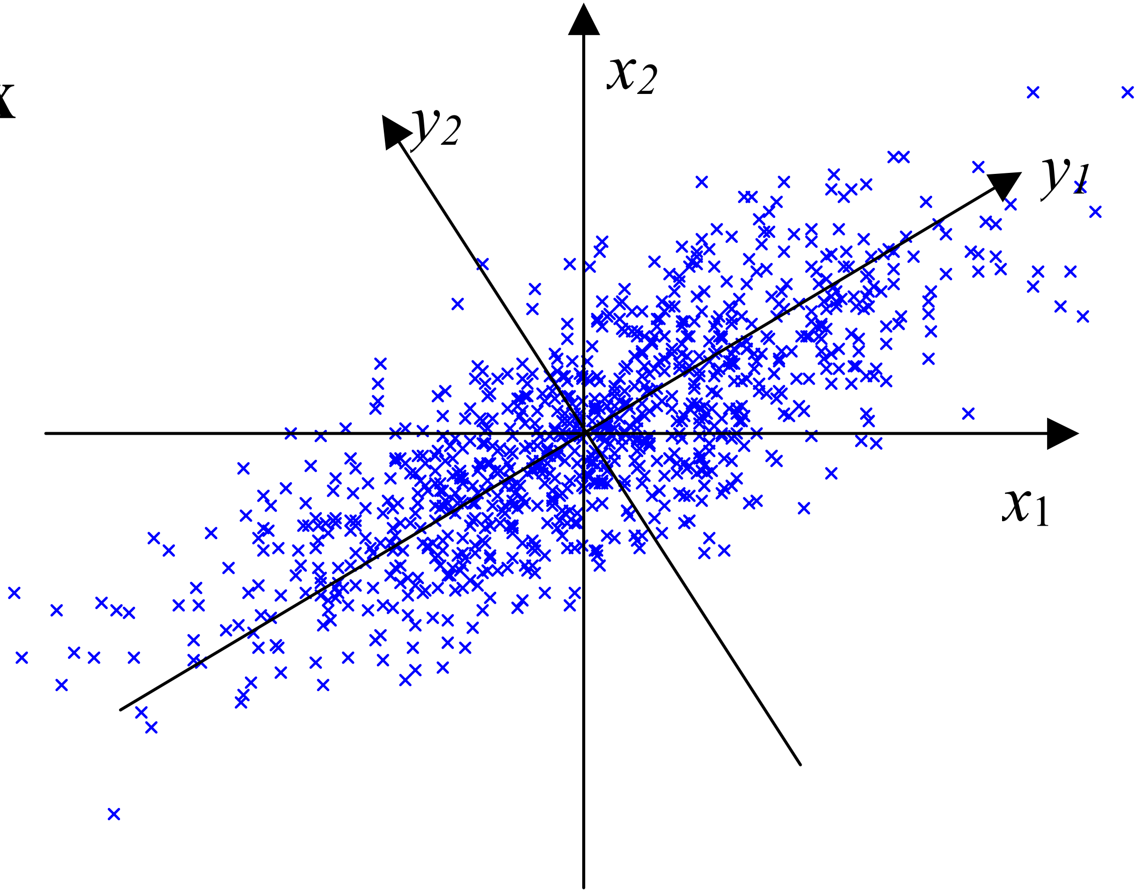
$$\frac{\partial\hat{\varepsilon}}{\partial\varphi_i}=\sum_{i=M+1}^{N}\left[2\mathbf{C}_{\mathbf{xx}}\varphi_i-2\lambda_i\varphi_i\right]=\mathbf{0}$$

$$\mathbf{C}_{\mathbf{xx}}\varphi_i=\lambda_i\varphi_i \qquad \varepsilon^2=\sum_{\imath=M+1}^{N}\varphi_i^T\mathbf{C}_{\mathbf{xx}}\varphi_i=\sum_{i=M+1}^{N}\varphi_i^T\lambda_i\varphi_i=\sum_{i=M+1}^{N}\lambda_i$$

Data compression, PCA networks

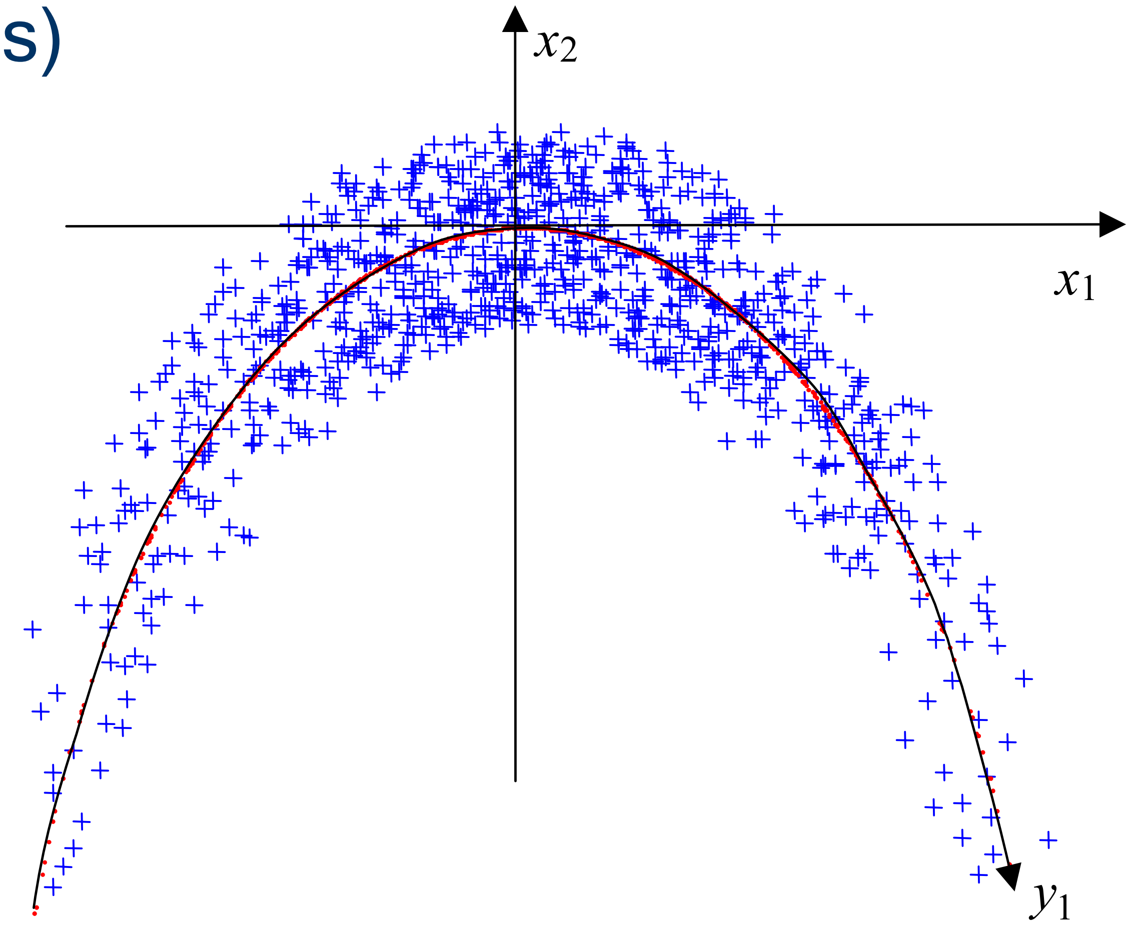
- Principal component analysis (Karhunen-Loève tranformation

$$\mathbf{y} = \mathbf{\Phi x}$$



Nonlinear data compression

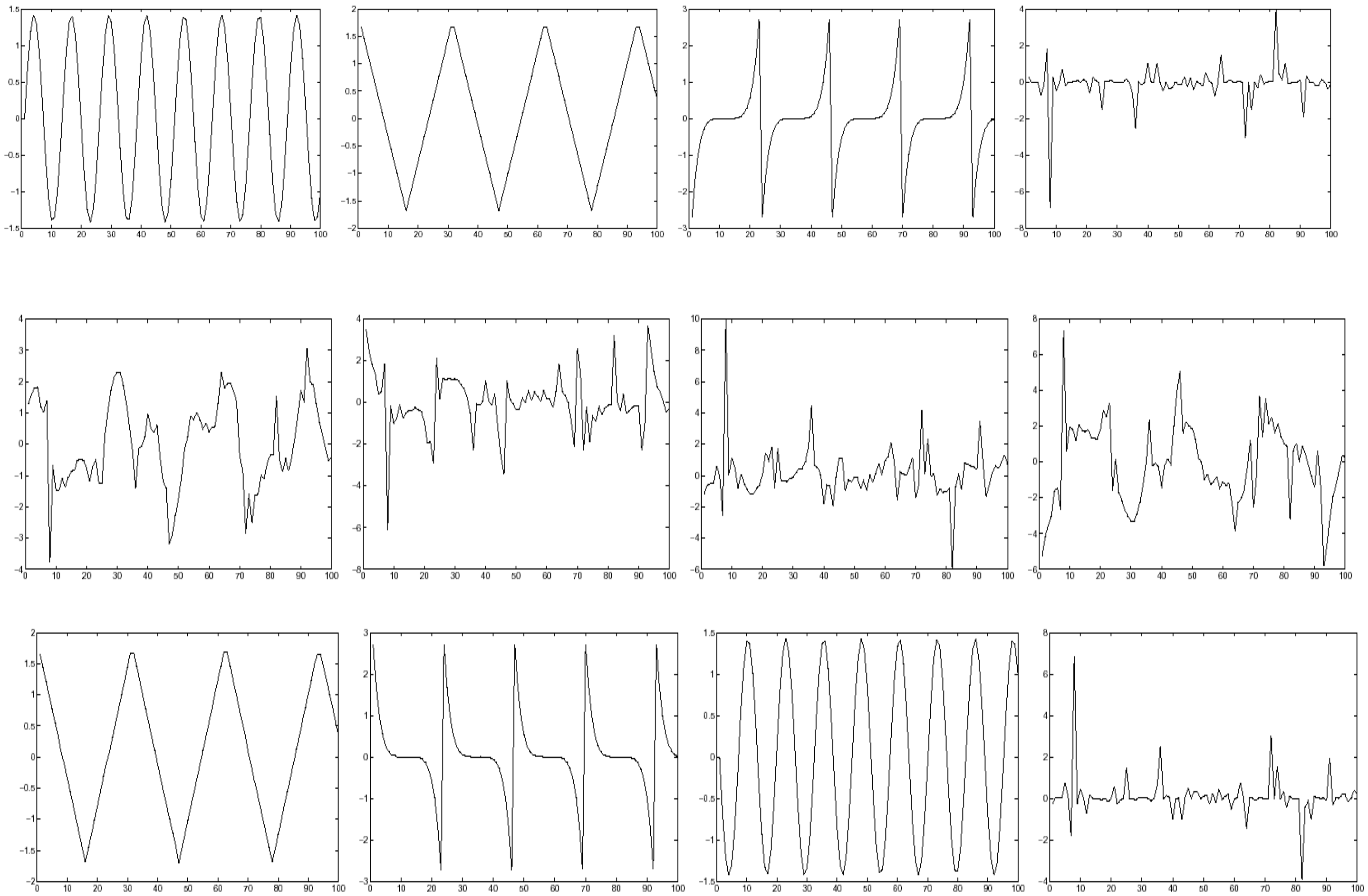
- Non-linear problem (curvilinear component analysis)



ICA networks

- Such linear transformation is looked for that restores the original components from mixed observations
- Many different approaches have been developed depending on the definition of independence (entropy, mutual information, Kullback-Leibleir information, non-Gaussianity)
- The weights can be obtained using nonlinear network (during training)
- Nonlinear version of the Oja rule

The task of independent component analysis



Pictures taken from: Aapo Hyvärinan: Survey of Independent Component Analysis

References and further readings

Brown, M. - Harris, C.J. and Parks, P "The Interpolation Capability of the Binary CMAC", *Neural Networks*, Vol. 6, pp. 429-440, 1993

Brown, M. and Harris, C.J. "Neurofuzzy Adaptive Modeling and Control" Prentice Hall, New York, 1994.

Hassoun, M. H.: "Fundamentals of Artificial Neural Networks", MIT Press, Cambridge, MA. 1995.

Haykin, S.: "Neural Networks. A Comprehensive Foundation" Prentice Hall, N. J.1999.

Hertz, J. - Krogh, A. - Palmer, R. G. "Introduction to the Theory of Neural Computation", Addison-Wesley Publishing Co. 1991.

Horváth, G. "CMAC: Reconsidering an Old Neural Network" *Proc. of the Intelligent Control Systems and Signal Processing, ICONS 2003*, Faro, Portugal. pp. 173-178, 2003.

Horváth, G. "Kernel CMAC with Improved Capability" *Proc. of the International Joint Conference on Neural Networks, IJCNN'2004*, Budapest, Hungary. 2004.

Lane, S.H. - Handelman, D.A. and Gelfand, J.J "Theory and Development of Higher-Order CMAC Neural Networks", *IEEE Control Systems*, Vol. Apr. pp. 23-30, 1992.

Miller, T.W. III. Glanz, F.H. and Kraft, L.G. "CMAC: An Associative Neural Network Alternative to Backpropagation" *Proceedings of the IEEE*, Vol. 78, pp. 1561-1567, 1990

Szabó, T. and Horváth, G. "Improving the Generalization Capability of the Binary CMAC" *Proc. of the International Joint Conference on Neural Networks, IJCNN'2000.* Como, Italy, Vol. 3, pp. 85-90, 2000.

Learning

Learning in neural networks

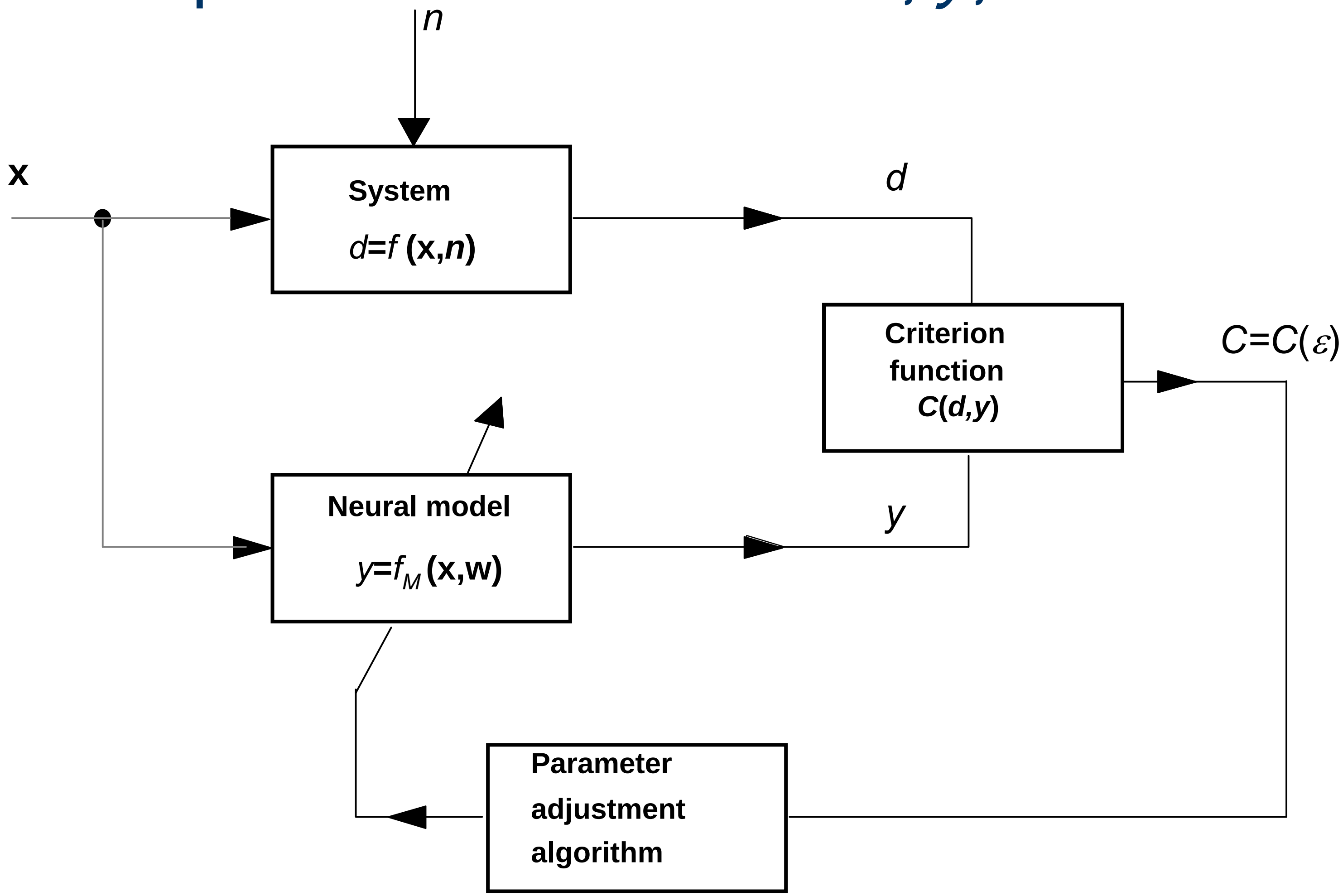
- Learning: parameter estimation
  - supervised learning, learning with a teacher

    x, y, d training set: $\{\mathbf{x}_i, \mathbf{d}_i\}_{i=1}^{P}$
  - unsupervised learning, learning without a teacher

    x, y
  - analytical learning

Supervised learning

- Model parameter estimation: **x**, *y*, *d*

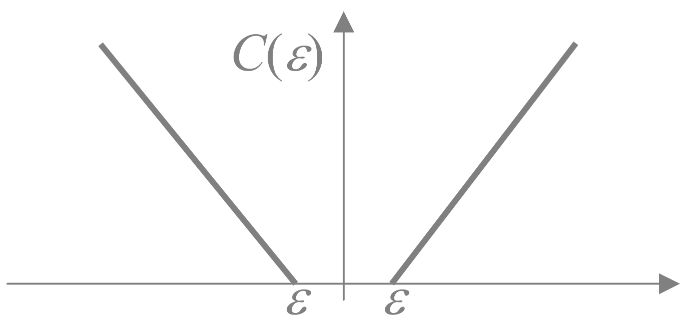


Supervised learning

- Criterion function
  - quadratic criterion function:

$$C(\mathbf{d},\mathbf{y}) = C(\boldsymbol{\varepsilon}) = \mathrm{E}\left\{(\mathbf{d}-\mathbf{y})^T(\mathbf{d}-\mathbf{y})\right\} = \mathrm{E}\left\{\sum_j \left(d_j - y_j\right)^2\right\}$$

  - other criterion functions
    - e.g. ε insensitive



  - regularized criterion functions: $C(\mathbf{d},\mathbf{y}) = C(\boldsymbol{\varepsilon}) + \lambda C_R$

    adding a penalty (regularization) term

Supervised learning

- Criterion minimization

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} C\big(\mathbf{d}, \mathbf{y}(\mathbf{w})\big)$$

- Analytical solution

  only in linear-in-the parameter cases

  e.g. linear networks: Wiener-Hopf equation

- Iterative solution
  - gradient methods
  - search methods
    - exhaustive search
    - random search
    - genetic search

Supervised learning

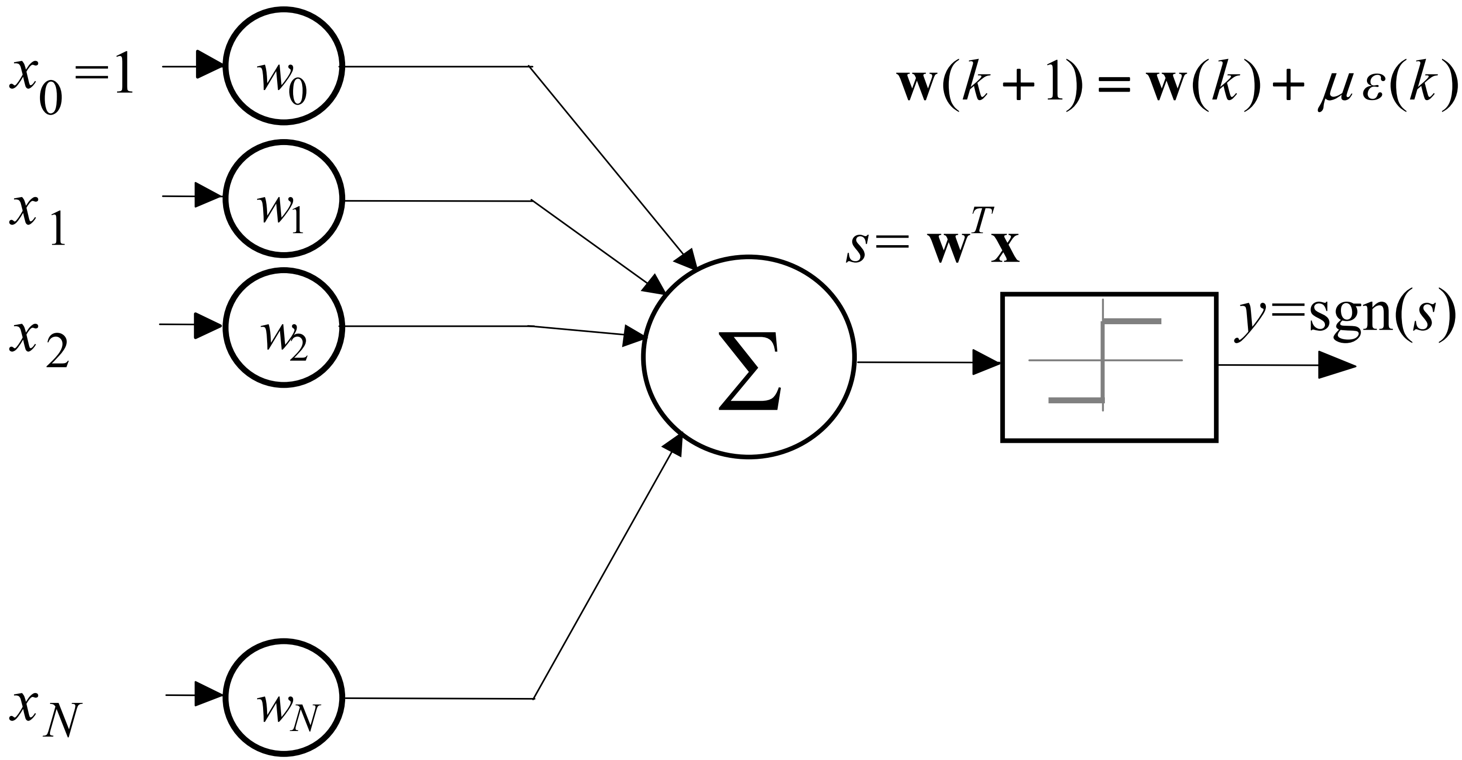
- Error correction rules
  - perceptron rule $\mathbf{w}(k+1)=\mathbf{w}(k)+\mu\varepsilon(k)\mathbf{x}(k)$
  - gradient methods $\mathbf{w}(k+1)=\mathbf{w}(k)+\mu\mathbf{Q}\big(-\nabla(k)\big)$
    - steepest descent $\mathbf{Q}=\mathbf{I}$
    - Newton $\mathbf{Q}=\mathbf{R}^{-1}$
    - Levenberg-Marquardt

$$\mathbf{w}(k+1)=\mathbf{w}(k)-\mathbf{H}(\mathbf{w}(k))^{-1}\nabla C(\mathbf{w}(k)). \qquad \mathbf{H}\cong \mathrm{E}\left\{\nabla y(\mathbf{w})\nabla y(\mathbf{w})^T\right\}+\lambda\boldsymbol{\Omega}$$

    - conjugate gradient

$$\mathbf{w}(k+1)=\mathbf{w}(k)+\alpha_k\mathbf{g}_k \qquad \mathbf{g}_j^T\mathbf{R}\mathbf{g}_k=0 \quad \text{if} \quad j\neq k$$

Perceptron training



$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\varepsilon(k)\mathbf{x}(k)$$

Converges in finite number of training steps if

we have a linearly separable two-class problem with finite number of samples with a finite upper bound $\|\mathbf{x}\| \leq M$ $\mu>0$

Gradient method

- Analytical solution
  - linear-in-the parameter model

$$y(k) = \mathbf{w}^T(k)\mathbf{x}(k).$$

  - quadratic criterion function

$$C(k) = \mathrm{E}\left\{\left(d(k) - \mathbf{w}^T(k)\mathbf{x}(k)\right)^2\right\}$$

$$= \mathrm{E}\left\{d^2(k)\right\} - 2\mathrm{E}\left\{d(k)\mathbf{x}^T(k)\right\}\mathbf{w}(k) + \mathbf{w}^T(k)\mathrm{E}\left\{\mathbf{x}(k)\mathbf{x}^T(k)\right\}\mathbf{w}(k)$$

$$= \mathrm{E}\left\{d^2(k)\right\} - 2\mathbf{p}^T\mathbf{w}(k) + \mathbf{w}^T(k)\mathbf{R}\mathbf{w}(k)$$

  - Wiener-Hopf equation

$$\mathbf{w}^* = \mathbf{R}^{-1}\mathbf{p}. \qquad \mathbf{R} = \mathrm{E}\left\{\mathbf{x}\mathbf{x}^T\right\} \qquad \mathbf{p} = \mathrm{E}\left\{\mathbf{x}y\right\}$$

Gradient method

- Iterative solution

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\left(-\nabla(k)\right).$$

  - gradient

$$\nabla(k) = \frac{\partial C(k)}{\partial \mathbf{w}(k)} = 2\mathbf{R}(\mathbf{w}(k) - \mathbf{w}^{*})$$

  - condition of convergence

$$0 < \mu < \frac{1}{\lambda_{\max}} \qquad \lambda_{\max} : \text{maximal eigenvalue of } \mathbf{R}$$

Gradient method

- LMS: iterative solution based on instantaneous error

$$\varepsilon(k) = d(k) - \mathbf{x}^T(k)\mathbf{w}(k) \qquad \hat{C}(k) = \varepsilon^2(k)$$

  - instantaneous gradient $$\hat{\nabla}(k) = \frac{\partial \hat{C}(k)}{\partial \mathbf{w}(k)} = 2\varepsilon(k)\frac{\partial \varepsilon(k)}{\partial \mathbf{w}(k)}$$

  - weight updating

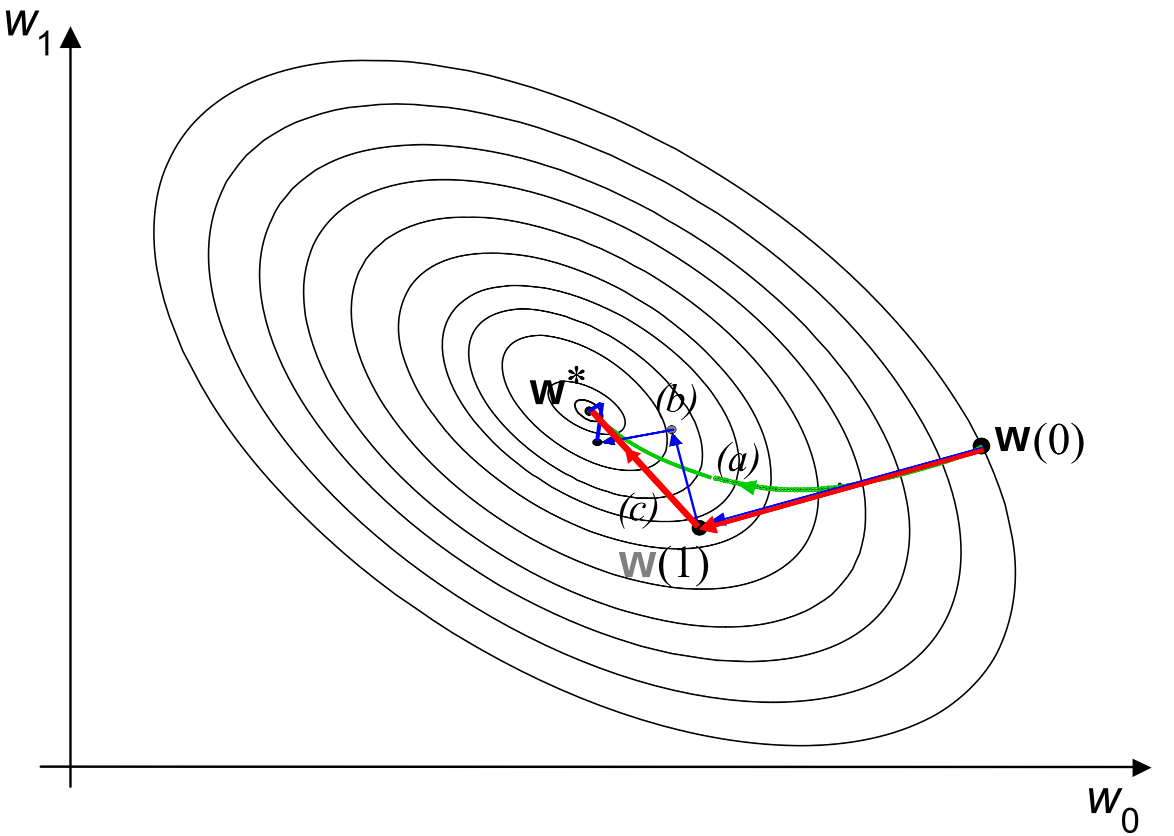
$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\left(-\hat{\nabla}(k)\right) = \mathbf{w}(k) + 2\mu\varepsilon(k)\mathbf{x}(k)$$

  - condition of convergence

$$0 < \mu < \frac{1}{\lambda_{\max}}$$

Gradient methods

- Example of convergence



a.) small μ b.) large μ c.) conjugate gradient

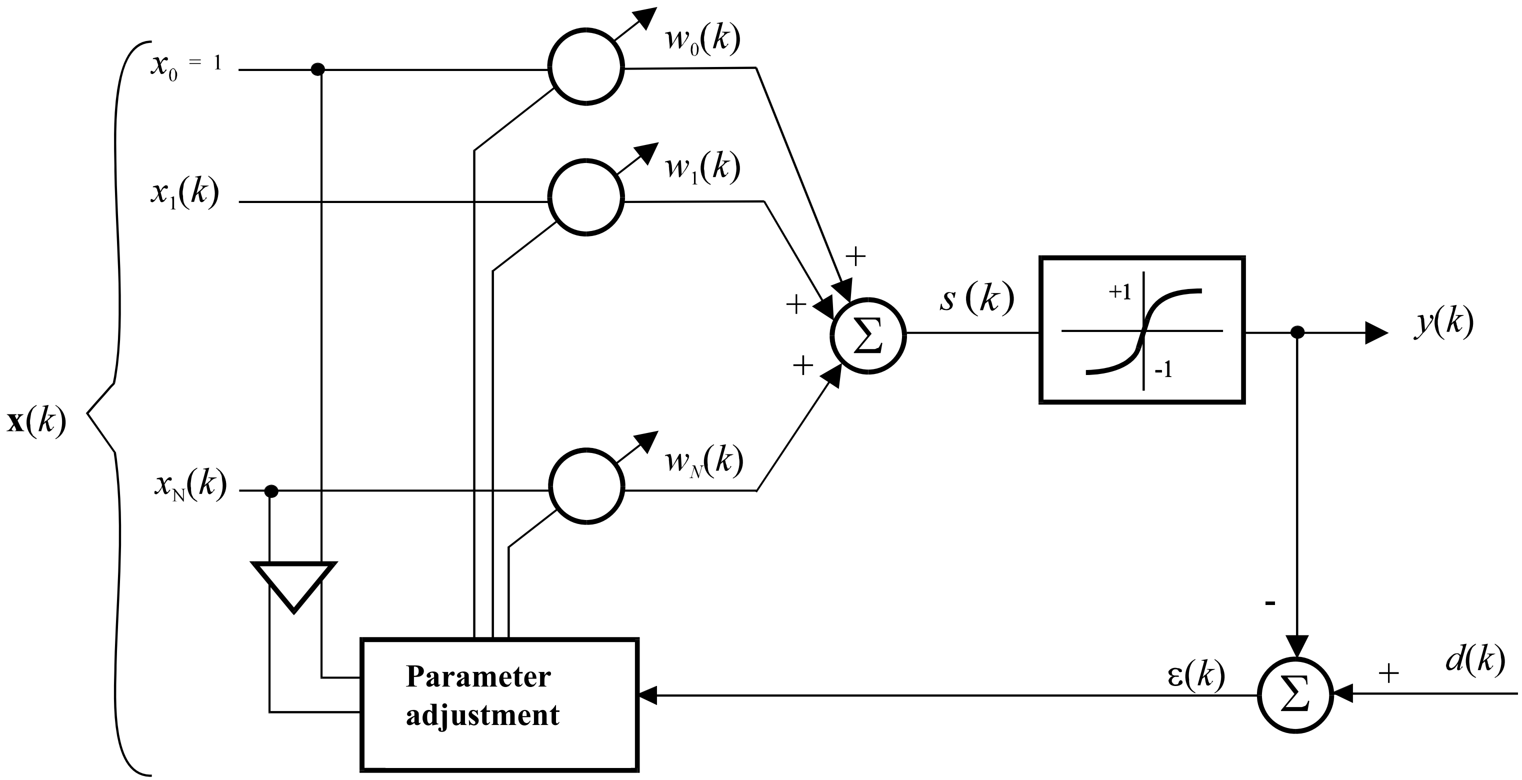
steepest descent

Gradient methods

- Single neuron with nonlinear activation function

$$\varepsilon(k) = d(k) - y(k) = d(k) - \mathrm{sgm}(s(k)) = d(k) - \mathrm{sgm}\left(\mathbf{w}^T(k)\mathbf{x}(k)\right)$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu(k)\,\varepsilon(k)\,\mathrm{sgm}'(s(k))\mathbf{x}(k) = \mathbf{w}(k) + 2\mu(k)\,\delta(k)\mathbf{x}(k)$$



Gradient methods

- Multi-layer network: error backpropagation (BP)

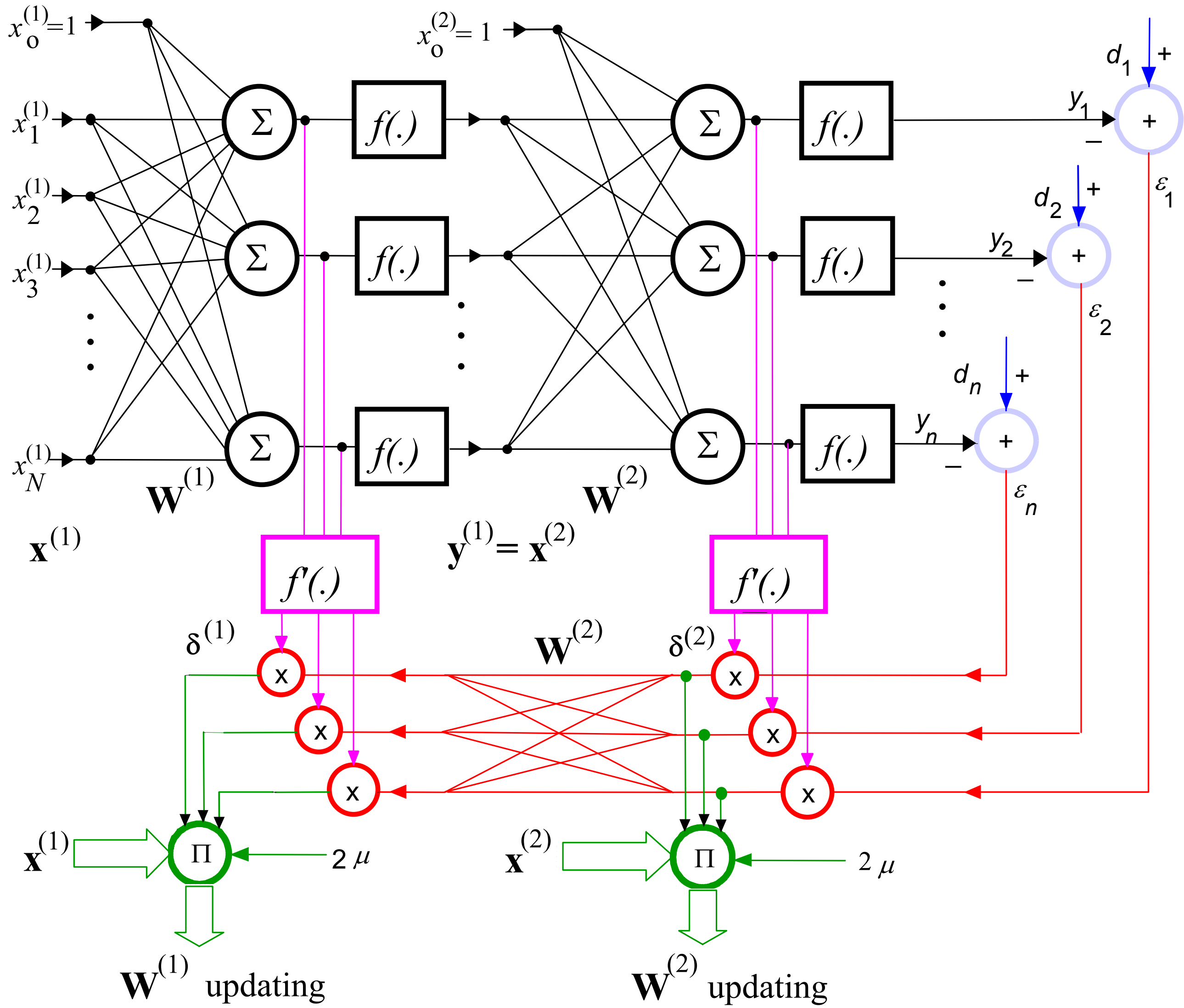
$$\mathbf{w}_i^{(l)}(k+1) = \mathbf{w}_i^{(l)}(k) + 2\mu\left(\sum_{r=1}^{N_{l+1}} \delta_r^{(l+1)}(k) w_{ri}^{(l+1)}(k)\right) \operatorname{sgm}'\left(s_i^{(l)}(k)\right)\mathbf{x}^{(l)}(k)$$

$$= \mathbf{w}_i^{(l)}(k) + 2\mu\delta_i^{(l)}(k)\mathbf{x}^{(l)}(k)$$

$$\delta_i^{(l)}(k) = \left(\sum_{r=1}^{N_{l+1}} \delta_r^{(l+1)}(k) w_{ri}^{(l+1)}(k)\right) \operatorname{sgm}'\left(s_i^{(l)}(k)\right)$$

l = layer index
i = processing element index
k = iteration index

MLP training: BP



Designing of an MLP

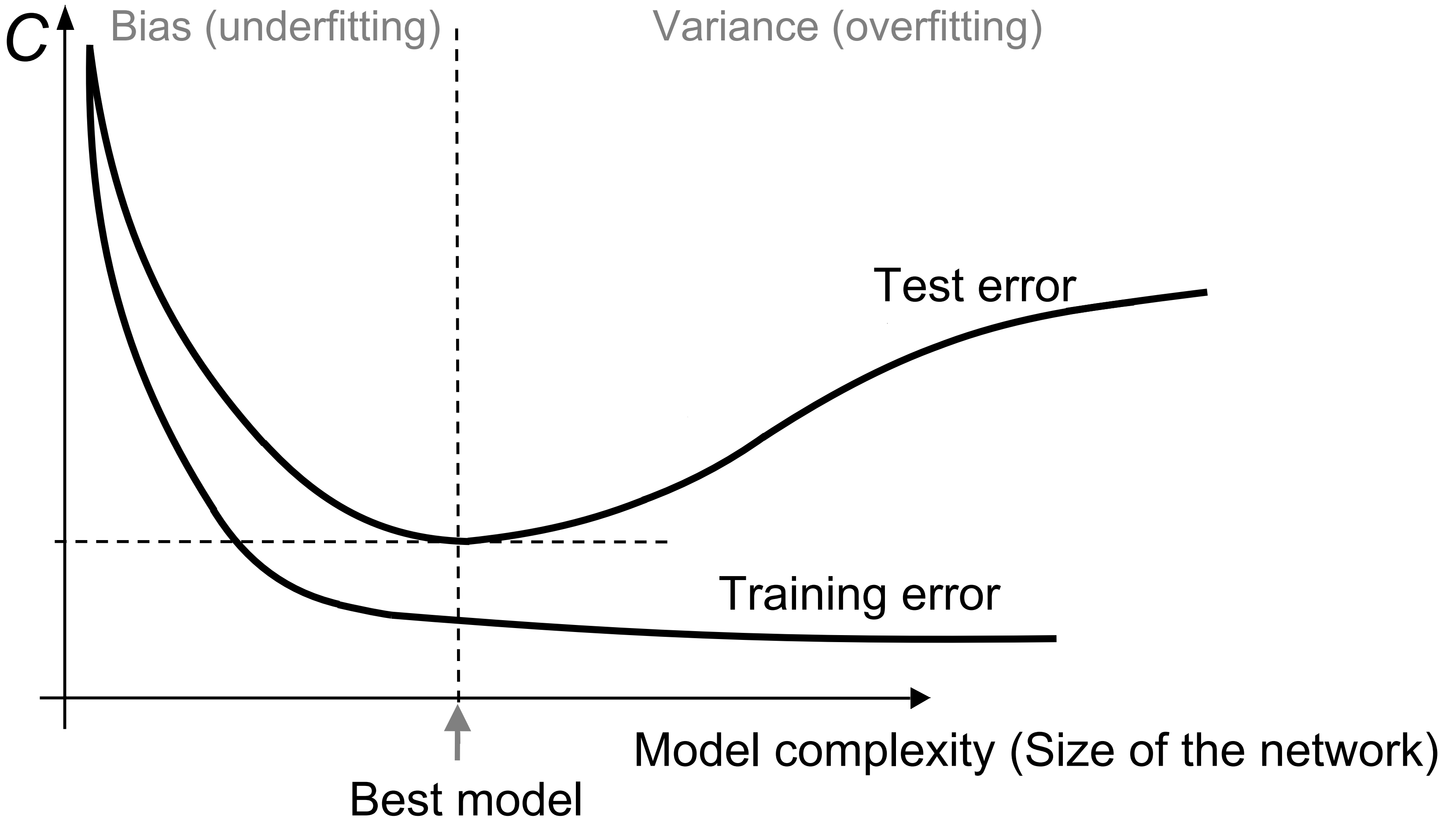
- important questions
  - the size of the network (model order: number of layers, number of hidden units)
  - the value of the learning rate, μ
  - initial values of the parameters (weights)
  - validation, cross validation learning and testing set selection
  - the way of learning, batch or sequential
  - stopping criteria

Designing of an MLP

- The size of the network: the number of hidden units (model order)
  - theoretical results: upper limits
- Practical approaches: two different strategies
  - from simple to complex
    - adding new neurons
  - from complex to simple
    - pruning
      - regularization
      - (OBD, OBS, etc)

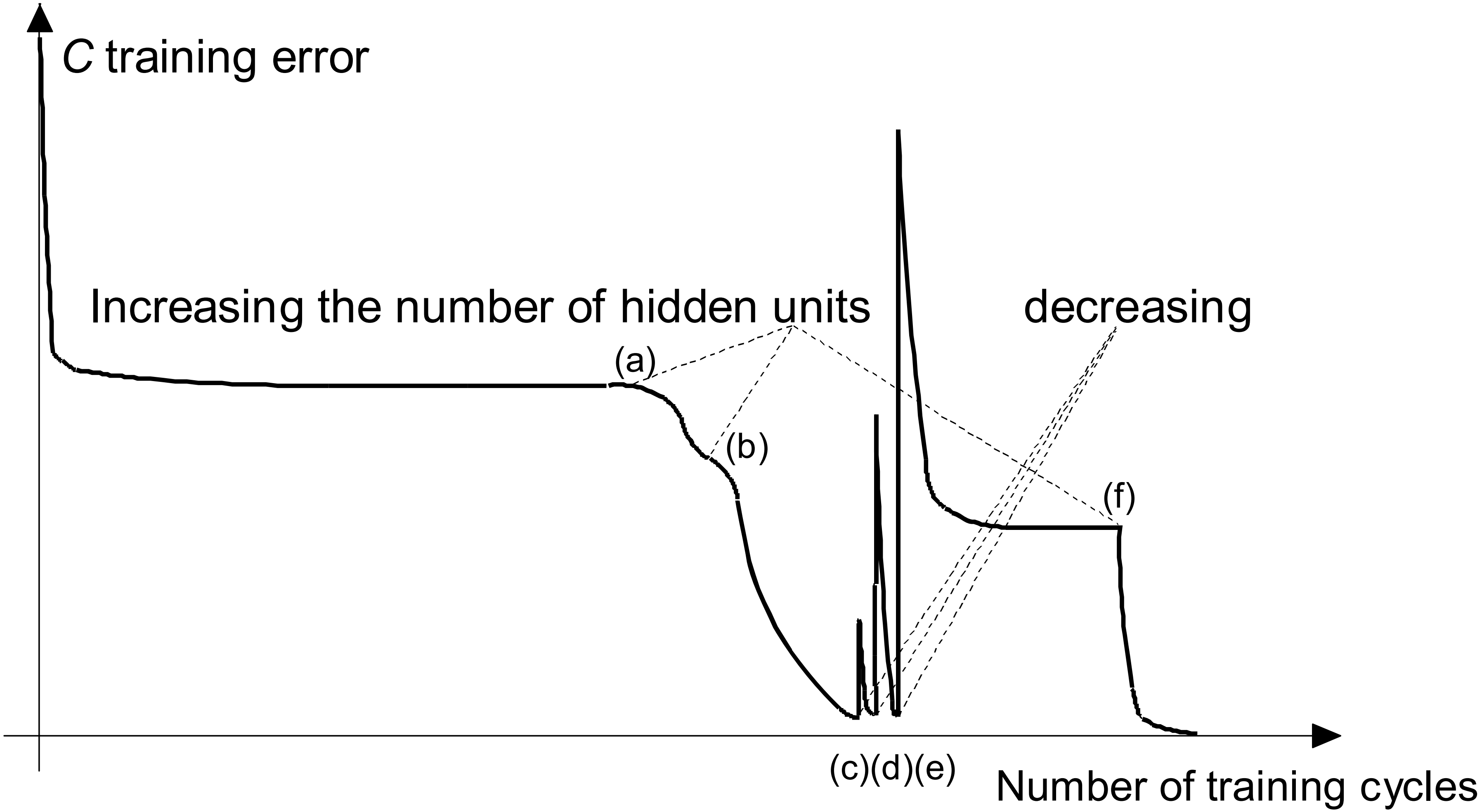
Designing of an MLP

- Cross validation for model selection



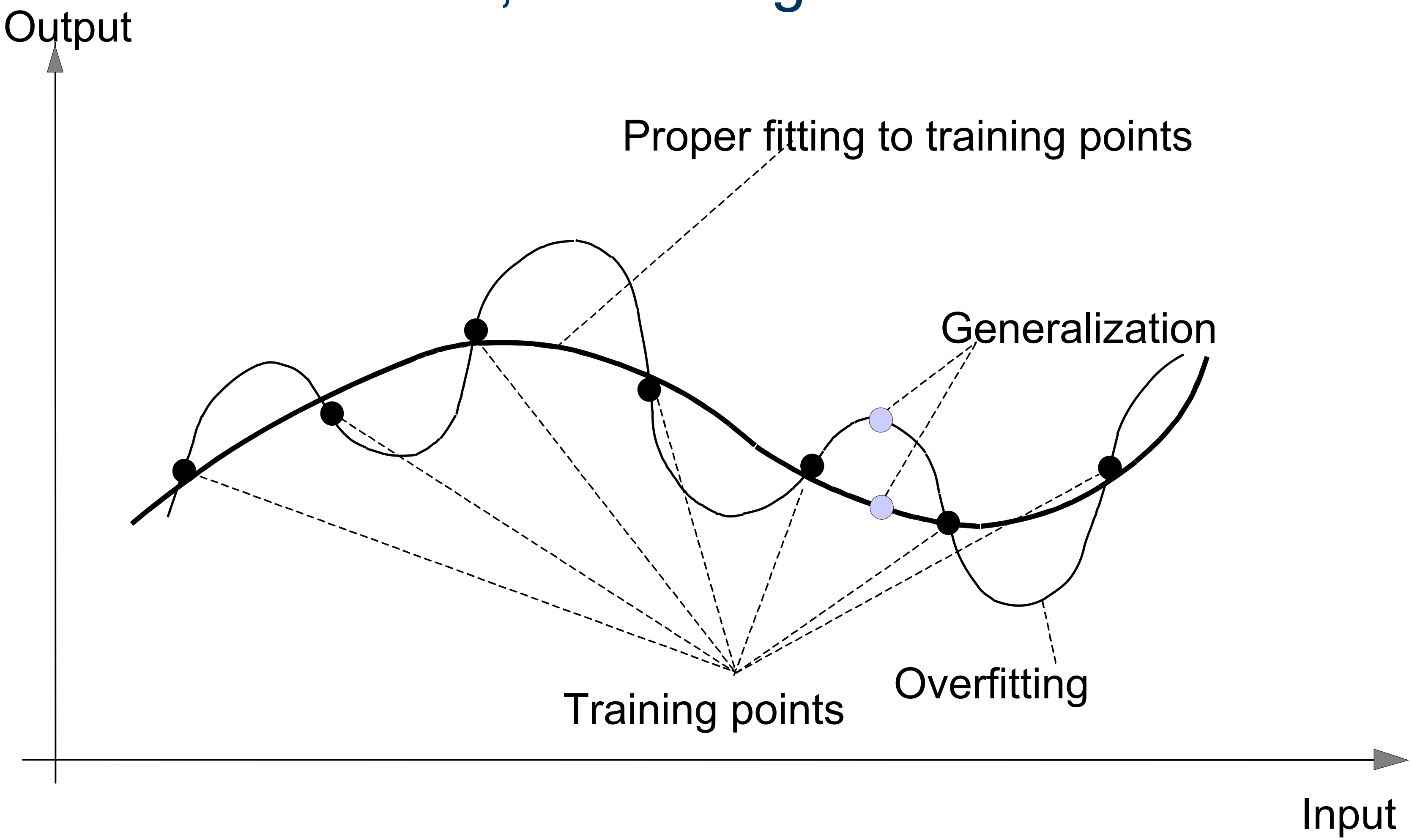
Designing of an MLP

- Structure selection



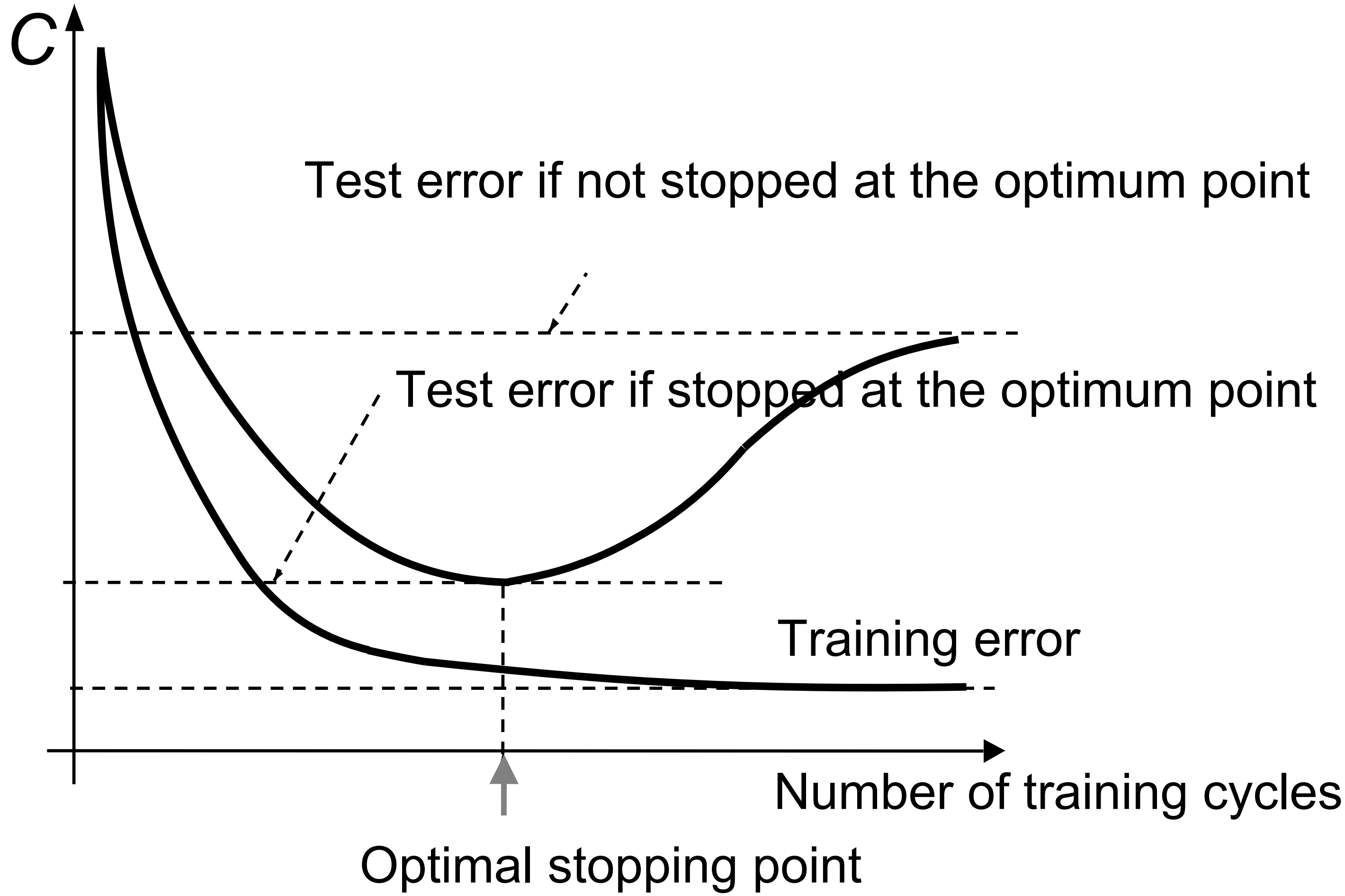
Designing of an MLP

- Generalization, overfitting



Designing of an MLP

- Early stopping for avoiding overfitting



Designing of an MLP

- Regularization
  - parametric penalty

$$C_r(\mathbf{w}) = C(\mathbf{w}) + \lambda \sum_{i,j} \left| w_{ij} \right|$$

$$\Delta w_{ij} = \mu \left(-\frac{\partial C}{\partial w_{ij}} \right) - \mu \lambda \,\mathrm{sgn}(w_{ij})$$

$$C_r(\mathbf{w}) = C(\mathbf{w}) + \boldsymbol{\lambda} \sum_{\left| w_{ij} \right| \le \Theta} \left| w_{ij} \right|$$

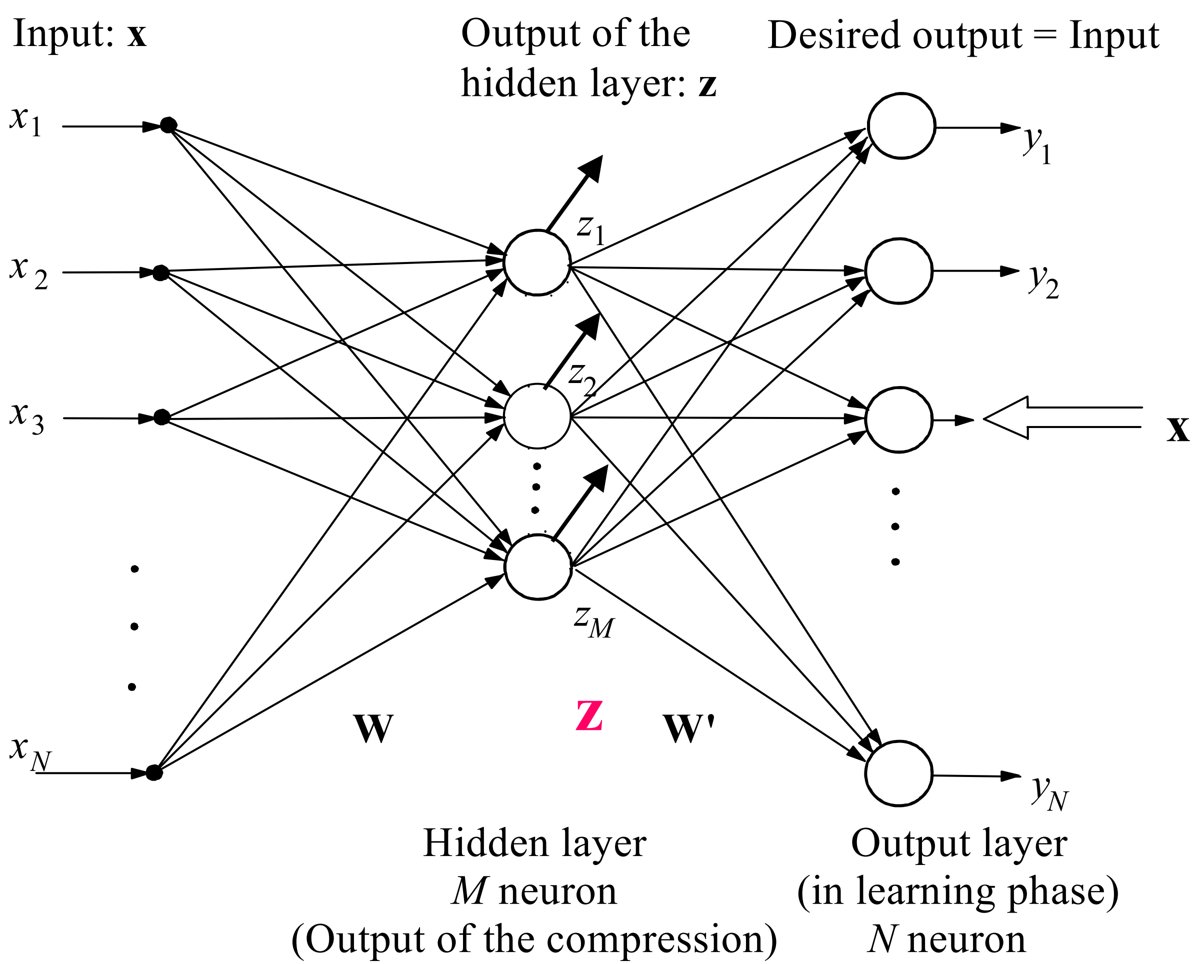
  - nonparametric penalty

$$C_r(\mathbf{w}) = C(\mathbf{w}) + \lambda \Phi\left(\hat{f}(x)\right)$$

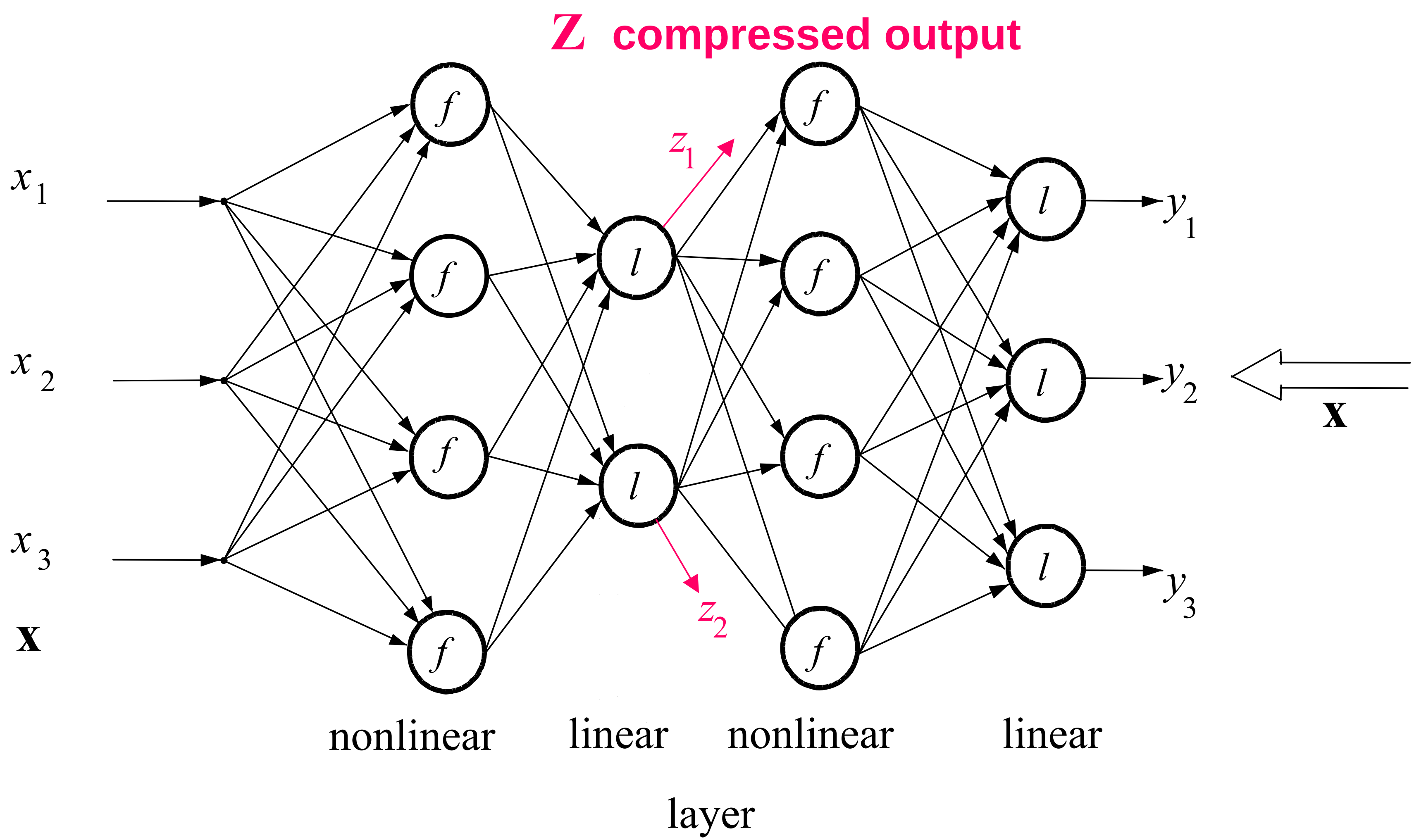
where $\Phi\left(\hat{f}(x)\right)$ is some measure of smoothness

MLP as linear data compressor
(autoassociative network)

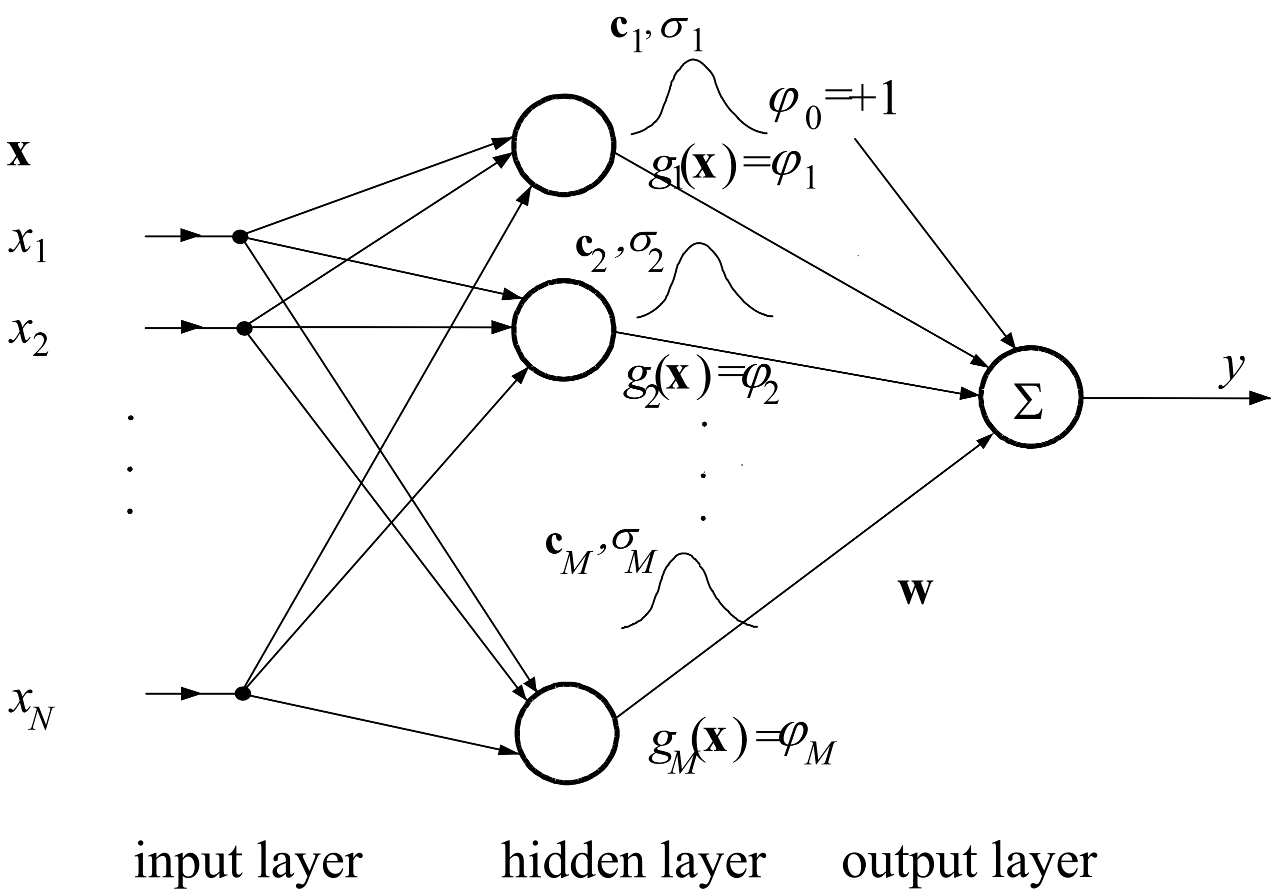
- Subspace transformation



Nonlinear data compression (autoassociative network)



RBF (Radial Basis Function)



$$y = \sum_i w_i g_i(\mathbf{x}) = \sum_i w_i g\left(\left\|\mathbf{x} - \mathbf{c}_i\right\|\right) = \mathbf{w}^T \mathbf{x} \qquad g_i(\mathbf{x}) = \exp\left[-\left\|\mathbf{x} - \mathbf{c}_i\right\|^2 / 2\sigma_i^2\right]$$

RBF training

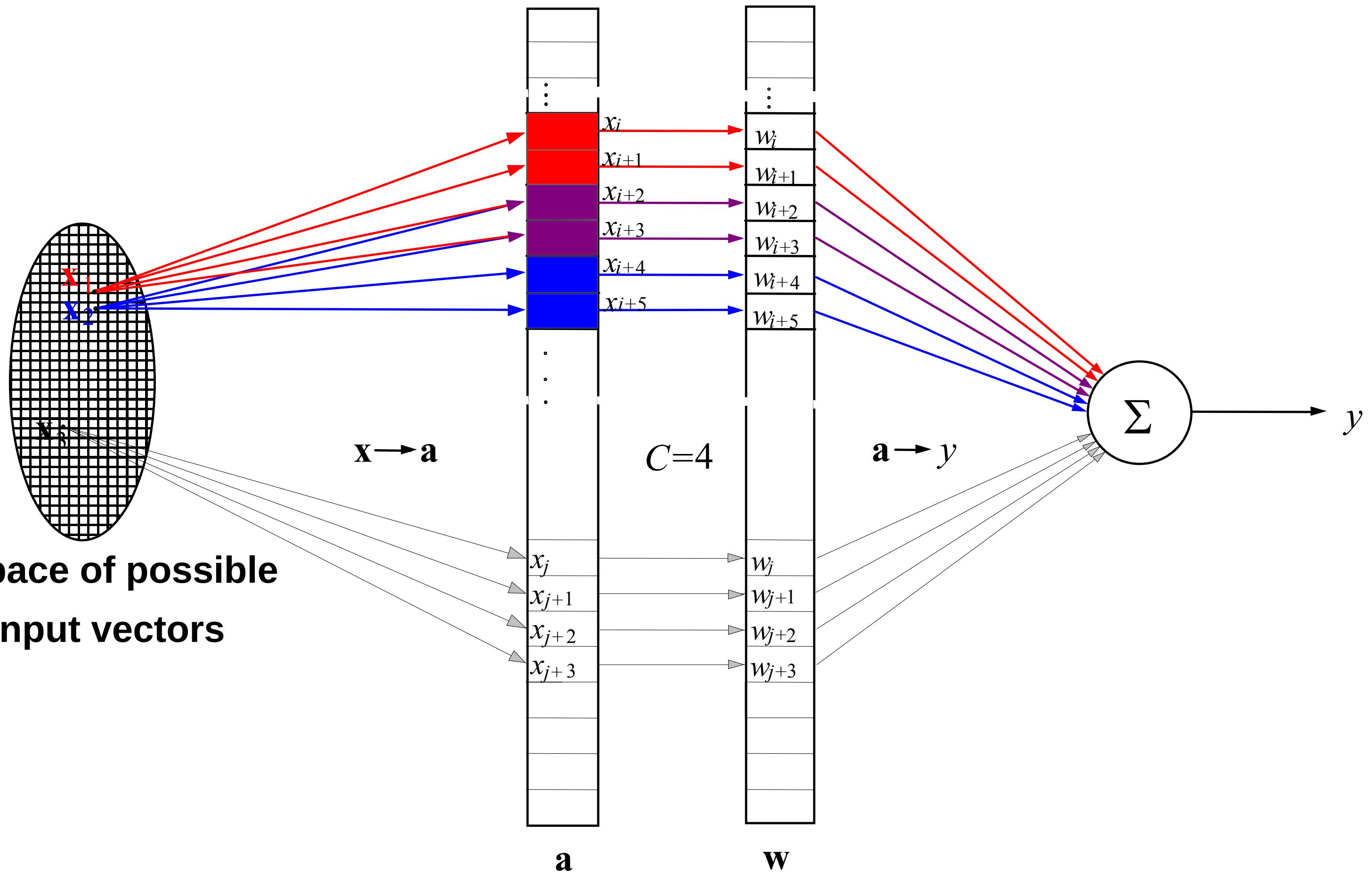
- Linear-in-the parameter structure
  - analytical solution
  - LMS
- Cetres (nonlinear-in-the-parameters)
  - *K*-means
  - clustering
  - unsupervised learning

Designing of an RBF

- Important questions
  - the size of the network (number of hidden units) (model order)
  - the value of learning rate, μ
  - initial values of parameters (centres, weights)
  - validation, learning and testing set selection
  - the way of learning, batch or sequential
  - stopping criteria

CMAC network

- Network with one trainable layer

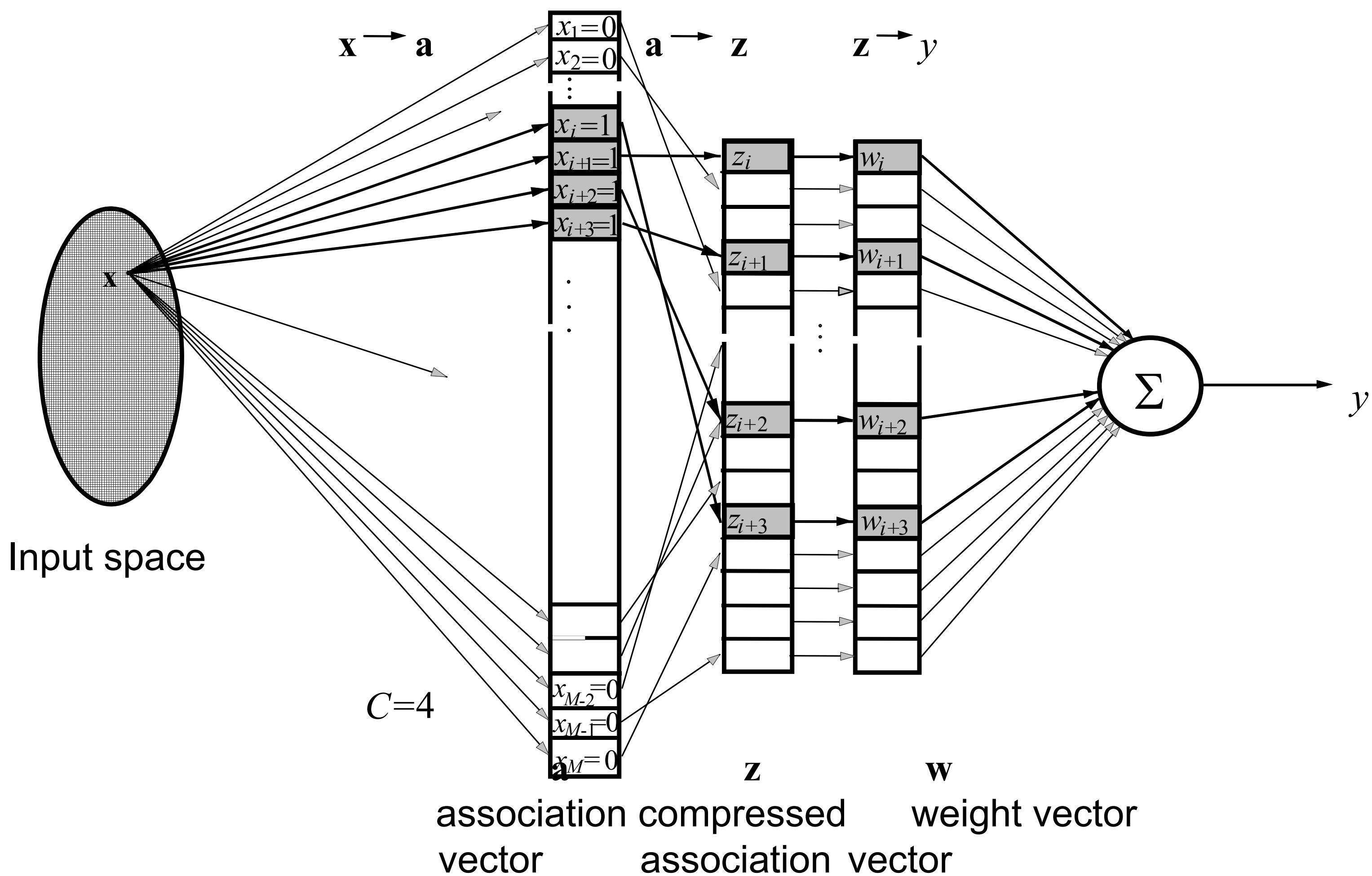


Space of possible input vectors

association vector **weight vector (trainable)**

CMAC network

- Network with hash-coding



CMAC modeling capability

- **Analytical solution**
- **Iterative algorithm (LMS)**

$$y(\mathbf{u}_i)=\mathbf{a}(\mathbf{u}_i)^T\mathbf{w} \quad i=1, 2, \ldots, P \qquad \mathbf{y}=\mathbf{A}\mathbf{w}$$

$$\mathbf{w}^* = \mathbf{A}^{\dagger}\mathbf{d} \qquad \mathbf{A}^{\dagger} = \mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1}$$

for univariate cases: $M \geq P$ for multivariate cases: $M < P$

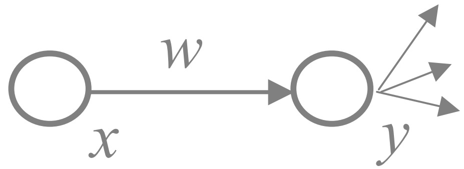
Networks with unsupervised learning

- Selforganizing network
  - Hebbian rule
  - Competitive learning
  - Main task
    - clustering, detection of similarities (normalized Hebbian + competitive)
    - data compression (PCA, KLT) (nomalized Hebbian)

Unsupervised learning

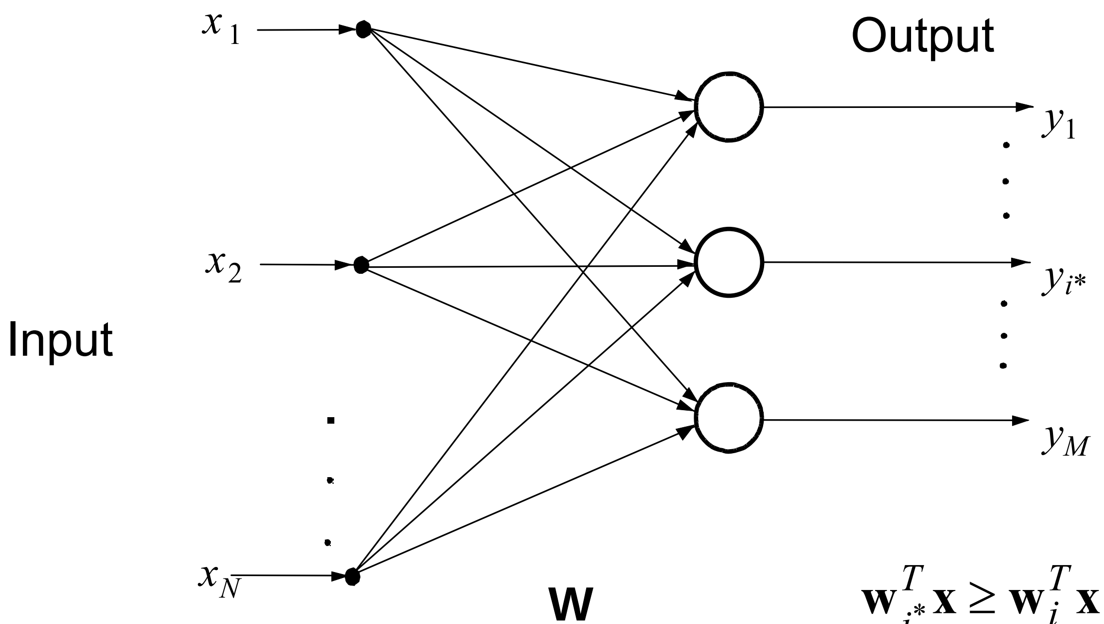
- Hebbian learning

$$\Delta w = \eta\, xy$$



- Competitive learning

Normalized Hebbian rule

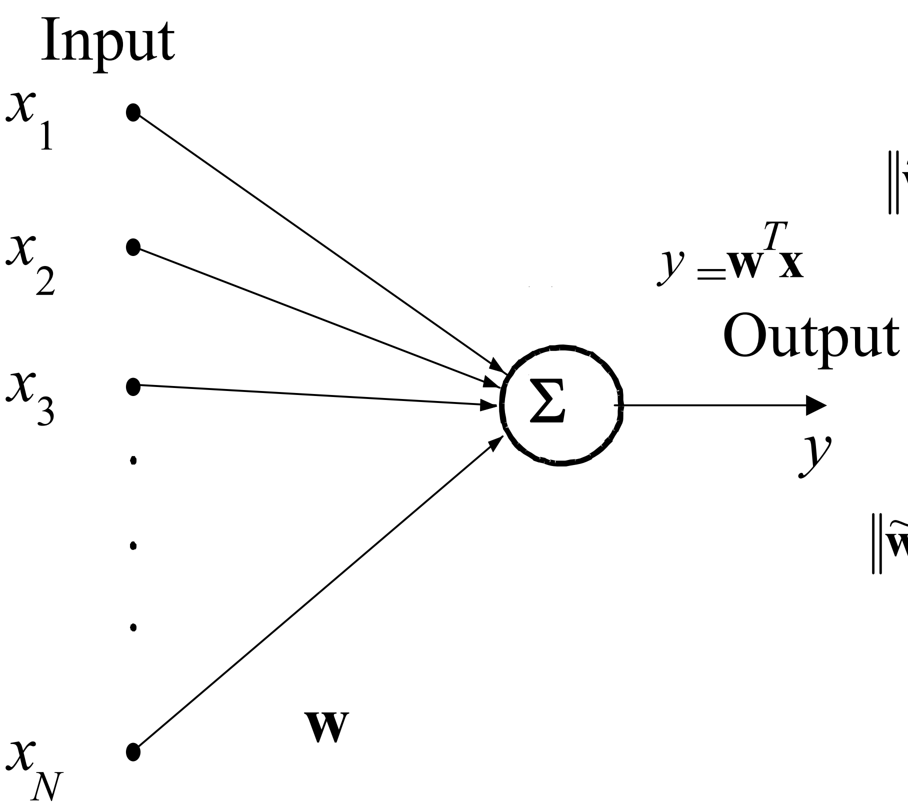


$$\Delta \mathbf{w}_{i*} = \mu\left(\mathbf{x} - \mathbf{w}_{i*}\right)$$

$$\mathbf{w}_{i^*}^T \mathbf{x} \geq \mathbf{w}_i^T \mathbf{x} \quad \forall i$$

PCA network

- Oja rule



$$\tilde{\mathbf{w}}(k+1)=\mathbf{w}(k)+\mu\mathbf{x}y$$

$$\mathbf{w}(k+1)=\frac{\tilde{\mathbf{w}}(k+1)}{\left\|\tilde{\mathbf{w}}(k+1)\right\|}=\tilde{\mathbf{w}}(k+1)\left\|\tilde{\mathbf{w}}(k+1)\right\|^{-1}$$

$$\begin{aligned}\left\|\tilde{\mathbf{w}}(k+1)\right\|^{2} &= \left\|\mathbf{w}(k)\right\|^{2}+2\mu\tilde{\mathbf{w}}(k)^{T}\mathbf{x}(k)y(k)+O\left(\mu^{2}\right)\\ &= \left\|\mathbf{w}(k)\right\|^{2}+2\mu\left[y(k)\right]^{2}+O\left(\mu^{2}\right)\end{aligned}$$

$$\left\|\tilde{\mathbf{w}}(k+1)\right\|^{-1}=\left(\left\|\tilde{\mathbf{w}}(k+1)\right\|^{2}\right)^{-1/2}=1-\mu\, y^{2}(k)+O\left(\mu^{2}\right)$$

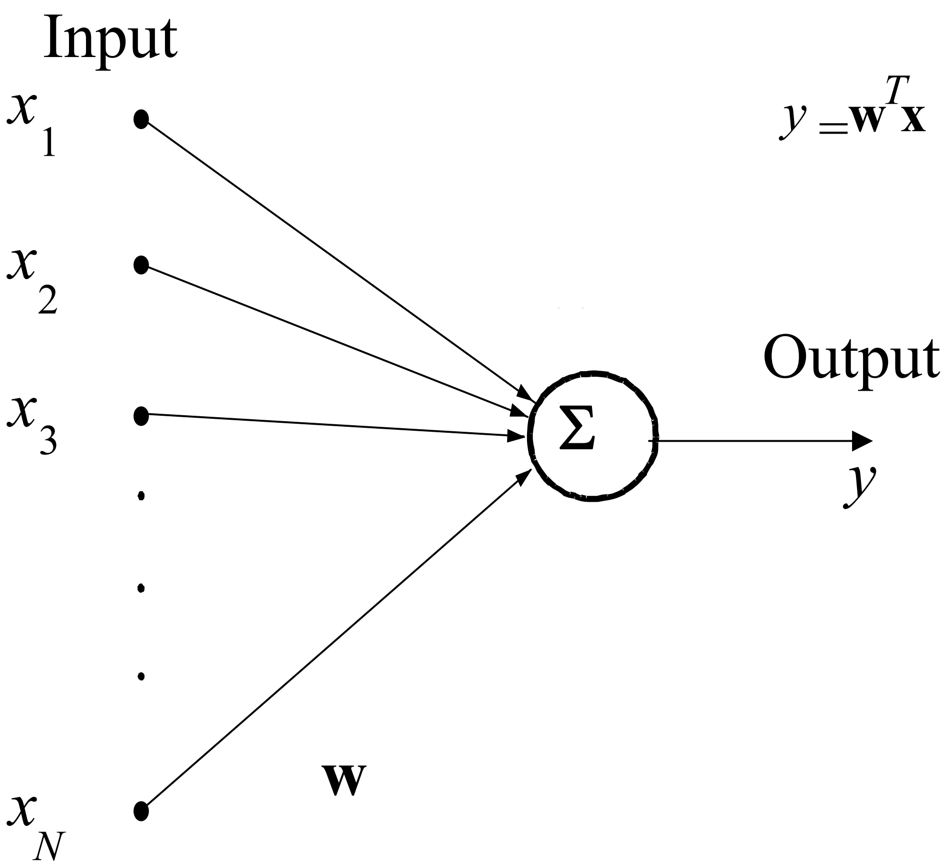
$$\Delta\mathbf{w}=\mu\, y\left(\mathbf{x}-y\,\mathbf{w}\right)=\mu\left(y\mathbf{x}-y^{2}\,\mathbf{w}\right)$$

$$\begin{aligned}\mathbf{w}(k+1) &= \left[\mathbf{w}(k)+\mu\mathbf{x}(k)y(k)\right]\left[1-\mu y^{2}(k)+O\left(\mu^{2}\right)\right]\\ &\cong \mathbf{w}(k)+\mu y(k)\left[\mathbf{x}(k)-\mathbf{w}(k)y(k)\right]\end{aligned}$$

It can be proofed: **w** converges to the largest eigenvector

PCA network

- Oja rule as a maximum problem (gradient search)



$$f(\mathbf{w}) = \frac{\mathrm{E}\{y^2\}}{\mathbf{w}^T\mathbf{w}} = \frac{\mathbf{w}^T\mathbf{R}\mathbf{w}}{\mathbf{w}^T\mathbf{w}}$$

$$\nabla_{f(\mathbf{w})} = 2\mathbf{R}\mathbf{w} - \left(\mathbf{w}^T\mathbf{R}\mathbf{w}\right)2\mathbf{w}$$

$$\nabla_{f(\mathbf{w})} = 2\mathrm{E}\{\mathbf{x}\mathbf{x}^T\}\mathbf{w} - 2\mathrm{E}\{\mathbf{w}^T\mathbf{x}\mathbf{x}^T\mathbf{w}\}\mathbf{w}$$
$$= 2\mathrm{E}\{\mathbf{x}y\} - 2\mathrm{E}\{y^2\}\mathbf{w}$$

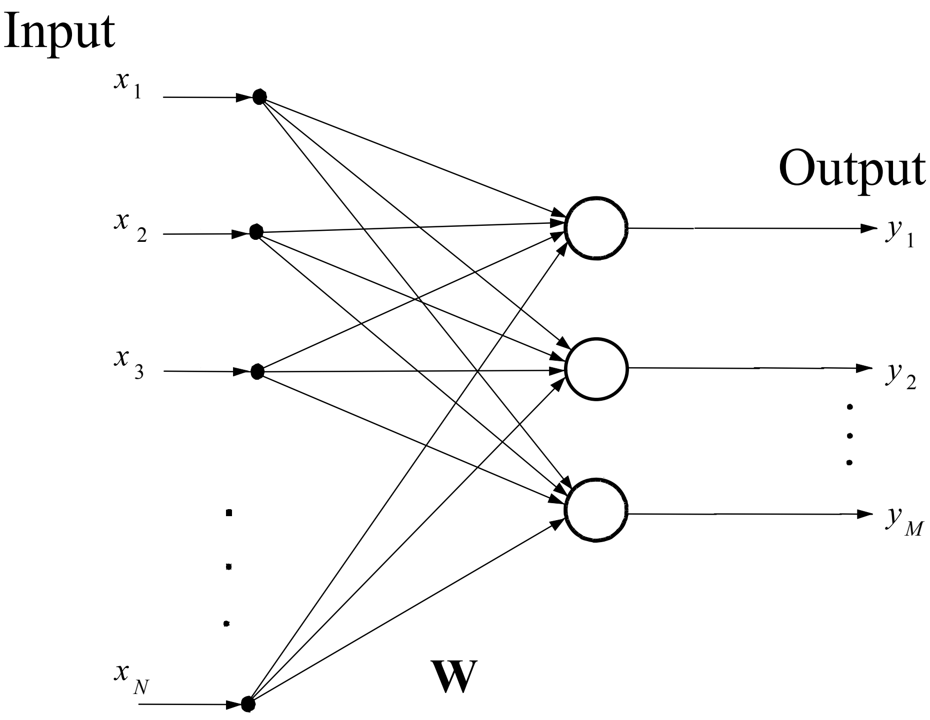
$$\nabla_{f(\mathbf{w})} = 2\mathbf{x}y - 2y^2\mathbf{w} = 2y(\mathbf{x} - \mathbf{w}y)$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu y(k)\left[\mathbf{x}(k) - \mathbf{w}(k)y(k)\right]$$

Solution: gradient method with instantenous gradient

PCA networks

- GHA network (Sanger network)



$$\Delta \mathbf{w}_1 = \mu\left(y_1 \mathbf{x}^{(1)} - y_1^2 \mathbf{w}_1\right)$$

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} - \left(\mathbf{w}_1^T \mathbf{x}^{(1)}\right)\mathbf{w}_1 = \mathbf{x}^{(1)} - y_1 \mathbf{w}_1$$

$$\Delta \mathbf{w}_2 = \mu\left(y_2 \mathbf{x}^{(2)} - y_2^2 \mathbf{w}_2\right) = \mu\left(y_2 \mathbf{x}^{(1)} - y_1 y_2 \mathbf{w}_1 - y_2^2 \mathbf{w}_2\right)$$

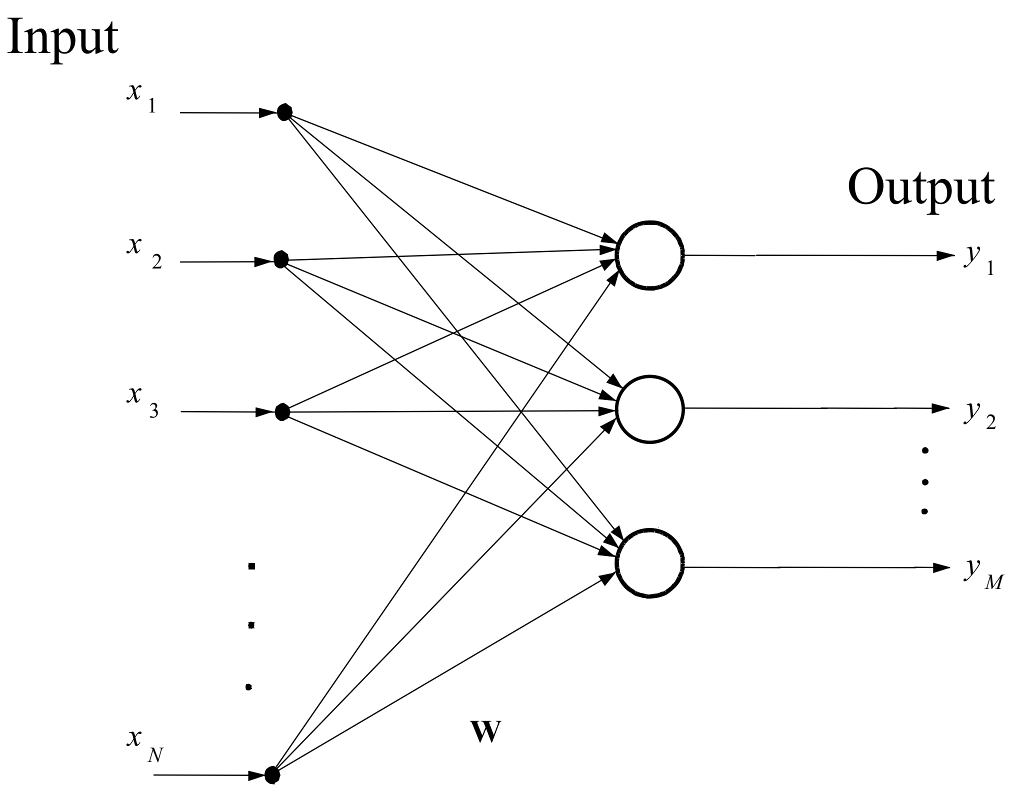
$$\begin{aligned}\Delta \mathbf{w}_i &= \mu\left(y_i \mathbf{x}^{(1)} - y_i^2 \mathbf{w}_i\right) \\ &= \mu\left(y_i \mathbf{x}^{(1)} - y_1 y_2 \mathbf{w}_i - \ldots - y_i^2 \mathbf{w}_i\right) \\ &= \mu\left(y_i \mathbf{x}^{(1)} - \sum_{j=1}^{i-1} y_i y_j \mathbf{w}_i - y_i^2 \mathbf{w}_i\right)\end{aligned}$$

$$\Delta \mathbf{W} = \eta\left[\mathbf{y}\mathbf{x}^T - \mathrm{LT}\left(\mathbf{y}\mathbf{y}^T\right)\mathbf{W}\right]$$

Oja rule + Gram-Schmidt orthogolarization

PCA networks

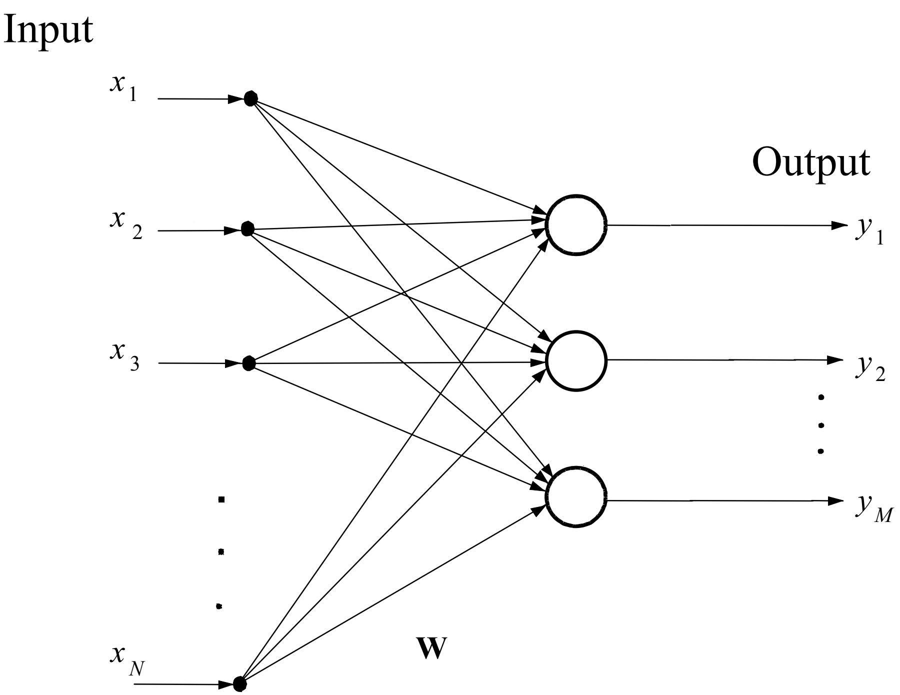
- Oja rule for multi-output (subspace problem)



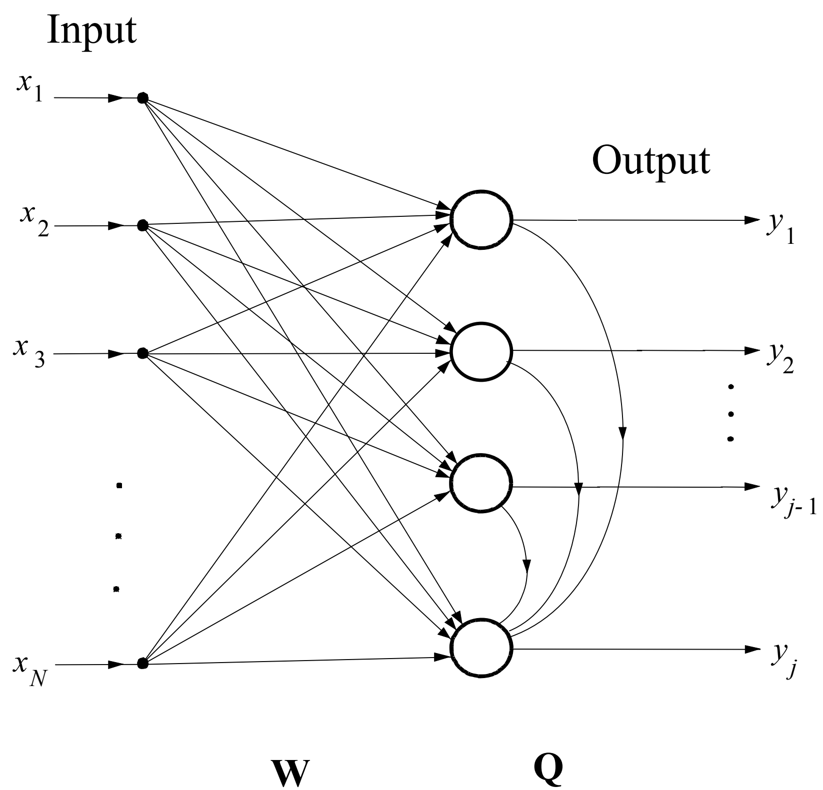
$$\Delta \mathbf{W} = \eta\left[\mathbf{y}\,\mathbf{x}^T - \left(\mathbf{y}\,\mathbf{y}^T\right)\mathbf{W}\right]$$

Output variance maximization rule

PCA networks



GHA (Sanger) network



Apex network

ICA networks

- Such linear transformation is looked for that restores the original components from mixed observations
- Many different approaches have been developed depending on the definition of independence (entropy, mutual information, Kullback-Leibleir information, non-Gaussianity)
- The weights can be obtained using nonlinear network (during training)
- Nonlinear version of the Oja rule

ICA training rule (one of the possible methods)

$$\mathbf{x}(k) = \mathbf{A}\mathbf{s}(k) + \mathbf{n}(k) = \sum_{i=1}^{M} \mathbf{a}_i s_i(k) + \mathbf{n}(k) \qquad \mathbf{y}(k) = \mathbf{B}\mathbf{x}(k) = \hat{\mathbf{s}}(k)$$

First step: whitening

$$\mathbf{v}(k) = \mathbf{V}\mathbf{x}(k) \qquad \mathrm{E}\left\{\mathbf{v}(k)\mathbf{v}(k)^T\right\} = \mathbf{I}$$

$$\mathbf{V}(k+1) = \mathbf{V}(k) - \mu(k)[\mathbf{v}(k)\mathbf{v}(k)^T - \mathbf{I}]\mathbf{V}(k)$$

Second step: separation

$$C(\mathbf{y}) = \sum_{i=1}^{M} \left|\mathrm{cum}\left(y_i^{\,4}\right)\right| = \sum_{i=1}^{M} \left|\mathrm{E}\left\{y_i^{\,4}\right\} - 3\mathrm{E}^2\left\{y_i^{\,2}\right\}\right|$$

$$\mathbf{B}(k) = \mathbf{W}(k)\mathbf{V}(k)$$

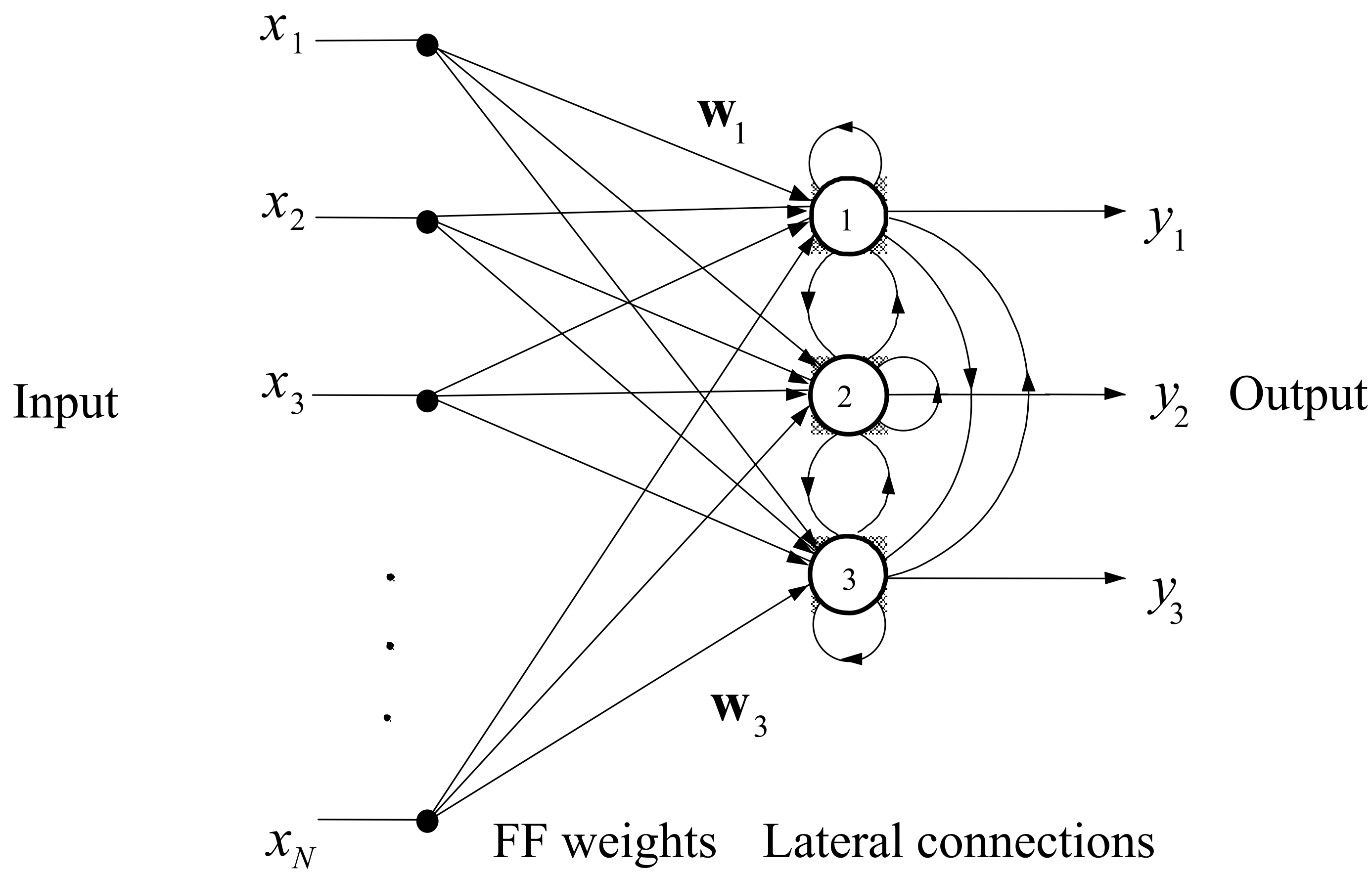
$$\mathbf{W}(k+1) = \mathbf{W}(k) + \eta(k)\left[\mathbf{v}(k) - \mathbf{W}(k)\,\mathbf{g}(\mathbf{y}(k))\right]\mathbf{g}\left(\mathbf{y}^T(k)\right) \qquad g(t)=\tanh(t)$$

Networks with unsupervised learning

- clustering
- detection of similarities
- data compression (PCA, KLT)
- Independent component analysis (ICA)

Networks with unsupervised learning

- Kohonen network: clustering

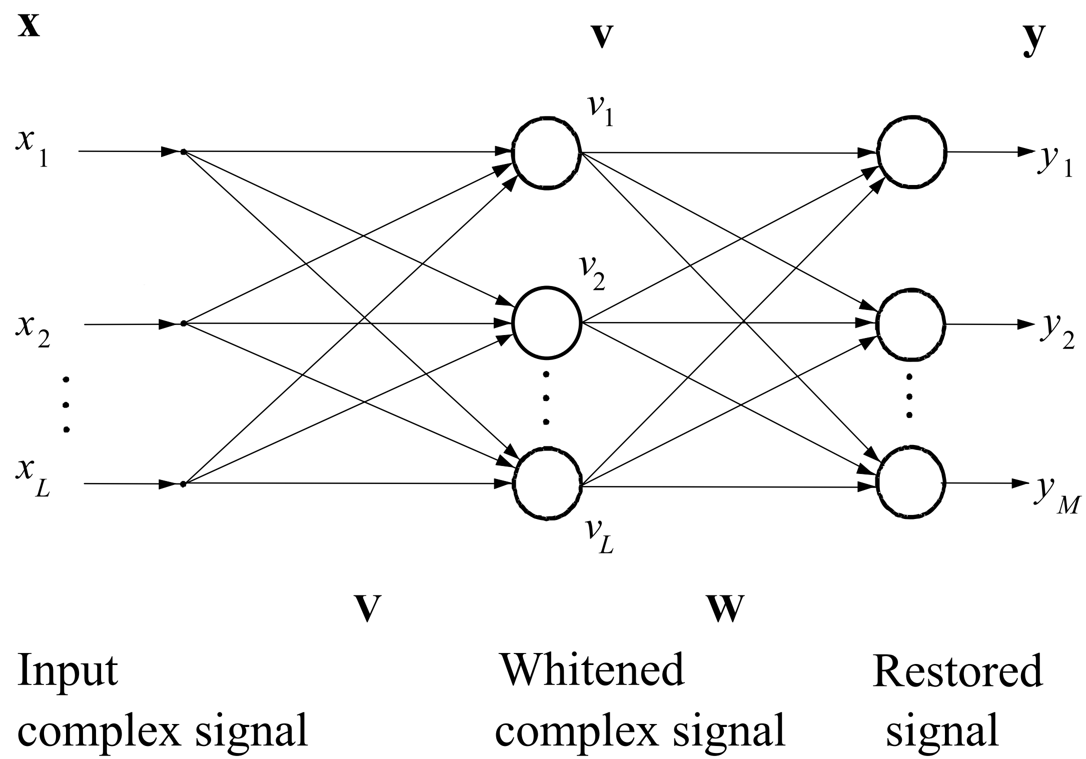


Independent component analysis

$$\mathbf{x}(k) = \mathbf{As}(k) + \mathbf{n}(k) = \sum_{i=1}^{M} \mathbf{a}_i s_i(k) + \mathbf{n}(k)$$

$$\mathbf{y}(k) = \mathbf{Bx}(k) = \hat{\mathbf{s}}(k)$$

ICA network architecture



References and further readings

Diamantaras, K. I .- Kung, S. Y. "Principal Component Neural Networks Theory and Applications", John Wiley and Sons, New York. 1996.

Girolami, M - Fyfe, C. "Higher Order Cumulant Maximisation Using Nonlinear Hebbian and Anti-Hebbian Learning for Adaptive Blind Separation of Source Signals", Proc. of the IEEE/IEE International Workshop on Signal and Image Processing, IWSIP-96, Advances in Computational Intelligence, Elsevier publishing, pp. 141 - 144, 1996.

Hassoun, M. H.: "Fundamentals of Artificial Neural Networks", MIT Press, Cambridge, MA. 1995.

Haykin, S.: "Neural Networks. A Comprehensive Foundation" Prentice Hall, N. J.1999.

Hertz, J. - Krogh, A. - Palmer, R. G. "Introduction to the Theory of Neural Computation", Addison-Wesley Publishing Co. 1991.

Karhunen, J. - Joutsensalo, J. "Representation and Separation of Signals using Nonlinear PCA Type Learning", Neural Networks, Vol. 7. No. 1. 1994. pp. 113-127.

Karhunen, J. - Hyvärinen, A. - Vigario, R. - Hurri, J. - and Oja, E. "Applications of Neural Blind Source Separation to Signal and Image Processing", Proceedings of the IEEE 1997 International Conference on Acoustics, Speech, and Signal Processing (ICASSP'97), April 21 - 24. 1997. Munich, Germany, pp. 131-134.

Kung, S. Y. - Diamantaras, C.I. "A Neural Network Learning Algorithm for Adaptive Principal Component Extraction (APEX)", International Confer-ence on Acoustics, Speech and Signal Processing, Vol. 2. 1990. pp. 861-864.

Narendra, K. S. - Parthasaraty, K. „Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks" IEEE Trans. on Neural networks, vol. 2. No. 2. pp. 252-262. 1991.

Sanger, T. "Optimal Unsupervised Learning in a Single-layer Linear Feedforward Neural Network", Neural Networks, Vol. 2. No. 6. 1989. pp. 459-473.

Widrow, B. - Stearns, S. D. "Adaptive Signal Processing", Prentice-Hall, Englewood Cliffs, N. J. 1985.

Dynamic neural architectures

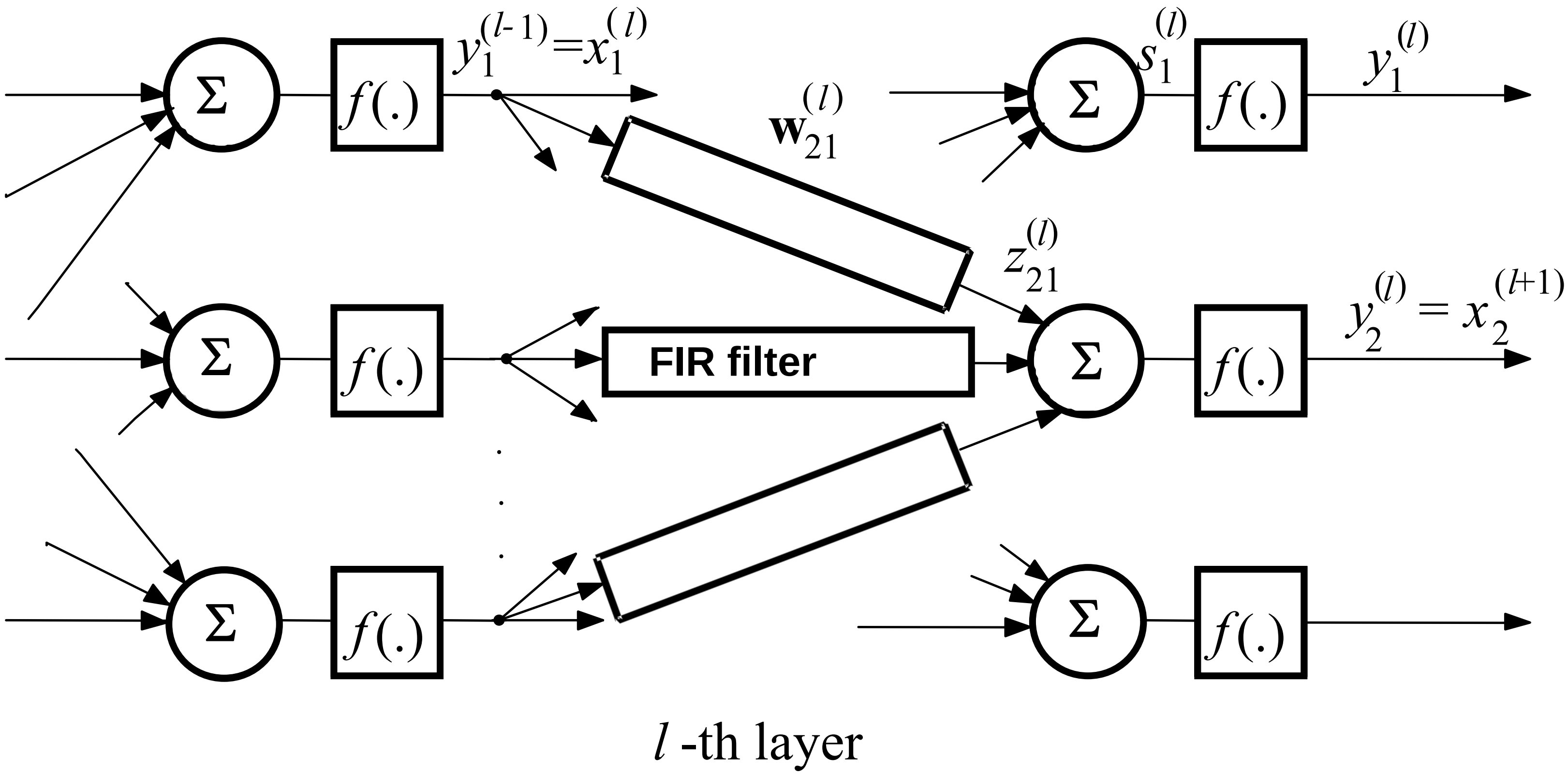
Dynamic neural structures

- Feed-forward networks
  - NFIR: FIR-MLP, FIR-RBF,etc.
  - NARX
- Feedback networks
  - RTRL
  - BPTT
- Main differences from static networks
  - time dependence (for all dynamic networks)
  - feedback (for feedback networks: NOE, NARMAX, etc.)
  - training: not for single sample pairs, but for sample sequences (sequantial networks)

Feed-forward architecture

- NFIR: FIR-MLP

  (winner of the Santa Fe competition for laser signal prediction)



Feed-forward architecture

FIR-MLP training: temporal backpropagation

$$\varepsilon^2 = \sum_{k=1}^{K} \varepsilon^2(k) \qquad \frac{\partial \varepsilon^2}{\partial \mathbf{w}_{ij}^{(l)}} = \sum_k \frac{\partial \varepsilon^2(k)}{\partial \mathbf{w}_{ij}^{(l)}} \qquad \frac{\partial \varepsilon^2}{\partial \mathbf{w}_{ij}^{(l)}} = \sum_k \frac{\partial \varepsilon^2}{\partial s_i^{(l)}(k)} \frac{\partial s_i^{(l)}(k)}{\partial \mathbf{w}_{ij}^{(l)}}$$

$$\frac{\partial \varepsilon^2(k)}{\partial \mathbf{w}_{ij}^{(l)}} \neq \frac{\partial \varepsilon^2}{\partial s_i^{(l)}(k)} \frac{\partial s_i^{(l)}(k)}{\partial \mathbf{w}_{ij}^{(l)}}$$

- output layer

$$\mathbf{w}_{ij}^{(L)}(k+1) = \mathbf{w}_{ij}^{(L)}(k) + 2\mu\varepsilon_i f'\left(s_i^{(L)}(k)\right)\mathbf{x}_j^{(L)}(k)$$

- hidden layer

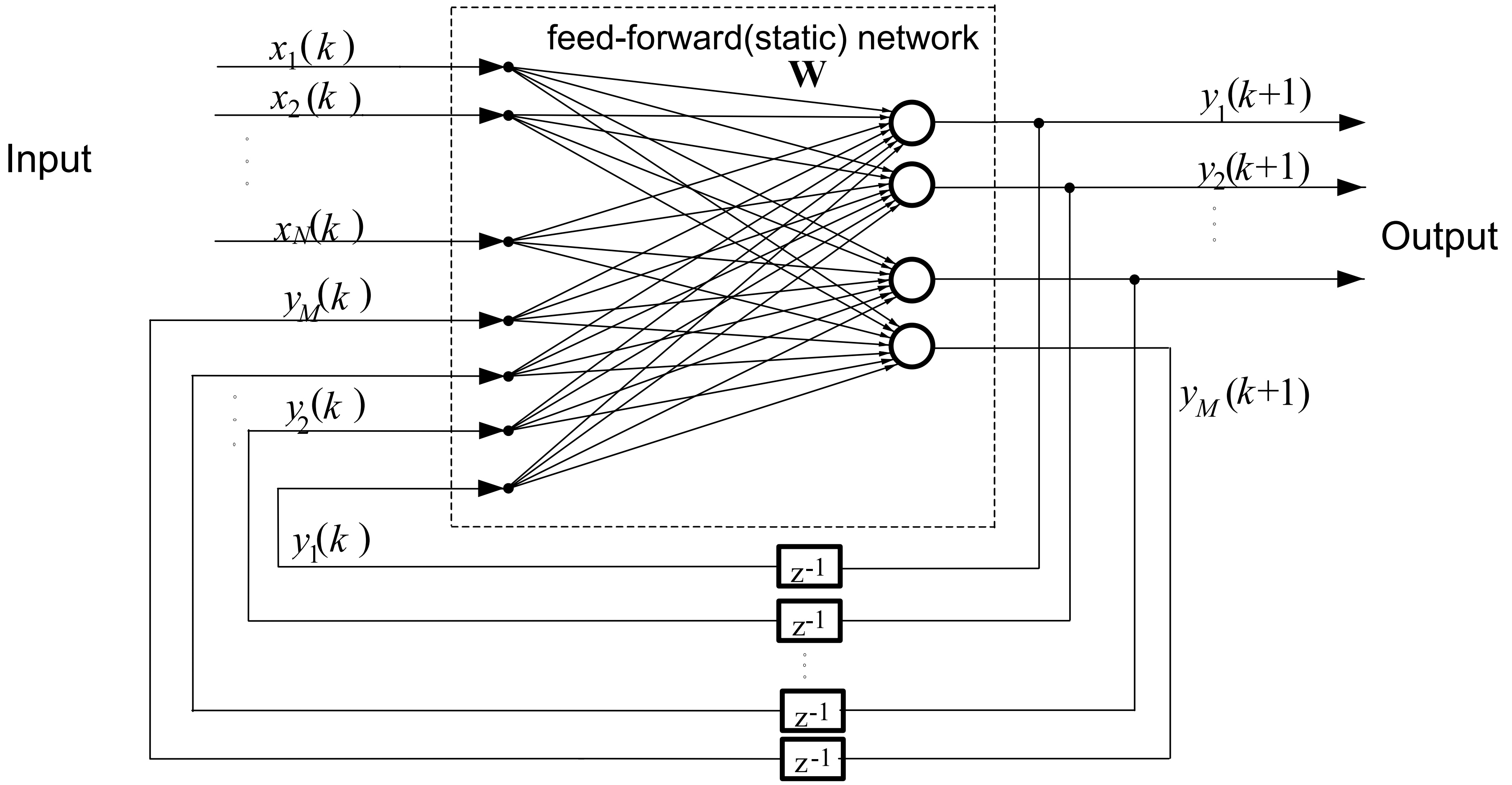
$$\mathbf{w}_{ij}(k+1) = \mathbf{w}_{ij}(k) + 2\mu\delta_i\,(k-M)\,\mathbf{x}_j(k-M)$$

$$\delta_i\,(k-M) = f'(s_i(k)) \sum_m \Delta_m^T(k-M)\,\mathbf{w}_{mi}$$

$$\Delta_m^{(l)}(k) = \left[\delta_m^{(l)}(k), \delta_m^{(l)}(k+1), \ldots, \delta_m^{(l)}\left(k + M_m^{(l)}\right)\right]$$

Recurrent network

- Architecture



Recurrent network

- Training: real-time recursive learning (RTRL)

$$\frac{\partial \varepsilon^2(k)}{\partial w_{ij}(k)} = \sum_{l\in \mathbf{C}} \frac{\partial \varepsilon_l^2(k)}{\partial w_{ij}(k)};\qquad \Delta w_{ij}(k) = -\mu \frac{\partial \varepsilon^2(k)}{\partial w_{ij}(k)};\qquad \frac{\partial \varepsilon^2(k)}{\partial w_{ij}(k)} = 2\sum_{l\in \mathbf{C}} \varepsilon_l(k)\frac{\partial \varepsilon_l(k)}{\partial w_{ij}(k)}$$

$$= -2\sum_{l\in \mathbf{C}} \varepsilon_l(k)\frac{\partial y_l(k)}{\partial w_{ij}(k)}$$

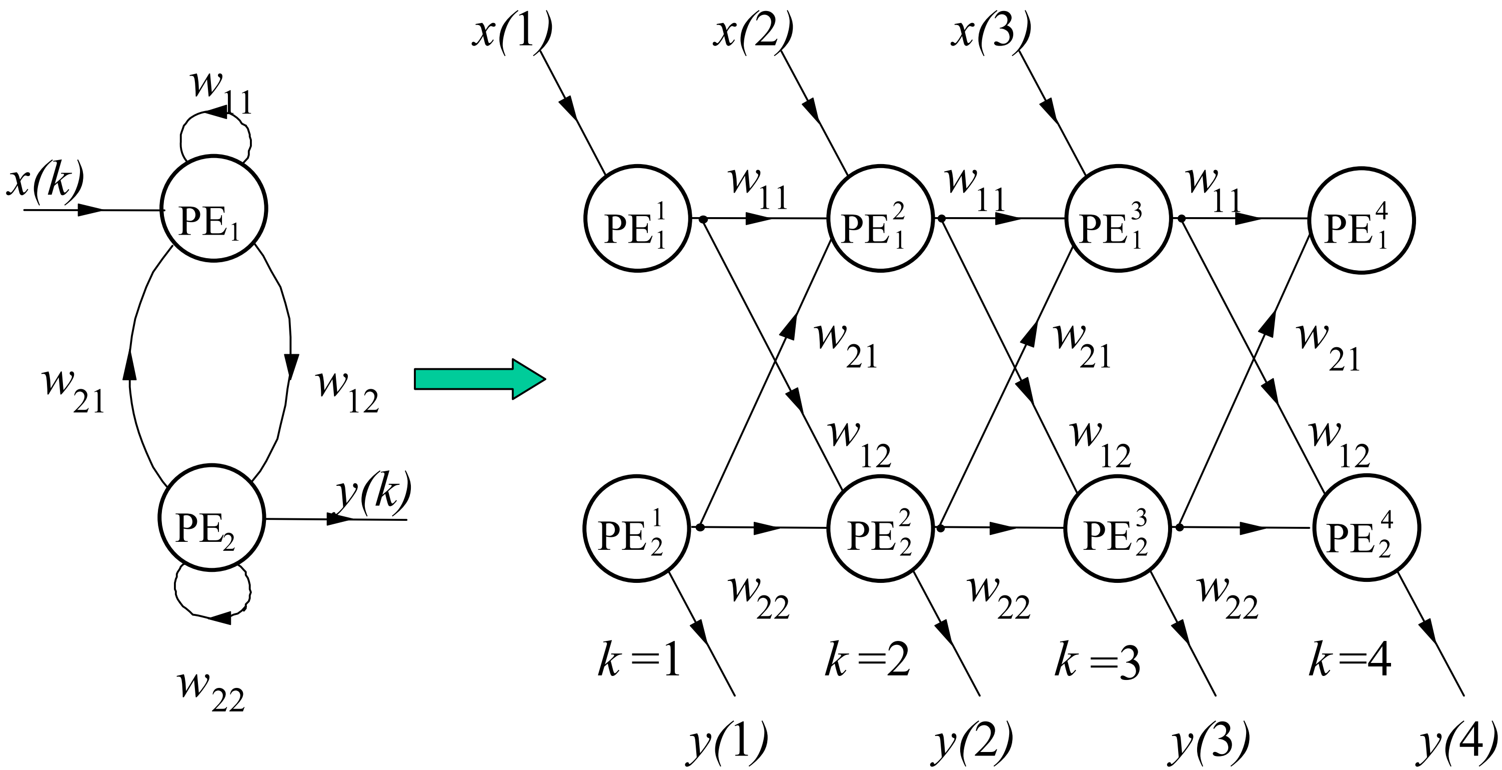
$$\frac{\partial y_l(k+1)}{\partial w_{ij}(k)} = f'(s_l(k))\left[\sum_{r\in \mathbf{B}} w_{lr}(k)\frac{\partial y_r(k)}{\partial w_{ij}(k)} + \delta_{li}u_j(k)\right]$$

$$w_{ij}(k+1) = w_{ij}(k) + 2\mu \sum_{l\in \mathbf{C}}\left\{\varepsilon_l(k) f'(s_l(k))\left[\sum_{r\in \mathbf{B}} w_{lr}(k)\frac{\partial y_r(k)}{\partial w_{ij}(k)} + \delta_{li}u_j(k)\right]\right\}$$

$$u_i(k) = \begin{cases} x_i(k) & \text{if } i\in \boldsymbol{A} \\ y_i(k) & \text{if } i\in \boldsymbol{B}\end{cases}\qquad \varepsilon_i^2(k) = \begin{cases}(d_i(k) - y_i(k))^2 & \text{if } i\in \mathbf{C}(k) \\ 0 & \text{otherwise}\end{cases}$$

Recurrent network

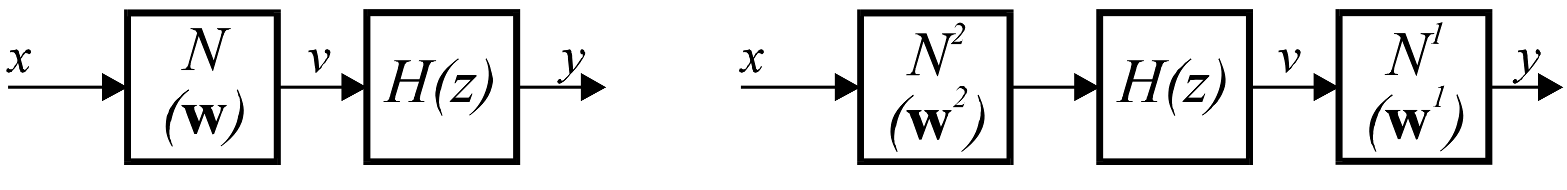
- Training: backpropagation through time (BPTT) unfolding in time



a.) b.)

Dynamic neural structures

- Combined linear dynamic and non-linear dynamic architectures

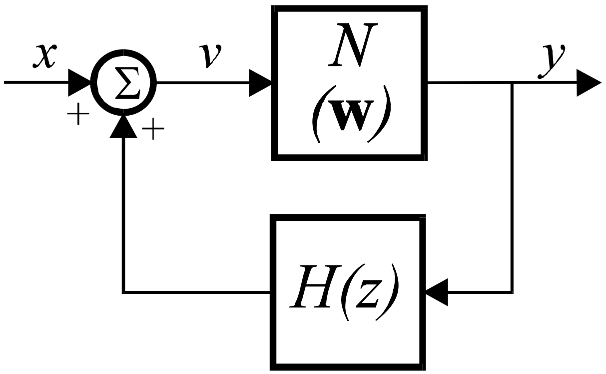


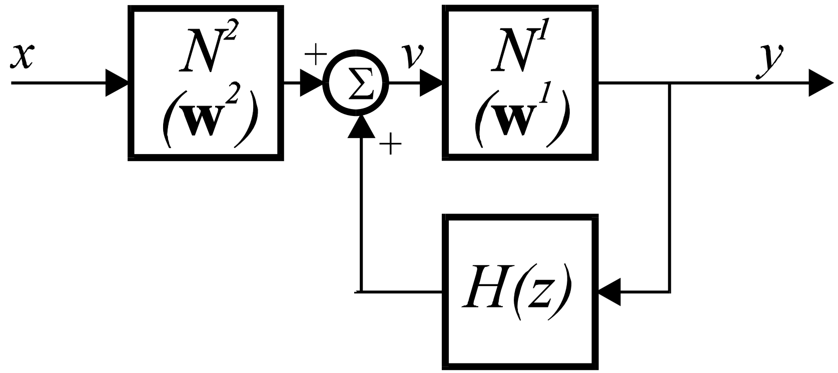
feed-forward architectures

$$\frac{\partial \boldsymbol{\varepsilon}(k)}{\partial w_{ij}} = \frac{\partial \mathbf{y}(k)}{\partial w_{ij}} = H(z)\frac{\partial \mathbf{v}}{\partial w_{ij}} \qquad \frac{\partial \varepsilon(k)}{\partial w_{ij}^{(2)}} = \frac{\partial y(k)}{\partial w_{ij}^{(2)}} = \sum_{l} \frac{\partial y(k)}{\partial v_l}\frac{\partial v_l}{\partial w_{ij}^{(2)}}$$

Dynamic backpropagation

Dynamic neural structures





a.) feedback arhitectures b.)

a.)

$$\frac{\partial \boldsymbol{\varepsilon}(k)}{\partial w_{ij}} = \frac{\partial \mathbf{y}(k)}{\partial w_{ij}} = \frac{\partial N(\mathbf{v})}{\partial \mathbf{v}} H(z) \frac{\partial \mathbf{y}(k)}{\partial w_{ij}} + \frac{\partial N(\mathbf{v})}{\partial w_{ij}}$$

b.)

$$\frac{\partial \varepsilon(k)}{\partial w_{ij}} = \frac{\partial y(k)}{\partial w_{ij}} = \frac{\partial N^1(v)}{\partial v}\left(\frac{\partial N^2(u)}{\partial w_{ij}} + H(z)\frac{\partial y(k)}{\partial w_{ij}}\right)$$

Dynamic system modeling

- **Example:** modeling of a discrete time system

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + f[u(k)]$$

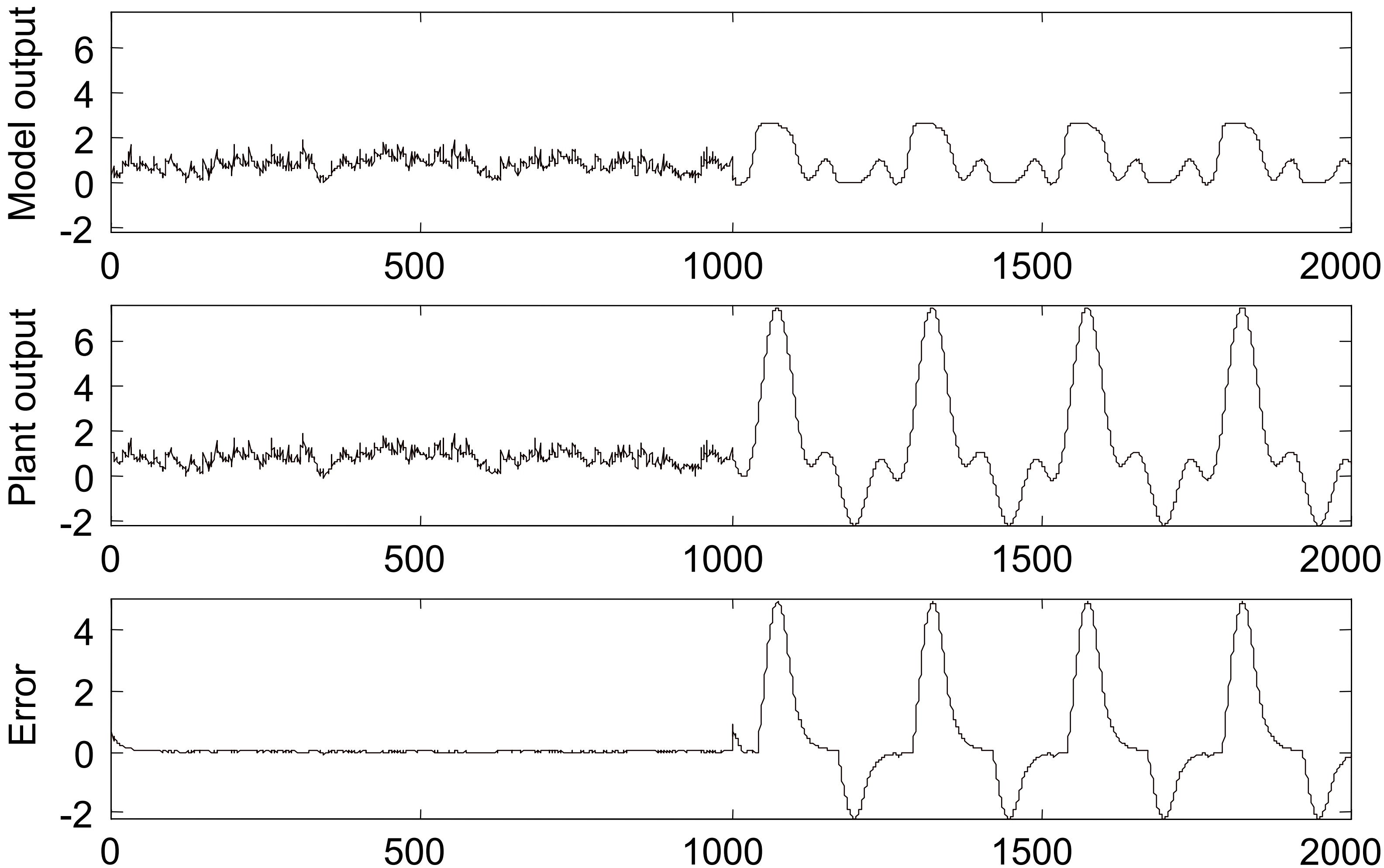
  - where

$$f(u) = u^3 + 0.3u^2 - 0.4u$$

  - Training signal: uniform, **random**, two different amplitudes

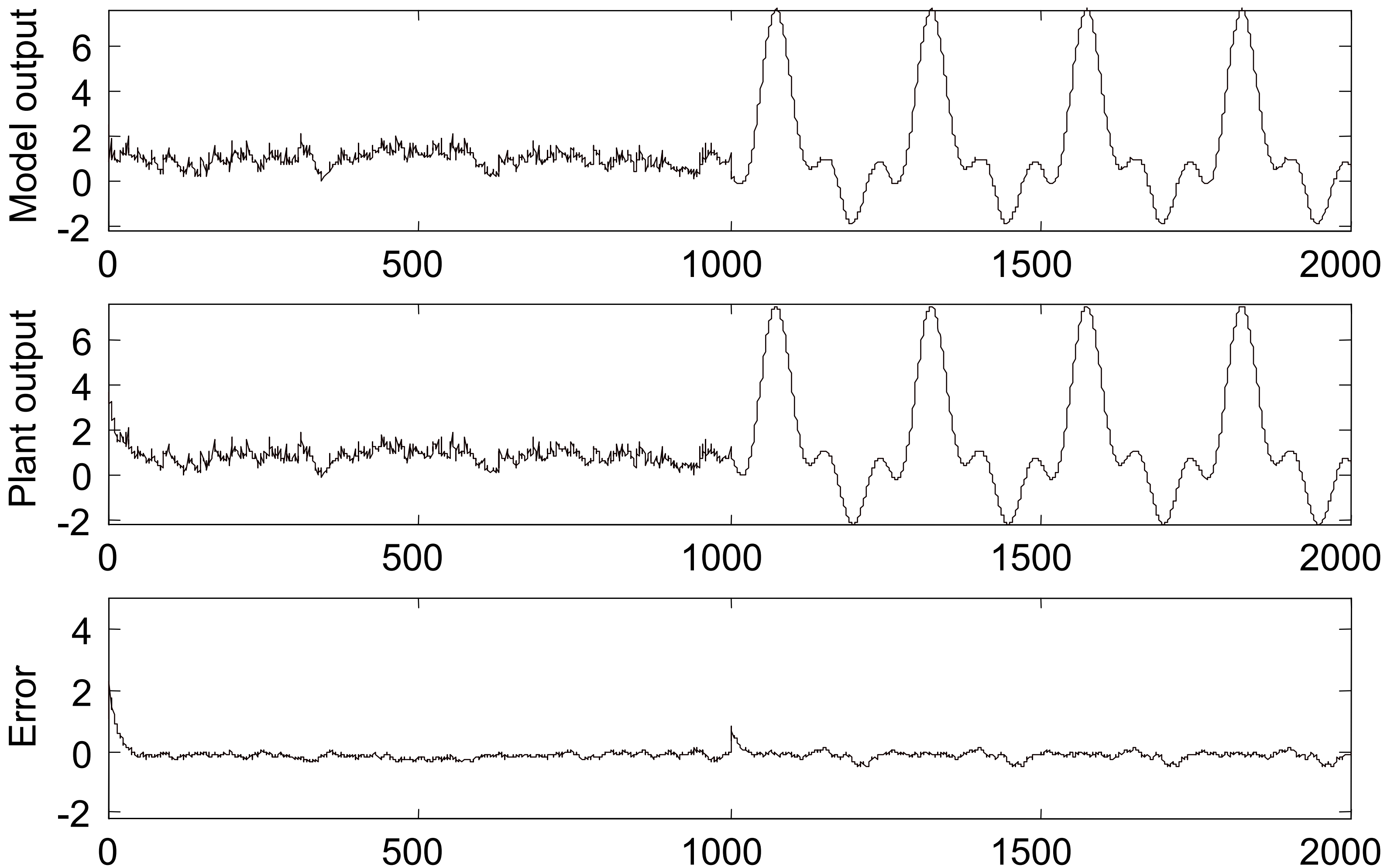
Dynamic system modeling

- The role of excitation: small excitation signal



Dynamic modeling

- The role of excitation: large excitation signal



References and further readings

Hassoun, M. H.: "Fundamentals of Artificial Neural Networks", MIT Press, Cambridge, MA. 1995.

Haykin, S.: "Neural Networks. A Comprehensive Foundation" Prentice Hall, N. J.1999.

Hertz, J. - Krogh, A. - Palmer, R. G. "Introduction to the Theory of Neural Computation", Addison-Wesley Publishing Co. 1991.

Widrow, B. - Stearns, S. D. "Adaptive Signal Processing", Prentice-Hall, Englewood Cliffs, N. J. 1985.

Narendra, K. S. - Pathasarathy, K. "Identification and Control of Dynamical Systems Using Neural Networks," IEEE Trans. Neural Networks, Vol. 1. 1990. pp. 4-27.

Narendra, K. S. - Pathasarathy, K. "Identification and Control of Dynamic Systems Using Neural Networks", IEEE Trans. on Neural Networks, Vol. 2. 1991. pp. 252-262.

Wan, E. A. "Temporal Backpropagation for FIR Neural Networks", Proc. of the 1990 IJCNN, Vol. I. pp. 575-580.

Weigend, A. S. - Gershenfeld, N. A. "Forecasting the Future and Under-standing the Past" Vol.15. Santa Fe Institute Studies in the Science of Complexity, Reading, MA. Addison-Wesley, 1994.

Williams, R. J. - Zipser, D. "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks", Neural Computation, Vol. 1. 1989. pp. 270-280.

Support vector machines

Outline

- Why we need a new approch
- Support vector machines
  - SVM for classification
  - SVM for regression
  - Other kernel machines
- Statistical learning theory
  - Validation (measure of quality: risk)
  - Vapnik-Chervonenkis dimension
  - Generalization

Support vector machines

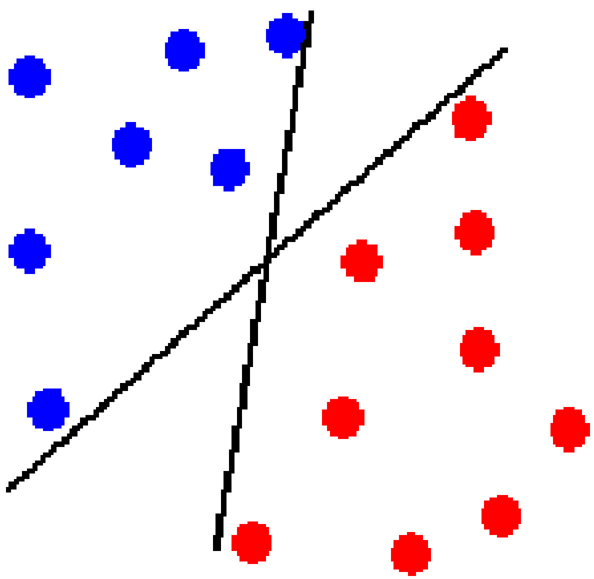
- A new approach:

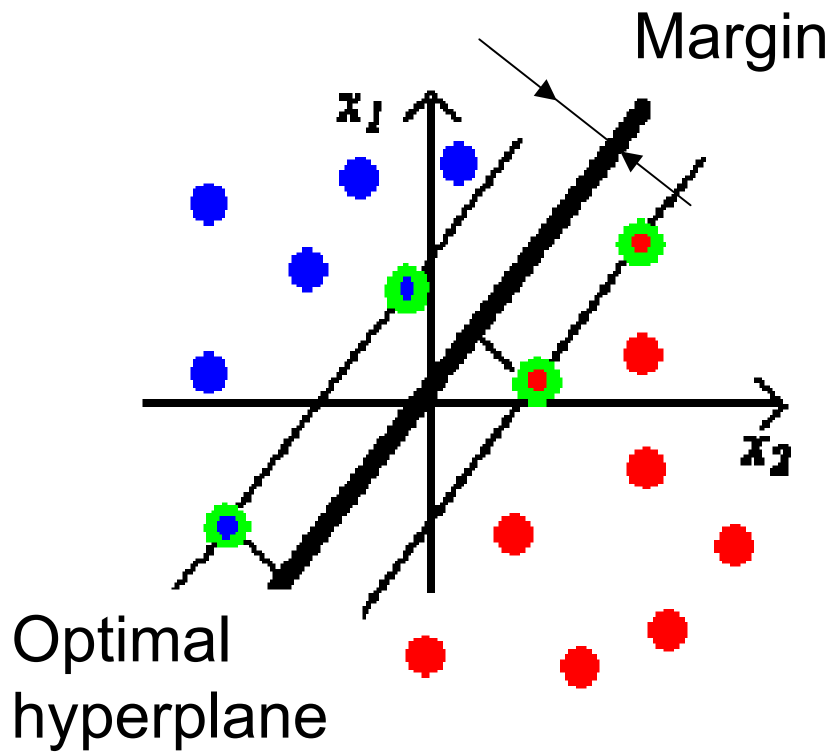
  gives answers for questions not solved using the classical approach

  - the size of the network
  - the generalization capability

Support vector machines

- Classification





Classical neural learning (perceptron)

Support Vector Machine

Support vector machines

- Linearly separable two-class problem

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^{P} \qquad \mathbf{x}_i \in X^1 \;\; y_i = +1, \;\; \mathbf{x}_i \in X^2 \quad y_i = -1$$

separating hyperpalne

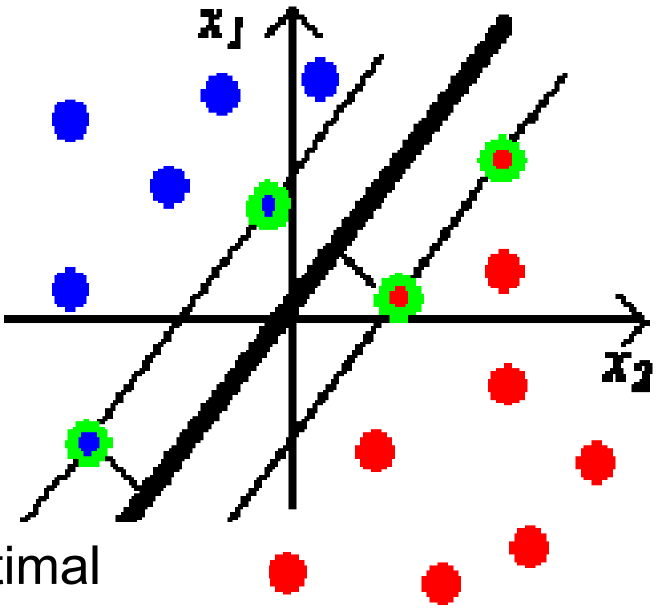
$$\mathbf{w}^T \mathbf{x} + b = 0$$

$$\mathbf{w}^T \mathbf{x}_i + b \geq +1, \text{ if } \; \mathbf{x}_i \in X^1$$

and

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1, \text{ if } \; \mathbf{x}_i \in X^2$$

$$(\mathbf{w}^T \mathbf{x}_i + b) y_i \geq 1, \qquad \forall i$$



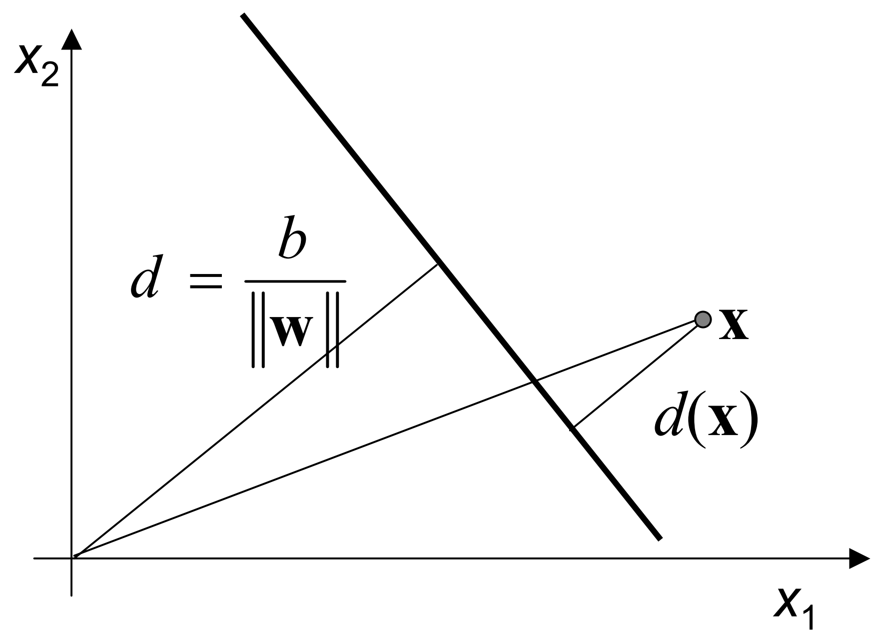
Optimal hyperplane

Support vector machines

- Geometric interpretation

  The formulation of optimality

$$d(\mathbf{w}, b, \mathbf{x}) = \frac{\left|\mathbf{w}^T\mathbf{x} + b\right|}{\|\mathbf{w}\|}$$



$$\rho(\mathbf{w}, b) = \min_{\{\mathbf{x}_i; y_i=1\}} d(\mathbf{w}, b, \mathbf{x}_i) + \min_{\{\mathbf{x}_i; y_i=-1\}} d(\mathbf{w}, b, \mathbf{x}_i) =$$

$$= \min_{\{\mathbf{x}_i; y_i=1\}} \frac{\left|\mathbf{w}^T\mathbf{x}_i + b\right|}{\|\mathbf{w}\|} + \min_{\{\mathbf{x}_i; y_i=-1\}} \frac{\left|\mathbf{w}^T\mathbf{x}_i + b\right|}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

Support vector machines

- Criterion function (primal problem)

$$\min \|\mathbf{w}\|^2 \rightarrow \max \text{margin}$$

$\Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$ with the conditions $(\mathbf{w}^T\mathbf{x}_i + b)y_i \geq 1, \quad \forall i$

a constrained optimization problem

(KKT conditions, saddle point)

$$J(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{P} \alpha_i \{[\mathbf{w}^T\mathbf{x}_i + b]y_i - 1\} \qquad \max_{\alpha} \min_{\mathbf{w}, b} J(\mathbf{w}, b, \alpha)$$

conditions

$$\frac{\partial J}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{P} \alpha_i \mathbf{x}_i y_i = 0 \qquad \frac{\partial J}{\partial b} = \sum_{i=1}^{P} \alpha_i y_i = 0 \qquad \mathbf{w} = \sum_{i=1}^{P} \alpha_i \mathbf{x}_i y_i \qquad \sum_{i=1}^{P} \alpha_i y_i = 0$$

Support vector machines

- Lagrange function (dual problem)

$$\max_{\alpha} W(\alpha) = \max_{\alpha} \{-\frac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P}\alpha_i\alpha_j y_i y_j (\mathbf{x}_i\mathbf{x}_j) + \sum_{i=1}^{P}\alpha_i\}$$

$$\sum_{i=1}^{P}\alpha_i y_i = 0 \qquad \alpha_i \geq 0 \quad \text{for all } i$$

support vectors

$$\mathbf{x}_i : \alpha_i > 0$$

optimal hyperplane

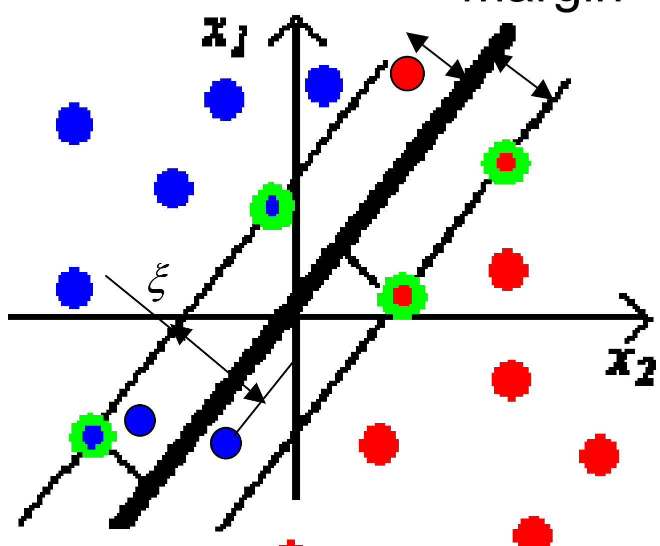
$$\mathbf{w}^* = \sum_{i=1}^{P}\alpha_i y_i \mathbf{x}_i$$

output

$$y(\mathbf{x}) = \mathbf{w}^{*T}\mathbf{x} + b = \sum_{i=1}^{P}\alpha_i y_i \left(\mathbf{x}_i^T\mathbf{x}\right) + b$$

Support vector machines

- Linearly nonseparable case
  (slightly nonlinear case)



separating hyperplane

$$y_i[\mathbf{w}^T\mathbf{x}_i + b] \geq 1 - \xi_i \qquad i = 1,\dots,P$$

criterion function (slack variable ξ)

$$\Phi(w,\xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{P}\xi_i$$

Lagrange function

$$J(\mathbf{w},b,\xi,\alpha,\beta) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{P}\xi_i - \sum\alpha_i\{y_i[\mathbf{w}^T\mathbf{x}_i + b] - 1 + \xi_i\} - \sum_{i=1}^{P}\beta_i\xi_i$$

$$0 \leq \alpha_i \leq C$$

support vectors $\mathbf{x}_i: \quad \alpha_i > 0$ optimal hyperplane $\mathbf{w}^* = \sum_{i=1}^{P}\alpha_i y_i \mathbf{x}_i$

Support vector machines

- Nonlinear separation, feature space
  - separating hypersurface (hyperplane in the φ space)

$$\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) + b = 0 \qquad \sum_{j=0}^{M} w_j \varphi_j(\mathbf{x}) = 0$$

  - decision surface

$$\sum_{i=1}^{P} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) = \sum_{i=1}^{P} \left(\alpha_i y_i \sum_{j=0}^{M} \varphi_j(\mathbf{x}_i) \varphi_j(\mathbf{x}) \right) = 0$$

  - kernel function (Mercer conditions)

$$K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\varphi}^T(\mathbf{x}_i) \boldsymbol{\varphi}(\mathbf{x}_j)$$

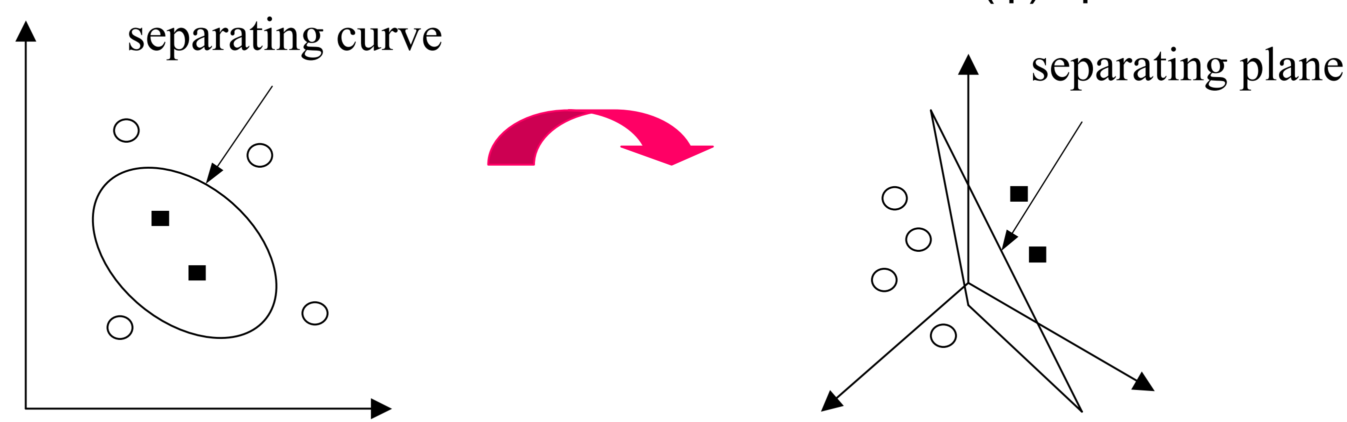
  - criterion function

$$W(\alpha) = \sum_{i=1}^{P} \alpha_i - \frac{1}{2} \sum_{i=1}^{P} \sum_{j=1}^{P} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i \mathbf{x}_j)$$

Feature space

input (x) space

(higher dimenzional)
feature (φ) space



$$y = \sum_{j=0}^{M} w_j \varphi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\varphi}(\mathbf{x})$$

$$\mathbf{w} = [w_0, w_1, \ldots, w_M]^T \qquad \boldsymbol{\varphi} = [\varphi_0(\mathbf{x}), \varphi_1(\mathbf{x}), \ldots, \varphi_M(\mathbf{x})]^T$$

Kernel space

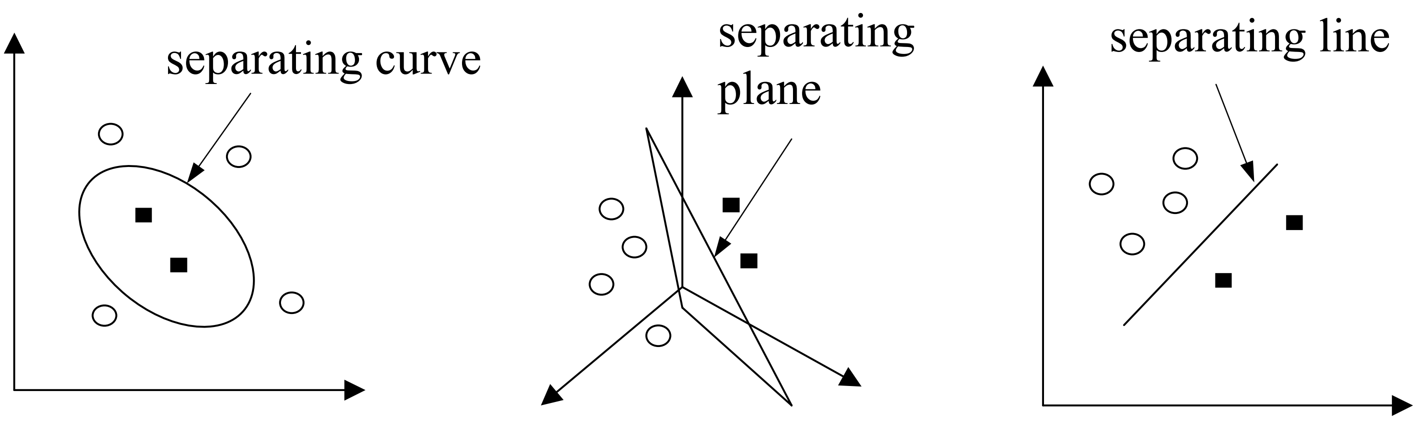
- Kernel trick

feature representation (nonlinear transformation) is not used

kernel function values (scalar values are used)

$$K(\mathbf{x}_i,\mathbf{x}_j)=\boldsymbol{\varphi}^T(\mathbf{x}_i)\boldsymbol{\varphi}(\mathbf{x}_j)$$

$$y(\mathbf{x})=\mathbf{w}^{*^T}\boldsymbol{\varphi}(\mathbf{x})+b=\sum_{i=1}^{P}\alpha_i y_i\left(\boldsymbol{\varphi}^{\mathrm{T}}(\mathbf{x}_i)\boldsymbol{\varphi}(\mathbf{x})\right)+b=\sum_{i=1}^{P}\alpha_i y_i K(\mathbf{x}_i\mathbf{x})+b$$



input space

feature space

kernel space

Support vector machines

- Examples of kernel functions
  - Polynomial

$$K(\mathbf{x},\mathbf{x}_i) = (\mathbf{x}^T\mathbf{x}_i + 1)^d \ , \quad d = 1,\ldots$$

  - RBF

$$K(\mathbf{x},\mathbf{x}_i) = \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}-\mathbf{x}_i\right\|^2\right)$$

  - MLP (only for certain β_o and β_1)

$$K(\mathbf{x},\mathbf{x}_i) = \tanh\left(\beta_0\mathbf{x}^T\mathbf{x}_i + \beta_1\right)$$

  - CMAC B-spline

Support vector machines

- Example: polynomial basis and kernel function
  - basis functions

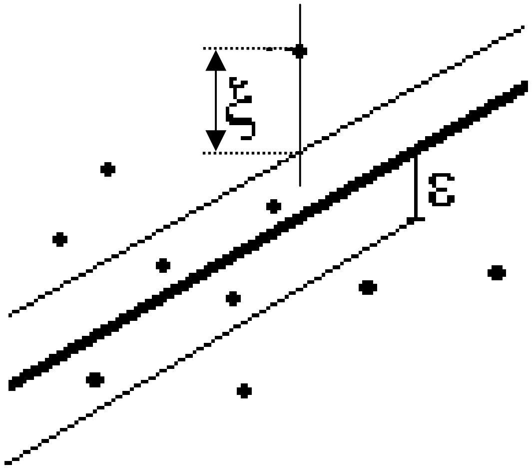
$$\varphi(\mathbf{x}_i) = [1,\ x_{i1}^2,\ \sqrt{2}x_{i1}x_{i2},\ x_{i2}^2,\ \sqrt{2}x_{i1},\ \sqrt{2}x_{i2}]^T$$

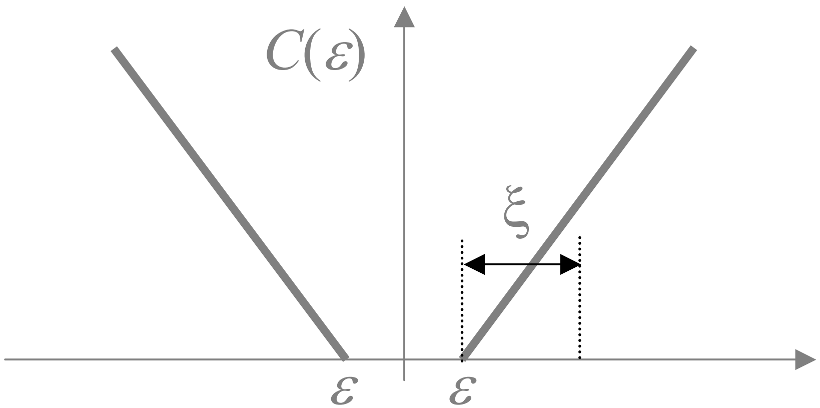
  - kernel function

$$K(\mathbf{x}, \mathbf{x}_i) = 1 + x_1^2 x_{i1}^2 + 2x_1\, x_2 x_{i1} x_{i2} + x_2^2 x_{i2}^2 + 2x_1\, x_{i1} + 2x_2 x_{i2}$$

SVR (regression)

ε-insensitive loss(criterion) function





$$C_\varepsilon(y, f(\mathbf{x},\alpha)) = \left| y - f(\mathbf{x},\alpha) \right|_\varepsilon = \begin{cases} 0 & \text{ha } \left| y - f(\mathbf{x},\alpha) \right| \le \varepsilon \\ \left| y - f(\mathbf{x},\alpha) \right| - \varepsilon & \text{otherwise} \end{cases}$$

SVR (regression)

$$f(\mathbf{x}) = \sum_{j=0}^{M} w_j \varphi_j(\mathbf{x})$$

Constraints:

$$y_i - \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) \leq \varepsilon + \xi_i,$$

$$\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) - y_i \leq \varepsilon + \xi'_i,$$

$$i = 1,2,\ldots,\ P$$

$$\xi_i \geq 0,$$

$$\xi'_i \geq 0,$$

Minimize:

$$\Phi(\mathbf{w}, \xi, \xi') = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\left(\sum_{i=1}^{P}(\xi_i + \xi'_i)\right)$$

SVR (regression)

Lagrange function

$$J\left(\mathbf{w},\xi,\xi',\alpha,\alpha',\gamma,\gamma'\right)=C\sum_{i=1}^{P}(\xi_i+\xi_i')+\frac{1}{2}\mathbf{w}^T\mathbf{w}-\sum_{i=1}^{P}\alpha_i\left[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i)-y_i+\varepsilon+\xi_i\right]-$$

$$-\sum_{i=1}^{P}\alpha_i'\left[y_i-\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i)+\varepsilon+\xi_i'\right]-\sum_{i=1}^{P}(\gamma_i\xi_i+\gamma_i'\xi')$$

$$\gamma_i=C-\alpha_i \qquad \gamma_i'=C-\alpha_i'$$

SVR (regression)

- Dual problem

$$W(\alpha_i\alpha_i') = \sum_{i=1}^{P} y_i(\alpha_i - \alpha_i') - \varepsilon\sum_{i=1}^{P}(\alpha_i + \alpha_i') - \frac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P}(\alpha_i - \alpha_i')(\alpha_j - \alpha_j')K(\mathbf{x}_i, \mathbf{x}_j)$$

constraints

support vectors

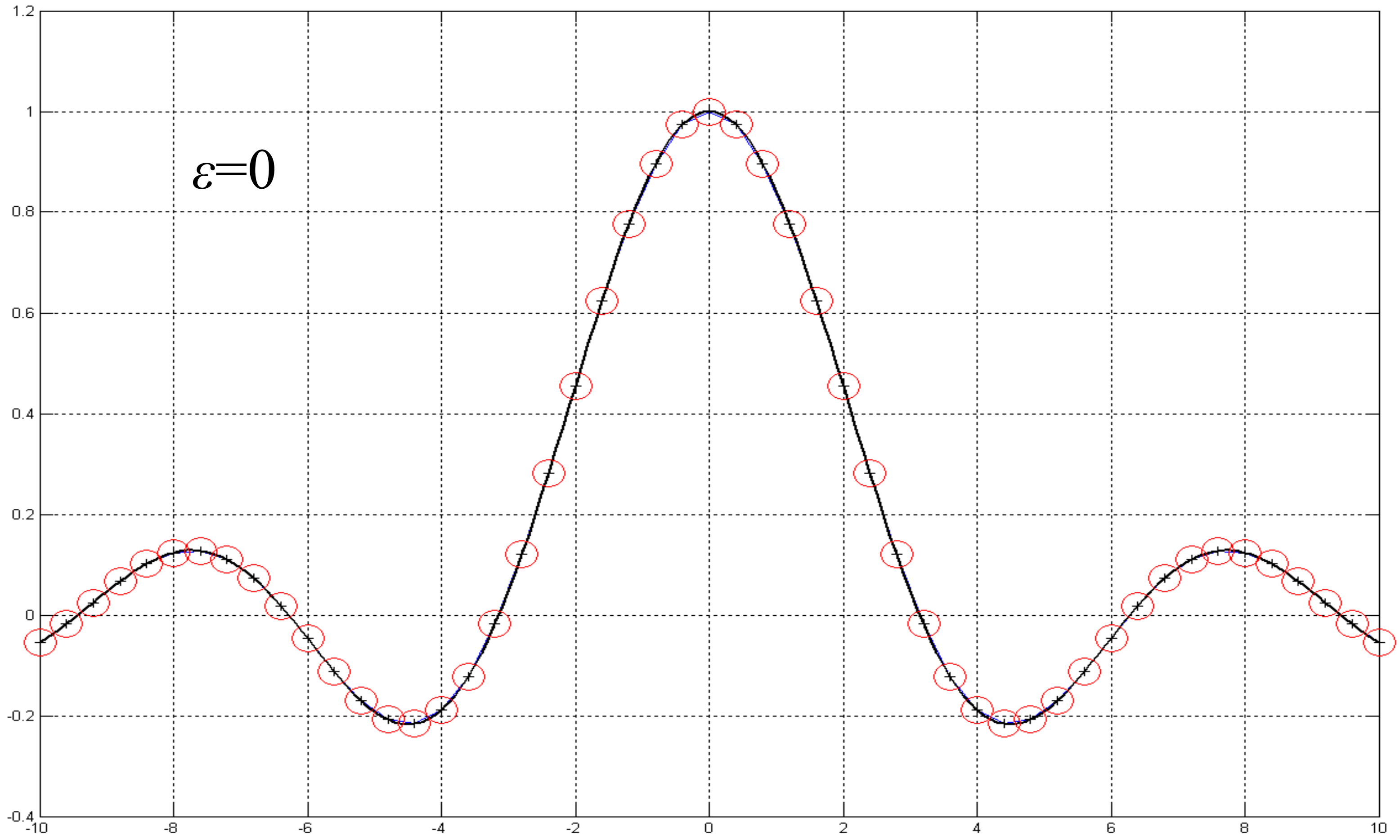
$$\sum_{i=1}^{P}(\alpha_i - \alpha_i') = 0 \qquad 0 \le \alpha_i \le C, \qquad 0 \le \alpha_i' \le C, \qquad \mathbf{x}_i : \alpha_i \ne \alpha_i'$$

solution

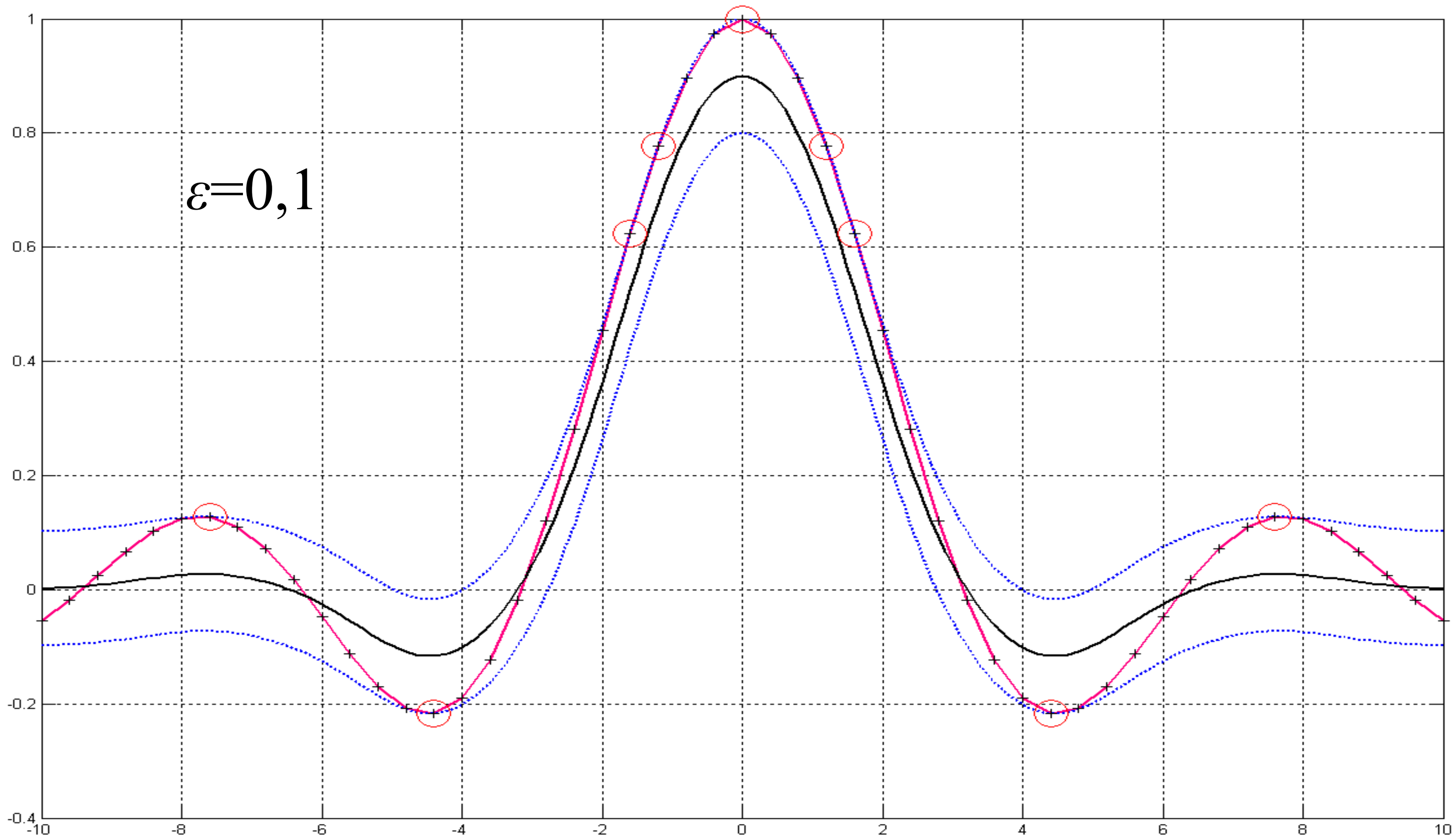
$$\mathbf{w}^* = \sum_{i=1}^{P}(\alpha_i - \alpha_i')\,\boldsymbol{\varphi}(\mathbf{x}_i)$$

$$y(\mathbf{x}) = \mathbf{w}^{*T}\boldsymbol{\varphi}(\mathbf{x}) = \sum_{i=1}^{P}(\alpha_i - \alpha_i')\left(\boldsymbol{\varphi}^{\mathrm{T}}(\mathbf{x}_i)\boldsymbol{\varphi}(\mathbf{x})\right) = \sum_{i=1}^{P}(\alpha_i - \alpha_i')K(\mathbf{x}_i\mathbf{x})$$

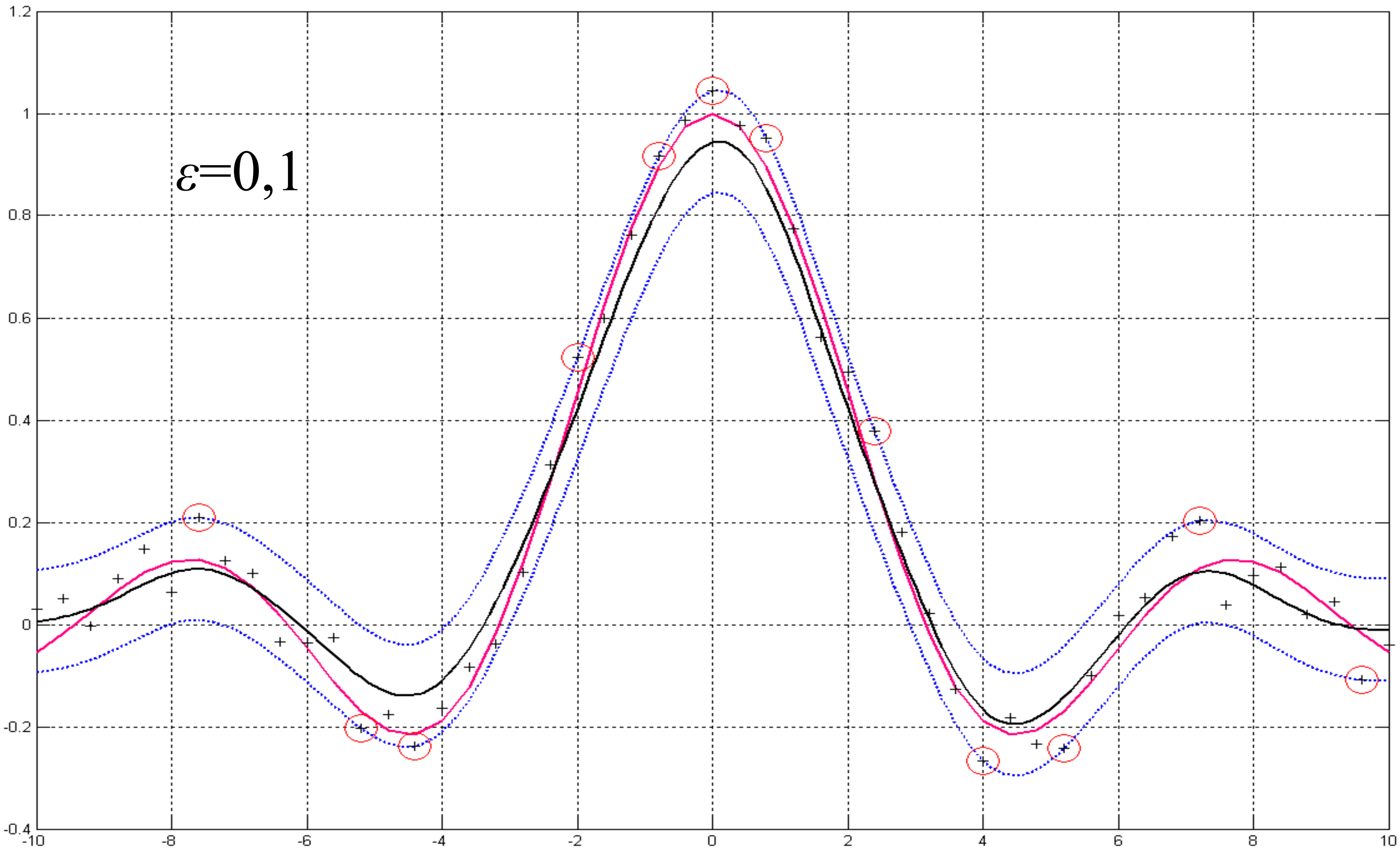
SVR (regression example)



SVR (regression example)



SVR (regression)



Support vector machines

- Main advantages
  - automatic model complexity (network size)
  - relevant training data point selection
  - allows tolerance (ε)
  - high-dimensional feature space representation is not used directly (kernel trick)
  - upper limit of the generalization error (see soon)
- Main difficulties
  - quadratic programming to solve dual problem
  - hyperparameter (C, ε, σ) selection
  - batch processing (there are on-line versions too)

SVM versions

- Classical Vapnik's SVM (drawbacks)
- LS-SVM

classification

$$\Phi(\mathbf{w},\xi,\xi')=\frac{1}{2}\mathbf{w}^T\mathbf{w}+C\left(\sum_{i=1}^{P}e_i^2\right)$$

equality constraints

$$y_i[\mathbf{w}^T\mathbf{x}_i+b]=1-e_i \qquad i=1,\dots,P$$

regression

$$\Phi(\mathbf{w},\xi,\xi')=\frac{1}{2}\mathbf{w}^T\mathbf{w}+C\left(\sum_{i=1}^{P}e_i^2\right)$$

$$y_i=\mathbf{w}^T\varphi(\mathbf{x}_i)+b+e_i \qquad i=1,\dots,P$$

no quadratic optimization to be solved : a linear set of equations

- Ridge regression

similar to LS-SVM

LS-SVM

Lagrange equation

$$L(\mathbf{w},b,\mathbf{e};\boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{2}\gamma\sum_{k=1}^{P} e_k^2 - \sum_{k=1}^{P}\alpha_k\left\{\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_k) + b + e_k - y_k\right\}$$

The results

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{0} \quad \rightarrow \quad \mathbf{w} = \sum_{k=1}^{P}\alpha_k\boldsymbol{\varphi}(\mathbf{x}_k)$$

$$\frac{\partial L}{\partial b} = 0 \quad \rightarrow \quad \sum_{k=1}^{P}\alpha_k = 0$$

$$\frac{\partial L}{\partial e_k} = 0 \quad \rightarrow \quad \alpha_k = \gamma\, e_k \qquad k = 1,\ldots,P$$

$$\frac{\partial L}{\partial \alpha_k} = 0 \quad \rightarrow \quad \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_k) + b + e_k - y_k = 0 \quad k = 1,\ldots,P$$

LS-SVM

Linear equation

Regression

$$\begin{bmatrix} 0 & \vec{\mathbf{1}}^T \\ \vec{\mathbf{1}} & \boldsymbol{\Omega}+\gamma^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}$$

Clasification

$$\begin{bmatrix} 0 & \mathbf{y}^T \\ \mathbf{y} & \boldsymbol{\Omega}+\gamma^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \vec{\mathbf{1}} \end{bmatrix}$$

where

$$\boldsymbol{\Omega}_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\varphi}^T(\mathbf{x}_i)\boldsymbol{\varphi}(\mathbf{x}_j)$$

$$\boldsymbol{\Omega}_{i,j} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

the response of the network

$$y(\mathbf{x}) = \sum_{k=1}^{N} \alpha_k K(\mathbf{x}, \mathbf{x}_k) + b$$

$$y(\mathbf{x}) = \sum_{k=1}^{N} \alpha_k y_k K(\mathbf{x}, \mathbf{x}_k) + b$$

Main features of LS-SVM and ridge regression

- Benefits
  - Easy to solve (no quadratic programming , only a linear equation set)
  - On-line, adaptive version (important in system identification)
- Drawbacks
  - Not sprase solution, all training points are used („there are no „support vectors”)
  - No „tolerance parameter” (ε)
  - No proved upper limit of the generalization error
  - Large kernel matrix if many training points are available

Improved LS Kernel machines

- There are sparse versions of the LS-SVM
  - The training points are ranked and only the most important ones are used (iterative solution)
  - The kernel matrix can be reduced (a tolerance parameter is introduced again)
  - Detailes: see the references
- Additional contraints can be used for special applications (see e.g. regularized kernel CMAC)

Kernel CMAC (an example)

- Goal
  - General goal:

    to show that additional constraints can be used in the framework of LS-SVM

    here: the additional constraint is a weight-smoothing term
  - Special goal:

    to show that kernel approach can be used for improving the modelling capability of the CMAC

General goal

- Introducing new constraints
- General LS-SVM problem

  Criterion function: two terms

  weight minimization term + error term

  Lagrange function

  criterion function+ Lagrange term

  Extension

  adding new constraint to the criterion function

  Extended Lagrange function

  new criterion function (with the new constraint) + Lagrange term

Special goal: improving the capability of the CMAC

- Difficulties with multivariate CMAC:
  - too many basis functions (too large weight memory)
  - poor **modelling** and **generalization** capability
- Improved **generalization**: regularization
- Improved **modelling** capability:
  - more basis functions:

    difficulties with the implementation

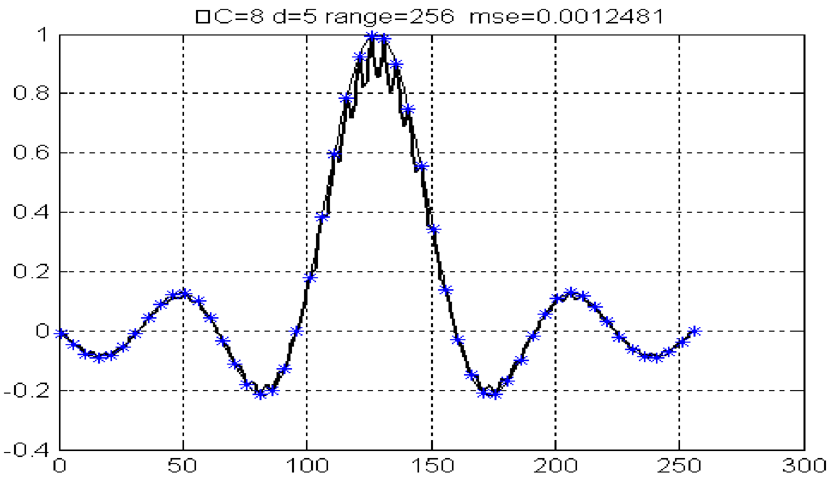
    kernel trick, kernel CMAC
- Improved modelling and generalization capability
  - regularized kernel CMAC

Regularized CMAC

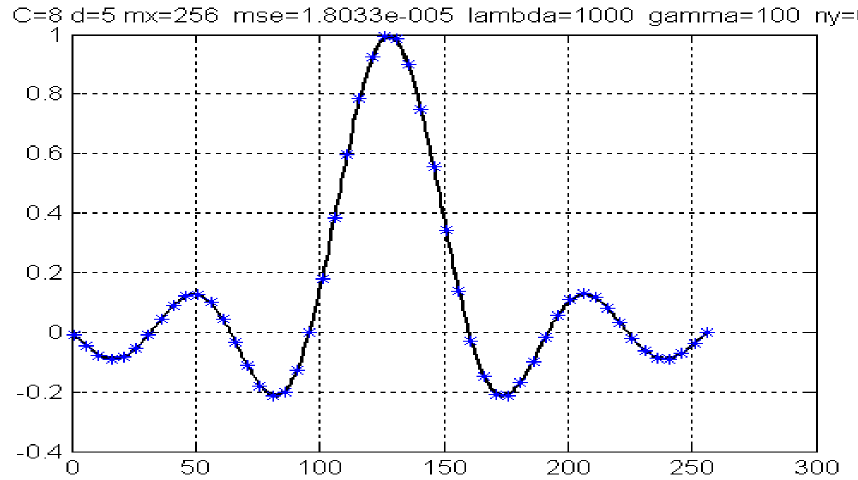
- Regularized criterion function (weight smoothing)

$$J(k)=\frac{1}{2}\left(y_d(k)-y(k)\right)^2+\frac{\lambda}{2}\left(\frac{y_d(k)}{C}-w_i(k)\right)^2$$

$$w_i(k+1)=w_i(k)+\mu(k)e(k)+\lambda\left(\frac{y_d(k)}{C}-w_i(k)\right)$$



without



with

regularization

Kernel CMAC

- Classical Albus CMAC: analytical solution

$$y_{d_k} = \mathbf{w}^T \mathbf{a}_k \qquad k = 1,\dots,P \qquad \mathbf{y}_d = \mathbf{A}\mathbf{w}$$

$$\mathbf{w}^* = \mathbf{A}^\dagger \mathbf{y}_d \qquad \mathbf{A}^\dagger = \mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1} \qquad y(\mathbf{x}) = \mathbf{a}^T(\mathbf{x})\mathbf{w}^* = \mathbf{a}^T(\mathbf{x})\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{y}_d$$

- Kernel version

criterion function (LS)

$$\min_{\mathbf{w}} J(\mathbf{w},\mathbf{e}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{\gamma}{2}\sum_{k=1}^{P} e_k^2$$

constraint

$$y_{d_k} = \mathbf{w}^T \mathbf{a}_k + e_k$$

Lagrangian

$$L(\mathbf{w},\mathbf{e},\boldsymbol{\alpha}) = J(\mathbf{w},\mathbf{e}) - \sum_{k=1}^{P} \alpha_k \left(\mathbf{w}^T \mathbf{a}_k + e_k - y_{d_k}\right)$$

Kernel CMAC (ridge regression)

- Using the derivatives the resulted equations

$$\begin{cases} \dfrac{\partial L(\mathbf{w},\mathbf{e},\boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{0} \;\rightarrow\; \mathbf{w} = \sum_{k=1}^{P} \alpha_k \mathbf{a}_k \\ \dfrac{\partial L(\mathbf{w},\mathbf{e},\boldsymbol{\alpha})}{\partial e_k} = 0 \;\rightarrow\; \alpha_k = \gamma e_k \qquad k = 1,\ldots,P \\ \dfrac{\partial L(\mathbf{w},\mathbf{e},\boldsymbol{\alpha})}{\partial \alpha_k} = 0 \;\rightarrow\; \mathbf{w}^T \mathbf{a}(\mathbf{x}_k) + e_k - y_{d_k} = 0 \qquad k = 1,\ldots,P \end{cases}$$

$$\left[\mathbf{K} + \frac{1}{\gamma}\mathbf{I}\right]\boldsymbol{\alpha} = \mathbf{y}_d \qquad \mathbf{K} = \mathbf{A}\mathbf{A}^T$$

$$y(\mathbf{x}) = \mathbf{a}^T(\mathbf{x})\mathbf{w} = \mathbf{a}^T(\mathbf{x}) \sum_{k=1}^{P} \alpha_k \mathbf{a}_k = \sum_{i=1}^{P} \alpha_k K(\mathbf{x}, \mathbf{x}_k) = \mathbf{K}^T(\mathbf{x})\boldsymbol{\alpha}$$

Kernel CMAC with regularization

Extended criterion function:

$$\min_{\mathbf{w}} J(\mathbf{w},\mathbf{e}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{\gamma}{2}\sum_{k=1}^{P} e_k^2 + \frac{\lambda}{2}\sum_{k=1}^{P}\sum_{i}\left(\frac{y_{d_k}}{C} - w_k(i)\right)^2$$

Lagrange function

$$L(\mathbf{w},\mathbf{e},\alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{\gamma}{2}\sum_{k=1}^{P} e_k^2 + \frac{\lambda}{2}\sum_{k=1}^{P}\sum_{i}\left(\frac{y_{d_k}}{C} - w_k(i)\right)^2 - \sum_{k=1}^{P}\alpha_k\left(\mathbf{w}^T\mathbf{a}_k + e_k - y_{d_k}\right)$$

$$L(\mathbf{w},\mathbf{e},\alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{\gamma}{2}\sum_{k=1}^{P} e_k^2 - \sum_{k=1}^{P}\alpha_k\left(\mathbf{a}_k^T diag(\mathbf{a}_k)\mathbf{w} + e_k - y_{d_k}\right)$$

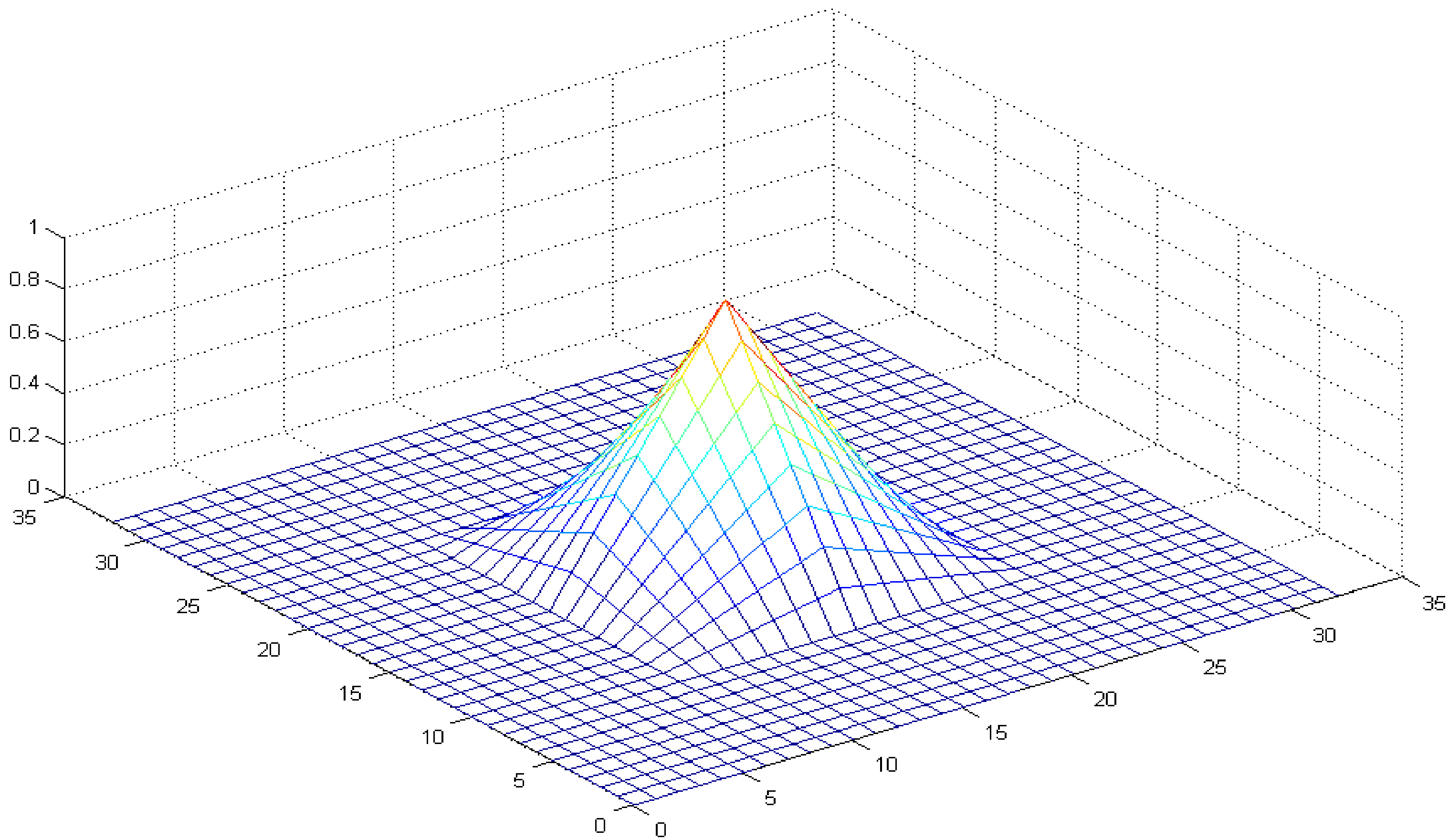
$$+\frac{\lambda}{2}\sum_{k=1}^{P}\frac{y_{d_k}^2}{C} - \lambda\sum_{k=1}^{P}\frac{d_k}{C}\mathbf{a}_k^T diag(\mathbf{a}_k)\mathbf{w} + \frac{\lambda}{2}\sum_{k=1}^{P}\mathbf{w}^T diag(\mathbf{a}_k)\mathbf{w}$$

Output

$$y(\mathbf{x}) = \mathbf{a}^T(\mathbf{x})\left(\mathbf{I} + \lambda\mathbf{D}\right)^{-1}\mathbf{A}^T\left[\boldsymbol{\alpha} + \frac{\lambda}{C}\mathbf{y}_d\right] \qquad \text{where} \qquad \mathbf{D} = \sum_{k=1}^{P} diag(\mathbf{a}_k)$$

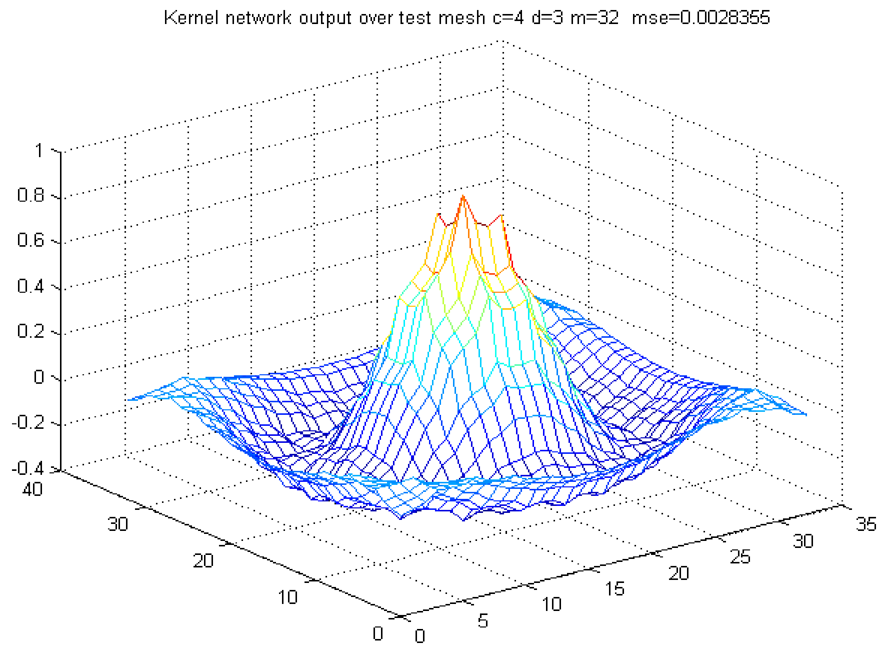
Kernel CMAC with regularization

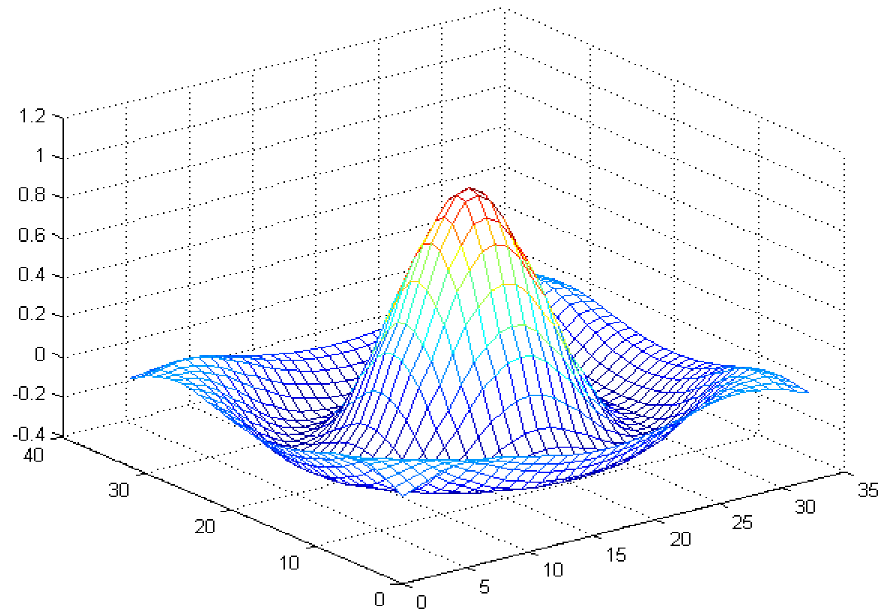
- Kernel function for two-dimensional kernel CMAC



Regularized Kernel CMAC (example)

- 2D sinc





without

with

regularization

References and further readings

Haykin, S.: "Neural Networks. A Comprehensive Foundation" Prentice Hall, N. J.1999.

Saunders, C.- Gammerman, A. and Vovk, V. "Ridge Regression Learning Algorithm in Dual Variables. Machine Learning", *Proc. of the Fifteenth International Conference on Machine Learning*, pp. 515-521, 1998.

Schölkopf, B. and Smola, P. "Learning with Kernels. Support Vector Machines, Regularization, Optimization and Beyond" MIT Press, Cambridge, MA, 2002.

Suykens, J.A.K., Van Gestel, T, De Brabanter, J., De Moor, B. and Vandewalle, J. "Least Squares Support Vector Machines", World Scientific, Singapore, 2002.

Vapnik, V. "Statistical Learning Theory", Wiley, New York, 1995.

Horváth, G. "CMAC: Reconsidering an Old Neural Network" *Proc. of the Intelligent Control Systems and Signal Processing, ICONS 2003*, Faro, Portugal. pp. 173-178, 2003.

Horváth, G. "Kernel CMAC with Improved Capability" *Proc. of the International Joint Conference on Neural Networks, IJCNN'2004*, Budapest, Hungary. 2004.

Lane, S.H. - Handelman, D.A. and Gelfand, J.J "Theory and Development of Higher-Order CMAC Neural Networks", *IEEE Control Systems*, Vol. Apr. pp. 23-30, 1992.

Szabó, T. and Horváth, G. "Improving the Generalization Capability of the Binary CMAC" *Proc. of the International Joint Conference on Neural Networks, IJCNN'2000.* Como, Italy, Vol. 3, pp. 85-90, 2000.

Statistical learning theory

- Main question: how can the quality of a learning machine be estimated
- Generalization measure based on the empirical risk (error).
- Empirical risk: the error determined in the training points

Statistical learning theory

- Goal: to find a solution that minimizes the risk

$$R(\mathbf{w}) = \int l(\mathbf{x},\mathbf{w})\ p(\mathbf{x},y)d\mathbf{x}dy = \int \left[y - f(\mathbf{x},\mathbf{w})\right]^2 p(\mathbf{x},y)d\mathbf{x}dy \qquad R(\mathbf{w}^*|P)$$

- Difficulties: joint density function is unknown

Only the empirical risk can be determined

$$R_{\mathrm{emp}}(\mathbf{w}) = \frac{1}{P}\sum_{l=1}^{P}\left[y_i - f(\mathbf{x}_l,\mathbf{w})\right]^2 \qquad R_{\mathrm{emp}}(\mathbf{w}^*|P)$$

optimal value

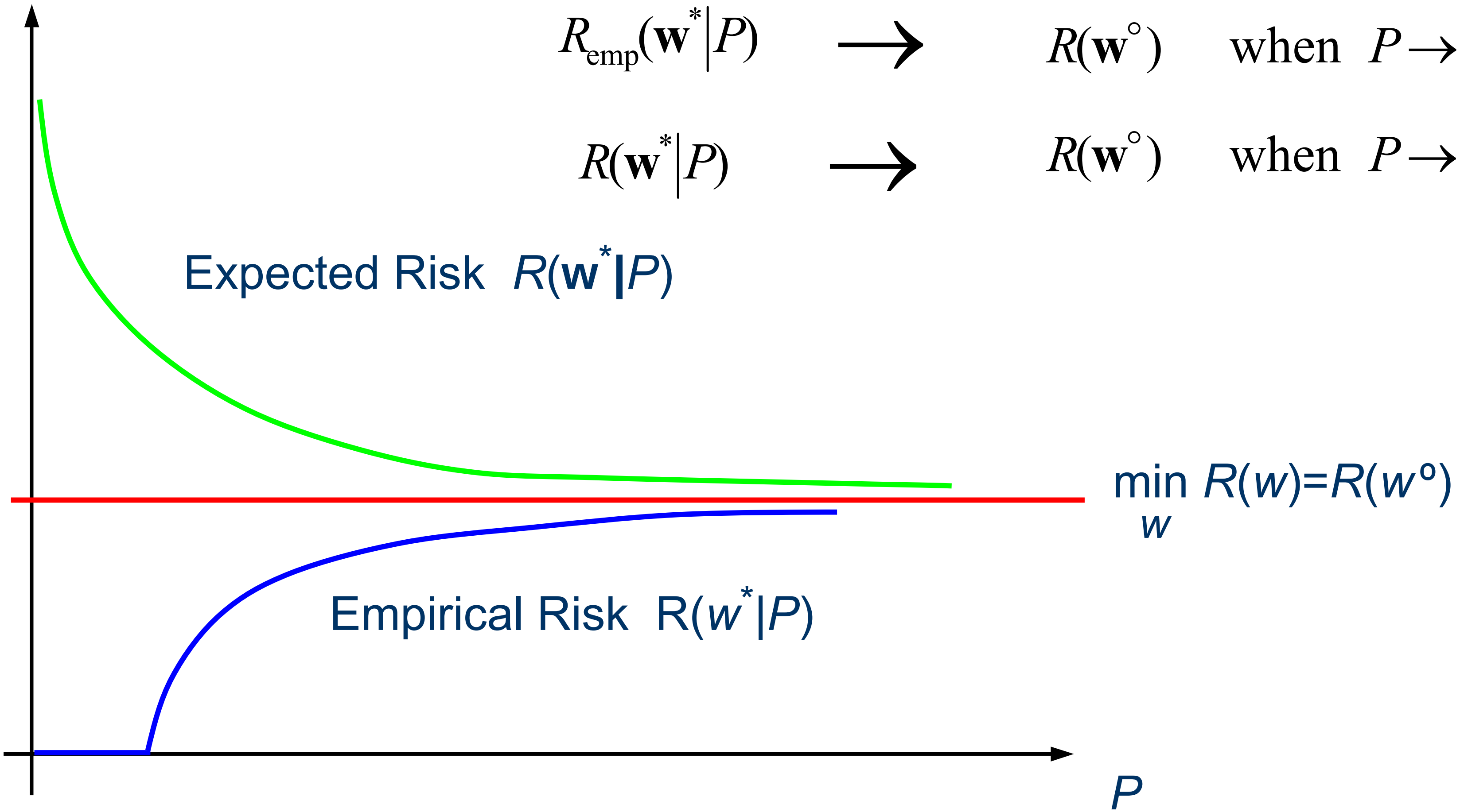
minimizing the empirical risk

Statistical learning theory: ERM principle

- Asymptotic consistency of empirical risk

$$R_{\text{emp}}(\mathbf{w}^*|P) \quad \longrightarrow \quad R(\mathbf{w}^\circ) \quad \text{when} \quad P \to \infty$$

$$R(\mathbf{w}^*|P) \quad \longrightarrow \quad R(\mathbf{w}^\circ) \quad \text{when} \quad P \to \infty$$



Statistical learning theory

- Condition of consistency of the ERM principle

  Necessarry and sufficient condition: *finite VC dimension*

  Also: this is a sufficient condition of *fast convergence*

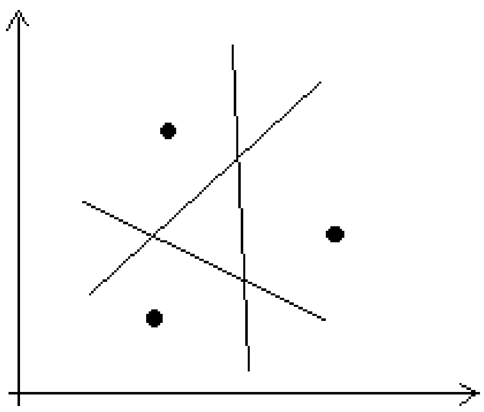
- VC (Vapnik-Chervonenkis) dimension:

  A set of function has VC dimension h if there exist h samples ***that can be shattered*** (can be separated into two classes in all possible ways: all 2^h possible ways) by this set of functions but there do not exist $h+1$ samples that can be shattered by the same set of functions.

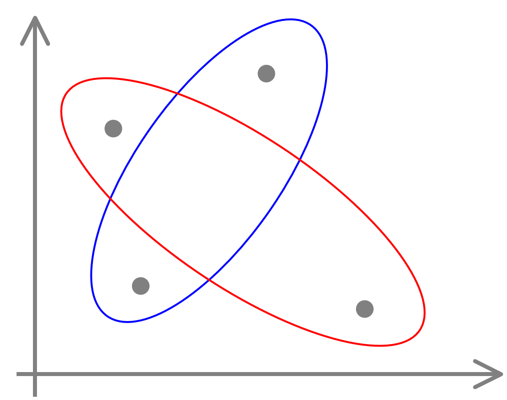
Model complexity, VC dimension

- VC dimension of a set of indicator functions
  - definition

    VC dimension is the ***maximum number*** of samples for which ***all possible binary labellings*** can be induced by a set of functions
  - illustration



linear separation



no linear separation

VC dimension

- Based on VC dimension *upper bounds of the risk* can be obtained
- Calculating the VC dimension
  - general case: rather difficult
    e.g for MLP VC-dimension can be infinite
  - special cases: e.g. linear function set
- VC dimension of a set of linear functions (linear separating task)

  $h = N + 1$ (N: input space dimension)

An important statement: *It can be proved that the VC dimension can be less than N +1*

Generalization error

- Bound on generalization
  - Classification: with probability of at least 1-η (confidence level; η is a given value within the additional term)

$$R(\mathbf{w}) \le R_{\text{emp}}(\mathbf{w}) + \text{additional term}(h)$$ (confidence interval)

$$h \le \min\left(R^2 / M^2, N\right) + 1;$$

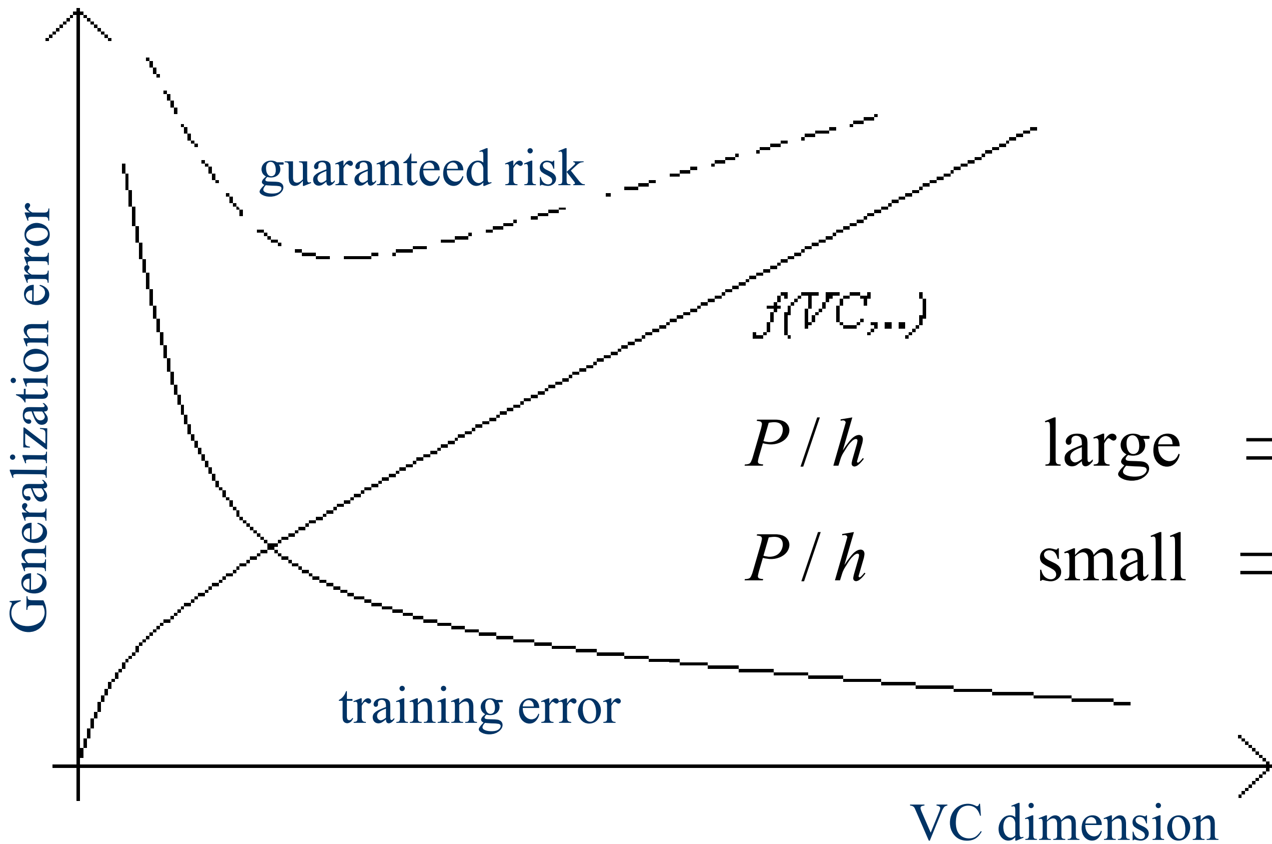
$$R = \text{Radius of a sphere containing all data points}$$

$$M = \frac{1}{\|\mathbf{w}\|} \text{ margin of classification}$$

  - regression

$$R(\mathbf{w}) \le \frac{R_{\text{emp}}(\mathbf{w})}{\left(1 - c\sqrt{\varepsilon(h)}\right)_+} \qquad \varepsilon = a_1 \frac{h\left[\log\left(a_2 N / h\right) + 1\right] - \log\left(\eta / 4\right)}{N}$$

Generalization error



$$P/h \quad \text{large} \quad \Rightarrow \quad R \sim R_{\text{emp}}$$

$$P/h \quad \text{small} \quad \Rightarrow \quad R >> R_{\text{emp}}$$

Tradeoff between the quality of approximation and the complexity of the approximating function

Structural risk minimization principle

- Good generalization: *both terms should be minimized*

  S set of approximating functions

  The elements of S, nested subset of S_k with finite VC dimension h_k

  $$S_1 \subset S_2 \subset \ldots \subset S_k \subset \ldots$$

  The ordering of complexity of the elements

  $$h_1 \leq h_2 \leq \ldots \leq h_k \leq \ldots$$

  Based on a priori information S is specified.

  For a given data set the optimal model estimation:

  - selection of an element of the set (model selection)
  - estimating the model from this subset (training the model)
  - there is an upper bound on the prediction risk with a given confidence level

Constructing a learning algorithm

- Structural risk minimization
  - Such S_k will be selected for which the guaranted risk is minimal
  - SRM principle suggests a *tradeoff between the quality of approximation and the complexity of the approximating function* (model selection problem)
  - Both terms are controlled:
    - the empirical risk with training
    - the complexity with the selection of S_k

$$R(\mathbf{w}_P^k) \leq R_{\text{emp}}(\mathbf{w}_P^k) + \text{additional term}(P / h_k)$$

confidence interval

SVM

- Support vector machines are such a learning machines that minimize the length of the weight vector

⇓

- They minimize the VC dimension. The upper bounds are valid for SVMs.
- For SVMs not only the structure (the size of the network) can be determined, but an estimate of its generalization error can be obtained.

References and further readings

Haykin, S.: "Neural Networks. A Comprehensive Foundation" Prentice Hall, N. J.1999.

Vapnik, V. "Statistical Learning Theory", Wiley, New York, 1998.

Cherkassky, V., Mulier, F. „Learning from Data, Concepts, Theory, and Methods”, John Wiley and Sons,1998.

Modular network architectures

Modular solution

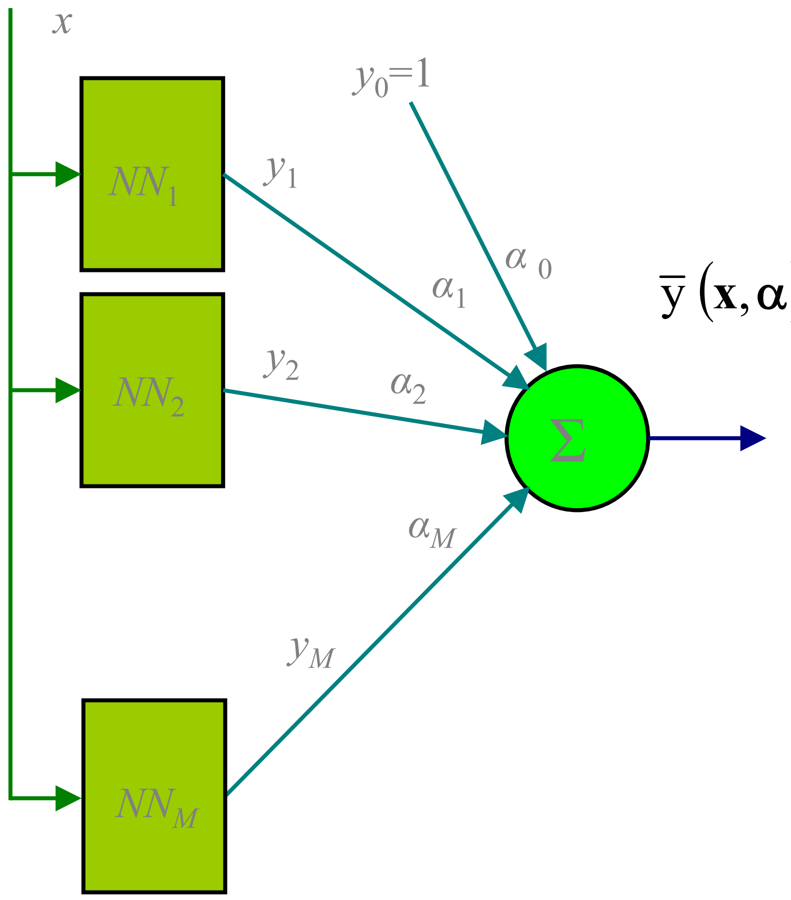
- A set of networks: competition/cooperation
  - all networks solve the same problem (competition/cooperation)
  - the whole problem is decomposed: the different networks solve different part of the whole problem (cooperation)
- Ensemble of networks
  - linear combination of networks
- Mixture of experts
  - using the same paradigm (e.g. neural networks)
  - using different paradigms (e.g. neural networks + symbolic systems, neural networks + fuzzy systems)

Cooperative networks

Ensemble of cooperating networks
(classification/regression)

- The motivation
  - Heuristic explanation
    - Different experts together can solve a problem better
    - Complementary knowledge
  - Mathematical justification
    - Accurate and diverse modules

Linear combination of networks



$$\bar{y}(\mathbf{x},\boldsymbol{\alpha}) = \sum_{j=0}^{M} \alpha_j y_j(\mathbf{x})$$

Ensemble of networks

- Mathematical justification
  - Ensemble output $$\bar{y}(\mathbf{x},\alpha)=\sum_{j=0}^{M}\alpha_j y_j(\mathbf{x})$$
  - Ambiguity (diversity) $$a_j(\mathbf{x})=\left[y_j(\mathbf{x})-\bar{y}(\mathbf{x},\alpha)\right]^2$$
  - Individual error $$\varepsilon_j(\mathbf{x})=\left[d(\mathbf{x})-y_j(\mathbf{x})\right]^2$$
  - Ensemble error $$\varepsilon(\mathrm{x})=\left[d(\mathbf{x})-\bar{y}(\mathbf{x},\alpha)\right]^2$$
  - Constraint $$\sum_{j=1}^{M}\alpha_j=1$$

Ensemble of networks

- Mathematical justification (cont'd)
  - Weighted error $\bar{\varepsilon}(\mathbf{x},\alpha)=\sum_{j=0}^{M}\alpha_j\varepsilon_j(\mathbf{x})$
  - Weighted diversity $\bar{a}(\mathbf{x},\alpha)=\sum_{j=0}^{M}\alpha_j a_j(\mathbf{x})$
  - Ensemble error $\varepsilon(\mathbf{x})=\left[d(\mathbf{x})-\bar{y}(\mathbf{x},\alpha)\right]^2 \quad =\bar{\varepsilon}(\mathbf{x},\alpha)-\bar{a}(\mathbf{x},\alpha)$
  - Averaging over the input distribution

$$E=\int_{\mathbf{x}}\varepsilon(\mathbf{x},\alpha)f(\mathbf{x})d\mathbf{x} \qquad \bar{E}=\int_{\mathbf{x}}\bar{\varepsilon}(\mathbf{x},\alpha)f(\mathbf{x})d\mathbf{x} \qquad \bar{A}=\int_{\mathbf{x}}\bar{a}(\mathbf{x},\alpha)f(\mathbf{x})d\mathbf{x}$$

$$E=\bar{E}-\bar{A}$$

Solution: Ensemble of accurate and diverse networks

Ensemble of networks

- How to get accurate and diverse networks
  - different structures: more than one network structure (e.g. MLP, RBF, CCN, etc.)
  - different size, different complexity networks (number of hidden units, number of layers, nonlinear function, etc.)
  - different learning strategies (BP, CG, random search,etc.) batch learning, sequential learning
  - different training algorithms, sample order, learning samples
  - different training parameters
  - different initial parameter values
  - different stopping criteria

Linear combination of networks

- Computation of optimal (fix) coefficients
  - $\alpha_k = \frac{1}{M}, \quad k = 1 \ldots M$ $\rightarrow$ simple average
  - $\alpha_k = 1, \; \alpha_j = 0, \;\; j \neq k$, k depends on the input

    for different input domains different network (alone) gives the output
  - optimal values using the constraint $\sum_{\kappa=1}^{M} \alpha_k = 1$
  - optimal values without any constraint

    Wiener-Hopf equation $\boldsymbol{\alpha}^*_{(1)} = \mathbf{R}_y^{-1}\mathbf{P}$

$$\mathbf{R}_y = \mathrm{E}\left[\bar{\mathbf{y}}(\mathbf{x})\,\bar{\mathbf{y}}(\mathbf{x})^T\right] \qquad \mathbf{P} = \mathrm{E}\left[\bar{\mathbf{y}}(\mathbf{x})\,d(\mathbf{x})\right]$$

References and further readings

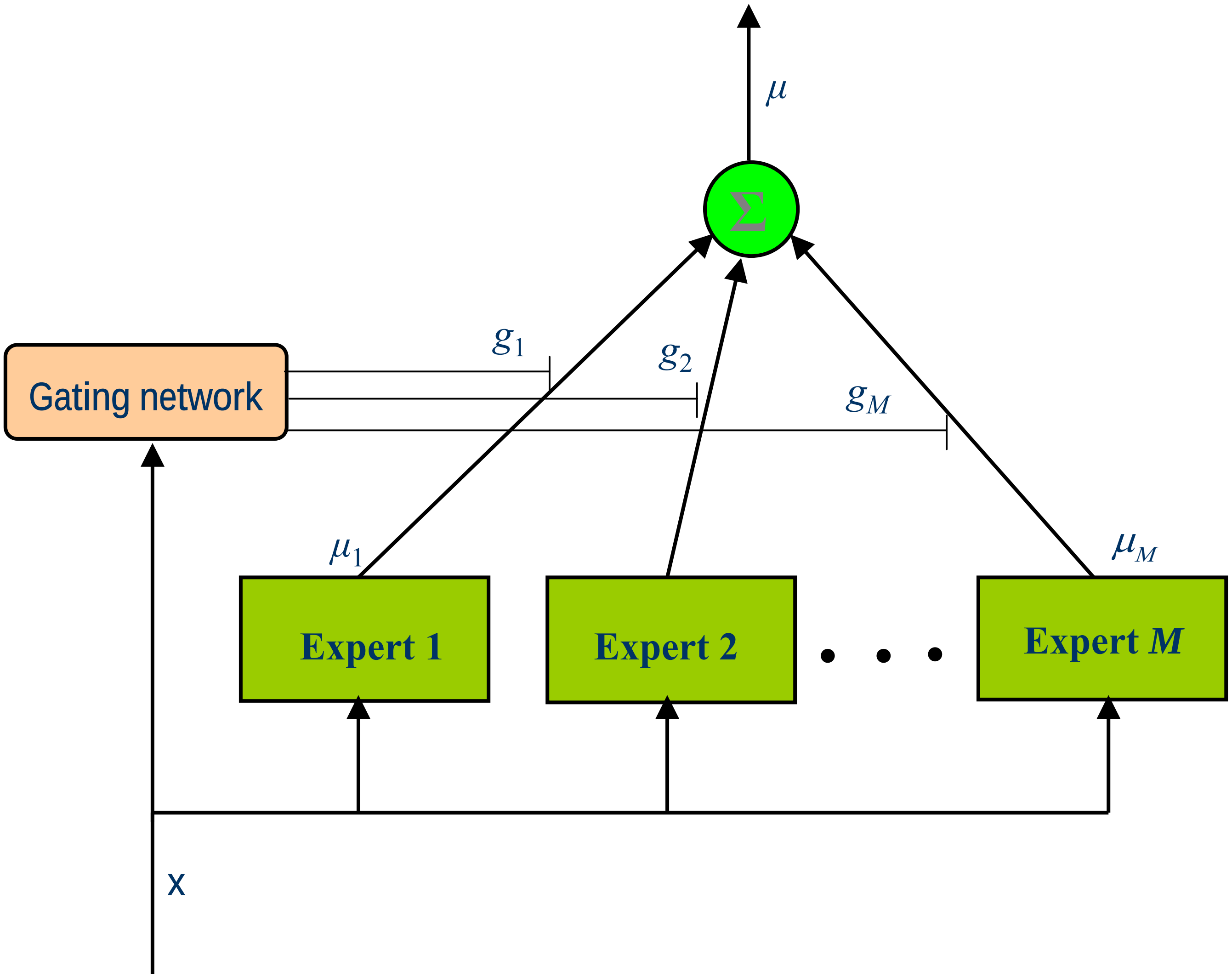
Haykin, S.: "Neural Networks. A Comprehensive Foundation" Prentice Hall, N. J.1999.

Hashem, S. “Optimal Linear Combination of Neural Networks” Neural Networks, Vol. 10. No. 4. pp. 599-614, 1997.

Krogh, A, Vedelsby, J.: “Neural Network Ensembles Cross Validation and Active Learning” In Tesauro, G, Touretzky, D, Leen, T. Advances in Neural Information Processing Systems, 7. Cambridge, MA. MIT Press pp. 231-238.

Gasser Auda and Mohamed Kamel: „Modular Neural Networks: A survey” Pattern Analysis andíMachine Intelligence Lab. Systems Design Engineering Department, University of Waterloo, Canada.

Mixture of Experts (MOE)



Mixture of Experts (MOE)

- The output is the weighted sum of the outputs of the experts

$$\mu = \sum_{i=1}^{M} g_i \mu_i \qquad \mu_i = f(\mathbf{x}, \mathbf{\Theta}_i) \qquad \sum_{i=1}^{M} g_i = 1 \qquad g_i \geq 0 \quad \forall i$$

$\mathbf{\Theta}_i$ is the parameter of the *i*-th expert

- The output of the gating network: “softmax” function

$$g_i = \frac{e^{\xi_i}}{\sum_{j=1}^{M} e^{\xi_j}} \qquad \xi_i = \mathbf{v}_i^T \mathbf{x}$$

- $\mathbf{v}_i^T$ is the parameter of the gating network

Mixture of Experts (MOE)

- Probabilistic interpretation

$$\mu_i = E[y \mid \mathbf{x}, \Theta_i] \qquad g_i = P(i \mid \mathbf{x}, \mathbf{v}_i)$$

the probabilistic model with true parameters

$$P(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\Theta}^0) = \sum_i g_i(\mathbf{x}, \mathbf{v}_i^0) P(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\Theta}_i^0)$$

a priori probability $\quad g_i(\mathbf{x}, \mathbf{v}_i^0) = P(i|\mathbf{x}, \mathbf{v}_i^0)$

Mixture of Experts (MOE)

- Training
  - Training data $X = \left\{\left(\mathbf{x}^{(l)}, \mathbf{y}^{(l)}\right)\right\}_{l=1}^{P}$
  - Probability of generating output from the input

$$P(\mathbf{y}^{(l)}|\mathbf{x}^{(l)}, \mathbf{\Theta}) = \sum_i P(i|\mathbf{x}^{(l)}, \mathbf{v}_i) P(\mathbf{y}^{(l)}|\mathbf{x}^{(l)}, \mathbf{\Theta}_i)$$

$$P(\mathbf{y} \mid \mathbf{x}, \mathbf{\Theta}) = \prod_{l=1}^{P} P(\mathbf{y}^{(l)} \mid \mathbf{x}^{(l)}, \mathbf{\Theta}) = \prod_{l=1}^{P} \left[\sum_i P(i \mid \mathbf{x}^{(l)}, \mathbf{v}_i) P(\mathbf{y}^{(l)} \mid \mathbf{x}^{(l)}, \mathbf{\Theta}_i) \right]$$

  - The log likelihood function (maximum likelihood estimation)

$$\mathcal{L}(\mathbf{x}, \mathbf{\Theta}) = \sum_l \log \left[\sum_i P(i \mid \mathbf{x}^{(l)}, \mathbf{v}_i) P(\mathbf{y}^{(l)} \mid \mathbf{x}^{(l)}, \mathbf{\Theta}_i) \right]$$

Mixture of Experts (MOE)

- Training (cont'd)
  - Gradient method

$$\frac{\partial \mathcal{L}(\mathbf{x},\boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}_i} = \mathbf{0} \qquad \text{and} \qquad \frac{\partial \mathcal{L}(\mathbf{x},\boldsymbol{\Theta})}{\partial \mathbf{v}_i} = \mathbf{0}$$

  - $$\frac{\partial \mathcal{L}(\mathbf{x},\boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}_i} = \frac{\partial \mathcal{L}(\mathbf{x},\boldsymbol{\Theta})}{\partial \mu_i}\frac{\partial \mu_i}{\partial \boldsymbol{\Theta}_i} \qquad \frac{\partial \mathcal{L}(\mathbf{x},\boldsymbol{\Theta})}{\partial \mathbf{v}_i} = \frac{\partial \mathcal{L}(\mathbf{x},\boldsymbol{\Theta})}{\partial \xi_i}\frac{\partial \xi_i}{\partial \mathbf{v}_i}$$

  - The parameter of the expert network

$$\boldsymbol{\Theta}_i(k+1) = \boldsymbol{\Theta}_i(k) + \eta \sum_{l=1}^{P} h_i^{(l)} (\mathbf{y}^{(l)} - \mu_i) \frac{\partial \mu_i}{\partial \boldsymbol{\Theta}_i}$$

  - The parameter of the gating network

$$\mathbf{v}_i(k+1) = \mathbf{v}_i(k) + \eta \sum_{l=1}^{P} \left(h_i^{(l)} - g_i^{(l)}\right)\mathbf{x}^{(l)}$$

Mixture of Experts (MOE)

- Training (cont'd)
  - A priori probability

$$g_i^{(l)} = g_i(\mathbf{x}^{(l)}, \mathbf{v}_i) = P(i \mid \mathbf{x}^{(l)}, \mathbf{v}_i)$$

  - A posteriori probability

$$h_i^{(l)} = \frac{g_i^{(l)} P(\mathbf{y}^{(l)} | \mathbf{x}^{(l)}, \Theta_i)}{\sum_j g_j^{(l)} P(\mathbf{y}^{(l)} | \mathbf{x}^{(l)}, \Theta_j)}$$

Mixture of Experts (MOE)

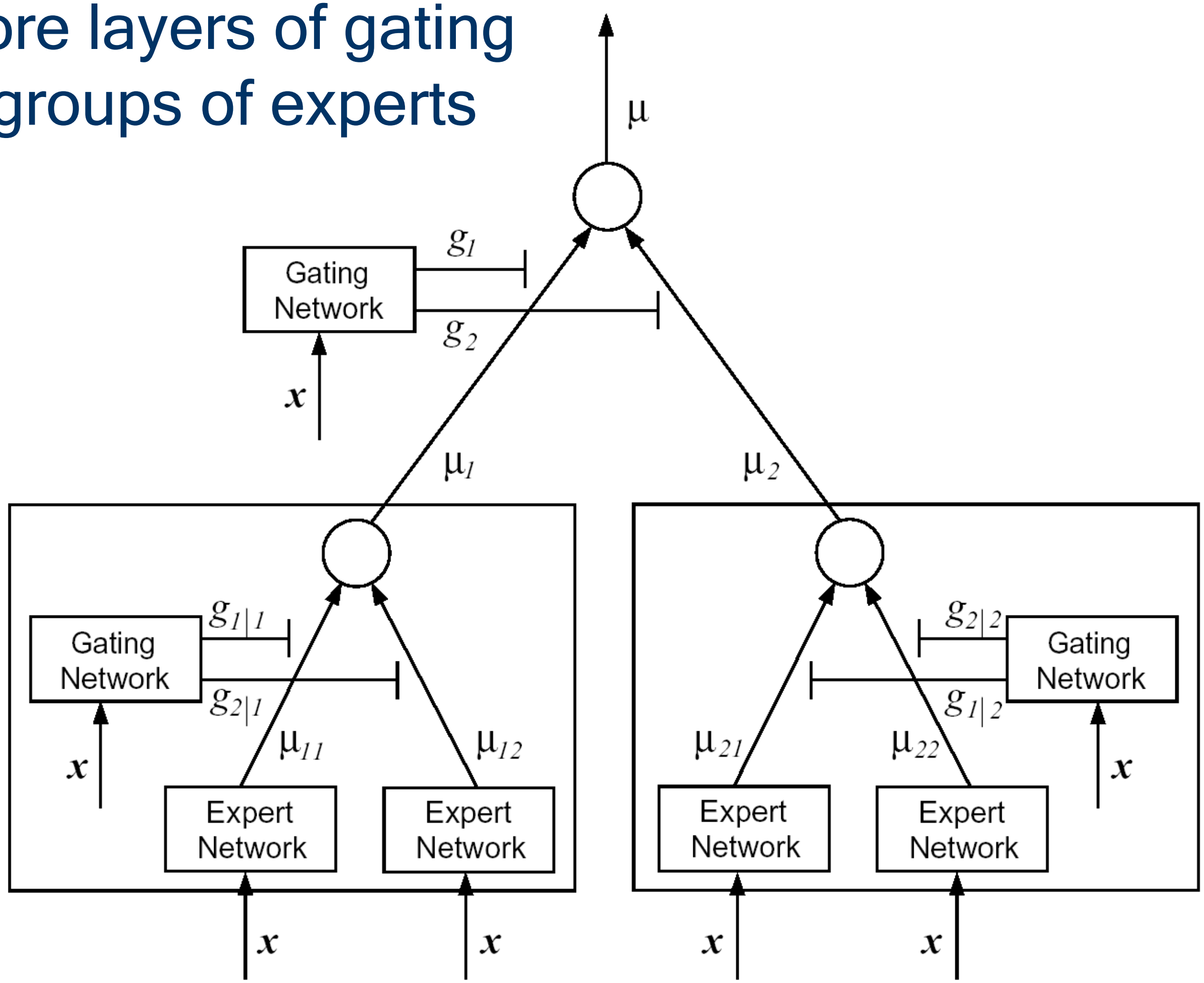
- Training (cont'd)
  - EM (Expectation Maximization) algorithm

    A general iterative technique for maximum likelihood estimation

    - Introducing hidden variables
    - Defining a log-likelihood function
  - Two steps:
    - Expectation of the hidden variables
    - Maximization of the log-likelihood function

Hierarchical Mixture of Experts (HMOE)

HMOE: more layers of gating networks, groups of experts



Mixture of Experts (MOE)

- MOE construction
- Cross-validation can be used to find the proper architecture
- CART (Clasification And Regression Tree) for initial hierarchical MOE (HMOE) architecture and for the initial expert and gating network parameters
- MOE based on SVMs: different SVMs with different hyperparameters

References and further readings

Haykin, S.: "Neural Networks. A Comprehensive Foundation" Prentice Hall, N. J.1999.

Jordan, M. I., Jacobs, R. A.: "Hierarchical Mixture of Experts and the EM Algorithm" Neural Computation Vol. 6. pp. 181-214, 1994.

Bilmes, J. A. et al "A Gentle Tutorial of the EM Algorithm and its application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models" 1998.

Dempster A.P. et al "Maximum-likelihood from incomplete data via the EM algorithm", 1977.

Moon, T.K."The Expectation-Maximization Algorithm" IEEE Trans Signal Processing, 1996.

Application: modelling an industrial plant (steel converter)

Overview

- Introduction
- Modeling approaches
- Building neural models
- Data base construction
- Model selection
- Modular approach
- Hybrid approach
- Information system
- Experiences with the advisory system
- Conclusions

Overview

- Introduction
- Modeling approaches
- Building neural models
- Data base construction
- Model selection
- Modular approach
- Hybrid approach
- Information system
- Experiences with the advisory system
- Conclusions

Introduction to the problem

- Task
  - to develop an advisory system for a Linz-Donawitz steel converter
  - to propose component composition
  - to support the factory staff in supervising the steel-making process
- A model of the process is required: first a system modelling task should be solved

LD Converter modeling

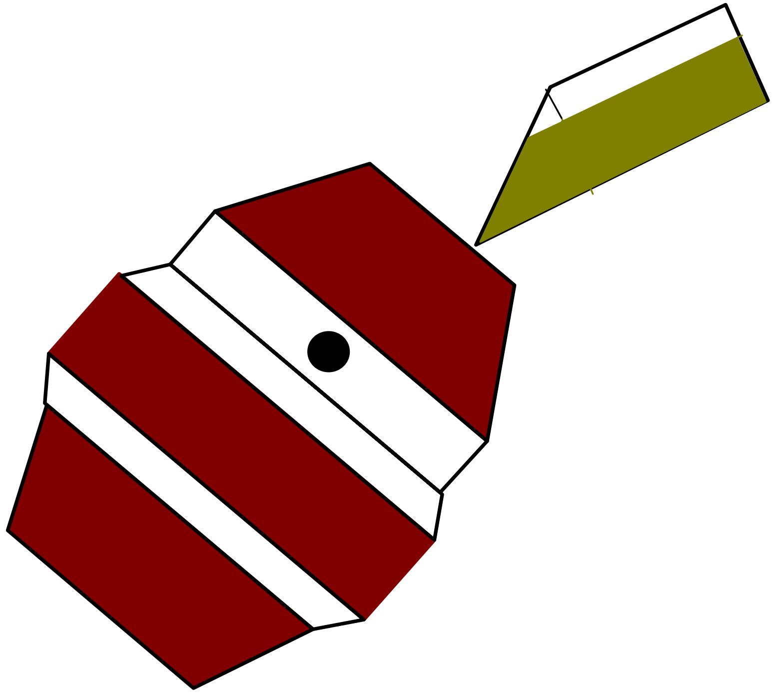
The Linz-Donawitz converter in Hungary (Dunaferr Co.)

Basic Oxigen Steelmaking (BOS)

Linz-Donawitz converter

Phases of steelmaking

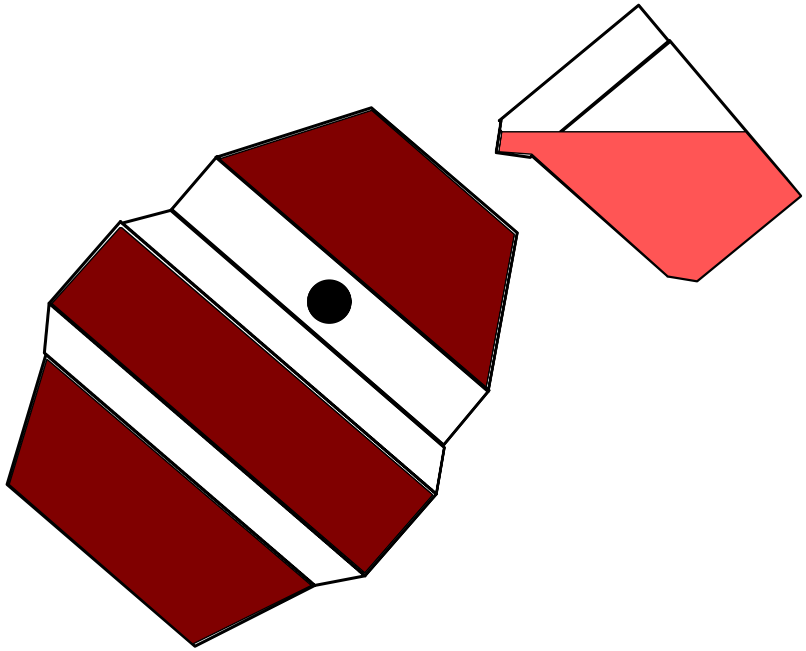
- **1. Filling of waste iron**
- 2. Filling of pig iron
- 3. Blasting with pure oxygen
- 4. Supplement additives
- 5. Sampling for quality testing
- 6. Tapping of steel and slag



Linz-Donawitz converter

Phases of steelmaking

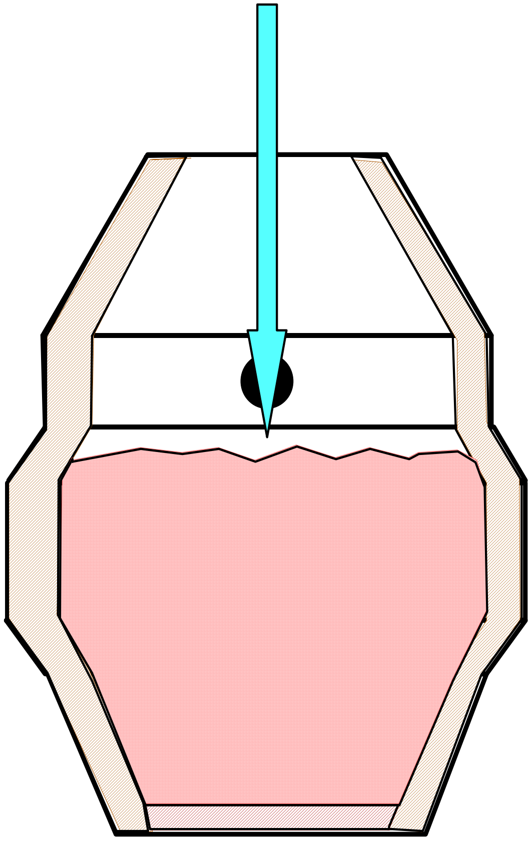
- 1. Filling of waste iron
- **2. Filling of pig iron**
- 3. Blasting with pure oxygen
- 4. Supplement additives
- 5. Sampling for quality testing
- 6. Tapping of steel and slag



Linz-Donawitz converter

Phases of steelmaking

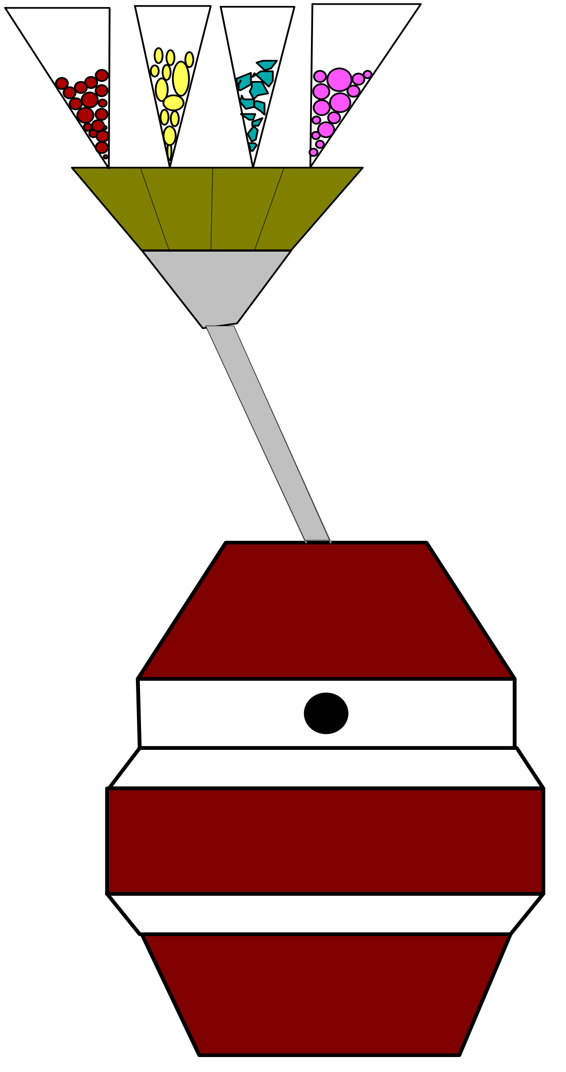
- 1. Filling of waste iron
- 2. Filling of pig iron
- **3. Blasting with pure oxygen**
- 4. Supplement additives
- 5. Sampling for quality testing
- 6. Tapping of steel and slag



Linz-Donawitz converter

Phases of steelmaking

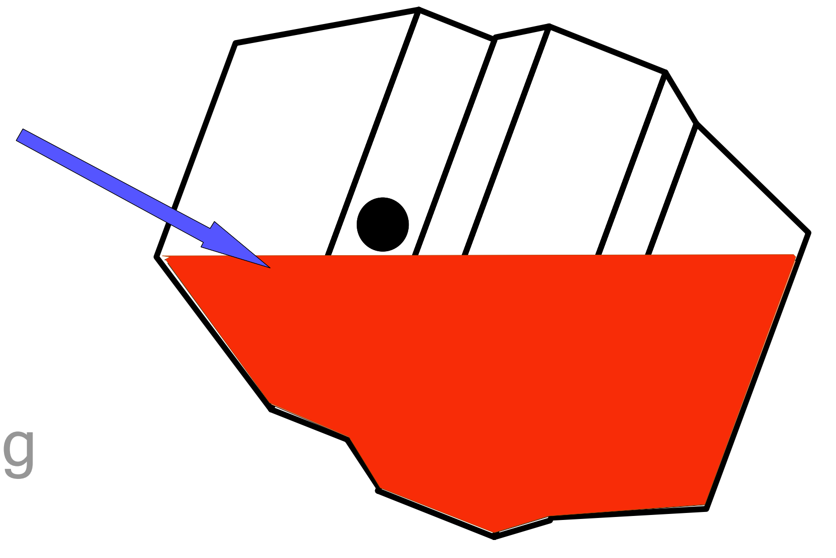
- 1. Filling of waste iron
- 2. Filling of pig iron
- 3. Blasting with pure oxygen
- **4. Supplement additives**
- 5. Sampling for quality testing
- 6. Tapping of steel and slag



Linz-Donawitz converter

Phases of steelmaking

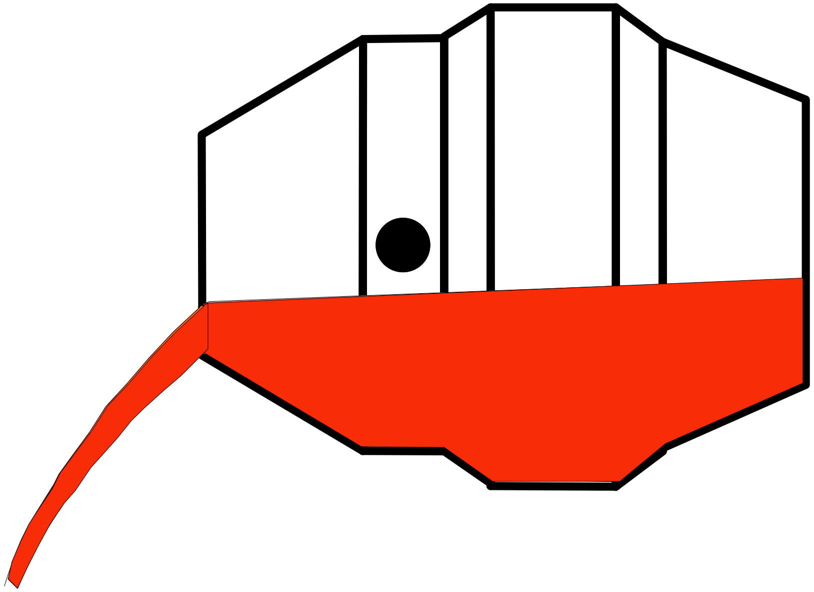
- 1. Filling of waste iron
- 2. Filling of pig iron
- 3. Blasting with pure oxygen
- 4. Supplement additives
- **5. Sampling for quality testing**
- 6. Tapping of steel and slag



Linz-Donawitz converter

Phases of steelmaking

- 1. Filling of waste iron
- 2. Filling of pig iron
- 3. Blasting with pure oxygen
- 4. Supplement additives
- 5. Sampling for quality testing
- **6. Tapping of steel and slag**



Main featutes of the process

- Nonlinear input-output relation between many inputs and two outputs
- input parameters (~50 different parameters)
  - certain features “measured” during the process
- The main output parameters (output measured values of the produced steel)
  - temperature (1640-1700 C^{O} -10 … +15 C^{O})
  - carbon content (0.03 - 0.70 %)
- More than 5000 records of data

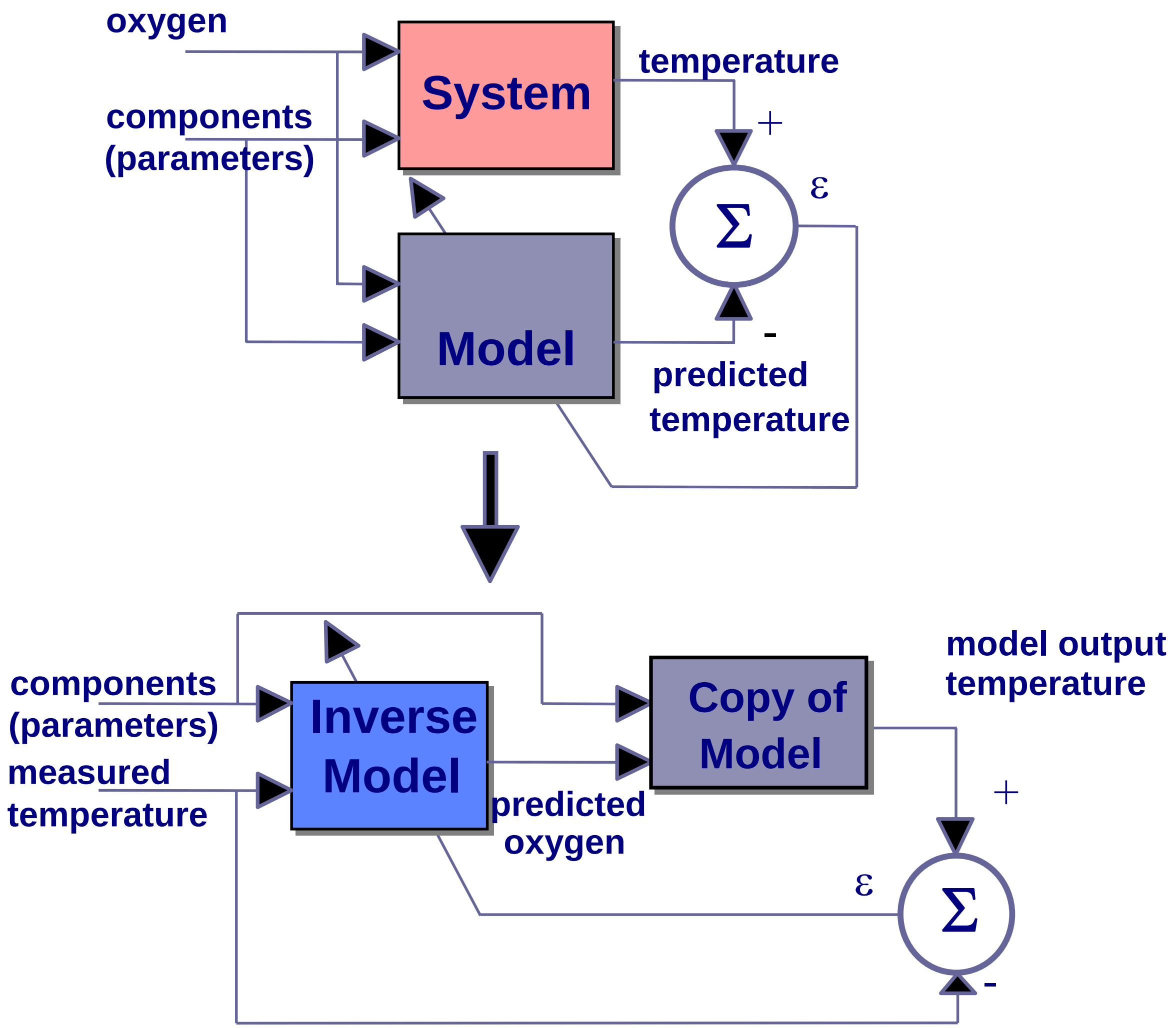
Modeling task

- The difficulties of model building
  - High complexity nonlinear input-output relationship
  - No (or unsatisfactory) physical insight
  - Relatively few measurement data
  - There are unmeasurable parameters of the process
  - Noisy, imprecise, unreliable data
  - Classical approach (heat balance, mass balance) gives no acceptable results

Modeling approaches

- Theoretical model - based on chemical and physical equations
- Input - output behavioral model
  - Neural model - based on the measured process data
  - Rule based system - based on the experimental knowledge of the factory staff
  - Combined neural - rule based system: a hybrid model

The modeling task

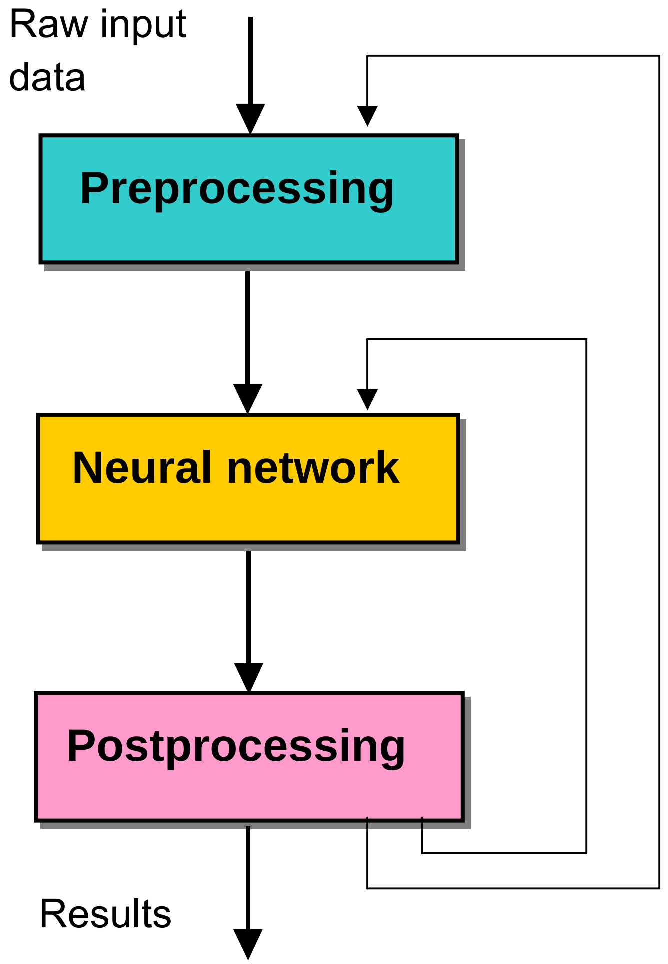


Overview

- Introduction
- Modeling approaches
- Building neural models
- Data base construction
- Model selection
- Modular approach
- Hybrid approach
- Information system
- Experiences with the advisory system
- Conclusions

„Neural” solution

- The steps of solving a practical problem



Building neural models

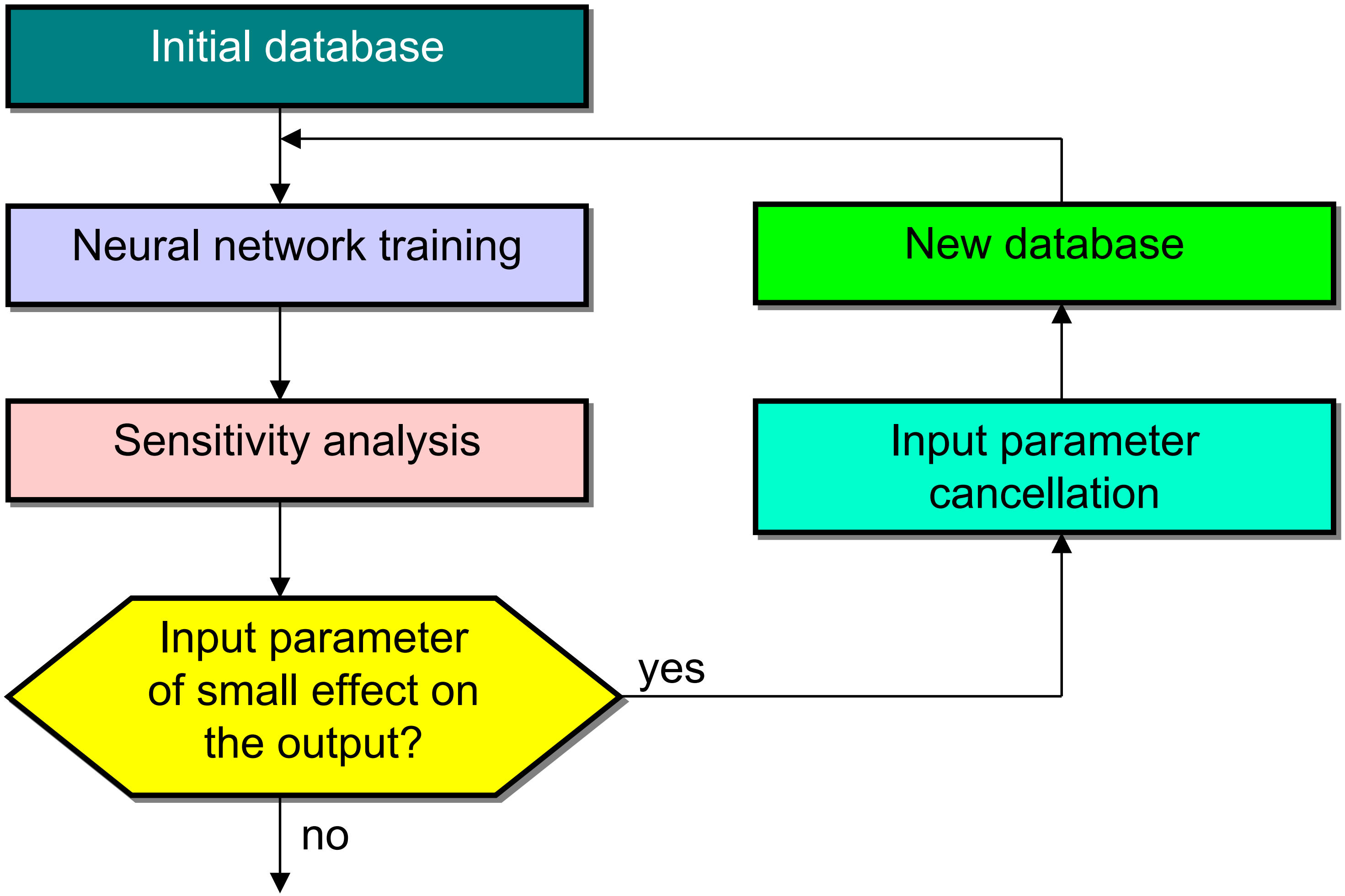
- Creating a reliable database
  - the problem of noisy data
  - the problem of missing data
  - the problem of uneven data distribution
- Selecting a proper neural architecture
  - static network (size of the network)
  - dynamic network
    - size of the network: nonlinear mapping
    - regressor selection + model order selection
- Training and validating the model

Creating a reliable database

- Input components
  - measure of importance
    - physical insight
    - sensitivity analysis (importance of the input variables)
    - mathematical methods: dimension reduction (e.g. PCA)
- Normalization
  - input normalization
  - output normalization
- Missing data
  - artificially generated data
- Noisy data
  - preprocessing, filtering,
  - errors-in-variables criterion function, etc.

Building database

- Selecting input components, sensitivity analysis

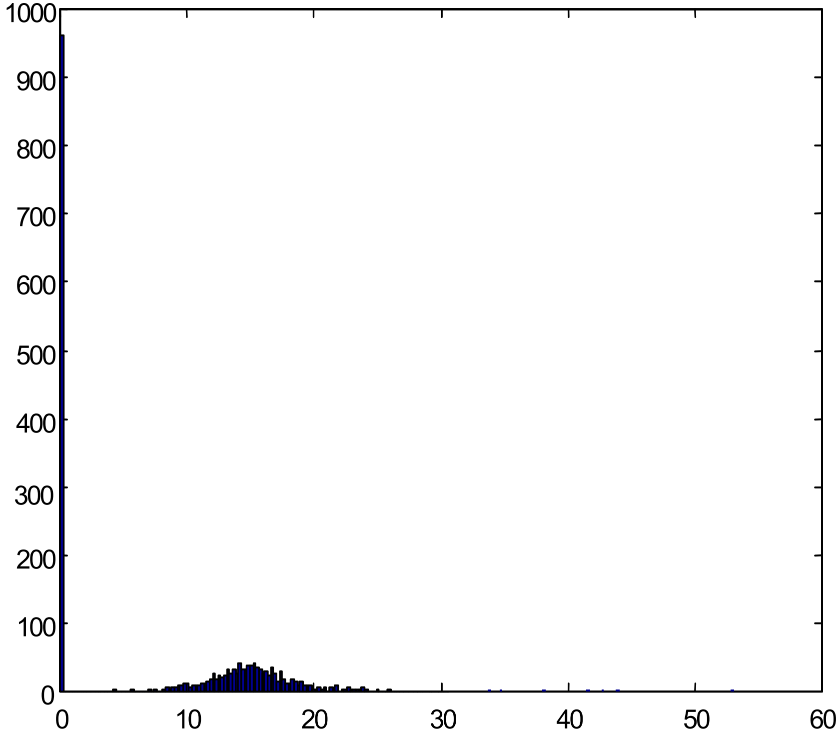


Building database

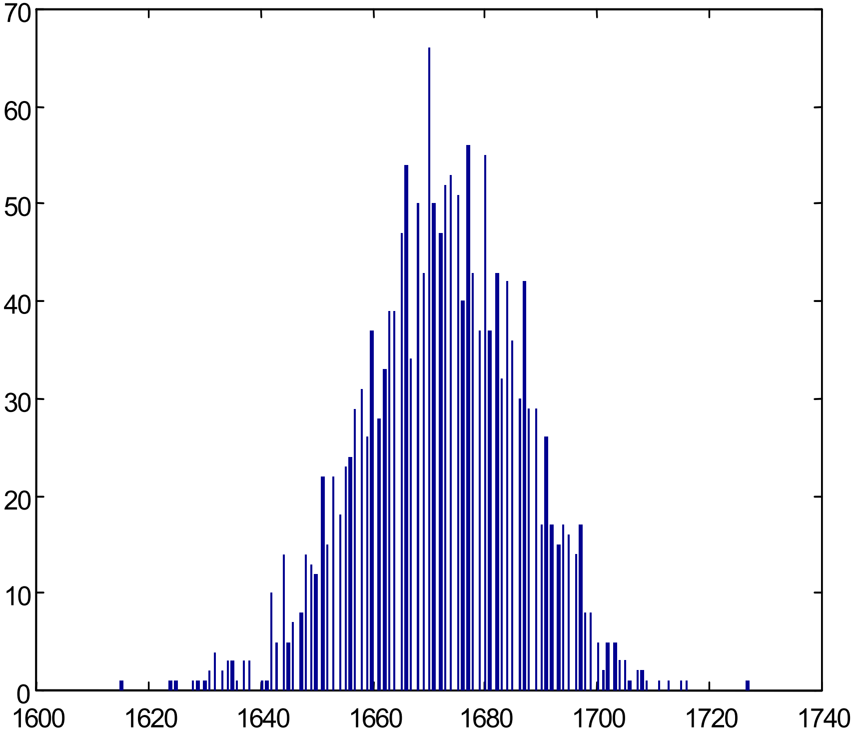
- Dimension reduction: mathematical methods
  - PCA
  - Non-linear PCA, Kernel PCA
  - ICA
- Combined methods

The effect of data distribution

- Typical data distributions



Uneven distribution



Approximately Gaussian distribution

Normalization

- Zero mean, unit standard deviation

$$\bar{x}_i = \frac{1}{P}\sum_{p=1}^{P} x_i^{(p)} \qquad \sigma_i^2 = \frac{1}{P-1}\sum_{p=1}^{P}\left(x_i^{(p)} - \bar{x}_i\right)^2 \qquad \tilde{x}_i^{(p)} = \frac{x_i^{(p)} - \bar{x}_i}{\sigma_i}$$

- Normalization into [0,1]

$$\tilde{x}_i = \frac{x_i - \min\{x_i\}}{\max\{x_i\} - \min\{x_i\}}$$

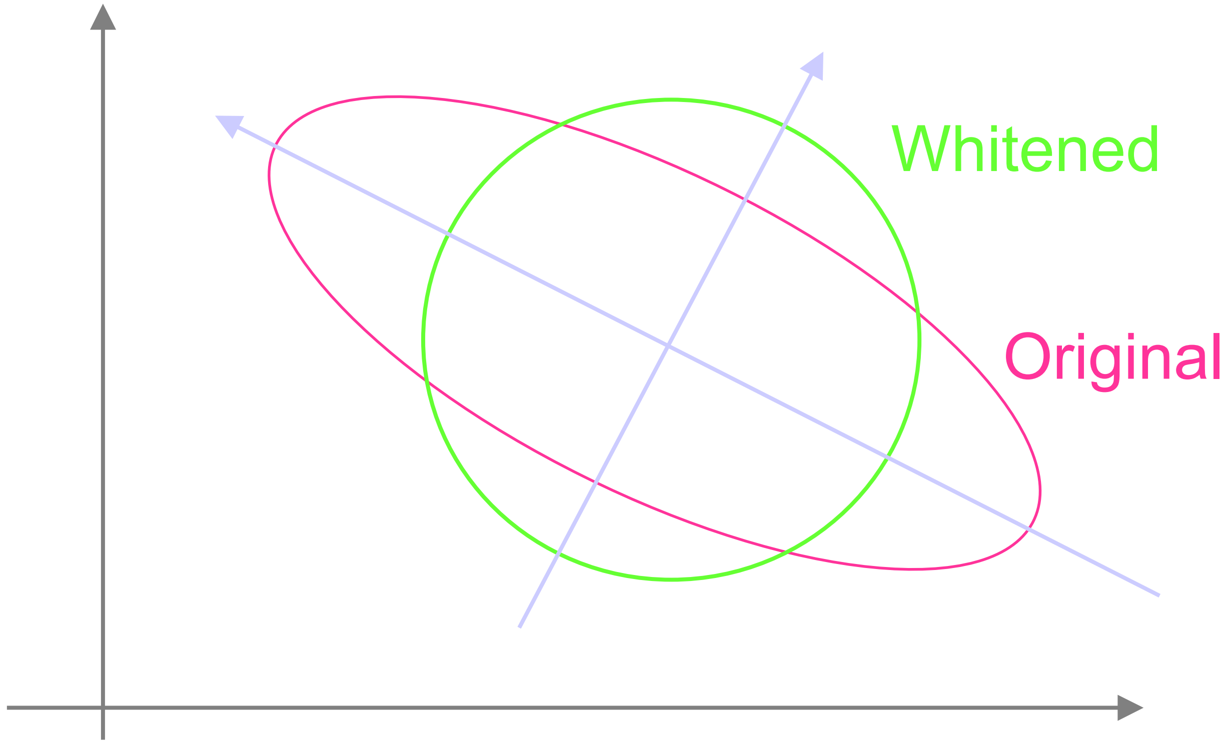
- Decorrelation + normalization

$$\boldsymbol{\Sigma} = \frac{1}{P-1}\sum_{p=1}^{P}\left(\mathbf{x}^{(p)} - \bar{\mathbf{x}}\right)\left(\mathbf{x}^{(p)} - \bar{\mathbf{x}}\right)^{\mathrm{T}} \qquad \boldsymbol{\Sigma}\boldsymbol{\varphi}_j = \lambda_j \boldsymbol{\varphi}_j \qquad \Lambda = \mathrm{diag}(\lambda_1 \ldots \lambda_N)$$

$$\tilde{\mathbf{x}}^{(p)} = \Lambda^{-1/2}\boldsymbol{\Phi}^{\mathrm{T}}\left(\mathbf{x}^{(p)} - \bar{\mathbf{x}}\right) \qquad \boldsymbol{\Phi} = \left[\boldsymbol{\varphi}_1\,\boldsymbol{\varphi}_2\,\ldots\,\boldsymbol{\varphi}_N\right]^T$$

Normalization

- Decorrelation + normalization = Whitening transformation



Missing or few data

- Filling in the missing values
  - based on available information
- Artificially generated data
  - using trends
  - using correlation
  - using realistic transformations

Overview

- Introduction
- Modeling approaches
- Building neural models
- Data base construction
- Model selection
- Modular approach
- Hybrid approach
- Information system
- Experiences with the advisory system
- Conclusions

Missing or few data

- Filling in the missing values based on:

  correlation coefficient between x_i and x_j $\quad \tilde{\mathbf{C}}(i,j) = \dfrac{\mathbf{C}(i,j)}{\sqrt{\mathbf{C}(i,i)\,\mathbf{C}(j,j)}}$

  previous (other) values $\quad \hat{x}_i = \bar{x}_i + \sigma_i \xi$

  other parameters $\hat{x}_i^{(k)} = \hat{f}(x_j^{(k)})$ or $\hat{x}_i^{(k)} = \hat{f}(x_j^{(k)}, x_j^{(k)}, \ldots)$

  time dependence (dynamic problem) $\quad \mathrm{R}_i(t,\tau) = E\{x_i(t)\, x_i(t+\tau)\}$

- Artificially generated data
  - using trends
  - using correlation
  - using realistic transformations

Few data

- Artificial data generation
  - using realistic transformations
  - using sensitivity values: data generation around various working points (a good example: ALVINN)

(ALVINN = *Autonomous Land Vehicle In a Neural Net* an onroad neural network navigation solution developed in CMU)

Noisy data

Inherent noise suppression

- classical neural nets have noise suppression property (inherent regularization)
- Regularization (smoothing regularization)
- averaging (modular approach)

• SVM

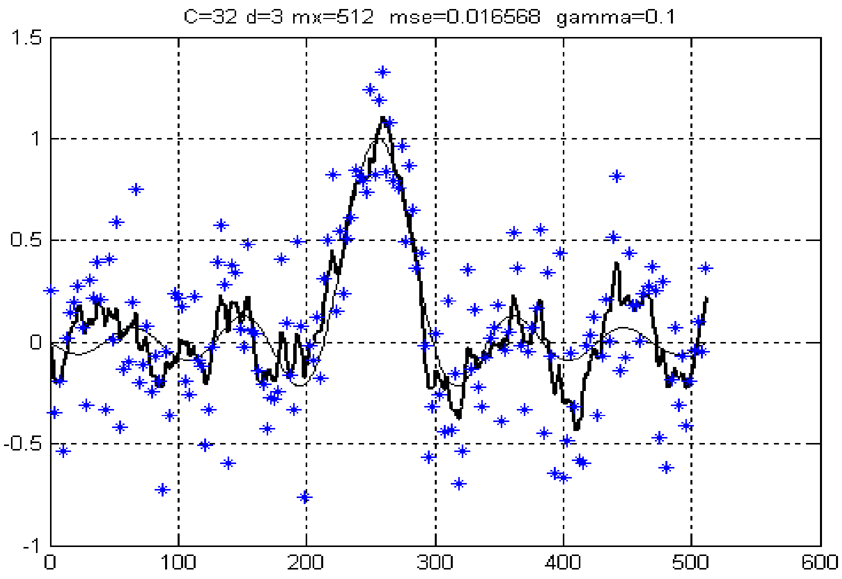
- ε-insensitive criterion function

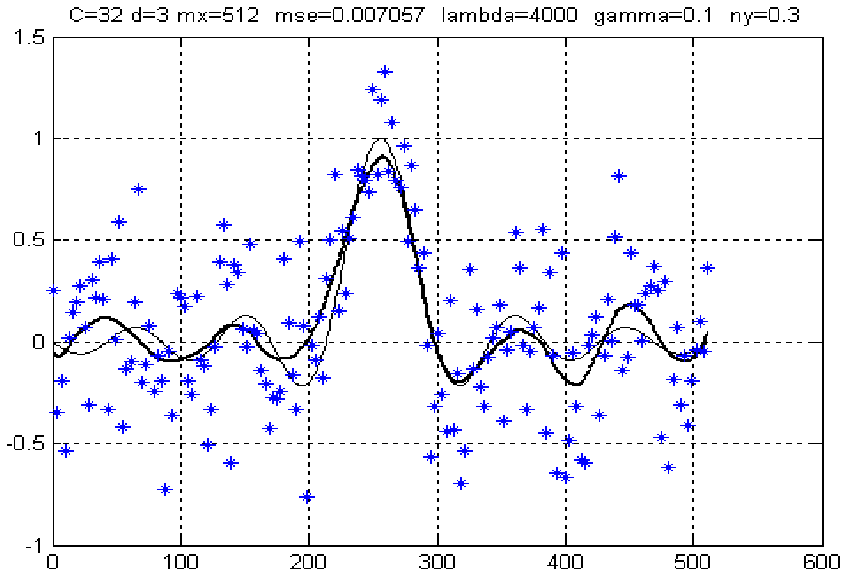
• EIV

- input and output noise are taken into consideration
- modified criterion function

Reducing the effect of output noise

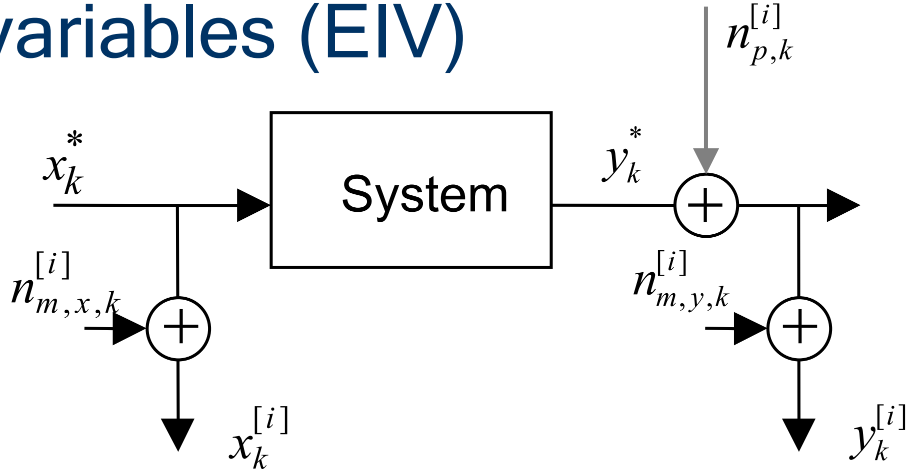
- Inherent regularization of MLP (smooth sigmoidal function)
- SVM with ε - insensitive loss function
- Regularization: example regularized (kernel) CMAC





Reducing the effect of input and output noise

- Errors in variables (EIV)



$$\bar{x}_k = \frac{1}{M}\sum_{i=1}^{M} x_k^{[i]}$$

$$\bar{y}_k = \frac{1}{M}\sum_{i=1}^{M} y_k^{[i]}$$

$$\sigma_{x,k}^2 = \frac{1}{M-1}\sum_{i=1}^{M}\left(x_k^{[i]} - \bar{x}_k\right)^2$$

$$\sigma_{y,k}^2 = \frac{1}{M-1}\sum_{i=1}^{M}\left(y_k^{[i]} - \bar{y}_k\right)^2$$

EIV

- LS vs EIV criterion function

$$C_{LS} = \frac{1}{P}\sum_{k=1}^{P}(y_k^* - f_{NN}(x_k^*, \mathbf{W}))^2$$

$$C_{EIV} = \frac{1}{P}\sum_{k=1}^{P}\left(\frac{(y_k - f_{NN}(x_k, \mathbf{W}))^2}{\sigma_{y,k}^2} + \frac{(x^*{}_k - x_k)^2}{\sigma_{x,k}^2}\right)$$

- EIV training

$$\Delta\mathbf{W}_j = \eta\frac{M}{2P}\sum_{k=1}^{P}\frac{e_{f,k}}{\sigma_{y,k}^2}\frac{\partial f_{NN}(x_k, \mathbf{W})}{\partial \mathbf{W}_j} \qquad \Delta x_k = \eta\frac{M}{2}\left[\frac{e_{f,k}}{\sigma_{y,k}^2}\frac{\partial f_{NN}(x_k, \mathbf{W})}{\partial x_k} + \frac{e_{x,k}}{\sigma_{x,k}^2}\right]$$

$$e_{f,k} = y_k - f_{NN}(x_k, \mathbf{W})$$

- Danger : overfitting $\rightarrow$ early stopping

Noisy data

- Output noise is easier to suppress than input noise
- SVM, regularization can reduce the effect of output noise
- EIV (and similar other methods) can take into consideration the input noise
- EIV results in only slightly better approximation
- EIV is rather prone to overfitting (much more free parameters) $\rightarrow$ early stopping

Overview

- Introduction
- Modeling approaches
- Building neural models
- Data base construction
- **Model selection**
- Modular approach
- Hybrid approach
- Information system
- Experiences with the advisory system
- Conclusions

Model selection

- Static or dynamic
  - why better a dynamic model

- Dynamic model class
  - regressor selection
  - basis function selection

- Size of the network
  - number of layers
  - number of hidden neurons
  - model order

Model selection

- NFIR
- NARX
- NOE
- NARMAX

NARX model, NOE model: model order selection

$$y_M(k) = f\left[\mathbf{x}(k), \mathbf{x}(k-1), \mathbf{x}(k-2), \ldots, \mathbf{x}(k-n), y(k-1), y(k-2), \ldots, y(k-m)\right]$$

Model order: the input dimension of the static network

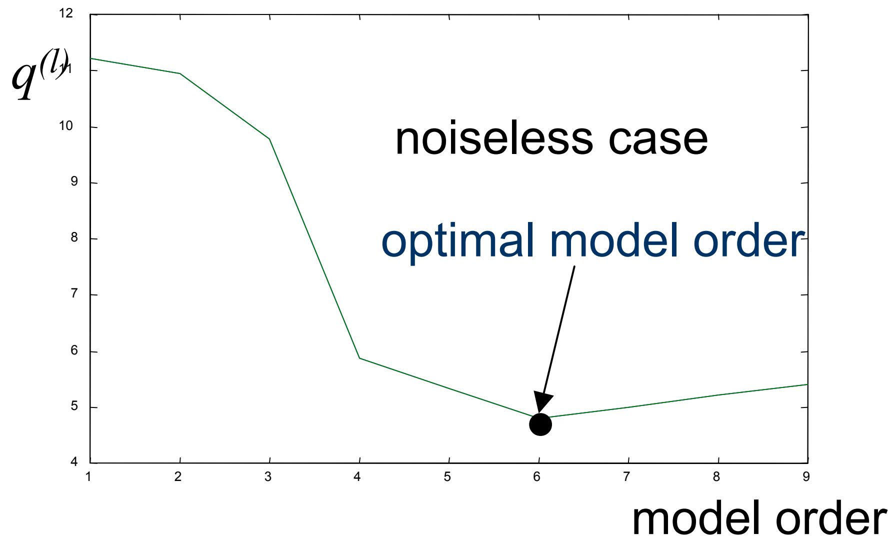
Model order selection

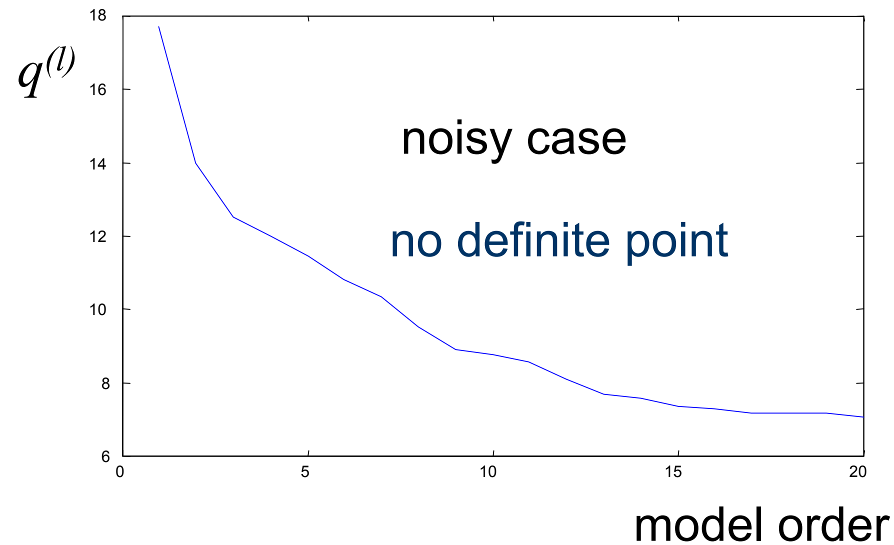
- AIC, MDL, NIC, Lipschitz number

$$y(k) = f\left[\mathbf{x}(k), \mathbf{x}(k-1), \mathbf{x}(k-2), \ldots, \mathbf{x}(k-n), y(k-1), y(k-2), \ldots, y(k-m)\right]$$

- Lipschitz number, Lipschitz quotient

$$q^{(l)} = \left(\prod_{k=1}^{p} \sqrt{n}\, q^{(l)}(k) \right)^{1/p} \qquad q_{ij} = \frac{\left| y_i - y_i \right|}{\left| \mathbf{x}_i - \mathbf{x}_j \right|},$$





Model order selection

- Lipschitz quotient

general nonlinear input-output relation, f(.) continuous, smooth

multivariable function $y = f\left[x_1, x_2, \ldots, x_n\right]$

bounded derivatives $\left|f_i^{'}\right| = \left|\frac{\partial f}{\partial x_i}\right| \leq M \quad i = 1, 2, \ldots, n$

Lipschitz quotient

$$q_{ij} = \frac{\left|y_i - y_i\right|}{\left|\mathbf{x}_i - \mathbf{x}_j\right|}, \quad i \neq j \qquad 0 \leq q_{ij} \leq L$$

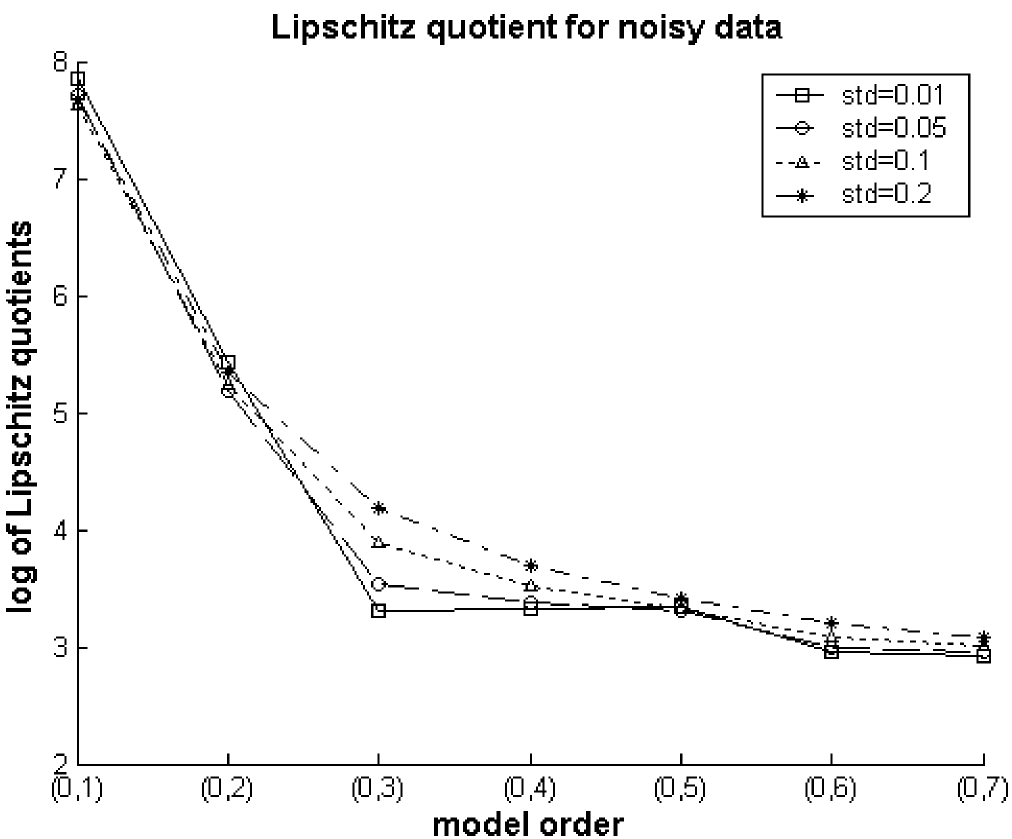
Sensitivity analysis

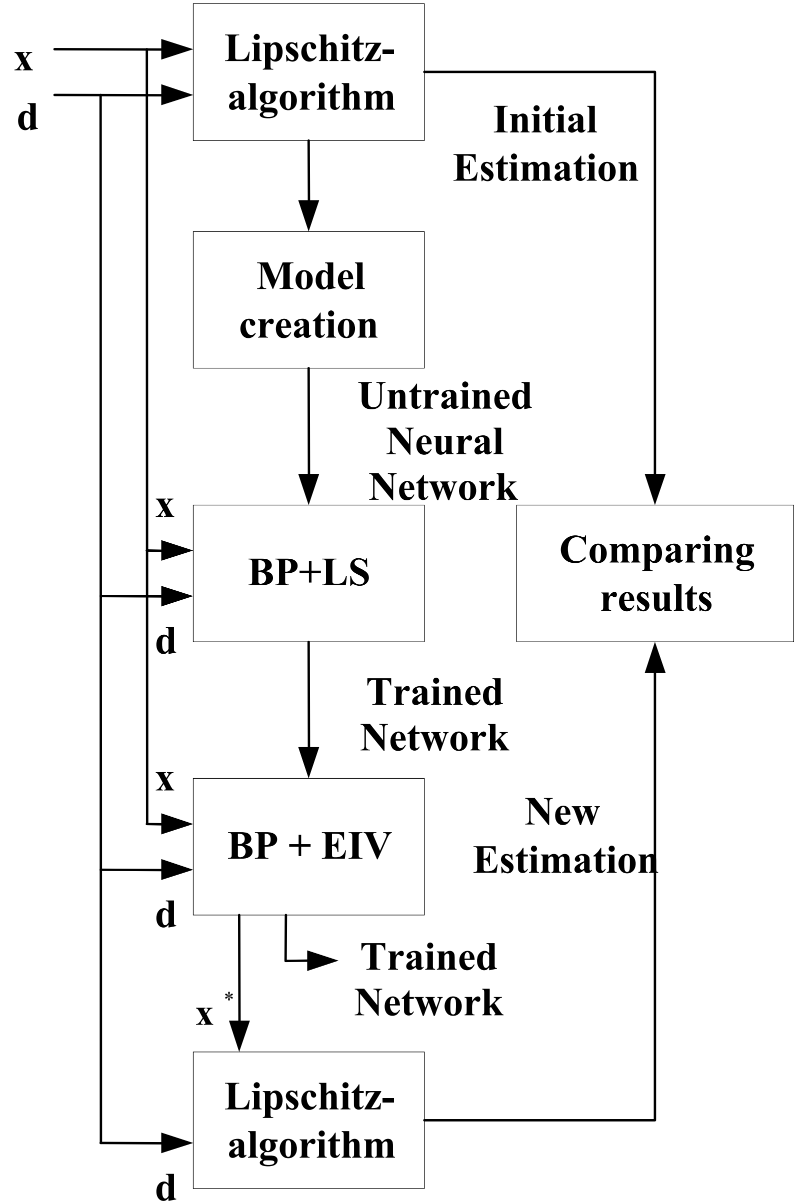
$$\Delta y = \frac{\partial f}{\partial x_1}\Delta x_1 + \frac{\partial f}{\partial x_2}\Delta x_2 + \ldots + \frac{\partial f}{\partial x_n}\Delta x_n = f_1^{'}\Delta x_1 + f_2^{'}\Delta x_2 + \ldots + f_n^{'}\Delta x_n$$

Model order selection

- Noisy case
  Combined method
  of Lipschitz + EIV

x





Correlation based model order selection

- Model order 2...4 because of prcatical problems
- Too many input components
- (2...4) * (number of input components + outputs)
- Too large network
- Too few training data
- The problem of missing data
- Network size: cross-validation

References and further readings

Berényi, P.,, Horváth, G., Pataki, B., Strausz, Gy. : "Hybrid-Neural Modeling of a Complex Industrial Process" Proc. of the IEEE Instrumentation and Measurement Technology Conference, IMTC'2001. Budapest, May 21-23. Vol. III. pp. 1424-1429.

Horváth, G., Pataki, B. Strausz, T.: "Neural Modeling of a Linz-Donawitz Steel Converter: Difficulties and Solutions" Proc. of the EUFIT'98, 6th European Congress on Intelligent Techniques and Soft Computing. Aachen, Germany. 1998. Sept. pp.1516-1521

Horváth, G. Pataki, B. Strausz, Gy.: "Black box modeling of a complex industrial process", Proc. Of the 1999 IEEE Conference and Workshop on Engineering of Computer Based Systems, Nashville, TN, USA. 1999. pp. 60-66

Pataki, B., Horváth, G., Strausz, Gy., Talata, Zs. "Inverse Neural Modeling of a Linz-Donawitz Steel Converter" e & i Elektrotechnik und Informationstechnik, Vol. 117. No. 1. 2000. pp.

Strausz, Gy., G. Horváth, B. Pataki : "Experiences from the results of neural modelling of an industrial process" Proc. of Engineering Application of Neural Networks, EANN'98, Gibraltar 1988. pp. 213-220

Akaike, H. "A New Look at the Statistical Model Identification", *IEEE Transaction on Aut. Control* Vol. AC-19, No.6., 1974, pp.716-723

Rissanen, J. "Estimation of Structure by Minimum Description Length", *Circuits, Systems and Signal Processing*, special issue on Rational Approximations, Vol. 1, No. 3-4, 1982, pp. 395-406.

Akaike, H. "Statistical Predictor Identification", *Ann. Istitute of Statistical Mathematics*, Vol. 22., 1970, pp. 203-217.

He, X. and Asada, H."A New Method for Identifying Orders of Input-output Models for Nonlinear Dynamic Systems" *Proceedings of the American Control Conference*, San Francisco, California, June 1993, pp. 2520-2523.

Overview

- Introduction
- Modeling approaches
- Building neural models
- Data base construction
- Model selection
- Modular approach
- Hybrid approach
- Information system
- Experiences with the advisory system
- Conclusions

Modular solution

- More neural modell for the different working conditions
- Processing of special cases
- Depending on the distribution of inputparameters
- Cooperative or competitive modular architecture

Hybrid solution

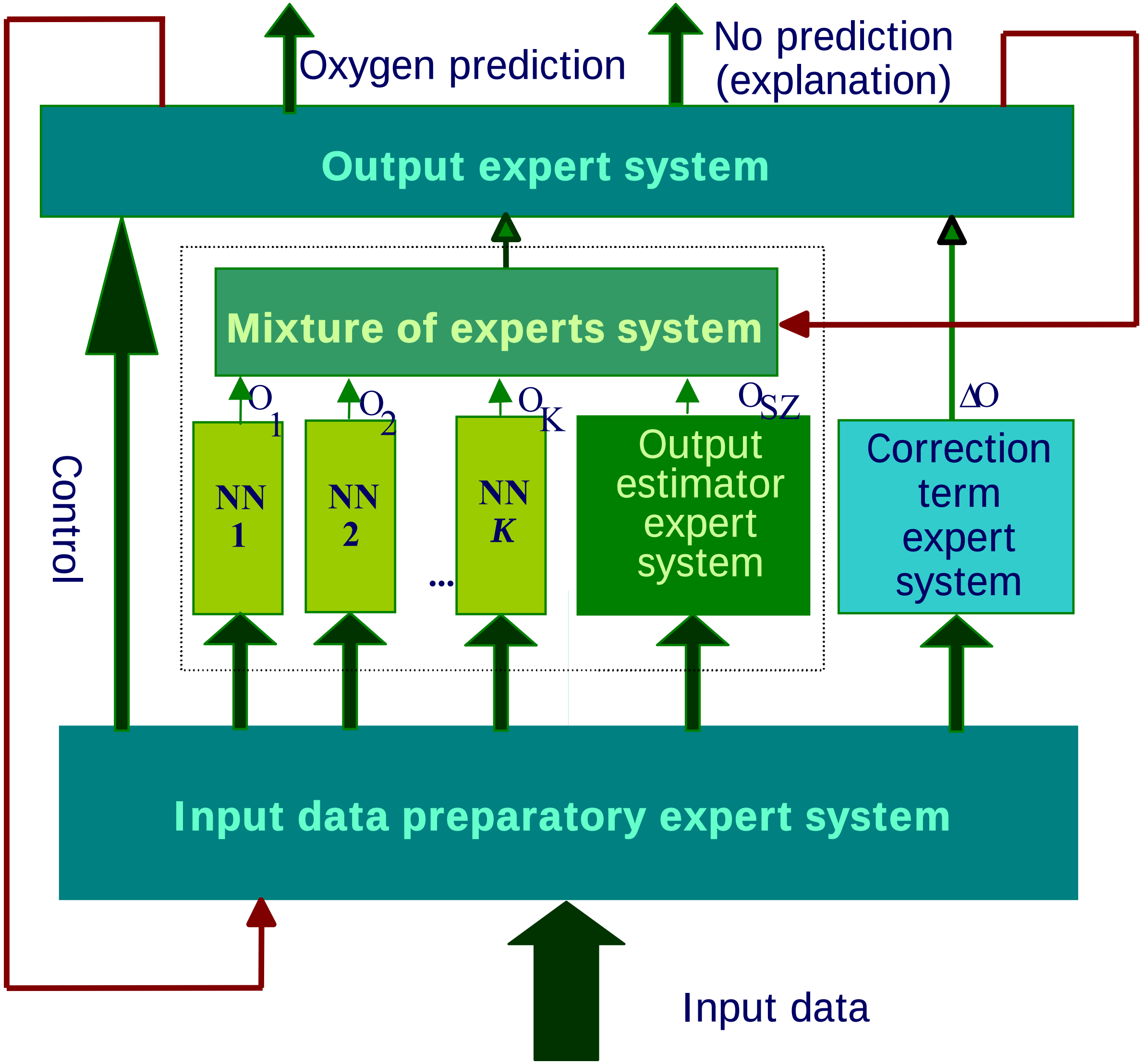
- Utilization of different forms of information
  - measurement, experimental data
  - symbolic rules
  - mathematical equations, physical knowledge

The hybrid information system

- Solution:
  - integration of measurement information and experimental knowledge about the process results
- Realization:
  - development system – supports the design and testing of different hybrid models
  - advisory system
    - hybrid models using the current process state and input information,
    - experiences collected by the rule-base system can be used to update the model.

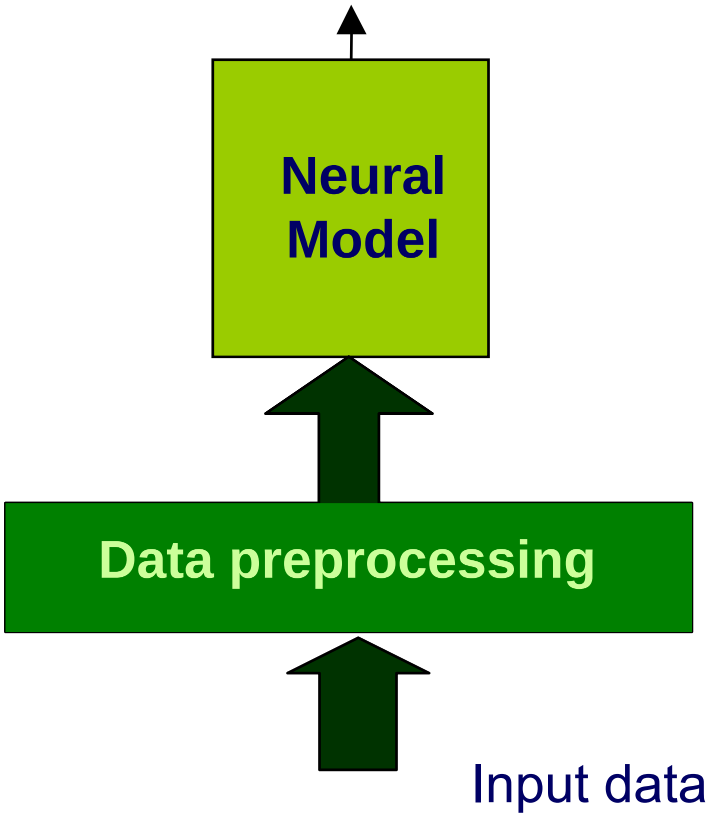
The hybrid-neural system

Information processing



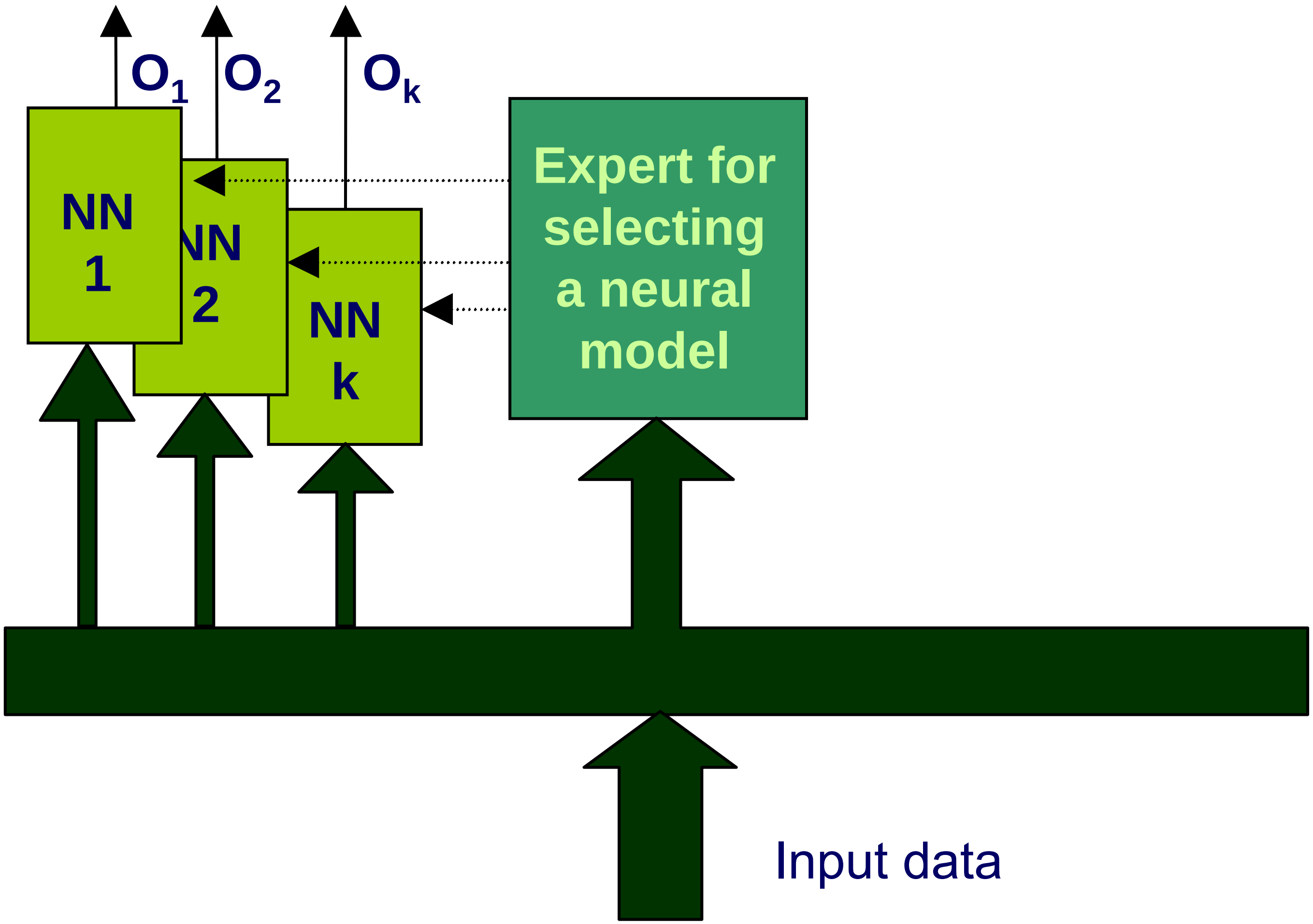
The hybrid-neural system

Data preprocessing and correction

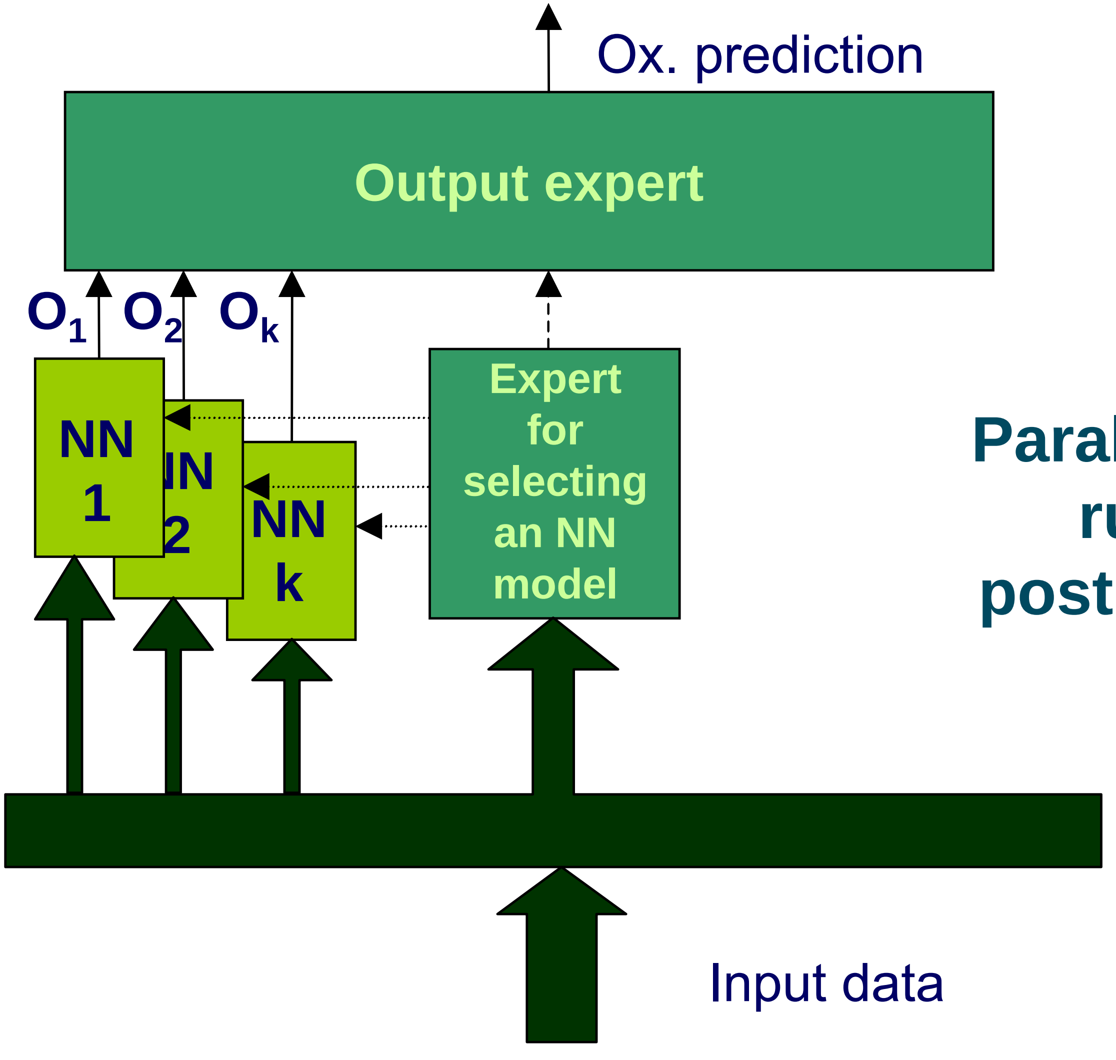


The hybrid-neural system

Conditional network running



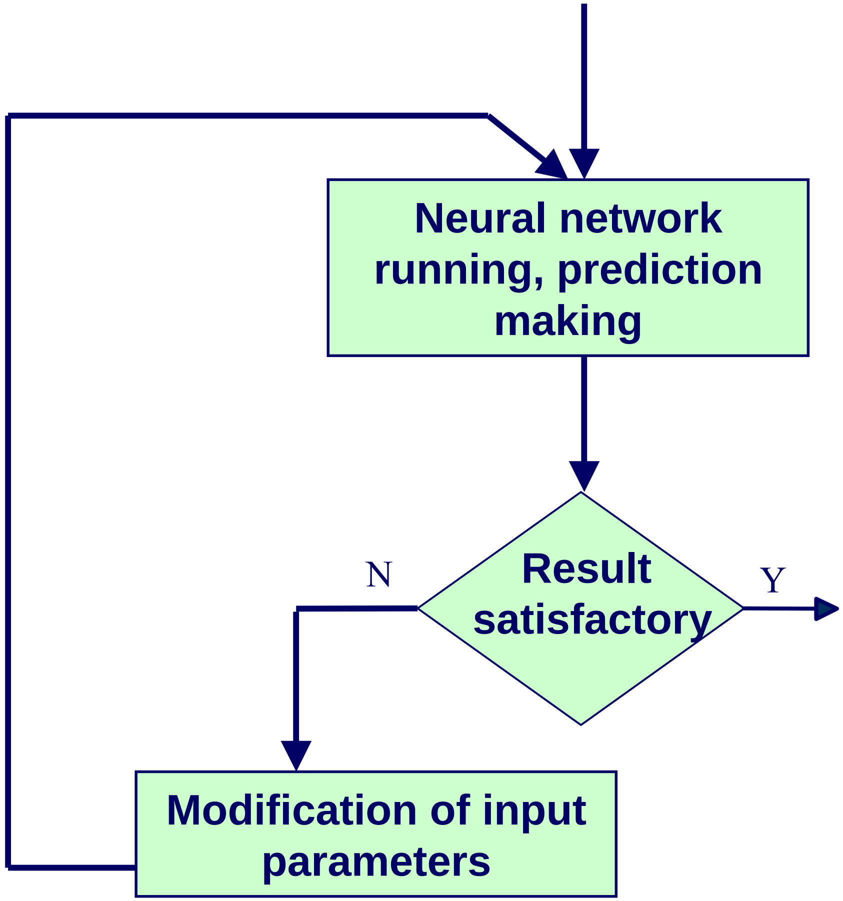
The hybrid-neural system



Parallel network running - postprocessing

The hybrid-neural system

Iterative network running



Validation

- Model selection
  - iterative process
  - utilization of domain knowledge
- Cross validation
  - fresh data
  - on-site testing

Experiences

- The hit rate is increased by + 10%
- Most of the special cases can be handled
- Further rules for handling special cases should be obtained
- The accuracy of measured data should be increased

Conclusions

- For complex industrial problems all available information have to be used
- Thinking about NNs as universal modeling devices alone
- Physical insight is important
- The importance of preprocessing and post-processing
- Modular approach:
  - decomposition of the problem
  - cooperation and competition
  - “experts” using different paradigms
- The hybrid approach to the problem provided better results

Summary

- Main questions
- Open questions
- Final conclusions

Main questions

- Neural modeling: black-box or not?
- When to apply neural approach?
- How to use neural networks?
- The role of prior knowledge
- How to use prior knowledge?
- How to validate the results?

Open (partly open) questions

- Model class selection
- Model order selection
- Validation, generalization capability
- Sample size, training set, test set, validation set
- Missing data, noisy data, few data
- Data consistency

Final coclusions

- Neural networks are especially important and proper architectures for (nonlinear) system modelling
- General solutions: NN and fuzzy-neural systems are universal modeling devices (universal approximators)
- The importance of the theoretical results, theoretical background
- The difficulty of the application of theoretical results in practice
- The role of data base
- The importance of prior information, physical insight
- The importance of preprocessing and post-processing
- Modular approach:
  - decomposition of the problem
  - cooperation and competition
  - “experts” using different paradigms
- Hybrid solutions: combination of rule based, fuzzy, neural, mathematical

References and further readings

Parag H. Batavia, Dean A. Pomerleau, Charles E. Thorpe.: „Applying Advanced Learning Algorithms to ALVINN”, Technical Report, CMU-RI-TR-96-31 Robotics Institute Carnegie Mellon University Pittsburgh, Pennsylvania 15213-3890

Berényi, P.,, Horváth, G., Pataki, B., Strausz, Gy. : "Hybrid-Neural Modeling of a Complex Industrial Process" Proc. of the IEEE Instrumentation and Measurement Technology Conference, IMTC'2001. Budapest, May 21-23. Vol. III. pp. 1424-1429.

Berényi P., Valyon J., Horváth, G. : "Neural Modeling of an Industrial Process with Noisy Data" IEA/AIE-2001, The Fourteenth International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, June 4-7, 2001, Budapest in Lecture Notes in Computer Sciences, 2001, Springer, pp. 269-280.

Bishop, C, M.: “Neural Networks for Pattern Recognition” Clanderon Press, Oxford, 1995.

Horváth, G., Pataki, B. Strausz, T.: "Neural Modeling of a Linz-Donawitz Steel Converter: Difficulties and Solutions" Proc. of the EUFIT'98, 6th European Congress on Intelligent Techniques and Soft Computing. Aachen, Germany. 1998. Sept. pp.1516-1521

Horváth, G. Pataki, B. Strausz, Gy.: "Black box modeling of a complex industrial process", Proc. Of the 1999 IEEE Conference and Workshop on Engineering of Computer Based Systems, Nashville, TN, USA. 1999. pp. 60-66

Pataki, B., Horváth, G., Strausz, Gy., Talata, Zs. "Inverse Neural Modeling of a Linz-Donawitz Steel Converter" e & i Elektrotechnik und Informationstechnik, Vol. 117. No. 1. 2000. pp.

References and further readings

Strausz, Gy., G. Horváth, B. Pataki : "Experiences from the results of neural modelling of an industrial process" Proc. of Engineering Application of Neural Networks, EANN'98, Gibraltar 1988. pp. 213-220

Strausz, Gy., G. Horváth, B. Pataki : "Effects of database characteristics on the neural modeling of an industrial process" Proc. of the International ICSC/IFAC Symposium on Neural Computation / NC'98, Sept. 1998, Vienna pp. 834-840.

Bishop, C, M.: "Neural Networks for Pattern Recognition" Clanderon Press, Oxford, 1995.

Horváth, G (ed.)," Neural Networks and Their Applications", Publishing house of the Budapest University of Technology and Economics, Budapest, 1998. (in Hungarian)

Jang, J. -S. R., Sun, C. -T. and Mizutani: E. „Neuro-Fuzzy and Soft Computing. A Computational Approach to Learning and Machine Intelligence", Prentice Hall, 1997.

Jang, J. -S. R: „ANFIS: Adaptive-Network-Based Fuzzy Inference System" IEEE Trans. on Sysytem Man, and Cybernetics. Vol. 23. No.3. pp. 665-685, 1993.

Nguyen, D. and Widrow, B. (1989). "The Truck Backer-Upper: An Example of Self-Learning in Neural Networks," in *Proceedings of the International Joint Conference on Neural Networks* (Washington, DC 1989), vol. II, 357-362.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986b). "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. I, D. E. Rumelhart, J. L. McClelland, and the PDP Research Group. MIT Press, Cambridge (1986).